Distributed machine learning with communication constraints

Yuchen Zhang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-47
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-47.html

May 11, 2016

Distributed Machine Learning with Communication Constraints

by

Yuchen Zhang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael I. Jordan, Co-chair
Professor Martin J. Wainwright, Co-chair
Associate Professor Benjamin Recht
Assistant Professor Aditya Guntuboyina

Spring 2016

Distributed Machine Learning with Communication Constraints

Abstract

Distributed Machine Learning with Communication Constraints

by

Yuchen Zhang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Michael I. Jordan, Co-chair

Professor Martin J. Wainwright, Co-chair

Distributed machine learning bridges the traditional fields of distributed systems and machine learning, nurturing a rich family of research problems. Classical machine learning algorithms process the data by a single-thread procedure, but as the scale of the dataset and the complexity of the models grow rapidly, it becomes prohibitively slow to process on a single machine. The usage of distributed computing involves several fundamental trade-offs. On one hand, the computation time is reduced by allocating the data to multiple computing nodes. But since the algorithm is parallelized, there are compromises in terms of accuracy and communication cost. Such trade-offs puts our interests in the intersection of multiple areas, including statistical theory, communication complexity theory, information theory and optimization theory.

In this thesis, we explore theoretical foundations of distributed machine learning under communication constraints. We study the trade-offs between communication and computation, as well as the trade-off between communication and learning accuracy. In particular settings, we are able to design algorithms that don't compromise on either side. We also establish fundamental limits that apply to all distributed algorithms. In more detail, this thesis makes the following contributions:

- We propose communication-efficient algorithms for statistical optimization. These algorithms achieve the best possible statistical accuracy and suffer the least possible computation overhead.
- We extend the same algorithmic idea to non-parametric regression, proposing an algorithm which also guarantees the optimal statistical rate and superlinearly reduces the computation time.
- In the general setting of regularized empirical risk minimization, we propose a distributed optimization algorithm whose communication cost is independent of the data size, and is only weakly dependent on the number of machines.

- We establish lower bounds on the communication complexity of statistical estimation and linear algebraic operations. These lower bounds characterize the fundamental limits of any distributed algorithm.
- We design and implement a general framework for parallelizing sequential algorithms. The framework consists of a programming interface and an execution engine. The programming interface allows machine learning experts to implement the algorithm without concerning any detail about the distributed system. The execution engine automatically parallelizes the algorithm in a communication-efficient manner.

To my parents

Contents

Acknowledgments

There are so many people that I want to sincerely thank for their presence during the five years of my PhD career. Without their help, I would not have been able to enjoy research and life at Berkeley as much as I did. I am grateful to be able to acknowledge their contributions here, and apologize if I forgot to mention some of them in the coming paragraphs.

The two people that influenced me the most are my research advisers: Michael Jordan and Martin Wainwright. Although both of them are prestigious professors, I got the feeling that they are my peers during the discussion of research problems — they are extremely friendly, patient, open-minded and curious to new ideas. The culture in Mike and Martin's groups is unique. I was encouraged by both advisers to pursue different research directions, and enjoy the complete freedom to choose topics that interest me. During my earlier years at Berkeley, I was surprised to find out that the group's interests are so diversified, that it is difficult to describe my advisers' research field in short words. Later I realized that it was this vibrant culture that creates an ideal environment for conducting high-quality research. Outside of the academic world, I got enormous positive energy from my advisers regarding the attitude towards life. Mike taught me to make career decisions following my internal passion, instead of driven by secondary factors such as money or people's perspective. Martin convinced me of the same thing during our chats in his office and at the restaurant. In addition, both of them demonstrated me the ideal way of securing a work-life balance.

I also want to thank John Duchi, who helped me so much during my first two years at Berkeley when both of us were members of Mike and Martin's group. As a junior graduate student, I had very little idea on research, writing and presentation. It was John who guided me to take the first step, and until now he is my role model in being a successful young researcher. In addition, I would like to thank Pieter Abbeel who was my temporary adviser for the first year, and Aditya Guntuboyina, who meets me regularly every week to teach me tons of statistics.

I had three wonderful internships at Google, Microsoft Research and Baidu. I must thank my mentor Vanja Josifovski at Google, who shared his great insight with me in addressing real problems. I appreciated the constructive discussions with Alex Smola and Amr Ahmed. At Microsoft, I enjoyed a wonderful summer internship with Lin Xiao. Lin is extremely nice, humorous, and always has a sharp theoretical insight. The collaboration with Lin has renewed my understanding on the field of optimization. I also thank Denny Zhou, who is a fantastic collaborator and friend at MSR. At Baidu, I want to thank my mentor Lin Yan and all colleagues in the fresh-search team. Although the internship was short, I have learnt a lot in contributing to real products. I am also grateful for Baidu for funding my PhD through scholarships.

I own credit to many people who are at Berkeley and who used to be at Berkeley. First, I want to express gratitude for the collaboration with Xi Chen. Xi is not only a great collaborator, but also a great friend who shared with me his invaluable understandings on the academic job market. I also want to thank Sivaraman Balakrishnan, Chi Jin and Jason Lee. It was a great pleasure to work with you guys on theoretical problems. I am grateful

for Ben Recht for being the chair of my Qual Exam committee and standing in my thesis committee, and for providing extremely useful feedbacks. I would like to thank my peer fellows in the EECS department and the Statistics department: Andre Wibisono, Xinghao Pan, Robert Nishihara, Philipp Moritz, Ahmed El Alaoui, Nihar Shah, Mert Pilanci, Yuting Wei, Fanny Yang, Aaditya Ramdas, Yudong Chen and Yun Yang: I have learnt much from every one of you. I want to thank my roommate, Qi Zhang, for our endless debates have made the life so much fun.

Finally, I cannot find the right words to thank my parents who have loved and supported me throughout this PhD. If it weren't for their constant motivation and belief, I would have probably never made this far and I owe all my success to them. A very special thanks to Wei, I am very fortunate to have your love and support, as well as your companion in Beijing, San Francisco Bay Area, Lijiang, Japan and New York.

Part I

Introduction and background

Chapter 1

Introduction

The marriage of distributed computing and machine learning nurtures a rich family of research problems. On one side, since traditional distributed systems were designed to serve as a general computing platform, distributed computing theorists study relevant topics — communication complexity [110], consensus [112, 111] and coding theory [19] — to understand the properties of such systems. On the other side, machine learning researchers care about finding regular patterns from noisy datasets, thus this necessities the study of statistical theory [74, 205], information theory [53] and optimization algorithms [154]. Distributed machine learning bridges the two fields, thus puts our interests in the intersection of the above two groups of topics. In this thesis, we explore the theoretical foundation of distributed machine learning, and in particular show how theoretical understanding leads to novel algorithms and practical systems.

1.1 Motivations

Machine learning techniques have helped scientists to discover principles of the nature which were often hidden behind large amounts of noisy observations. For instance, the Large Hadron Collider (LHC) was built to record collisions between protons and ions, which happen billions of times per second. Scientists are interested in some particular physics process, such as when a Higgs boson decaying into two tau particles, but this decay is a small signal buried in background noise. Machine learning techniques can help identifying such subtle events from the overwhelmingly noisy observations [212, 3]. Machine learning was also heavily used to process satellite data in atmospheric physics [108], and to analyze the genome sequencing data in genomics [51, 123].

The common character of the above applications is their need to process massive amount of data. Since the datasets are automatically collected, they easily reach terabytes or petabytes in scale. For example, the worldwide LHC computing grid can generate one petabyte of data per month [193], taking up to 250,000 standard DVDs to record. As another example, the size of sequencing data generated on human genomes will reach 250

petabytes in 3-5 years [61]. Even a subset of these datasets cannot fit into a single computer. Thus, distributed computing becomes an inevitable choice.

The same scalability issue arises in the artificial intelligence research where the Internet serves as a major data source. For instance, to learn a ranking function for a search engine, the learning algorithm resorts to the search logs that were collected from billions of users [36, 165, 44]. Storing and processing such large-scale data necessitates distributed computing. In applications like computer vision and natural language processing, researchers train very complex models that involves billions of parameters, such as convolutional neural networks for image classification [109, 201, 194] or long-short term memory models for machine translation [90, 11]. Distributed machine learning algorithms are employed to learn such models across thousands of machines [57, 46, 1, 148].

Traditionally, the performance of machine learning algorithms have been evaluated in two dimensions. The first is the statistical accuracy, measuring how close the acquired knowledge is to the truth, or how accurate the prediction is made on new instances. The second is the computational efficiency, measured by the CPU time on a single processor. The study of distributed machine learning adds a third dimension called *communication efficiency*. It measures the overhead of exchanging messages across the network. Studies on a variety of distributed computing platforms confirm that communication can become a significant bottleneck on algorithmic efficiency [218, 178, 140, 40, 70]. In particular, the overhead for a single message exchange can be long enough for thousands or more floating-point operations [218]. Without a carefully designed protocol, the communication time will dominate the computation time.

In this thesis, we study algorithms, theories and systems for machine learning when the communication efficiency is a constraint. We examine how the classical design and analysis of learning algorithms can often fail to address the challenge of distributed computing, then we seek to obtain new solutions to address the challenge, as well as rigorous theoretical interpretation to these solutions.

1.2 Trade-offs in distributed computing

Given the distributed nature of the system, there are fundamental trade-offs which must be taken into account. In this section, we interpret how these trade-offs correspond to critical challenges that we will address in this thesis.

1.2.1 Communication versus computation

There is an fundamental trade-off between computation and communication. Indeed, when a sequential algorithm is parallelized across multiple machines, all computing nodes need to synchronize their states during the execution of the algorithm, which incurs communication cost. Although parallelization reduces the computation time, it often incurs nonneglible communication overhead.

We illustrate this trade-off using a concrete example. A typical example is the application of the stochastic gradient descent (SGD). Suppose that we want to minimize an objective function $f : \Theta \to \mathbb{R}$ with respect to parameter $\theta \in \Theta$. The SGD algorithm starts from a fixed initial parameter $\theta^{(0)}$. At iteration t, it takes a noisy gradient vector g_t with respect to the parameter $\theta^{(t)}$, such that $\mathbb{E}[g_t] = \nabla f(\theta^{(t)})$, then the algorithm updates the vector by:

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t g_t, \tag{1.1}$$

where η_t is the stepsize of iteration t. Typically, the SGD algorithm terminates when the parameter $\theta^{(t)}$ converges to a stationary point.

Typically SGD takes a large number of iterations to converge. To utilize the computation power of a distributed system, we want to perform the update formula (1.1) in parallel. More precisely, we consider the following approach: at the t-th iteration, every machine computes an independent copy of the noisy gradient based on its local data. The stochastic gradient is then computed as an average of local noisy gradients: $g^{(t)} = \frac{1}{m}\sum_{i=1}^{m} g_i^{(t)}$ where $g_i^{(t)}$ represents the noisy gradient on the i-th machine. Since averaging reduces the variance, if $m > 1$, then the averaged gradient $g^{(t)}$ will have less variance than any of the local gradients. It is known that using gradients with reduced variance accelerates the SGD convergence [see e.g. 59]. Thus, for achieving any pre-specified error bound, the computation time of SGD will be reduced by using $m > 1$ machines.

However, since computing the averaged gradient $g^{(t)}$ requires aggregating the local information on m machines, it is necessary for these machines to synchronize. At each iteration, the noisy gradients are aggregated at a master node, then the updated parameter is broadcasted from the master node to all machines. Since each round of communication is an all-reduce operation [56], it can be implemented in $\mathcal{O}(m)$ time for star-shaped networks, or in $\mathcal{O}(\log(m))$ time for fully connected networks. In either case, the communication time grows with the number of machines. It means that the more parallel machines we use, the less iterations that SGD requires to converge to a desired accuracy, but the more time that we have to spend on communication.

The communication overhead can become a severe performance bottleneck on commodity clusters. For example, an Amazon EC2 node takes a few micro-seconds to process a single data point for SGD, but it takes hundreds of micro-seconds to perform one round of map-reduce operation. Thus, if the stochastic gradients are computed on single data points, then the time of communication will be two orders-of-magnitude greater than that of computation. This phenomenon raises open questions on algorithm design. If we want to fully utilize the computation resource in a distributed system, we have to be careful in preserving the communication efficiency.

1.2.2 Communication versus accuracy

Even without the concern of computation overhead, there is a trade-off between the communication cost and the statistical accuracy. This is because that communication is an essential

step to aggregate useful information across separate locations. Let's consider examples in a typical statistical estimation setting. In this setting, the data is stored on multiple machines, but they are are i.i.d. drawn from a common distribution P_θ, where θ is a parameter that characterizes the properties of the distribution. The goal of statistical estimation is to estimate θ. Without communication, the estimation can be performed on a single machine. It can still be a reasonable estimation because the data is identically distributed on all machines. However, the small amount of local data makes the estimation less accurate. To achieve a high accuracy, it is necessary to aggregate information across machines, which incurs communication cost. If the communication cost is a main bottleneck on algorithmic efficiency, then we want to impose a budget B on communication. More precisely, at most B bits of data is allowed to be exchanged across the network. Under this constraint, we want to answer the following question: what is the best possible accuracy that can be achieved for statistical estimation? And further more, what is the algorithm that achieves this optimal rate?

The trade-off between communication and accuracy is a more practical concern when the data is high-dimensional. For a concrete example, suppose that a design matrix $X \in \mathbb{R}^{n\times d}$ is stored separately on many machines, where n is the number of data points and d is the dimension of each point. The goal is to compute the rank of X in order to explore the structure of the data. Although the computation result can be encoded in $\mathcal{O}(\log(d))$ bits, the communication cost is substantially higher than that. In particular, we assume that the i-th machine has n_i data points, so that its local design matrix is $X_i \in \mathbb{R}^{n_i\times d}$. A simple algorithm is the following: the i-th machine computes the covariance matrix $X_i^T X_i$ and send it to the master node, then the master node aggregates the local covariance matrices to compute $X^T X$, using which the rank of X is obtained. Although this algorithm computes the exact rank of the matirx, its communication cost is $\mathcal{O}(d^2)$, which might be too expensive for large d. To give the reader a concrete idea of how expensive it is, the URL Reputation dataset in the UCI machine learning repository [28] contains $n = 2,396,130$ data points with $d = 3,231,961$ dimensions. As a result, roughly 10 trillion matrix entries will be communicated. An alternative algorithm is to transmit all data points to a single node to compute the rank, but it is still communication inefficient. Given the situation, it is reasonable to consider approximate algorithms, which allows the result to be inexact, but is a good enough approximation to the exact rank. Again, we want to impose a budget B on the total number of bits exchanged across the network, and asking what is the best possible approximation rate under the communication constraint.

1.2.3 Efficiency versus ease of implementation

Another important trade-off is between the algorithmic efficiency and the ease of implementation. Remember that to implement a distributed algorithm, one has to manage the collaboration between all computing nodes in the distributed system. Converting an existing single-thread algorithm to a distributed algorithm is challenging, even given pre-implemented message passing interfaces [e.g. 79, 215]. This is because that many decisions must be cor-

rectly made to build an efficient implementation. These decisions include the design of the data partitioning scheme, the design of communication protocols and conflict management, fault tolerance, etc. Thus, it is desirable to have a general and user-friendly programming interface that can convert existing algorithms to parallellized algorithms. In addition, there should be an execution engine that can efficiently execute the converted algorithms on a distributed system.

1.3 Connections to existing work

In this section, we survey some of the existing lines of research that explore themes related to the work in this thesis. Since the topics covered by this thesis are broad, we will start by surveying the core problem – distributed statistical optimization. Then we review related areas, including non-parametric regression, minimax theory, communication complexity theory and machine learning systems.

1.3.1 Distributed statistical optimization

Many machine learning tasks can be formulated as a statistical optimization problem. Given the following objective function:

$$f(\theta) = \frac{1}{n}\sum_{j=1}^{n} \phi(\theta; z_j), \tag{1.2}$$

the goal is to find a minimizer of the population risk $\mathbb{E}[f(\theta)]$, where the expectation is taken with respect to the random data points z_j. Here, the data points $\{z_j\}_{j=1}^n$ are i.i.d. sampled from an underlying distribution. The function $\phi(\cdot; z_j)$ is the loss function on the j-th data point. For example, we obtain linear regression if ϕ is chosen as the least-square loss; we obtain logistic regression or support vector machine if ϕ is chosen as the logistic loss or the hinge loss.

Since the population risk $\mathbb{E}[f(\theta)]$ is unavailable, a popular work-around is to compute a minimizer of the empirical function f. This approach is called empirical risk minimization (ERM) and it is well-studied in literature. In particular, approaches based on VC theory [208, 207], metric entropy [105, 74], Rademacher and Gaussian complexities [16, 117, 202] have all contributed to characterizing the statistical property of the empirical risk minimizer.

If the function $\phi(\theta; z)$ is convex respective to θ, then the empirical risk minimizer can be computed via convex optimization methods. Among other algorithms, the full gradient method and its accelerated variants [156, 180, 17, 155] are easy to parallelize under the map-reduce framework. But for objective functions with large condition numbers (see definition in Section 1.4.4), these methods often requires taking many passes over the data before converging to a satisfactory accuracy. It thus makes them less efficient in processing large-scale datasets. The stochastic gradient method and its variants [226, 31, 195, 114, 216]

achieves the same accuracy by taking a substantial less number of passes over the data. But because of the incremental nature of these algorithms, each individual update relies on the outcome of all previous updates, which makes them difficult to parallelize.

Recent years witnessed a flurry of research on distributed approaches to solving large-scale statistical optimization problems. Nedić and Ozdaglar [151], Ram et al. [168] and Johansson et al. [98] studied incremental sub-gradient methods, which involve every machine minimizing its own objective function while exchanging information locally with other machines in the network over a time-varying topology. Duchi et al. [64] studied a distributed dual averaging algorithm. Dekel et al. [59] proposed a mini-batch training approach to accelerate the optimization on smooth functions, and Duchi et al. [65] extended the method to nonsmooth optimization. Recht et al. [172] and Agarwal and Duchi [4] studied the parallelized SGD algorithm with asynchronous communication. Jaggi et al. [97] proposed a distributed dual coordinate ascent algorithm. Shamir et al. [189] proposed a distributed approximate newton-type method, which communicates like a first-order method, but converges like a second-order method for quadratic optimization.

Despite the rich literature, most algorithms involve high communication cost. In particular, their iteration complexity have similar or worse dependency on the condition number as the classical accelerated gradient method. This suggests researchers to look into further structures of the problem. In statistical optimization setting, we have made the critical assumption that the data points $\{z_j\}_{j=1}^n$ are i.i.d. sampled. Under this assumption, we will present novel alogrithms that enjoy better communication efficiency, both in theory and in practice.

1.3.2 Non-parametric regression

Non-parametric regression is a classical problem in machine learning and statistics. The goal is to fit an "infinite dimensional" model, such as a function, from a finite dataset. This topic is interesting to the theme of this thesis, because the traditional algorithms for non-parametric methods are often computationally expensive, thus using distributed computing may substantially improve the computation efficiency.

The goal of non-parametric regression is to find a function $f \in \mathcal{F}$ which minimizes the population risk $\mathbb{E}[(f(x) - y)^2]$. Here, we assume that the pair (x, y) are randomly sampled from an underlying distribution, and the function f belongs to a predefined function class $\mathcal{F}$. When the population risk is usually unavailable, one computes an estimate of function f based on a finite set of n samples. Researchers have studied a wide range of such estimators (see the books by Gyorfi et al. [82], Wasserman [211] and van de Geer [74] for examples). One class of methods, known as regularized M-estimators [74], are based on minimizing a regularized version of the empirical risk. If the function class $\mathcal{F}$ is a reproducing kernel Hilbert space (RKHS), then the corresponding non-parametric regression problem is called kernel ridge regression (KRR). It is one of the most widely-used non-parametric method in practice [see e.g. 87, 192].

The standard implementation of KRR requires $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^3)$ memory, which is prohibitively expensive for large sample size n. As a consequence, approximations have been designed to avoid the expense of finding an exact minimizer. One family of approaches is based on low-rank approximation of the kernel matrix; examples include kernel PCA [183], the incomplete Cholesky decomposition [71], or Nystrom sampling [213]. These methods reduce the time complexity to $\mathcal{O}(dn^2)$ or $\mathcal{O}(d^2n)$ where $d \ll n$ is the preserved rank. In this thesis, we propose a communication-efficient distributed algorithm for solving this problem. The algorithm has rigorous theoretical guarantees, and enjoys better computation efficiency then existing approaches.

1.3.3 Minimax theory

The fundamental limit of statistical estimation has been studied by the minimax theory. We review the related literature as it is relavant to our study of fundamental limits under the communication constraint.

In a statistical estimation setting, the data points are sampled from an underlying distribution $P_\theta \in \mathcal{P}$ characterized by the parameter θ. The goal is to estimate θ based on the data points. The minimax error of an estimation problem can be described in the following game-theoretic setting: the statistician chooses an optimal data-based estimator $\widehat{\theta}$, then the adversary chooses a worst-case parameter θ so that the distribution P_θ generates the data $Z \in \mathcal{Z}^n$ consisting of n i.i.d. samples. The minimax error is then defined as:

$$\mathfrak{M}_n(\mathcal{P}, \Theta) := \inf_{\widehat{\theta}} \sup_{\theta \in \Theta} \mathbb{E}_\theta[\|\widehat{\theta}(Z) - \theta\|_2^2]. \tag{1.3}$$

There are a variety of techniques for providing lower bounds on the minimax risk (1.3). These techniques use information theoretic arguments to prove that, as long as the sample size n is finite, the data doesn't provide sufficient information to distinguish the true parameter θ with some alternative parameter θ' which is close to θ. As a consequence, any estimator suffers a positive estimation error. When the lower bound on the minimax error is tight (when it matches the upper bound achieved by a concrete estimator), it characterizes the intrinsic hardness of the estimation problem.

The minimax error can be lower bounded by several techniques, including the classicial Le Cam's method [115, 223]. The Le Cam's method reduces the parameter estimation problem to a binary hypothesis testing problem, then uses an information theoretic argument to establish a lower bound on the testing error. A more general method for lower bounding the testing error is based on Fano's method, beginning with the pioneering work of Hasminskii and Ibragimov [104, 96], and followed by more recent works in a variety of settings [e.g. 24, 223, 219, 25, 170, 81, 37]. Fano's method is often more general than the Le Cam's method because it is capable of addressing multiple-hypothesis testing. In special settings, the Assouads lemma [7] is also a useful tool for proving minimax lower bounds. See Yu's paper [223] for a comprehensive survey on Le Cam's method, Fano's method and Assound's lemma.

In this thesis, we borrow ideas from the minimax theory, but the problem to be solved is different. In the classical setting (i.e., equation (1.3)), the estimator $\widehat{\theta}$ can perform arbitrary computation on the data Z. In our setting, the data is separately stored, thus computing $\widehat{\theta}$ involves inter-machine communication. We are interested in characterizing the minimax error rate under a pre-defined communication constraint.

1.3.4 Communication complexity theory

The research on communication complexity has a long history, dating back to the seminal work of Yao [221] and Abelson [2]. In the basic setting of communication complexity, two players Alice and Bob wish to compute a function $f : X \times Y \to \{0, 1\}$ where X and Y are arbitrary finite sets. Alice holds an input $x \in X$, Bob $y \in Y$, and they wish to evaluate $f(x, y)$ while minimizing the number of bits communicated. See the books by Kushilevitz and Nisan [110], Lee and Shraibman [118] and Hromkovič [94] for a complete survey on this topic.

In this thesis, we are interested in a particular case: the inputs are matrices and the output is a linear algebraic function on the matrix. For example, suppose that, x and y are n-by-n matrices and the function f is the rank of the matrix $x + y$. Note that the rank of the matrix can be encoded in $\mathcal{O}(\log(n))$ bits, but the communication cost for computing this rank may be substantially higher than that. Bounding the communication complexity is challenging because the target function has no closed-form expression.

Characterizing the communication complexity of linear algebraic operations is a fundamental question, and here we review existing literatures. For the problem of rank testing, Chu and Schnitger [47, 48] prove the $\Omega(n^2)$ communication complexity lower bound for deterministically testing the singularity of integer-valued matrices. A successful algorithm for this task is required to distinguish two types of matrices—the singular matrices and the non-singular matrices with arbitrarily small eigenvalues—a requirement that is often too severe for practical applications. Luo and Tsitsiklis [135] prove an $\Omega(n^2)$ lower bound for computing one entry of A^{-1}, applicable to exact algorithms (with no form of error allowed). However, the lower bound doesn't apply to approximate algorithms which are widely used in practice. For randomized algorithms, Li et al. [200, 122] prove $\Omega(n^2)$ lower bounds for the problems of rank testing, computing a matrix inverse, and solving a set of linear equations over finite fields. To the best of our knowledge, it is not known whether the same lower bounds hold for matrices in the real field. In other related work, Clarkson and Woodruff [49] give an $\Omega(r^2)$ space lower bound in the streaming model for distinguishing between matrices of rank r and $r - 1$. However, such a space lower bound in the streaming model does not imply a communication complexity lower bound in the interactive communication model studied in this thesis.

In this thesis, we establish upper and lower bound for computing the rank of a distributively stored matrix. We allow the algorithm to output approximate solutions. In this setting, there is an algorithm achieving much better communication efficiency than the lower bounds mentioned above. We also establish lower bounds that matches the new upper

bound, and demonstrate that the performance gap between deterministic algorithms and randomized algorithms is big.

1.3.5 Distributed machine learning systems

Distributed machine learning systems have been implemented for a variety of applications and are based on different programming paradigms. Related frameworks include parameter servers [164, 57, 6, 121], Petuum [217], Naiad [149] and GraphLab [131]. There are also machine learning systems built on existing platforms, including Mahout [73] based on Hadoop [72] and MLI [196] based on Apache Spark [224]. These systems provide simple programming interfaces, so that the user can implement an efficient distributed algorithm without dealing with low-level communication protocols.

Most of these frameworks are efficient in parallelizing batch algorithms, that is, the algorithm which processes a large bulk of data points in every iteration. If the batch is big, then the computation time will be higher than the communication time, so that communication will no longer be an efficiency bottleneck. However, none of these systems are explicitly designed for parallelizing (sequential) stochastic algorithms (e.g. SGD), which are most popular for processing large-scale datasets. Stochastic algorithms are difficult to parallelize because the algorithmic step on every data point relies on the outcome of processing all previous data points. One contribution of this thesis will be designing a user-friendly programming interface for parallelizing stochastic algroithms on Apache Spark [224].

Before presenting our framework, we provide a detailed review of existing works. Mahout and MLI, both adopting the iterative MapReduce [56] framework, are designed for batch algorithms. The parameter servers, Petuum and Naiad provide user-definable update primitives such as (`get`, `set`) on variables or (`pull`, `push`) on messages, under which a distributed stochastic algorithm can be implemented. However, a typical stochastic algorithm updates its parameters in every iteration, which involves expensive inter-node communication. In practice, we found that the per-iteration computation usually takes a few microseconds, but pushing an update from one Amazon EC2 node to another takes milliseconds. Thus, the communication cost dominates the computation cost. If the communication is asynchronous, then the algorithm will easily diverge because of the significant latency.

GraphLab asynchronously schedules communication using a graph abstraction, which guarantees the serializability of the algorithm. Many stochastic algorithms can be written as a graph-based program. For the SGD algorithm, one constructs a vertex for every sample and every feature, and connects an edge between the vertices if the sample processes a particular feature. However, when the individual feature is shared among many samples, running SGD on this graph will cause many conflicts, which significantly restricts the degree of parallelism. Such a paradigm is efficient only if the feature is very sparse.

1.4 Contributions of this thesis

This section highlights our contribution to address the challenges in Section 1.2. At high level, the goal of this thesis is to solve the following problems:

- Designing communication-efficient algorithms for machine learning. The algorithm should be able to achieve optimal statistical accuracy, utilizing the power of distributed computing to reduce the computation overhead, and make sure that the communication overhead won't be an efficiency bottleneck.
- Characterizing the fundamental trade-offs between communication and accuracy. For a pre-specified communication budget, we want to characterize the best possible accuracy that can be achieved by any algoithm satisfying the budget constraint.
- Designing and implementing a programming paradigm for distributed computing, so that the machine learning experts can implement their favorite algorithms without knowing details about the distributed system. The system should be able to automatically parallelize the algorithm in a communication efficient manner.

To achieve the goals of this thesis, we present novel algorithms, theories and systems. The rest of this section describes the details of our contribution.

1.4.1 Distributed algorithms

The first two chapters are devoted to divide-and-conquer methods. Divide-and-conquer is a natural approach to distributed computation. It splits a big problem into smaller subproblems, solving them separately before merging into a global solution. It is a communication-efficient approach because the algorithm requires only one round of synchronization at the end. However, the merged solution is not equal to the solution computed by a centralized program, thus the algorithm usually suffers sub-optimal performance.

Perhaps the simplest divide-and-conquer algorithm for distributed statistical optimization is what we term the *average mixture* (Avgm) algorithm. It is an appealingly simple method: given m different machines and a dataset of size N, first assign to each machine a (distinct) dataset of size $n = N/m$, then have each machine i compute the empirical minimizer θ_i on its fraction of the data, and finally average all the parameter estimates θ_i across the machines. This approach has been studied for some classification and estimation problems by Mcdonald et al. [142] and McDonald et al. [143], as well as for certain stochastic approximation methods by Zinkevich et al. [238]. To the best of our knowledge, however, no work has shown rigorously that the Avgm procedure generally has greater efficiency than the naive approach of using $n = N/m$ samples on a single machine.

In Chapter 3, we present two main results for divide-and-conquer statistical optimization. First, we provide a sharp analysis of the Avgm algorithm, showing that under a reasonable set of conditions on the population risk, and under the condition that $m < n$, it can achieve

the optimal statistical accuracy using all N samples. Second, we develop a novel extension of simple averaging. It is based on an appropriate form of resampling [68, 84, 161], which we refer to as the *subsampled average mixture* (SAVGM) approach. At a high level, the SAVGM algorithm distributes samples evenly among m processors or computers as before, but instead of simply returning the empirical minimizer, each processor further subsamples its own dataset in order to estimate the bias of its own estimate, and returns a subsample-corrected estimate. We establish that the SAVGM algorithm has mean-squared error decaying as $\mathcal{O}(m^{-1}n^{-1}+n^{-3})$. As long as $m < n^2$, the subsampled method again matches the optimal statistical rate. Thus the SAVGM method is more robust to a high degree of parallelism than AVGM. The theories presented in this section are complemented by experiments on a large-scale click prediction problem.

In Chapter 4, we extend the idea of divide-and-conquer to non-parametric regression. The algorithm is still simple: we partition the dataset of size N randomly into m equal sized subsets, and we compute the kernel ridge regression estimate $\widehat{f}_i$ for each of the $i = 1, \ldots, m$ subsets independently, with a *careful choice* of the regularization parameter. The estimates are then averaged via $\bar{f} = (1/m)\sum_{i=1}^{m} \widehat{f}_i$. Our main theoretical result gives conditions under which the average $\bar{f}$ achieves the minimax rate of convergence over the underlying Hilbert space. Even using naive implementations of KRR and with single-thread computing, this decomposition gives time and memory complexity scaling as $\mathcal{O}(N^3/m^2)$ and $\mathcal{O}(N^2/m^2)$, respectively, while the traditional algorithm requires $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ time and space (see Section 1.3.2). As concrete examples, our theory guarantees that the number of subsets m may grow nearly linearly for finite-rank or Gaussian kernels and polynomially in N for Sobolev spaces, which in turn allows for substantial reductions in computational cost. Our approach dovetails naturally with parallel and distributed computation: we are guaranteed superlinear speedup with m parallel processors (though we must still communicate the function estimates from each processor). The theoretical results are complemented by experiments on the Yahoo! music year prediction problem, which confirms that the new algorithm is orders-of-magnitude faster than the traditional approach for KRR.

In Chapter 5, we explore a more general setting of statistical optimization where the assumptions made in Chapter 3 fails. In particular, the AVGM algorithm requires the objective function to be strongly convex. Here, we consider a more general setup, where the objective function is not strongly convex but regularized by the squared ℓ_2-norm. The regularization parameter is decreasing at the rate $1/\sqrt{N}$ as in the standard setting of supervised learning. In this setting, the AVGM algorithm will suffer from sub-optimal performance. In this chapter, we propose a communication-efficient distributed algorithm called DiSCO to minimize the regularized loss. We analyze its iteration complexity and communication efficiency, and discuss the results for distributed ridge regression, logistic regression and binary classification with a smoothed hinge loss. We prove that the required number of communication rounds of the algorithm does not increase with the sample size, and only grows slowly with the number of machines. Experiments confirm that, the DiSCO algorithm achieves the state-of-the-art efficiency on large-scale distributed optimization, outperforming the ADMM algorithm [34], the accelerated full gradient method [156], and the L-BFGS quasi-Newton method [159].

1.4.2 Theories of distributed computing

While there is a rich literature on statistical minimax theory [e.g. 96, 223, 219, 205], little of it characterizes the effects of limiting communication. In other areas, ranging from theoretical computer science [221, 2, 110], decentralized detection and estimation [204, 136], to information theory [e.g. 85, 69], there is a substantial literature on communication complexity, but they don't share the same setups as in machine learning problems. Our theoretical results targets at bridging this gap.

In Chapter 6, we formulate and study the problem of distributed statistical estimation. We consider two variants, one based on protocols that engage in only a single round of message-passing, and the other based on interactive protocols that can use multiple rounds of communication. The main question of interest is the following: how must the communication budget B scale as a function of the sample size n at each machine, the total number of machines m, and the problem dimension d so that the distributed protocol matches the accuracy of the best centralized estimator?

A trivial lower bound on the communication budget is the number of bits reqiured to encode the problem solution. However, for some important problems, we demonstrate that the communication requirement must be exponentially greater. For example, we show that for problems such as location estimation in Gaussian and binomial families, the amount of communication must scale linearly in the product dm of the dimension number of machines m, which is exponentially larger than the $O(d \log m)$ bits required to specify the problem or communicate its solution. The same conclusion can be extended to linear regression and probit regression. To exhibit these gaps, we provide lower bounds using novel information-theoretic techniques. We also establish sharp upper bounds to demonstrate the tightness of the lower bounds.

In Chapter 7, we study an alternative setting, where the goal is to compute the generalized rank of a matrix: given an $n \times n$ matrix and a constant $c \geq 0$, estimate the number of eigenvalues that are greater than c. We demonstrate that the rank estimation problem is of essential importance in practice, and it is connected to several other important problems in linear algebra and convex optimization. In the distributed setting, the matrix of interest is the sum of m matrices held by separate machines. The question is how many bits have to be communicated to yield a good enough approximate solution. Differing from traditional communication complexity setups, the quantity that we compute here doesn't have a closed form expression.

For this seemingly simple problem, we show that any deterministic algorithm must communicate $\Omega(n^2)$ bits, which is order-equivalent to transmitting the whole matrix. It implies that no deterministic algorithm can be communication efficient. In contrast, we propose a randomized algorithm that communicates only $O(n)$ bits. We demonstrate the sharpness of the upper bound by proving a $\Omega(n)$ lower bound on the randomized communication complexity.

1.4.3 Distributed systems

As described in Section 1.3.5, stochastic algorithms are efficient approaches to solving machine learning and optimization problems, but no existing distributed machine learning framework is explicitly designed for parallelizing stochastic algorithms. In Chapter 8, we present a general framework called Splash for parallelizing stochastic algorithms on multi-node distributed systems. Splash consists of a programming interface and an execution engine. Using the programming interface, the user develops sequential stochastic algorithms without concerning any detail about distributed computing. The algorithm is then automatically parallelized by a communication-efficient execution engine.

Theoretically, we prove that Splash achieves the optimal rate of convergence for parallelizing SGD, assuming that the objective function is smooth and strongly convex. We conduct extensive experiments on a variety of stochastic algorithms, including algorithms for logistic regression, collaborative filtering and topic modeling. The experiments verify that Splash can yield orders-of-magnitude speedups over single-thread stochastic algorithms and over state-of-the-art batch algorithms.

Besides its performance, Splash is a contribution on the distributed computing systems front, providing a flexible interface for the implementation of stochastic algorithms. We build Splash on top of Apache Spark [224], a popular distributed data-processing framework for batch algorithms. Splash takes the standard Resilient Distributed Dataset (RDD) of Spark as input and generates an RDD as output. The data structure also supports default RDD operators such as Map and Reduce, ensuring convenient interaction with Spark. Because of this integration, Splash works seamlessly with other data analytics tools in the Spark ecosystem, enabling a single system to address the entire analytics pipeline.

1.4.4 Previously published works

The results in this thesis are based on the previously published works with several collaborators. Chapter 3 and Chapter 4 are based on joint works with John Duchi and Martin Wainwright [231, 233]. Chapter 5 is a joint work with Lin Xiao [230]. Chapter 6 is based on a joint work with John Duchi, Michael Jordan and Martin Wainwright [232]. Chapter 7 is based on a joint work with Michael Jordan and Martin Wainwright [234]. Chapter 8 is a joint work with Michael Jordan [228].

Chapter 2

Background

In this chapter, we set up concepts and backgrounds that will be frequently used throughout the thesis. The primary goal is to present high-level concepts and lemmas that serve as a preliminary for a broad class of problems. In later chapters, the formulation of specific problems will be presented in more details. The reader is encouraged to read this chapter before delving into the technical details of later chapters.

2.1 Background on empirical risk minimization

In many chapters of this thesis, we will be focusing on the empirical risk minimization (ERM) approach for solving statistical optimization problems. Consider the problem

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}} \quad \mathbb{E}_z[\phi(\theta, z)], \tag{2.1}$$

where z is a random vector whose probability distribution is supported on a set $\mathcal{Z} \subset \mathbb{R}^p$, and the cost function $\phi : \mathbb{R}^d \times \mathcal{Z} \to \mathbb{R}$ is convex in θ for every $z \in \mathcal{Z}$. The expected objective function $f_0 := \mathbb{E}_z[\phi(\theta, z)]$ is referred to as the *population risk*.

In general, evaluating the expected objective function with respect to z is intractable, even if the distribution is given. The idea of ERM is to approximate the solution to (2.1) by solving a deterministic problem defined over a large number of i.i.d. (independent and identically distributed) samples generated from the distribution of z [see, e.g. 191, Chapter 5]. Suppose our distributed computing system consists of m machines, and each has access to n samples $z_{i,1}, \ldots, z_{i,n}$, for $i = 1, \ldots, m$. Then each machine can evaluate a local empirical loss function

$$f_i(\theta) =: \frac{1}{n}\sum_{j=1}^{n} \phi(\theta, z_{i,j}), \qquad i = 1, \ldots, m.$$

Our goal is to minimize the overall empirical loss defined with all mn samples:

$$f(\theta) =: \frac{1}{m}\sum_{i=1}^{m} f_i(\theta) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \phi(\theta, z_{i,j}). \tag{2.2}$$

Examples As a concrete example, we consider ERM of linear predictors for supervised learning. In this case, each sample has the form $z_{i,j} = (x_{i,j}, y_{i,j}) \in \mathbb{R}^{d+1}$, where $x_{i,j} \in \mathbb{R}^d$ is a feature vector and $y_{i,j}$ can be a target response in $\mathbb{R}$ (for regression) or a discrete label (for classification). Examples of the loss function include

- linear regression: $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, and $\phi(\theta, (x, y)) = (y - \theta^T x)^2$.
- logistic regression: $x \in \mathbb{R}^d$, $y \in \{+1, -1\}$, and $\phi(\theta, (x, y)) = \log(1 + \exp(-y(\theta^T x)))$.
- hinge loss: $x \in \mathbb{R}^d$, $y \in \{+1, -1\}$, and $\phi(\theta, (x, y)) = \max\{0,\ 1 - y(\theta^T x)\}$.

For stability and generalization purposes, we often add a regularization term $(\lambda/2)\|\theta\|_2^2$ to make the empirical loss function strongly convex. More specifically, we modify the definition of $f_i(\theta)$ as

$$f_i(\theta) =: \frac{1}{n}\sum_{j=1}^{n} \phi(\theta, z_{i,j}) + \frac{\lambda}{2}\|\theta\|_2^2, \qquad i = 1, \ldots, m. \tag{2.3}$$

For example, when ϕ is the hinge loss, this formulation yields the support-vector machine [52].

Notations Before continuing, we define the general notations that will be used in solving ERM. We use ℓ_2 to denote the usual Euclidean norm $\|\theta\|_2 = (\sum_{j=1}^{d} \theta_j^2)^{\frac{1}{2}}$. The ℓ_2-operator norm of a matrix $A \in \mathbb{R}^{d_1 \times d_2}$ is its maximum singular value, defined by

$$|\!|\!| A |\!|\!|_2 := \sup_{v \in \mathbb{R}^{d_2}, \|v\|_2 \le 1} \|Av\|_2.$$

A convex function f is λ-*strongly convex* on a set $U \subseteq \mathbb{R}^d$ if for arbitrary $u, v \in U$ we have

$$f(u) \ge f(v) + \langle \nabla f(v), u - v \rangle + \frac{\lambda}{2}\|u - v\|_2^2.$$

where $\nabla f(v)$ or $f'(v)$ denotes the gradient of function f at point v. If f is not differentiable, we may replace ∇f with any subgradient of f. If f is twice differentiable, we use $\nabla^2 f(u)$ or f'' to denote the Hessian matrix at point u. Then the definition of λ-strongly convexity is equivalent of saying

$$\nabla^2 f(u) \succeq \lambda I, \quad \forall\, u \in U.$$

Similarly the function is called L-*smooth* if

$$\nabla^2 f(u) \preceq L I, \quad \forall\, u \in U.$$

The value $\kappa = L/\lambda \ge 1$ is called the *condition number* of f, which is a key quantity in characterizing the complexity of iterative optimization algorithms.

We let $\otimes$ denote the Kronecker product, and for a pair of vectors u, v, we define the outer product $u \otimes v = uv^\top$. For a three-times differentiable function f, we denote the third

derivative tensor by $\nabla^3 f$, so that for each $u \in \operatorname{dom} f$ the operator $\nabla^3 f(u) : \mathbb{R}^{d \times d} \to \mathbb{R}^d$ is linear and satisfies the relation

$$\left[\nabla^3 f(u)(v \otimes v)\right]_i = \sum_{j,k=1}^{d} \left(\frac{\partial^3}{\partial u_i \partial u_j \partial u_k} F(u) \right) v_j v_k.$$

We denote the indicator function of an event $\mathcal{E}$ by $1_{(\mathcal{E})}$, which is 1 if $\mathcal{E}$ is true and 0 otherwise.

2.2 Background on reproducing kernels

The method of kernel ridge regression is based on the idea of a reproducing kernel Hilbert space. We provide only a very brief coverage of the basics here, referring the reader to one of the many books on the topic [210, 192, 20, 80] for further details. Any symmetric and positive semidefinite kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defines a reproducing kernel Hilbert space (RKHS for short). For a given distribution $\mathbb{P}$ on $\mathcal{X}$, the Hilbert space is strictly contained in $L^2(\mathbb{P})$. For each $x \in \mathcal{X}$, the function $z \mapsto K(z, x)$ is contained with the Hilbert space $\mathcal{H}$; moreover, the Hilbert space is endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that $K(\cdot, x)$ acts as the representer of evaluation, meaning

$$\langle f, K(x, \cdot) \rangle_{\mathcal{H}} = f(x) \quad \text{for } f \in \mathcal{H}. \tag{2.4}$$

We let $\|g\|_{\mathcal{H}} := \sqrt{\langle g, g \rangle_{\mathcal{H}}}$ denote the norm in $\mathcal{H}$, and similarly $\|g\|_2 := (\int_{\mathcal{X}} g(x)^2 d\mathbb{P}(x))^{1/2}$ denotes the norm in $L^2(\mathbb{P})$. Under suitable regularity conditions, Mercer's theorem guarantees that the kernel has an eigen-expansion of the form

$$K(x, x') = \sum_{j=1}^{\infty} \mu_j \phi_j(x) \phi_j(x'),$$

where $\mu_1 \geq \mu_2 \geq \cdots \geq 0$ are a non-negative sequence of eigenvalues, and $\{\phi_j\}_{j=1}^{\infty}$ is an orthonormal basis for $L^2(\mathbb{P})$.

From the reproducing relation (2.4), we have $\langle \phi_j, \phi_j \rangle_{\mathcal{H}} = 1/\mu_j$ for any j and $\langle \phi_j, \phi_{j'} \rangle_{\mathcal{H}} = 0$ for any $j \neq j'$. For any $f \in \mathcal{H}$, by defining the basis coefficients $\theta_j = \langle f, \phi_j \rangle_{L^2(\mathbb{P})}$ for $j = 1, 2, \ldots$, we can expand the function in terms of these coefficients as $f = \sum_{j=1}^{\infty} \theta_j \phi_j$, and simple calculations show that

$$\|f\|_2^2 = \int_{\mathcal{X}} f^2(x) d\mathbb{P}(x) = \sum_{j=1}^{\infty} \theta_j^2, \quad \text{and} \quad \|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} = \sum_{j=1}^{\infty} \frac{\theta_j^2}{\mu_j}.$$

Consequently, we see that the RKHS can be viewed as an elliptical subset of the sequence space $\ell^2(\mathbb{N})$ as defined by the non-negative eigenvalues $\{\mu_j\}_{j=1}^{\infty}$.

Kernel ridge regression Suppose that we are given a data set $\{(x_i, y_i)\}_{i=1}^N$ consisting of N i.i.d. samples drawn from an unknown distribution $\mathbb{P}$ over $\mathcal{X} \times \mathbb{R}$. The goal of kernel ridge regression is to estimate the function that minimizes the mean-squared error $\mathbb{E}[(f(X)-Y)^2]$, where the expectation is taken jointly over (X, Y) pairs. It is well-known that the optimal function is the conditional mean $f^*(x) := \mathbb{E}[Y \mid X = x]$. In order to estimate the unknown function f^*, we consider an M-estimator that is based on minimizing a combination of the least-squares loss defined over the dataset with a weighted penalty based on the squared Hilbert norm,

$$\widehat{f} := \operatorname*{argmin}_{f \in \mathcal{H}} \left\{ \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\}, \tag{2.5}$$

where $\lambda > 0$ is a regularization parameter. When $\mathcal{H}$ is a reproducing kernel Hilbert space, then the estimator (2.5) is known as the *kernel ridge regression estimate*, or KRR for short. It is a natural generalization of the ordinary ridge regression estimate [91] to the non-parametric setting.

By the representer theorem for reproducing kernel Hilbert spaces [210], any solution to the KRR program (2.5) must belong to the linear span of the kernel functions $\{K(\cdot, x_i),\, i = 1, \ldots, N\}$. This fact allows the computation of the KRR estimate to be reduced to an N-dimensional quadratic program, involving the N^2 entries of the kernel matrix $\{K(x_i, x_j), i, j = 1, \ldots, n\}$. On the statistical side, a line of past work [74, 227, 39, 197, 95] has provided bounds on the estimation error of $\widehat{f}$ as a function of N and λ.

2.3 Background on self-concordant functions

The theory of self-concordant functions were developed by Nesterov and Nemirovski for the analysis of interior-point methods [157]. It will be a very useful tool for our analysis in Chapter 5. Roughly speaking, a function is called self-concordant if its third derivative can be controlled, in a specific way, by its second derivative. Suppose the function $f : \mathbb{R}^d \to \mathbb{R}$ has continuous third derivatives. We use $f''(w) \in \mathbb{R}^{d \times d}$ to denote its Hessian at $w \in \mathbb{R}^d$, and use $f'''(w)[u] \in \mathbb{R}^{d \times d}$ to denote the limit

$$f'''(w)[u] =: \lim_{t \to 0} \frac{1}{t} \big(f''(w + tu) - f''(w) \big).$$

Definition 1. *A convex function $f : \mathbb{R}^d \to \mathbb{R}$ is self-concordant with parameter M_f if the inequality*

$$\left| u^T (f'''(w)[u]) u \right| \leq M_f \left(u^T f''(w) u \right)^{3/2}$$

holds for any $w \in \operatorname{dom}(f)$ and $u \in \mathbb{R}^d$. In particular, a self-concordant function with parameter 2 is called standard self-concordant.

The reader may refer to the books [157, 154] for detailed treatment of self-concordance. In particular, the following lemma [154, Corollary 4.1.2] states that any self-concordant function can be rescaled to become standard self-concordant.

Lemma 1. *If a function f is self-concordant with parameter M_f, then $\frac{M_f^2}{4} f$ is standard self-concordant (with parameter 2).*

In the rest of this section, we show that several popular regularized empirical loss functions for linear regression and binary classification are either self-concordant or can be well approximated by self-concordant functions.

First we consider regularized linear regression (ridge regression) with

$$f(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \frac{\lambda}{2}\|w\|_2^2.$$

To simplify notation, here we use a single subscript i running from 1 to $N = mn$, instead of the double subscripts $\{i, j\}$ used in the introduction. Since f is a quadratic function, its third derivatives are all zero. Therefore, it is self-concordant with parameter 0, and by definition is also standard self-concordant.

For binary classification, we consider the following regularized empirical loss function

$$\ell(w) =: \frac{1}{N}\sum_{i=1}^{N}\varphi(y_i w^T x_i) + \frac{\gamma}{2}\|w\|_2^2, \tag{2.6}$$

where $x_i \in \mathcal{X} \subset \mathbb{R}^d$, $y_i \in \{-1, 1\}$, and $\varphi : \mathbb{R} \to \mathbb{R}$ is a convex surrogate function for the binary loss function which returns 0 if $y_i = \text{sign}(w^T x_i)$ and 1 otherwise. We further assume that the elements of $\mathcal{X}$ are bounded, that is, we have $\sup_{x\in\mathcal{X}} \|x\|_2 \le B$ for some finite B. Under this assumption, the following lemma shows that the regularized loss $\ell(w)$ is self-concordant.

Lemma 2. *Assume that $\gamma > 0$ and there exist $Q > 0$ and $\alpha \in [0, 1)$ such that $|\varphi'''(t)| \le Q(\varphi''(t))^{1-\alpha}$ for every $t \in \mathbb{R}$. Then:*

(a) The function $\ell(w)$ defined by equation (2.6) is self-concordant with parameter $\frac{B^{1+2\alpha}Q}{\gamma^{1/2+\alpha}}$.

(b) The scaled function $f(w) = \frac{B^{2+4\alpha}Q^2}{4\gamma^{1+2\alpha}}\ell(w)$ is standard self-concordant.

Proof We need to bound the third derivative of ℓ appropriately. Using equation (2.6) and the assumption on φ, we have

$$\begin{aligned}
\left|u^T(\ell'''(w)[u])u\right| &\leq \frac{1}{N}\sum_{i=1}^{N}\left|\varphi'''(y_i w^T x_i)(y_i u^T x_i)^3\right| \\
&\overset{(i)}{\leq} \frac{Q}{N}\sum_{i=1}^{N}\left((u^T x_i)^2\varphi''(y_i w^T x_i)\right)^{1-\alpha}(B\|u\|_2)^{1+2\alpha} \\
&\overset{(ii)}{\leq} B^{1+2\alpha}Q\left(\frac{1}{N}\sum_{i=1}^{N}(u^T x_i)^2\varphi''(y_i w^T x_i)\right)^{1-\alpha}(\|u\|_2)^{1+2\alpha} \\
&\overset{(iii)}{\leq} B^{1+2\alpha}Q\left(u^T\ell''(w)u\right)^{1-\alpha}(\|u\|_2)^{1+2\alpha}.
\end{aligned}$$

In the above derivation, inequality (i) uses the property that $|y_i| = 1$ and $|u^T x_i| \leq B\|u\|_2$, inequality (ii) uses Hölder's inequality and concavity of $(\cdot)^{1-\alpha}$, and inequality (iii) uses the fact that the additional regularization term in $\ell(w)$ is convex.

Since ℓ is γ-strongly convex, we have $u^T\ell''(w)u \geq \gamma\|u\|_2^2$. Thus, we can upper bound $\|u\|_2$ by $\|u\|_2 \leq \gamma^{-1/2}(u^T\ell''(w)u)^{1/2}$. Substituting this inequality into the above upper bound completes the proof of part (a). Given part (a), part (b) follows immediately from Lemma 1. □

It is important to note that the self-concordance of ℓ essentially relies on the regularization parameter γ being positive. If $\gamma = 0$, then the function will no longer be self-concordant, as pointed out by Bach [9] on logistic regression. Since we have the freedom to choose φ, Lemma 2 handles a broad class of empirical loss functions. Next, we take the logistic loss and a smoothed hinge loss as two concrete examples.

Logistic regression For logistic regression, we minimize the objective function (2.6) where φ is the logistic loss: $\varphi(t) = \log(1 + e^{-t})$. We can calculate the second and the third derivatives of $\varphi(t)$:

$$\begin{aligned}
\varphi''(t) &= \frac{e^t}{(e^t+1)^2}\,, \\
\varphi'''(t) &= \frac{e^t(1-e^t)}{(e^t+1)^3} = \frac{1-e^t}{1+e^t}\varphi''(t)\,.
\end{aligned}$$

Since $|\frac{1-e^t}{1+e^t}| \leq 1$ for all $t \in \mathbb{R}$, we conclude that $|\varphi'''(t)| \leq \varphi''(t)$ for all $t \in \mathbb{R}$. This implies that the condition in Lemma 2 holds with $Q = 1$ and $\alpha = 0$. Therefore, the regularized empirical loss $\ell(w)$ is self-concordant with parameter $B/\sqrt{\gamma}$, and the scaled loss function $f(w) = (B^2/(4\gamma))\ell(w)$ is standard self-concordant.

Smoothed hinge loss In classification tasks, it is sometimes more favorable to use the hinge loss $\varphi(t) = \max\{0, 1-t\}$ than using the logistic loss. We consider a family of smoothed

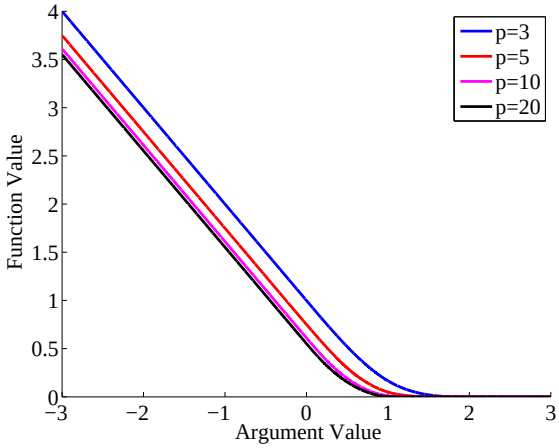


Figure 2.1: Smoothed hinge loss φ_p with $p = 3, 5, 10, 20$.

hinge loss functions φ_p parametrized by a positive number $p \geq 3$. The function is defined by

$$\varphi_p(t) = \begin{cases} \frac{3}{2} - \frac{p-2}{p-1} - t & \text{for } t < -\frac{p-3}{p-1}, \\ \frac{3}{2} - \frac{p-2}{p-1} - t + \frac{(t+(p-3)/(p-1))^p}{p(p-1)} & \text{for } -\frac{p-3}{p-1} \leq t < 1 - \frac{p-3}{p-1}, \\ \frac{p+1}{p(p-1)} - \frac{t}{p-1} + \frac{1}{2}(1-t)^2 & \text{for } 1 - \frac{p-3}{p-1} \leq t < 1, \\ \frac{(2-t)^p}{p(p-1)} & \text{for } 1 \leq t < 2, \\ 0 & \text{for } t \geq 2. \end{cases} \tag{2.7}$$

We plot the functions φ_p for $p = 3, 5, 10, 20$ on Figure 2.1. As the plot shows, $\varphi_p(t)$ is zero for $t > 2$, and it is a linear function with unit slope for $t < -\frac{p-3}{p-1}$. These two linear zones are connected by three smooth non-linear segments on the interval $[-\frac{p-3}{p-1}, 2]$.

The smoothed hinge loss φ_p satisfies the condition of Lemma 2 with $Q = p - 2$ and $\alpha = \frac{1}{p-2}$. To see this, we note that the third derivative of $\varphi_p(t)$ is nonzero only when $t \in [-\frac{p-3}{p-1}, 1 - \frac{p-3}{p-1}]$ and when $t \in [1, 2]$. On the first interval, we have

$$\varphi_p''(t) = \left(t + \frac{p-3}{p-1}\right)^{p-2}, \qquad \varphi_p'''(t) = (p-2)\left(t + \frac{p-3}{p-1}\right)^{p-3}.$$

On the second interval, we have

$$\varphi_p''(t) = (2-t)^{p-2}, \qquad \varphi_p'''(t) = -(p-2)(2-t)^{p-3}.$$

For both cases we have the inequality

$$|\varphi_p'''(t)| \leq (p-2)(\varphi_p''(t))^{1-\frac{1}{p-2}},$$

which means $Q = p - 2$ and $\alpha = \frac{1}{p-2}$. Therefore, according to Lemma 2, the regularized empirical loss $\ell(w)$ is self-concordant with parameter

$$M_p = \frac{(p-2)B^{1+\frac{2}{p-2}}}{\gamma^{\frac{1}{2}+\frac{1}{p-2}}}, \tag{2.8}$$

and the scaled loss function $f(w) = (M_p^2/4)\ell(w)$ is standard self-concordant.

2.4 Background on communication complexity

We provide some basic background on communication complexity theory; see the books [118, 110] for more details. The standard set-up in multi-party communication complexity is as follows: suppose that there are m players (equivalently, agents, machines, etc.), and for $i \in \{1, \ldots, m\}$, player i holds an input string x_i. In the standard form of communication complexity, the goal is to compute a joint function $F(x_1, \ldots, x_m)$ of all m input strings with as little communication between machines as possible. In this paper, we analyze a communication scheme known as the *public blackboard model*, in which each player can write messages on a common blackboard to be read by all other players. A distributed protocol Π consists of a coordinated order in which players write messages on the blackboard. Each message is constructed from the player's local input and the earlier messages on the blackboard. At the end of the protocol, some player outputs the value of $F(x_1, \ldots, x_m)$ based on the information she collects through the process. The communication cost of a given protocol Π, which we denote by $\mathcal{C}(\Pi)$, is the maximum number of bits written on the blackboard given an arbitrary input.

In a *deterministic protocol*, all messages must be deterministic functions of the local input and previous messages. The deterministic communication complexity computing function F, which we denote by $\mathcal{D}(F)$, is defined by

$$\mathcal{D}(F) := \min \Big\{ \mathcal{C}(\Pi) : \ \Pi \text{ is a deterministic protocol that correctly computes } F \Big\}. \tag{2.9}$$

In other words, the quantity $\mathcal{D}(F)$ is the communication cost of the most efficient deterministic protocol.

A broader class of protocols are those that allow some form of randomization. In the public randomness model, each player has access to an infinite-length random string, and their messages are constructed from the local input, the earlier messages and the random string. Let $\mathcal{P}_\epsilon(F)$ be the set of randomized protocols that correctly compute the function F on any input with probability at least $1 - \epsilon$. The *randomized communication complexity* of computing function F with failure probability ϵ is given by

$$\mathcal{R}_\epsilon(F) := \min \Big\{ \mathcal{C}(\Pi) \ \mid \Pi \in \mathcal{P}_\epsilon(F) \Big\}. \tag{2.10}$$

Example As a concrete example, let's assume that there are two players, Alice and Bob, holding binary strings x and y. The function $F(x, y)$ returns 1 if $x = y$ and returns 0 otherwise. Suppose that the length of x and y are both equal to n. A trivial upper bound on the deterministic communication complexity is equal to n, because Bob can send the whole string y to Alice, which costs n bits. It can be proved that n is also a lower bound on the deterministic communication complexity [118]. That is, there is no deterministic algorithm computing this function F whose communication cost is lower than n.

For randomized algorithms, both Alice and Bob have access to a random string. In fact, we can assume that they have access to a shared random string, which doesn't change the randomized communication complexity [118]. There is an communication-efficient protocol to compute $F(x, y)$: Bob uses the random string to compute a b-bit random hashing function $h(y)$, such that if $x \neq y$, then the probability that $h(x) \neq h(y)$ is equal to $1 - 2^{-b}$. Bob sends the string $h(y)$ to Alice, then Alice compares $h(x)$ to $h(y)$ to determine the value of $F(x, y)$. The random hashing function's property implies that this protocol computes the correct value of $F(x, y)$ with probability at least $1 - 2^{-b}$. Thus, we get an upper bound $\mathcal{O}(\log(1/\epsilon))$ on the randomized communication complexity. It can be proved that the lower bound also matches the order of $\log(1/\epsilon)$.

This example shows that some randomized algorithm can be order-of-magnitude more efficient than any deterministic algorithm. This conclusion not only holds for the equality comparison problem, but also holds for many other distributed computing problems.

Part II

Distributed algorithms

Chapter 3

Divide-and-conquer methods for statistical optimization

In this chapter, we study communication-efficient algorithms for statistical optimization. In a centralized setting, the problem of statistical optimization is often solved via empirical risk minimization (ERM). There are many procedures for solving ERM, among them are standard convex programming approaches [e.g. 33] as well as stochastic approximation and optimization algorithms [174, 88, 152]. When the size of the dataset becomes extremely large, however, it may be infeasible to store all of the data on a single computer, or at least to keep the data in memory. Accordingly, we need distributed and communication-efficient procedures for empirical risk minimization.

It can be difficult within a purely optimization-theoretic setting to show explicit benefits arising from distributed computation. In statistical settings, however, distributed computation can lead to gains in computational efficiency, as shown by a number of authors [4, 59, 172, 65]. Within the family of distributed algorithms, there can be significant differences in communication complexity: different computers must be synchronized, and when the dimensionality of the data is high, communication can be prohibitively expensive. Thus, the communication cost of a less carefully design algorithm can easily dominate the computation cost.

With this context, perhaps the simplest algorithm for distributed statistical estimation is what we term the *average mixture* (Avgm) algorithm. It is an appealingly simple method: given m different machines and a dataset of size N, first assign to each machine a (distinct) dataset of size $n = N/m$, then have each machine i compute the empirical minimizer θ_i on its fraction of the data, and finally average all the parameter estimates θ_i across the machines. This approach has been studied for some classification and estimation problems by Mcdonald et al. [142] and McDonald et al. [143], as well as for certain stochastic approximation methods by Zinkevich et al. [238]. Given an empirical risk minimization algorithm that works on one machine, the procedure is straightforward to implement and is extremely communication efficient, requiring only a single round of communication. It is also relatively robust to possible failures in a subset of machines and/or differences in speeds, since there is no

repeated synchronization. When the local estimators are all unbiased, it is clear that the the AVGM procedure will yield an estimate that is essentially as good as that of an estimator based on all N samples. However, many estimators used in practice are biased, and so it is natural to ask whether the method has any guarantees in a more general setting. To the best of our knowledge, however, no work has shown rigorously that the AVGM procedure generally has greater efficiency than the naive approach of using $n = N/m$ samples on a single machine.

In Section 3.2, we provide a sharp analysis of the AVGM algorithm, showing that under a reasonable set of conditions on the population risk, it can indeed achieve substantially better rates than the naive approach. More concretely, we provide bounds on the mean-squared error (MSE) that decay as $\mathcal{O}((nm)^{-1} + n^{-2})$. Whenever the number of machines m is less than the number of samples n per machine, this guarantee matches the best possible rate achievable by a centralized algorithm having access to all $N = nm$ samples. In the special case of optimizing log likelihoods, the pre-factor in our bound involves the trace of the Fisher information, a quantity well-known to control the fundamental limits of statistical estimation. We also show how the result extends to stochastic programming approaches, exhibiting a stochastic gradient-descent based procedure that also attains convergence rates scaling as $\mathcal{O}((nm)^{-1})$, but with slightly worse dependence on different problem-specific parameters.

Our second contribution is to develop a novel extension of simple averaging. It is based on an appropriate form of resampling, which we refer to as the *subsampled average mixture* (SAVGM) approach. At a high level, the SAVGM algorithm distributes samples evenly among m processors or computers as before, but instead of simply returning the empirical minimizer, each processor further subsamples its own dataset in order to estimate the bias of its own estimate, and returns a subsample-corrected estimate. We establish that the SAVGM algorithm has mean-squared error decaying as $\mathcal{O}(m^{-1}n^{-1} + n^{-3})$. As long as $m < n^2$, the subsampled method again matches the centralized gold standard in the first-order term, and has a second-order term smaller than the standard averaging approach.

In Sections 3.3 and 3.4, we perform a detailed empirical evaluation of both the AVGM and SAVGM procedures. Using simulated data from normal and non-normal regression models, we explore the conditions under which the SAVGM algorithm yields better performance than the AVGM algorithm; in addition, we study the performance of both methods relative to an oracle baseline that uses all N samples. We also study the sensitivity of the algorithms to the number of splits m of the data, and in the SAVGM case, we investigate the sensitivity of the method to the amount of resampling. These simulations show that both AVGM and SAVGM have favorable performance, even when compared to the unattainable "gold standard" procedure that has access to all N samples. In Section 3.4, we complement our simulation experiments with a large logistic regression experiment that arises from the problem of predicting whether a user of a search engine will click on an advertisement. This experiment is large enough—involving $N \approx 2.4 \times 10^8$ samples in $d \approx 740,000$ dimensions with a storage size of approximately 55 gigabytes—that it is difficult to solve efficiently on one machine. Consequently, a distributed approach is essential to take full advantage of this data set. Our experiments on this problem show that SAVGM—with the resampling and

correction it provides—gives substantial performance benefits over naive solutions as well as the averaging algorithm AVGM.

3.1 Problem set-up

We begin by setting up our decision-theoretic framework for empirical risk minimization, after which we describe our algorithms and the assumptions we require for our main theoretical results.

3.1.1 Empirical risk minimization

Let $\{\phi(\cdot; x),\ x \in \mathcal{X}\}$ be a collection of real-valued and convex loss functions, each defined on a set containing the convex set $\Theta \subseteq \mathbb{R}^d$. Let P be a probability distribution over the sample space $\mathcal{X}$. Assuming that each function $x \mapsto \phi(\theta; x)$ is P-integrable, the *population risk* $f_0 : \Theta \to \mathbb{R}$ is given by

$$f_0(\theta) := \mathbb{E}_P[\phi(\theta; X)] = \int_{\mathcal{X}} \phi(\theta; x) dP(x). \tag{3.1}$$

Our goal is to estimate the parameter vector minimizing the population risk, namely the quantity

$$\theta^* := \operatorname*{argmin}_{\theta \in \Theta} f_0(\theta) = \operatorname*{argmin}_{\theta \in \Theta} \int_{\mathcal{X}} \phi(\theta; x) dP(x), \tag{3.2}$$

which we assume to be unique. In practice, the population distribution P is unknown to us, but we have access to a collection S of samples from the distribution P. Empirical risk minimization is based on estimating θ^* by solving the optimization problem

$$\widehat{W}\theta \in \operatorname*{argmin}_{\theta \in \Theta} \Big\{ \frac{1}{|S|} \sum_{x \in S} \phi(\theta; x) \Big\}. \tag{3.3}$$

Throughout the chapter, we impose some regularity conditions on the parameter space, the risk function f_0, and the instantaneous loss functions $\phi(\cdot; x) : \Theta \to \mathbb{R}$. These conditions are standard in classical statistical analysis of M-estimators. Our first assumption deals with the relationship of the parameter space to the optimal parameter θ^*.

Assumption A (Parameters)**.** *The parameter space $\Theta \subset \mathbb{R}^d$ is a compact convex set, with $\theta^* \in \operatorname{int} \Theta$ and ℓ_2-radius $R = \max_{\theta \in \Theta} \|\theta - \theta^*\|_2$.*

In addition, the risk function is required to have some amount of curvature. We formalize this notion in terms of the Hessian of f_0:

Assumption B (Local strong convexity)**.** *The population risk is twice differentiable, and there exists a parameter $\lambda > 0$ such that $\nabla^2 f_0(\theta^*) \succeq \lambda I_{d \times d}$.*

Here $\nabla^2 f_0(\theta)$ denotes the $d \times d$ Hessian matrix of the population objective f_0 evaluated at θ, and we use $\succeq$ to denote the positive semidefinite ordering (i.e., $A \succeq B$ means that $A - B$ is positive semidefinite.) This local condition is milder than a global strong convexity condition and is required to hold only for the population risk f_0 evaluated at θ^*. It is worth observing that some type of curvature of the risk is required for any method to consistently estimate the parameters θ^*.

3.1.2 Averaging methods

Consider a data set consisting of $N = mn$ samples, drawn i.i.d. according to the distribution P. In the distributed setting, we divide this N-sample data set evenly and uniformly at random among a total of m processors. (For simplicity, we have assumed the total number of samples is a multiple of m.) For $i = 1, \ldots, m$, we let $S_{1,i}$ denote the data set assigned to processor i; by construction, it is a collection of n samples drawn i.i.d. according to P, and the samples in subsets $S_{1,i}$ and $S_{1,j}$ are independent for $i \neq j$. In addition, for each processor i we define the (local) empirical distribution $P_{1,i}$ and empirical objective $f_{1,i}$ via

$$P_{1,i} := \frac{1}{|S_1|} \sum_{x \in S_{1,i}} \delta_x, \quad \text{and} \quad f_{1,i}(\theta) := \frac{1}{|S_{1,i}|} \sum_{x \in S_{1,i}} \phi(\theta; x). \tag{3.4}$$

With this notation, the Avgm algorithm is very simple to describe.

Average mixture algorithm:

(1) For each $i \in \{1, \ldots, m\}$, processor i uses its local dataset $S_{1,i}$ to compute the local empirical minimizer

$$\theta_{1,i} \in \operatorname*{argmin}_{\theta \in \Theta} \Big\{ \underbrace{\frac{1}{|S_{1,i}|} \sum_{x \in S_{1,i}} \phi(\theta; x)}_{f_{1,i}(\theta)} \Big\}. \tag{3.5}$$

(2) These m local estimates are then averaged together—that is, we compute

$$\overline{\theta}_1 = \frac{1}{m} \sum_{i=1}^{m} \theta_{1,i}. \tag{3.6}$$

The subsampled average mixture (Savgm) algorithm is based on an additional level of sampling on top of the first, involving a fixed subsampling rate $r \in [0, 1]$. It consists of the following additional steps:

Subsampled average mixture algorithm:

(1) Each processor i draws a subset $S_{2,i}$ of size $\lceil rn \rceil$ by sampling uniformly at random without replacement from its local data set $S_{1,i}$.

(2) Each processor i computes both the local empirical minimizers $\theta_{1,i}$ from equation (3.5) and the empirical minimizer

$$\theta_{2,i} \in \operatorname*{argmin}_{\theta \in \Theta} \Big\{ \underbrace{\frac{1}{|S_{2,i}|} \sum_{x \in S_{2,i}} \phi(\theta; x)}_{f_{2,i}(\theta)} \Big\}. \tag{3.7}$$

(3) In addition to the previous average (3.6), the SAVGM algorithm computes the bootstrap average $\overline{\theta}_2 := \frac{1}{m} \sum_{i=1}^m \theta_{2,i}$, and then returns the weighted combination

$$\overline{\theta}_{\text{SAVGM}} := \frac{\overline{\theta}_1 - r\overline{\theta}_2}{1 - r}. \tag{3.8}$$

The intuition for the weighted estimator (3.8) is similar to that for standard bias correction procedures using the bootstrap or subsampling [68, 84, 161]. Roughly speaking, if $b_0 = \theta^* - \theta_1$ is the bias of the first estimator, then we may approximate b_0 by the subsampled estimate of bias $b_1 = \theta^* - \theta_2$. Then, we use the fact that $b_1 \approx b_0/r$ to argue that $\theta^* \approx (\theta_1 - r\theta_2)/(1 - r)$. The re-normalization enforces that the relative "weights" of $\overline{\theta}_1$ and $\overline{\theta}_2$ sum to 1.

The goal of this chapter is to understand under what conditions—and in what sense—the estimators (3.6) and (3.8) approach the *oracle performance*, by which we mean the error of a centralized risk minimization procedure that is given access to all $N = nm$ samples.

3.2 Theoretical results

Having described the AVGM and SAVGM algorithms, we now turn to statements of our main theorems on their statistical properties, along with some consequences and comparison to past work.

3.2.1 Smoothness conditions

In addition to our previously stated assumptions on the population risk, we require regularity conditions on the empirical risk functions. It is simplest to state these in terms of the functions $\theta \mapsto \phi(\theta; x)$, and we note that, as with Assumption B, we require these to hold only locally around the optimal point θ^*, in particular within some Euclidean ball $U = \{\theta \in \mathbb{R}^d \mid \|\theta^* - \theta\|_2 \leq \rho\} \subseteq \Theta$ of radius $\rho > 0$.

Assumption C (Smoothness). *There are finite constants G, H such that the first and the second partial derivatives of ϕ exist and satisfy the bounds*

$$\mathbb{E}[\|\nabla\phi(\theta; X)\|_2^8] \leq G^8 \quad \text{and} \quad \mathbb{E}[\left\|\nabla^2\phi(\theta; X) - \nabla^2 f_0(\theta)\right\|_2^8] \leq H^8 \quad \text{for all } \theta \in U.$$

In addition, for any $x \in \mathcal{X}$, the Hessian matrix $\nabla^2\phi(\theta; x)$ is $L(x)$-Lipschitz continuous, meaning that

$$\left\|\nabla^2\phi(\theta'; x) - \nabla^2\phi(\theta; x)\right\|_2 \leq L(x)\,\|\theta' - \theta\|_2 \quad \text{for all } \theta, \theta' \in U. \tag{3.9}$$

We require that $\mathbb{E}[L(X)^8] \leq L^8$ and $\mathbb{E}[(L(X) - \mathbb{E}[L(X)])^8] \leq L^8$ for some finite constant L.

It is an important insight of our analysis that some type of smoothness condition on the Hessian matrix, as in the Lipschitz condition (3.9), is *essential* in order for simple averaging methods to work. This necessity is illustrated by the following example:

Example 1 (Necessity of Hessian conditions). *Let X be a Bernoulli variable with parameter $\frac{1}{2}$, and consider the loss function*

$$\phi(\theta; x) = \begin{cases} \theta^2 - \theta & \text{if } x = 0 \\ \theta^2 1_{(\theta \leq 0)} + \theta & \text{if } x = 1, \end{cases} \tag{3.10}$$

where $1_{(\theta \leq 0)}$ is the indicator of the event $\{\theta \leq 0\}$. The associated population risk is $f_0(\theta) = \frac{1}{2}(\theta^2 + \theta^2 1_{(\theta \leq 0)})$. Since $|f_0'(w) - f_0'(v)| \leq 2|w - v|$, the population risk is strongly convex and smooth, but it has discontinuous second derivative. The unique minimizer of the population risk is $\theta^* = 0$, and by an asymptotic expansion given in Section 3.5.1, it can be shown that $\mathbb{E}[\theta_{1,i}] = \Omega(n^{-\frac{1}{2}})$. Consequently, the bias of $\overline{\theta}_1$ is $\Omega(n^{-\frac{1}{2}})$, and the Avgm *algorithm using $N = mn$ observations must suffer mean squared error $\mathbb{E}[(\overline{\theta}_1 - \theta^*)^2] = \Omega(n^{-1})$.*

The previous example establishes the necessity of a smoothness condition. However, in a certain sense, it is a pathological case: both the smoothness condition given in Assumption C and the local strong convexity condition given in Assumption B are relatively innocuous for practical problems. For instance, both conditions will hold for standard forms of regression, such as linear and logistic, as long as the *population* data covariance matrix is not rank deficient and the data has suitable moments. Moreover, in the linear regression case, one has $L = 0$.

3.2.2 Bounds for simple averaging

We now turn to our first theorem that provides guarantees on the statistical error associated with the Avgm procedure. We recall that θ^* denotes the minimizer of the population objective function f_0, and that for each $i \in \{1, \ldots, m\}$, we use S_i to denote a dataset of n independent samples. For each i, we use $\theta_i \in \operatorname{argmin}_{\theta \in \Theta}\{\frac{1}{n}\sum_{x \in S_i}\phi(\theta; x)\}$ to denote a minimizer of the empirical risk for the dataset S_i, and we define the averaged vector $\overline{\theta} = \frac{1}{m}\sum_{i=1}^m \theta_i$. The following result bounds the mean-squared error between this averaged estimate and the minimizer θ^* of the population risk.

Theorem 1. *Under Assumptions A through C, the mean-squared error is upper bounded as*

$$\mathbb{E}\left[\left\|\overline{\theta}-\theta^*\right\|_2^2\right] \leq \frac{2}{nm}\mathbb{E}\left[\left\|\nabla^2 f_0(\theta^*)^{-1}\nabla\phi(\theta^*;X)\right\|_2^2\right] + \frac{c}{\lambda^2 n^2}\left(H^2\log d + \frac{L^2G^2}{\lambda^2}\right)\mathbb{E}\left[\left\|\nabla^2 f_0(\theta^*)^{-1}\nabla\phi(\theta^*;X)\right\|_2^2\right] + \mathcal{O}(m^{-1}n^{-2}) + \mathcal{O}(n^{-3}), \tag{3.11}$$

where c is a numerical constant.

A slightly weaker corollary of Theorem 1 makes it easier to parse. In particular, note that

$$\left\|\nabla^2 f_0(\theta^*)^{-1}\nabla\phi(\theta^*;x)\right\|_2 \overset{(i)}{\leq} \left|\!\left|\!\left|\nabla^2 f_0(\theta^*)^{-1}\right|\!\right|\!\right|_2 \left\|\nabla\phi(\theta^*;x)\right\|_2 \overset{(ii)}{\leq} \frac{1}{\lambda}\left\|\nabla\phi(\theta^*;x)\right\|_2, \tag{3.12}$$

where step (i) follows from the inequality $\left|\!\left|\!\left|Ax\right|\!\right|\!\right|_2 \leq \left|\!\left|\!\left|A\right|\!\right|\!\right| \left\|x\right\|_2$, valid for any matrix A and vector x; and step (ii) follows from Assumption B. In addition, Assumption C implies $\mathbb{E}[\|\nabla\phi(\theta^*;X)\|_2^2] \leq G^2$, and putting together the pieces, we have established the following.

Corollary 1. *Under the same conditions as Theorem 1,*

$$\mathbb{E}\left[\left\|\overline{\theta}-\theta^*\right\|_2^2\right] \leq \frac{2G^2}{\lambda^2 nm} + \frac{cG^2}{\lambda^4 n^2}\left(H^2\log d + \frac{L^2G^2}{\lambda^2}\right) + \mathcal{O}(m^{-1}n^{-2}) + \mathcal{O}(n^{-3}). \tag{3.13}$$

This upper bound shows that the leading term decays proportionally to $(nm)^{-1}$, with the pre-factor depending inversely on the strong convexity constant λ and growing proportionally with the bound G on the loss gradient. Although easily interpretable, the upper bound (3.13) can be loose, since it is based on the relatively weak series of bounds (3.12).

The leading term in our original upper bound (3.11) involves the product of the gradient $\nabla\phi(\theta^*;X)$ with the inverse Hessian. In many statistical settings, including the problem of linear regression, the effect of this matrix-vector multiplication is to perform some type of standardization. When the loss $\phi(\cdot;x) : \Theta \to \mathbb{R}$ is actually the negative log-likelihood $\ell(x \mid \theta)$ for a parametric family of models $\{P_\theta\}$, we can make this intuition precise. In particular, under suitable regularity conditions [e.g. 119, Chapter 6], we can define the Fisher information matrix

$$I(\theta^*) := \mathbb{E}\left[\nabla\ell(X \mid \theta^*)\nabla\ell(X \mid \theta^*)^\top\right] = \mathbb{E}[\nabla^2\ell(X \mid \theta^*)].$$

Recalling that $N = mn$ is the total number of samples available, let us define the neighborhood $B_2(\theta, t) := \{\theta' \in \mathbb{R}^d : \|\theta' - \theta\|_2 \leq t\}$. Then under our assumptions, the Hájek-Le Cam minimax theorem [206, Theorem 8.11] guarantees for *any estimator* $\widehat{W}\theta_N$ based on N samples that

$$\lim_{c\to\infty}\liminf_{N\to\infty}\sup_{\theta\in B_2(\theta^*,c/\sqrt{N})} N\,\mathbb{E}_\theta\left[\left\|\widehat{W}\theta_N - \theta\right\|_2^2\right] \geq \operatorname{tr}(I(\theta^*)^{-1}).$$

In connection with Theorem 1, we obtain:

Corollary 2. *In addition to the conditions of Theorem 1, suppose that the loss functions $\phi(\cdot; x)$ are the negative log-likelihood $\ell(x \mid \theta)$ for a parametric family $\{P_\theta, \theta \in \Theta\}$. Then the mean-squared error is upper bounded as*

$$\mathbb{E}\left[\left\|\overline{\theta}_1 - \theta^*\right\|_2^2\right] \leq \frac{2}{N}\operatorname{tr}(I(\theta^*)^{-1}) + \frac{cm^2\operatorname{tr}(I(\theta^*)^{-1})}{\lambda^2 N^2}\left(H^2 \log d + \frac{L^2G^2}{\lambda^2}\right) + \mathcal{O}(mN^{-2}),$$

where c is a numerical constant.

Proof Rewriting the log-likelihood in the notation of Theorem 1, we have $\nabla\ell(x \mid \theta^*) = \nabla\phi(\theta^*; x)$ and all we need to note is that

$$\begin{aligned} I(\theta^*)^{-1} &= \mathbb{E}\left[I(\theta^*)^{-1}\nabla\ell(X \mid \theta^*)\nabla\ell(X \mid \theta^*)^\top I(\theta^*)^{-1}\right] \\ &= \mathbb{E}\Big[\left(\nabla^2 f_0(\theta^*)^{-1}\nabla\phi(\theta^*; X)\right)\left(\nabla^2 f_0(\theta^*)^{-1}\nabla\phi(\theta^*; X)\right)^\top\Big]. \end{aligned}$$

Now apply the linearity of the trace and use the fact that $\operatorname{tr}(uu^\top) = \|u\|_2^2$. □

Except for the factor of two in the bound, Corollary 2 shows that Theorem 1 essentially achieves the best possible result. The important aspect of our bound, however, is that we obtain this convergence rate without calculating an estimate on all $N = mn$ samples: instead, we calculate m independent estimators, and then average them to attain the convergence guarantee. We remark that an inspection of our proof shows that, at the expense of worse constants on higher order terms, we can reduce the factor of $2/mn$ on the leading term in Theorem 1 to $(1 + c)/mn$ for any constant $c > 0$; as made clear by Corollary 2, this is unimprovable, even by constant factors.

As noted in the introduction, our bounds are certainly to be expected for unbiased estimators, since in such cases averaging m independent solutions reduces the variance by $1/m$. In this sense, our results are similar to classical distributional convergence results in M-estimation: for smooth enough problems, M-estimators behave asymptotically like averages [206, 119], and averaging multiple independent realizations reduces their variance. However, it is often desirable to use biased estimators, and such bias introduces difficulty in the analysis, which we explore more in the next section. We also note that in contrast to classical asymptotic results, our results are applicable to finite samples and give explicit upper bounds on the mean-squared error. Lastly, our results are not tied to a specific model, which allows for fairly general sampling distributions.

3.2.3 Bounds for subsampled mixture averaging

When the number of machines m is relatively small, Theorem 1 and Corollary 1 show that the convergence rate of the AVGM algorithm is mainly determined by the first term in the bound (3.11), which is at most $\frac{G^2}{\lambda^2 mn}$. In contrast, when the number of processors m grows, the second term in the bound (3.11), in spite of being $\mathcal{O}(n^{-2})$, may have non-negligible effect.

This issue is exacerbated when the local strong convexity parameter λ of the risk f_0 is close to zero or the Lipschitz continuity constant H of $\nabla\phi$ is large. This concern motivated our development of the subsampled average mixture (SAVGM) algorithm, to which we now return.

Due to the additional randomness introduced by the subsampling in SAVGM, its analysis requires an additional smoothness condition. In particular, recalling the Euclidean ρ-neighborhood U of the optimum θ^*, we require that the loss function ϕ is (locally) smooth through its third derivatives.

Assumption D (Strong smoothness)**.** *For each $x \in \mathcal{X}$, the third derivatives of ϕ are $M(x)$-Lipschitz continuous, meaning that*

$$\left\| \left(\nabla^3 \phi(\theta; x) - \nabla^3 \phi(\theta'; x) \right) (u \otimes u) \right\|_2 \leq M(x) \left\| \theta - \theta' \right\|_2 \left\| u \right\|_2^2 \quad \text{for all } \theta, \theta' \in U, \text{ and } u \in \mathbb{R}^d,$$

where $\mathbb{E}[M^8(X)] \leq M^8$ for some constant $M < \infty$.

It is easy to verify that Assumption D holds for least-squares regression with $M = 0$. It also holds for various types of non-linear regression problems (e.g., logistic, multinomial etc.) as long as the covariates have finite eighth moments.

With this set-up, our second theorem establishes that bootstrap sampling yields improved performance:

Theorem 2. *Under Assumptions A through D, the output $\overline{\theta}_{\text{SAVGM}} = (\overline{\theta}_1 - r\overline{\theta}_2)/(1-r)$ of the bootstrap* SAVGM *algorithm has mean-squared error bounded as*

$$\begin{aligned} \mathbb{E}\left[\left\| \overline{\theta}_{\text{SAVGM}} - \theta^* \right\|_2^2 \right] \leq \frac{2+3r}{(1-r)^2} \cdot \frac{1}{nm} \mathbb{E}\left[\left\| \nabla^2 f_0(\theta^*)^{-1} \nabla \phi(\theta^*; X) \right\|_2^2 \right] \\ + c \left(\frac{M^2 G^6}{\lambda^6} + \frac{G^4 L^2 d \log d}{\lambda^4} \right) \left(\frac{1}{r(1-r)^2} \right) n^{-3} + \mathcal{O}\left(\frac{1}{(1-r)^2} m^{-1} n^{-2} \right) \end{aligned} \tag{3.14}$$

for a numerical constant c.

Comparing the conclusions of Theorem 2 to those of Theorem 1, we see that the the $\mathcal{O}(n^{-2})$ term in the bound (3.11) has been eliminated. The reason for this elimination is that subsampling at a rate r reduces the bias of the SAVGM algorithm to $\mathcal{O}(n^{-3})$, whereas in contrast, the bias of the AVGM algorithm induces terms of order n^{-2}. Theorem 2 suggests that the performance of the SAVGM algorithm is affected by the subsampling rate r; in order to minimize the upper bound (3.14) in the regime $m < N^{2/3}$, the optimal choice is of the form $r \propto C\sqrt{m}/n = Cm^{3/2}/N$ where $C \approx (G^2/\lambda^2) \max\{MG/\lambda, L\sqrt{d \log d}\}$. Roughly, as the number of machines m becomes larger, we may increase r, since we enjoy averaging affects from the SAVGM algorithm.

Let us consider the relative effects of having larger numbers of machines m for both the AVGM and SAVGM algorithms, which provides some guidance to selecting m in practice. We

define $\sigma^2 = \mathbb{E}[\|\nabla^2 f_0(\theta^*)^{-1}\nabla\phi(\theta^*; X)\|_2^2]$ to be the asymptotic variance. Then to obtain the optimal convergence rate of σ^2/N, we must have

$$\frac{1}{\lambda^2}\max\left\{H^2\log d, L^2G^2\right\}\frac{m^2}{N^2}\sigma^2 \leq \frac{\sigma^2}{N} \quad \text{or} \quad m \leq N^{\frac{1}{2}}\sqrt{\frac{\lambda^2}{\max\{H^2\log d, L^2G^2/\lambda^2\}}} \tag{3.15}$$

in Theorem 1. Applying the bound of Theorem 2, we find that to obtain the same rate we require

$$\max\left\{\frac{M^2G^2}{\lambda^6}, \frac{L^2 d\log d}{\lambda^4}\right\}\frac{G^4m^3}{rN^3} \leq \frac{(1+r)\sigma^2}{N} \text{ or } m \leq N^{\frac{2}{3}}\left(\frac{\lambda^4 r(1+r)\sigma^2}{\max\left\{M^2G^6/\lambda^2, G^4L^2 d\log d\right\}}\right)^{\frac{1}{3}}.$$

Now suppose that we replace r with $Cm^{3/2}/N$ as in the previous paragraph. Under the conditions $\sigma^2 \approx G^2$ and $r = o(1)$, we then find that

$$m \leq N^{\frac{2}{3}}\left(\frac{\lambda^2\sigma^2 m^{3/2}}{G^2\max\left\{MG/\lambda, L\sqrt{d\log d}\right\}N}\right)^{\frac{1}{3}} \text{ or } m \leq N^{\frac{2}{3}}\left(\frac{\lambda^2}{\max\left\{MG/\lambda, L\sqrt{d\log d}\right\}}\right)^{\frac{2}{3}}. \tag{3.16}$$

Comparing inequalities (3.15) and (3.16), we see that in both cases m may grow polynomially with the global sample size N while still guaranteeing optimal convergence rates. On one hand, this asymptotic growth is faster in the subsampled case (3.16); on the other hand, the dependence on the dimension d of the problem is more stringent than the standard averaging case (3.15). As the local strong convexity constant λ of the *population risk* shrinks, both methods allow less splitting of the data, meaning that the sample size per machine must be larger. This limitation is intuitive, since lower curvature for the population risk means that the local empirical risks associated with each machine will inherit lower curvature as well, and this effect will be exacerbated with a small local sample size per machine. Averaging methods are, of course, not a panacea: the allowed number of partitions m does not grow linearly in either case, so blindly increasing the number of machines proportionally to the total sample size N will not lead to a useful estimate.

In practice, an optimal choice of r may not be apparent, which may necessitate cross validation or another type of model evaluation. We leave as intriguing open questions whether computing multiple subsamples at each machine can yield improved performance or reduce the variance of the SAVGM procedure, and whether using estimates based on resampling the data with replacement, as opposed to without replacement as considered here, can yield improved performance.

3.2.4 Time complexity

In practice, the exact empirical minimizers assumed in Theorems 1 and 2 may be unavailable. Instead, we need to use a finite number of iterations of some optimization algorithm in order to obtain reasonable approximations to the exact minimizers. In this section, we sketch an

argument that shows that both the AVGM algorithm and the SAVGM algorithm can use such approximate empirical minimizers, and as long as the optimization error is sufficiently small, the resulting averaged estimate achieves the same order-optimal statistical error. Here we provide the arguments only for the AVGM algorithm; the arguments for the SAVGM algorithm are analogous.

More precisely, suppose that each processor runs a finite number of iterations of some optimization algorithm, thereby obtaining the vector θ_i' as an approximate minimizer of the objective function $f_{1,i}$. Thus, the vector θ_i' can be viewed as an approximate form of θ_i, and we let $\overline{\theta}' = \frac{1}{m}\sum_{i=1}^m \theta_i'$ denote the average of these approximate minimizers, which corresponds to the output of the approximate AVGM algorithm. With this notation, we have

$$\mathbb{E}\left[\left\|\overline{\theta}' - \theta^*\right\|_2^2\right] \overset{(i)}{\leq} 2\mathbb{E}[\left\|\overline{\theta} - \theta^*\right\|_2^2] + 2\mathbb{E}\left[\left\|\overline{\theta}' - \overline{\theta}\right\|_2^2\right] \overset{(ii)}{\leq} 2\mathbb{E}[\left\|\overline{\theta} - \theta^*\right\|_2^2] + 2\mathbb{E}[\|\theta_1' - \theta_1\|_2^2], \tag{3.17}$$

where step (i) follows by triangle inequality and the elementary bound $(a+b)^2 \leq 2a^2 + 2b^2$; step (ii) follows by Jensen's inequality. Consequently, suppose that processor i runs enough iterations to obtain an approximate minimizer θ_1' such that

$$\mathbb{E}[\|\theta_i' - \theta_i\|_2^2] = \mathcal{O}((mn)^{-2}). \tag{3.18}$$

When this condition holds, the bound (3.17) shows that the average $\overline{\theta}'$ of the approximate minimizers shares the same convergence rates provided by Theorem 1.

But how long does it take to compute an approximate minimizer θ_i' satisfying condition (3.18)? Assuming processing one sample requires one unit of time, we claim that this computation can be performed in time $\mathcal{O}(n\log(mn))$. In particular, the following two-stage strategy, involving a combination of stochastic gradient descent (see the following subsection for more details) and standard gradient descent, has this complexity:

(1) As shown in the proof of Theorem 1, with high probability, the empirical risk f_1 is strongly convex in a ball $B_\rho(\theta_1)$ of constant radius $\rho > 0$ around θ_1. Consequently, performing stochastic gradient descent on f_1 for $\mathcal{O}(\log^2(mn)/\rho^2)$ iterations yields an approximate minimizer that falls within $B_\rho(\theta_1)$ with high probability [e.g. 152, Proposition 2.1]. Note that the radius ρ for local strong convexity is a property of the population risk f_0 we use as a prior knowledge.

(2) This initial estimate can be further improved by a few iterations of standard gradient descent. Under local strong convexity of the objective function, gradient descent is known to converge at a geometric rate [see, e.g. 159, 33], so $\mathcal{O}(\log(1/\epsilon))$ iterations will reduce the error to order ϵ. In our case, we have $\epsilon = (mn)^{-2}$, and since each iteration of standard gradient descent requires $\mathcal{O}(n)$ units of time, a total of $\mathcal{O}(n\log(mn))$ time units are sufficient to obtain a final estimate θ_1' satisfying condition (3.18).

Overall, we conclude that the speed-up of the AVGM relative to the naive approach of processing all $N = mn$ samples on one processor, is at least of order $m/\log(N)$.

3.2.5 Stochastic gradient descent with averaging

The previous strategy involved a combination of stochastic gradient descent and standard gradient descent. In many settings, it may be appealing to use only a stochastic gradient algorithm, due to their ease of their implementation and limited computational requirements. In this section, we describe an extension of Theorem 1 to the case in which each machine computes an approximate minimizer using only stochastic gradient descent.

Stochastic gradient algorithms have a lengthy history in statistics, optimization, and machine learning [174, 163, 152, 167]. Let us begin by briefly reviewing the basic form of stochastic gradient descent (SGD). Stochastic gradient descent algorithms iteratively update a parameter vector θ^t over time based on randomly sampled gradient information. Specifically, at iteration t, a sample X_t is drawn at random from the distribution P (or, in the case of a finite set of data $\{X_1, \ldots, X_n\}$, a sample X_t is chosen from the data set). The method then performs the following two steps:

$$\theta^{t+\frac{1}{2}} = \theta^t - \eta_t \nabla \phi(\theta^t; X_t) \quad \text{and} \quad \theta^{t+1} = \operatorname*{argmin}_{\theta \in \Theta} \left\{ \left\| \theta - \theta^{t+\frac{1}{2}} \right\|_2^2 \right\}. \tag{3.19}$$

Here $\eta_t > 0$ is a stepsize, and the first update in (3.19) is a gradient descent step with respect to the random gradient $\nabla \phi(\theta^t; X_t)$. The method then projects the intermediate point $\theta^{t+\frac{1}{2}}$ back onto the constraint set Θ (if there is a constraint set). The convergence of SGD methods of the form (3.19) has been well-studied, and we refer the reader to the papers by Polyak and Juditsky [163], Nemirovski et al. [152], and Rakhlin et al. [167] for deeper investigations.

To prove convergence of our stochastic gradient-based averaging algorithms, we require the following smoothness and strong convexity condition, which is an alternative to the Assumptions B and C used previously.

Assumption E (Smoothness and Strong Convexity II)**.** *There exists a function $L : \mathcal{X} \to \mathbb{R}_+$ such that*

$$\left\| \nabla^2 \phi(\theta; x) - \nabla^2 \phi(\theta^*; x) \right\|_2 \le L(x) \left\| \theta - \theta^* \right\|_2 \quad \textit{for all } x \in \mathcal{X},$$

and $\mathbb{E}[L^2(X)] \le L^2 < \infty$. There are finite constants G and H such that

$$\mathbb{E}[\|\nabla \phi(\theta; X)\|_2^4] \le G^4, \quad \textit{and} \quad \mathbb{E}[\left\|\nabla^2 \phi(\theta^*; X)\right\|_2^4] \le H^4 \quad \textit{for each fixed } \theta \in \Theta.$$

In addition, the population function f_0 is λ-strongly convex over the space Θ, meaning that

$$\nabla^2 f_0(\theta) \succeq \lambda I_{d \times d} \quad \textit{for all } \theta \in \Theta. \tag{3.20}$$

Assumption E does not require as many moments as does Assumption C, but it does require each moment bound to hold globally, that is, over the entire space Θ, rather than only in a neighborhood of the optimal point θ^*. Similarly, the necessary curvature—in the form of the lower bound on the Hessian matrix $\nabla^2 f_0$—is also required to hold globally, rather than only locally. Nonetheless, Assumption E holds for many common problems; for instance, it holds for any linear regression problem in which the covariates have finite fourth moments

and the domain Θ is compact.

The averaged stochastic gradient algorithm (SGDAVGM) is based on the following two steps:

(1) Given some constant $c > 1$, each machine performs n iterations of stochastic gradient descent (3.19) on its local dataset of n samples using the stepsize $\eta_t = \frac{c}{\lambda t}$, then outputs the resulting local parameter θ_i'.

(2) The algorithm computes the average $\overline{\theta}^n = \frac{1}{m}\sum_{i=1}^m \theta_i'$.

The following result characterizes the mean-squared error of this procedure in terms of the constants

$$\alpha := 4c^2 \quad \text{and} \quad \beta := \max\left\{\left\lceil \frac{cH}{\lambda}\right\rceil, \frac{c\alpha^{3/4}G^{3/2}}{(c-1)\lambda^{5/2}}\left(\frac{\alpha^{1/4}LG^{1/2}}{\lambda^{1/2}} + \frac{4G+HR}{\rho^{3/2}}\right)\right\}.$$

Theorem 3. *Under Assumptions A and E, the output $\overline{\theta}^n$ of the* SAVGM *algorithm has mean-squared error upper bounded as*

$$\mathbb{E}\left[\left\|\overline{\theta}^n - \theta^*\right\|_2^2\right] \leq \frac{\alpha G^2}{\lambda^2 mn} + \frac{\beta^2}{n^{3/2}}. \tag{3.21}$$

Theorem 3 shows that the averaged stochastic gradient descent procedure attains the optimal convergence rate $\mathcal{O}(N^{-1})$ as a function of the total number of observations $N = mn$. The constant and problem-dependent factors are somewhat worse than those in the earlier results we presented in Theorems 1 and 2, but the practical implementability of such a procedure may in some circumstances outweigh those differences. We also note that the second term of order $\mathcal{O}(n^{-3/2})$ may be reduced to $\mathcal{O}(n^{(2-2k)/k})$ for any $k \geq 4$ by assuming the existence of kth moments in Assumption E; we show this in passing after our proof of the theorem in Section 3.5.4. It is not clear whether a bootstrap correction is possible for the stochastic-gradient based estimator; such a correction could be significant, because the term $\beta^2/n^{3/2}$ arising from the bias in the stochastic gradient estimator may be non-trivial. We leave this question to future work.

3.3 Performance on synthetic data

In this section, we report the results of simulation studies comparing the AVGM, SAVGM, and SGDAVGM methods, as well as a trivial method using only a fraction of the data available on a single machine. For each of our simulated experiments, we use a fixed total number of samples $N = 100{,}000$, but we vary the number of parallel splits m of the data (and consequently, the local dataset sizes $n = N/m$) and the dimensionality d of the problem solved.

For our experiments, we simulate data from one of three regression models:

$$y = \langle u, x \rangle + \varepsilon, \tag{3.22}$$

$$y = \langle u, x \rangle + \sum_{j=1}^{d} v_j x_j^3 + \varepsilon, \quad \text{or} \tag{3.23}$$

$$y = \langle u, x \rangle + h(x)|\varepsilon|, \tag{3.24}$$

where $\varepsilon \sim N(0, 1)$, and h is a function to be specified. Specifically, the data generation procedure is as follows. For each individual simulation, we choose fixed vector $u \in \mathbb{R}^d$ with entries u_i distributed uniformly in $[0, 1]$ (and similarly for v), and we set $h(x) = \sum_{j=1}^{d} (x_j/2)^3$. The models (3.22) through (3.24) provide points on a curve from correctly-specified to grossly misspecified models, so models (3.23) and (3.24) help us understand the effects of subsampling in the SAVGM algorithm. (In contrast, the standard least-squares estimator is unbiased for model (3.22).) The noise variable ε is always chosen as a standard Gaussian variate $N(0, 1)$, independent from sample to sample.

In our simulation experiments we use the least-squares loss

$$\phi(\theta; (x, y)) := \frac{1}{2}(\langle \theta, x \rangle - y)^2.$$

The goal in each experiment is to estimate the vector θ^* minimizing $f_0(\theta) := \mathbb{E}[\phi(\theta; (X, Y))]$. For each simulation, we generate N samples according to either the model (3.22) or (3.24). For each $m \in \{2, 4, 8, 16, 32, 64, 128\}$, we estimate $\theta^* = \arg\min_\theta f_0(\theta)$ using a parallel method with data split into m independent sets of size $n = N/m$, specifically

(i) The AVGM method

(ii) The SAVGM method with several settings of the subsampling ratio r

(iii) The SGDAVGM method with stepsize $\eta_t = d/(10(d+t))$, which gave good performance.

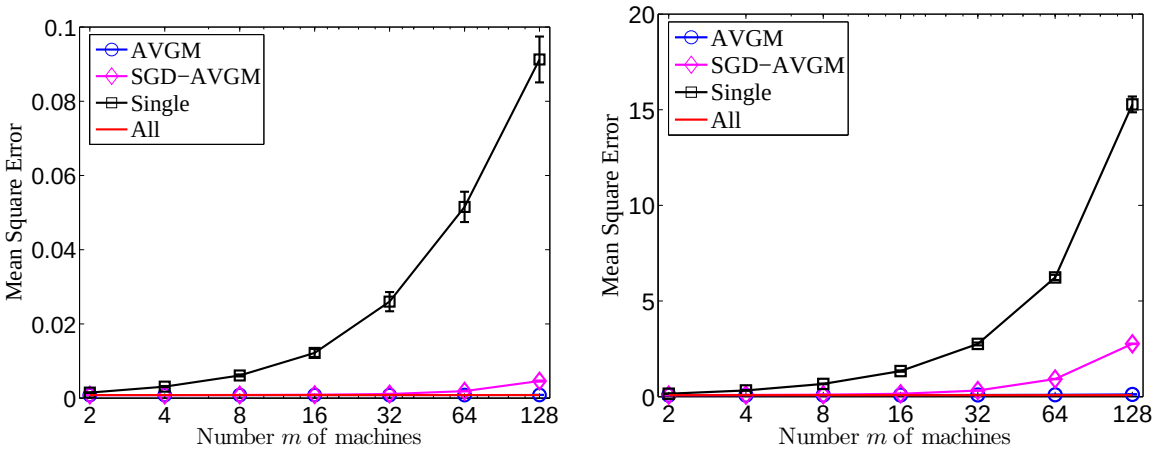
In addition to (i)–(iii), we also estimate θ^* with

(iv) The empirical minimizer of a single split of the data of size $n = N/m$

(v) The empirical minimizer on the full dataset (the oracle solution).

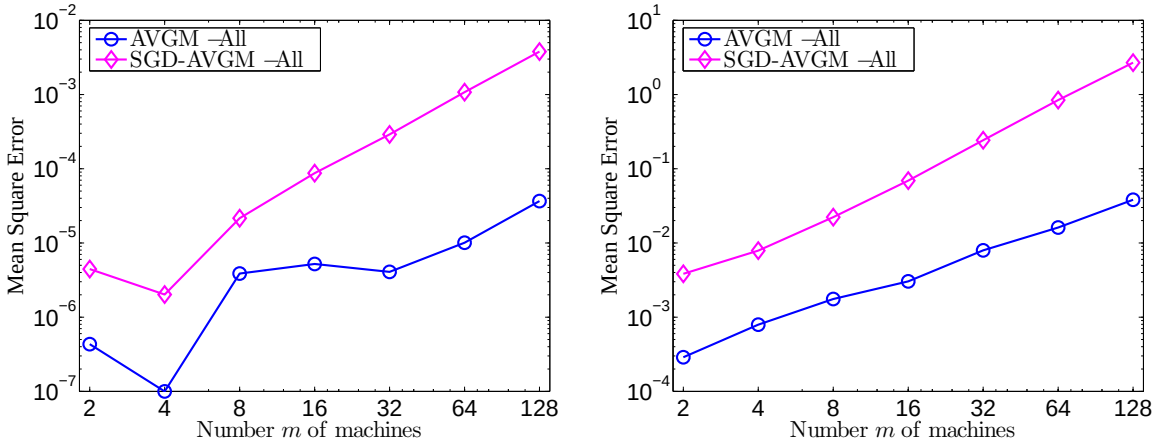
3.3.1 Averaging methods

For our first set of experiments, we study the performance of the averaging methods (AVGM and SAVGM), showing their scaling as the number of splits of data—the number of machines m—grows for fixed N and dimensions $d = 20$ and $d = 200$. We use the standard regression model (3.22) to generate the data, and throughout we let $\widehat{W}\theta$ denote the estimate returned by the method under consideration (so in the AVGM case, for example, this is the vector



(a) $d = 20$ (b) $d = 200$

Figure 3.1. The error $\|\widehat{W}\theta - \theta^*\|_2^2$ versus number of machines, with standard errors across twenty simulations, for solving least squares with data generated according to the normal model (3.22). The oracle least-squares estimate using all N samples is given by the line "All," while the line "Single" gives the performance of the naive estimator using only $n = N/m$ samples.



(a) $d = 20$ (b) $d = 200$

Figure 3.2. Comparison of AVGM and SGDAVGM methods as in Figure 3.1 plotted on logarithmic scale. The plot shows $\|\widehat{W}\theta - \theta^*\|_2^2 - \|\theta_N - \theta^*\|_2^2$, where θ_N is the oracle least-squares estimator using all N data samples.

$\widehat{W}\theta := \overline{\theta}_1$). The data samples consist of pairs (x, y), where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ is the target value. To sample each x vector, we choose five distinct indices in $\{1, \ldots, d\}$ uniformly at random, and the entries of x at those indices are distributed as $N(0, 1)$. For the model (3.22), the population optimal vector θ^* is u.

In Figure 3.1, we plot the error $\|\widehat{W}\theta - \theta^*\|_2^2$ of the inferred parameter vector $\widehat{W}\theta$ for the true parameters θ^* versus the number of splits m, or equivalently, the number of separate machines available for use. We also plot standard errors (across twenty experiments) for each curve. As a baseline in each plot, we plot as a red line the squared error $\|\widehat{W}\theta_N - \theta^*\|_2^2$ of the centralized "gold standard," obtained by applying a batch method to all N samples.

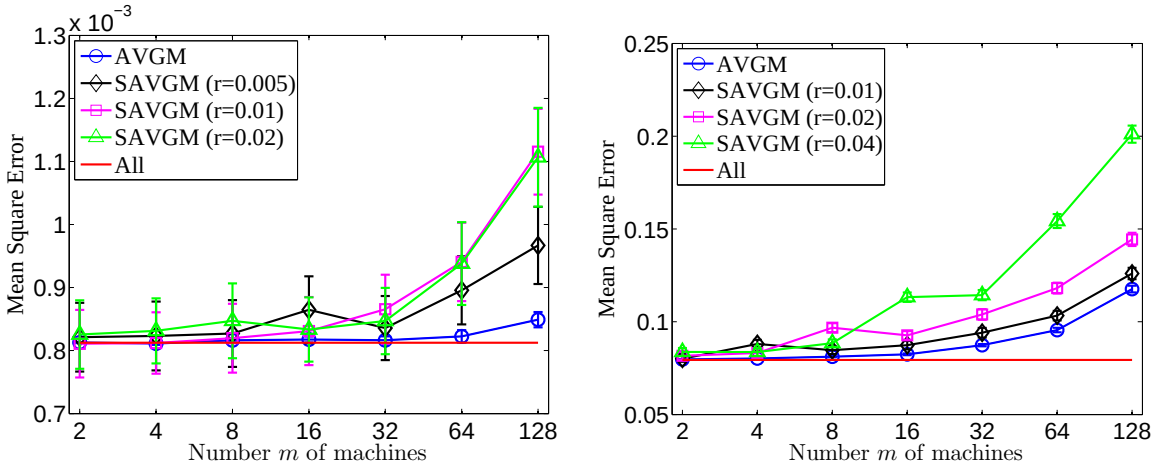
From the plots in Figure 3.1, we can make a few observations. The AVGM algorithm enjoys excellent performance, as predicted by our theoretical results, especially compared to the naive solution using only a fraction $1/m$ of the data. In particular, if $\widehat{W}\theta$ is obtained by the batch method, then AVGM is almost as good as the full-batch baseline even for m as large as 128, though there is some evident degradation in solution quality. The SGDAVGM (stochastic-gradient with averaging) solution also yields much higher accuracy than the naive solution, but its performance degrades more quickly than the AVGM method's as m grows. In higher dimensions, both the AVGM and SGDAVGM procedures have somewhat worse performance; again, this is not unexpected since in high dimensions the strong convexity condition is satisfied with lower probability in local datasets.

We present a comparison between the AVGM method and the SGDAVGM method with somewhat more distinguishing power in Figure 3.2. For these plots, we compute the gap between the AVGM mean-squared-error and the unparallel baseline MSE, which is the accuracy lost due to parallelization or distributing the inference procedure across multiple machines. Figure 3.2 shows that the mean-squared error grows polynomially with the number of machines m, which is consistent with our theoretical results. From Corollary 2, we expect the AVGM method to suffer (lower-order) penalties proportional to m^2 as m grows, while Theorem 3 suggests the somewhat faster growth we see for the SGDAVGM method in Figure 3.2. Thus, we see that the improved run-time performance of the SGDAVGM method—requiring only a single pass through the data on each machine, touching each datum only once—comes at the expense of some loss of accuracy, as measured by mean-squared error.

3.3.2 Subsampling correction

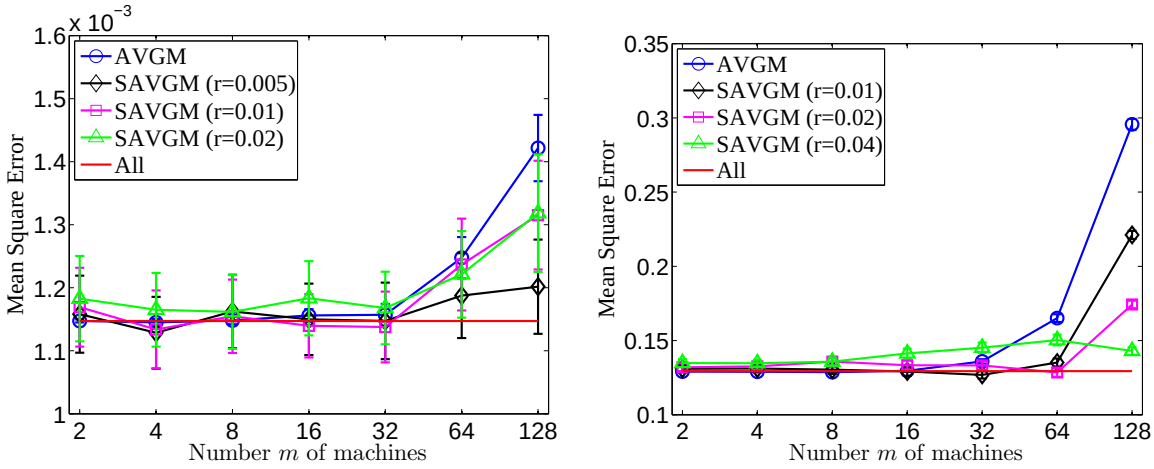
We now turn to developing an understanding of the SAVGM algorithm in comparison to the standard average mixture algorithm, developing intuition for the benefits and drawbacks of the method. Before describing the results, we remark that for the standard regression model (3.22), the least-squares solution is unbiased for θ^*, so we expect subsampled averaging to yield little (if any) improvement. The SAVGM method is essentially aimed at correcting the bias of the estimator $\overline{\theta}_1$, and de-biasing an unbiased estimator only increases its variance. However, for the mis-specified models (3.23) and (3.24) we expect to see some performance gains. In our experiments, we use multiple sub-sampling rates to study their effects, choosing $r \in \{0.005, 0.01, 0.02, 0.04\}$, where we recall that the output of the SAVGM algorithm is the vector $\widehat{W}\theta := (\overline{\theta}_1 - r\overline{\theta}_2)/(1-r)$.

We begin with experiments in which the data is generated as in the previous section. That is, to generate a feature vector $x \in \mathbb{R}^d$, choose five distinct indices in $\{1, \ldots, d\}$ uniformly at



(a) $d = 20$ (b) $d = 200$

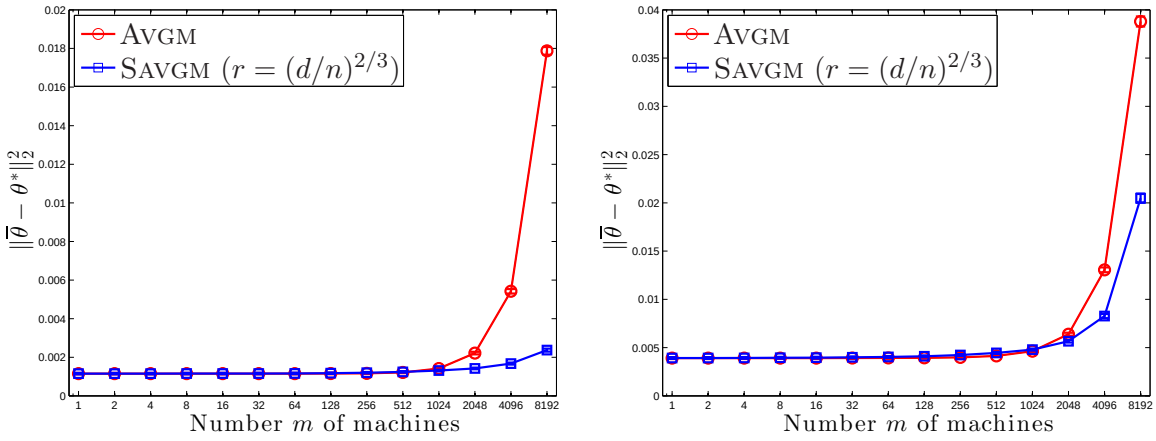
Figure 3.3. The error $\|\widehat{W}\theta - \theta^*\|_2^2$ plotted against the number of machines m for the AVGM and SAVGM methods, with standard errors across twenty simulations, using the normal regression model (3.22). The oracle estimator is denoted by the line "All."



(a) $d = 20$ (b) $d = 200$

Figure 3.4. The error $\|\widehat{W}\theta - \theta^*\|_2^2$ plotted against the number of machines m for the AVGM and SAVGM methods, with standard errors across twenty simulations, using the non-normal regression model (3.24). The oracle estimator is denoted by the line "All."

random, and the entries of x at those indices are distributed as $N(0, 1)$. In Figure 3.3, we plot the results of simulations comparing AVGM and SAVGM with data generated from the normal regression model (3.22). Both algorithms have have low error rates, but the AVGM method is slightly better than the SAVGM method for both values of the dimension d and all and sub-sampling rates r. As expected, in this case the SAVGM method does not offer improvement over AVGM, since the estimators are unbiased. (In Figure 3.3(a), we note that the standard error is in fact very small, since the mean-squared error is only of order 10^{-3}.)



(a) $d = 20$ (b) $d = 40$

Figure 3.5. The error $\|\widehat{W}\theta - \theta^*\|_2^2$ plotted against the number of machines m for the AVGM and SAVGM methods using regression model (3.23).

To understand settings in which subsampling for bias correction helps, in Figure 3.4, we plot mean-square error curves for the least-squares regression problem when the vector y is sampled according to the non-normal regression model (3.24). In this case, the least-squares estimator is biased for θ^* (which, as before, we estimate by solving a larger regression problem using $10N$ data samples). Figure 3.4 shows that both the AVGM and SAVGM method still enjoy good performance; in some cases, the SAVGM method even beats the oracle least-squares estimator for θ^* that uses all N samples. Since the AVGM estimate is biased in this case, its error curve increases roughly quadratically with m, which agrees with our theoretical predictions in Theorem 1. In contrast, we see that the SAVGM algorithm enjoys somewhat more stable performance, with increasing benefit as the number of machines m increases. For example, in case of $d = 200$, if we choose $r = 0.01$ for $m \leq 32$, choose $r = 0.02$ for $m = 64$ and $r = 0.04$ for $m = 128$, then SAVGM has performance comparable with the oracle method that uses all N samples. Moreover, we see that all the values of r—at least for the reasonably small values we use in the experiment—provide performance improvements over a non-subsampled distributed estimator.

For our final simulation, we plot results comparing SAVGM with AVGM in model (3.23), which is mis-specified but still a normal model. We use a simpler data generating mechanism, specifically, we draw $x \sim N(0, I_{d\times d})$ from a standard d-dimensional normal, and v is chosen uniformly in $[0, 1]$; in this case, the population minimizer has the closed form $\theta^* = u + 3v$. Figure 3.5 shows the results for dimensions $d = 20$ and $d = 40$ performed over 100 experiments (the standard errors are too small to see). Since the model (3.23) is not that badly mis-specified, the performance of the SAVGM method improves upon that of the AVGM method only for relatively large values of m, however, the performance of the SAVGM is always at least as good as that of AVGM.

Feature Name	Dimension	Description
Query	20000	Word tokens appearing in the query.
Gender	3	Gender of the user
Keyword	20000	Word tokens appearing in the purchase keywords.
Title	20000	Word tokens appearing in the ad title.
Advertiser	39191	Advertiser's ID
AdID	641707	Advertisement's ID.
Age	6	Age of the user
UserFreq	25	Number of appearances of the same user.
Position	3	Position of advertisement on search page.
Depth	3	Number of ads in the session.
QueryFreq	25	Number of occurrences of the same query.
AdFreq	25	Number of occurrences of the same ad.
QueryLength	20	Number of words in the query.
TitleLength	30	Number of words in the ad title.
DespLength	50	Number of words in the ad description.
QueryCtr	150	Average click-through-rate for query.
UserCtr	150	Average click-through-rate for user.
AdvrCtr	150	Average click-through-rate for advertiser.
WordCtr	150	Average click-through-rate for keyword advertised.
UserAdFreq	20	Number of times this user sees an ad.
UserQueryFreq	20	Number of times this user performs a search.

Table 3.1: Features used in online advertisement prediction problem.

3.4 Experiments with advertising data

Predicting whether a user of a search engine will click on an advertisement presented to him or her is of central importance to the business of several internet companies, and in this section, we present experiments studying the performance of the AVGM and SAVGM methods for this task. We use a large dataset from the Tencent search engine, `soso.com` [199], which contains 641,707 distinct advertisement items with $N = 235{,}582{,}879$ data samples.

Each sample consists of a so-called *impression*, which in the terminology of the information retrieval literature [e.g., see the book by 139], is a list containing a user-issued search, the advertisement presented to the user in response to the search, and a label $y \in \{+1, -1\}$ indicating whether the user clicked on the advertisement. The ads in our dataset were presented to 23,669,283 distinct users.

Transforming an impression into a useable set of regressors x is non-trivial, but the Tencent dataset provides a standard encoding. We list the features present in the data in Table 3.1, along with some description of their meaning. Each text-based feature—that is, those made up of words, which are Query, Keyword, and Title—is given a "bag-of-words" encoding [139]. This encoding assigns each of 20,000 possible words an index, and if the word

appears in the query (or Keyword or Title feature), the corresponding index in the vector x is set to 1. Words that do not appear are encoded with a zero. Real-valued features, corresponding to the bottom fifteen features in Table 3.1 beginning with "Age", are binned into a fixed number of intervals $[-\infty, a_1], a_1 a_2, \ldots, a_k \infty$, each of which is assigned an index in x. (Note that the intervals and number thereof vary per feature, and the dimension of the features listed in Table 3.1 corresponds to the number of intervals). When a feature falls into a particular bin, the corresponding entry of x is assigned a 1, and otherwise the entries of x corresponding to the feature are 0. Each feature has one additional value for "unknown." The remaining categorical features—gender, advertiser, and advertisement ID (AdID)—are also given $\{0, 1\}$ encodings, where only one index of x corresponding to the feature may be non-zero (which indicates the particular gender, advertiser, or AdID). This combination of encodings yields a binary-valued covariate vector $x \in \{0, 1\}^d$ with $d = 741{,}725$ dimensions. Note also that the features incorporate information about the user, advertisement, and query issued, encoding information about their interactions into the model.

Our goal is to predict the probability of a user clicking a given advertisement as a function of the covariates in Table 3.1. To do so, we use a logistic regression model to estimate the probability of a click response

$$P(y = 1 \mid x; \theta) := \frac{1}{1 + \exp(-\langle \theta, x \rangle)},$$

where $\theta \in \mathbb{R}^d$ is the unknown regression vector. We use the negative logarithm of P as the loss, incorporating a ridge regularization penalty. This combination yields instantaneous loss

$$\phi(\theta; (x, y)) = \log\left(1 + \exp(-y \langle \theta, x \rangle)\right) + \frac{\lambda}{2} \|\theta\|_2^2 . \tag{3.25}$$

In all our experiments, we assume that the population negative log-likelihood risk has local strong convexity as suggested by Assumption B. In practice, we use a small regularization parameter $\lambda = 10^{-6}$ to ensure fast convergence for the local sub-problems.

For this problem, we cannot evaluate the mean-squared error $\|\widehat{W}\theta - \theta^*\|_2^2$, as we do not know the true optimal parameter θ^*. Consequently, we evaluate the performance of an estimate $\widehat{W}\theta$ using log-loss on a held-out dataset. Specifically, we perform a five-fold validation experiment, where we shuffle the data and partition it into five equal-sized subsets. For each of our five experiments, we hold out one partition to use as the test set, using the remaining data as the training set for inference. When studying the AVGM or SAVGM method, we compute the local estimate θ_i via a trust-region Newton-based method [159] implemented by LIBSVM [43].

The dataset is too large to fit in the memory of most computers: in total, four splits of the data require 55 gigabytes. Consequently, it is difficult to provide an oracle training comparison using the full N samples. Instead, for each experiment, we perform 10 passes of stochastic dual coordinate ascent (SDCA) [185] and 10 passes of stochastic gradient descent (SGD) through the dataset to get two rough baselines of the performance attained by the

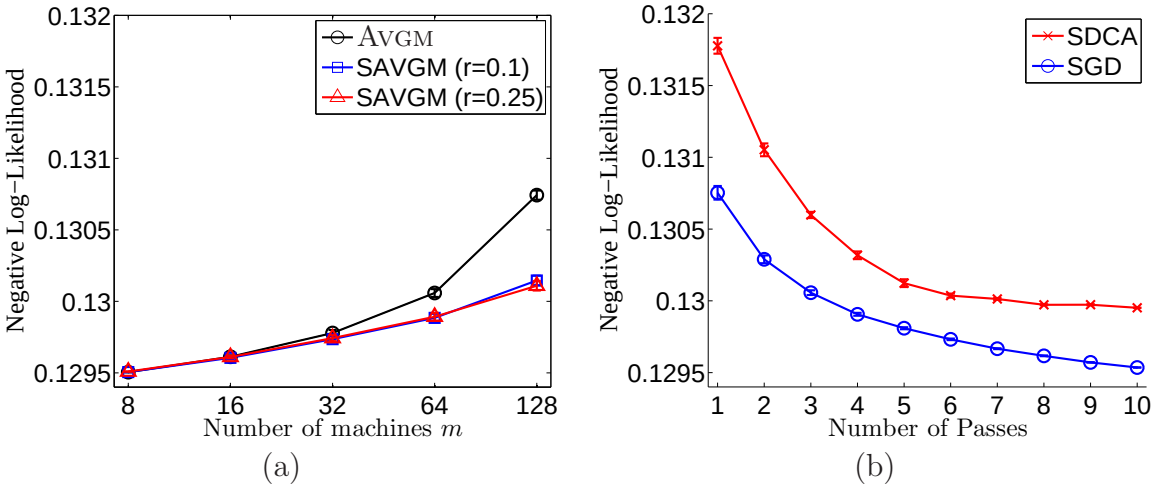


Figure 3.6. The negative log-likelihood of the output of the AVGM, SAVGM, and stochastic methods on the held-out dataset for the click-through prediction task. (a) Performance of the AVGM and SAVGM methods versus the number of splits m of the data. (b) Performance of SDCA and SGD baselines as a function of number of passes through the entire dataset.

empirical minimizer for the entire training dataset. Figure 3.6(b) shows the hold-out set log-loss after each of the sequential passes through the training data finishes. Note that although the SDCA enjoys faster convergence rate on the regularized empirical risk [185], the plot shows that the SGD has better generalization performance.

In Figure 3.6(a), we show the average hold-out set log-loss (with standard errors) of the estimator $\bar{\theta}_1$ provided by the AVGM method versus number of splits of the data m, and we also plot the log-loss of the SAVGM method using subsampling ratios of $r \in \{.1, .25\}$. The plot shows that for small m, both AVGM and SAVGM enjoy good performance, comparable to or better than (our proxy for) the oracle solution using all N samples. As the number of machines m grows, however, the de-biasing provided by the subsampled bootstrap method yields substantial improvements over the standard AVGM method. In addition, even with $m = 128$ splits of the dataset, the SAVGM method gives better hold-out set performance than performing two passes of stochastic gradient on the entire dataset of m samples; with $m = 64$, SAVGM enjoys performance as strong as looping through the data four times with stochastic gradient descent. This is striking, since doing even one pass through the data with stochastic gradient descent gives minimax optimal convergence rates [163, 5]. In ranking applications, rather than measuring negative log-likelihood, one may wish to use a direct measure of prediction error; to that end, Figure 3.7 shows plots of the area-under-the-curve (AUC) measure for the AVGM and SAVGM methods; AUC is a well-known measure of prediction error for bipartite ranking [139]. Broadly, this plot shows a similar story to that in Figure 3.6.

It is instructive and important to understand the sensitivity of the SAVGM method to the value of the resampling parameter r. We explore this question in Figure 3.8 using $m = 128$ splits, where we plot the log-loss of the SAVGM estimator on the held-out data set versus the subsampling ratio r. We choose $m = 128$ because more data splits provide more variable

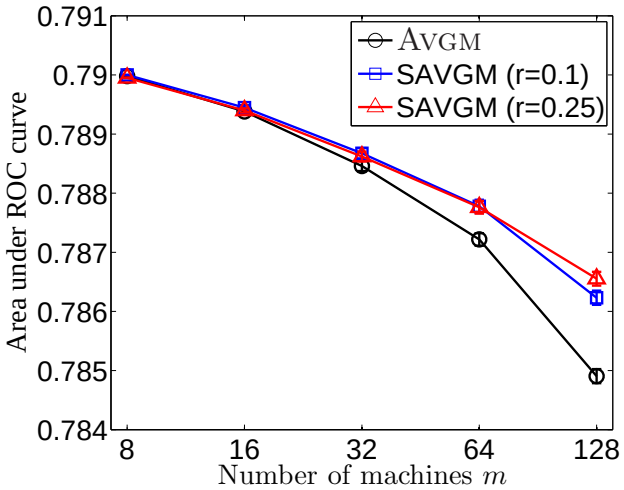


Figure 3.7. The area-under-the-curve (AUC) measure of ranking error for the output of the AVGM and SAVGM methods for the click-through prediction task.

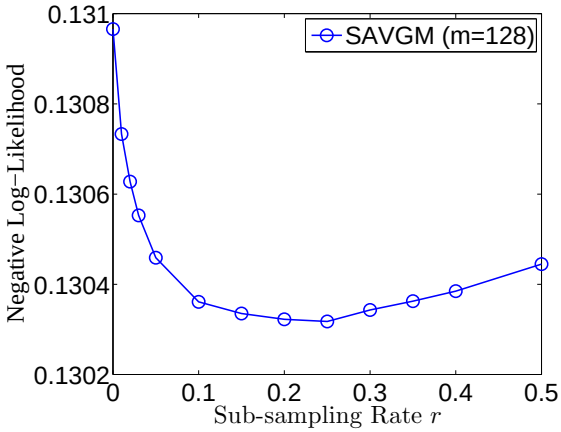


Figure 3.8. The log-loss on held-out data for the SAVGM method applied with $m = 128$ parallel splits of the data, plotted versus the sub-sampling rate r.

performance in r. For the `soso.com` ad prediction data set, the choice $r = .25$ achieves the best performance, but Figure 3.8 suggests that mis-specifying the ratio is not terribly detrimental. Indeed, while the performance of SAVGM degrades to that of the AVGM method, a wide range of settings of r give improved performance, and there does not appear to be a phase transition to poor performance.

3.5 Proofs of technical results

3.5.1 The necessity of smoothness

Here we show that some version of the smoothness conditions presented in Assumption C are necessary for averaging methods to attain better mean-squared error than using only the n samples on a single processor. Given the loss function (3.10), let $n_0 = \sum_{i=1}^n 1_{(X_i=0)}$ to be the count of 0 samples. Using θ_1 as shorthand for $\theta_{1,i}$, we see by inspection that the empirical minimizer θ_1 is

$$\theta_1 = \begin{cases} \frac{n_0}{n} - \frac{1}{2} & \text{when } n_0 \le n/2 \\ 1 - \frac{n}{2n_0} & \text{otherwise.} \end{cases}$$

For simplicity, we may assume that n is odd. In this case, we obtain that

$$\begin{aligned} \mathbb{E}[\theta_1] &= \frac{1}{4} + \mathbb{E}\left[\frac{n_0}{n} 1_{(n_0 < n/2)}\right] - \mathbb{E}\left[\frac{n}{2n_0} 1_{(n_0 > n/2)}\right] \\ &= \frac{1}{4} + \frac{1}{2^n} \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} \frac{i}{n} - \frac{1}{2^n} \sum_{i=\lceil n/2 \rceil}^{n} \binom{n}{i} \frac{n}{2i} = \frac{1}{4} + \frac{1}{2^n} \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} \left[\frac{i}{n} - \frac{n}{2(n-i)}\right] \end{aligned}$$

by the symmetry of the binomial. Adding and subtracting $\frac{1}{2}$ from the term within the braces, noting that $P(n_0 < n/2) = 1/2$, we have the equality

$$\mathbb{E}[\theta_1] = \frac{1}{2^n} \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} \left[\frac{i}{n} - \frac{n}{2(n-i)} + \frac{1}{2}\right] = \frac{1}{2^n} \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} \frac{i(n-2i)}{2n(n-i)}.$$

If Z is distributed normally with mean $1/2$ and variance $1/(4n)$, then an asymptotic expansion of the binomial distribution yields

$$\begin{aligned} \left(\frac{1}{2}\right)^n \sum_{i=0}^{\lfloor n/2 \rfloor} \binom{n}{i} \frac{i(n-2i)}{2n(n-i)} &= \mathbb{E}\left[\frac{Z(1-2Z)}{2-2Z} \mid 0 \le Z \le \frac{1}{2}\right] + o(n^{-1/2}) \\ &\ge \frac{1}{2}\mathbb{E}\left[Z - 2Z^2 \mid 0 \le Z \le \frac{1}{2}\right] + o(n^{-1/2}) = \Omega(n^{-\frac{1}{2}}), \end{aligned}$$

the final equality following from standard calculations, since $\mathbb{E}[|Z|] = \Omega(n^{-1/2})$.

3.5.2 Proof of Theorem 1

Although Theorem 1 is in terms of bounds on 8^{th} order moments, we prove a somewhat more general result in terms of a set of (k_0, k_1, k_2) moment conditions given by

$$\begin{aligned} &\mathbb{E}[\|\nabla \phi(\theta; X)\|_2^{k_0}] \le G^{k_0}, \qquad \mathbb{E}[\left\|\left|\nabla^2 \phi(\theta; X) - \nabla^2 f_0(\theta)\right|\right\|_2^{k_1}] \le H^{k_1}, \\ &\mathbb{E}[L(X)^{k_2}] \le L^{k_2} \quad \text{and} \quad \mathbb{E}[(L(X) - \mathbb{E}[L(X)])^{k_2}] \le L^{k_2} \end{aligned}$$

for $\theta \in U$. (Recall the definition of U prior to Assumption C). Doing so allows sharper control if higher moment bounds are available. The reader should recall throughout our arguments that we have assumed $\min\{k_0, k_1, k_2\} \geq 8$. Throughout the proof, we use f_1 and θ_1 to indicate the local empirical objective and empirical minimizer of machine 1 (which have the same distribution as those of the other processors), and we recall the notation $1_{(\mathcal{E})}$ for the indicator function of the event $\mathcal{E}$.

Before beginning the proof of Theorem 1 proper, we begin with a simple inequality that relates the error term $\overline{\theta} - \theta^*$ to an average of the errors $\theta_i - \theta^*$, each of which we can bound in turn. Specifically, a bit of algebra gives us that

$$\begin{aligned}
\mathbb{E}[\left\|\overline{\theta} - \theta^*\right\|_2^2] &= \mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^m \theta_i - \theta^*\right\|_2^2\right] \\
&= \frac{1}{m^2}\sum_{i=1}^m \mathbb{E}[\|\theta_i - \theta^*\|_2^2] + \frac{1}{m^2}\sum_{i \neq j} \mathbb{E}[\langle \theta_i - \theta^*, \theta_j - \theta^* \rangle] \\
&\leq \frac{1}{m}\mathbb{E}[\|\theta_1 - \theta^*\|_2^2] + \frac{m(m-1)}{m^2}\left\|\mathbb{E}[\theta_1 - \theta^*]\right\|_2^2 \\
&\leq \frac{1}{m}\mathbb{E}[\|\theta_1 - \theta^*\|_2^2] + \left\|\mathbb{E}[\theta_1 - \theta^*]\right\|_2^2 . \qquad (3.26)
\end{aligned}$$

Here we used the definition of the averaged vector $\overline{\theta}$ and the fact that for $i \neq j$, the vectors θ_i and θ_j are statistically independent, they are functions of independent samples. The upper bound (3.26) illuminates the path for the remainder of our proof: we bound each of $\mathbb{E}[\|\theta_i - \theta^*\|_2^2]$ and $\|\mathbb{E}[\theta_i - \theta^*]\|_2^2$. Intuitively, since our objective is locally strongly convex by Assumption B, the empirical minimizing vector θ_1 is a nearly unbiased estimator for θ^*, which allows us to prove the convergence rates in the theorem.

We begin by defining three events—which we (later) show hold with high probability—that guarantee the closeness of θ_1 and θ^*. In rough terms, when these events hold, the function f_1 behaves similarly to the population risk f_0 around the point θ^*; since f_0 is locally strongly convex, the minimizer θ_1 of f_1 will be close to θ^*. Recall that Assumption C guarantees the existence of a ball $U_\rho = \{\theta \in \mathbb{R}^d : \|\theta - \theta^*\|_2 < \rho\}$ of radius $\rho \in (0, 1)$ such that

$$\left\|\left\|\nabla^2\phi(\theta; x) - \nabla^2\phi(\theta'; x)\right\|\right\|_2 \leq L(x)\left\|\theta - \theta'\right\|_2$$

for all $\theta, \theta' \in U_\rho$ and any x, where $\mathbb{E}[L(X)^{k_2}] \leq L^{k_2}$. In addition, Assumption B guarantees that $\nabla^2 f_0(\theta^*) \succeq \lambda I$. Now, choosing the potentially smaller radius $\delta_\rho = \min\{\rho, \rho\lambda/4L\}$, we

can define the three "good" events

$$\begin{aligned}
\mathcal{E}_0 &:= \left\{ \frac{1}{n} \sum_{i=1}^{n} L(X_i) \leq 2L \right\}, \\
\mathcal{E}_1 &:= \left\{ \left| \left| \left| \nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*) \right| \right| \right|_2 \leq \frac{\rho\lambda}{2} \right\}, \quad \text{and} \\
\mathcal{E}_2 &:= \left\{ \|\nabla f_1(\theta^*)\|_2 \leq \frac{(1-\rho)\lambda\delta_\rho}{2} \right\}.
\end{aligned} \tag{3.27}$$

We then have the following lemma:

Lemma 3. *Under the events $\mathcal{E}_0$, $\mathcal{E}_1$, and $\mathcal{E}_2$ previously defined (3.27), we have*

$$\|\theta_1 - \theta^*\|_2 \leq \frac{2\|\nabla f_1(\theta^*)\|_2}{(1-\rho)\lambda}, \quad \textit{and} \quad \nabla^2 f_1(\theta) \succeq (1-\rho)\lambda I_{d\times d}.$$

The proof of Lemma 3 relies on some standard optimization guarantees relating gradients to minimizers of functions (e.g. [33], Chapter 9), although some care is required since smoothness and strong convexity hold only locally in our problem. As the argument is somewhat technical, we defer it to Appendix 3.5.5.

Our approach from here is to give bounds on $\mathbb{E}[\|\theta_1 - \theta^*\|_2^2]$ and $\|\mathbb{E}[\theta_1 - \theta^*]\|_2^2$ by careful Taylor expansions, which allows us to bound $\mathbb{E}[\|\bar{\theta}_1 - \theta^*\|_2^2]$ via our initial expansion (3.26). We begin by noting that whenever the events $\mathcal{E}_0$, $\mathcal{E}_1$, and $\mathcal{E}_2$ hold, then $\nabla f_1(\theta_1) = 0$, and moreover, by a Taylor series expansion of ∇f_1 between θ^* and θ_1, we have

$$0 = \nabla f_1(\theta_1) = \nabla f_1(\theta^*) + \nabla^2 f_1(\theta')(\theta_1 - \theta^*)$$

where $\theta' = \kappa\theta^* + (1-\kappa)\theta_1$ for some $\kappa \in [0,1]$. By adding and subtracting terms, we have

$$\begin{aligned}
0 = \nabla f_1(\theta^*) &+ (\nabla^2 f_1(\theta') - \nabla^2 f_1(\theta^*))(\theta_1 - \theta^*) \\
&+ (\nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*))(\theta_1 - \theta^*) + \nabla^2 f_0(\theta^*)(\theta_1 - \theta^*).
\end{aligned} \tag{3.28}$$

Since $\nabla^2 f_0(\theta^*) \succeq \lambda I$, we can define the inverse Hessian matrix $\Sigma^{-1} := [\nabla^2 f_0(\theta^*)]^{-1}$, and setting $\Delta := \theta_1 - \theta^*$, we multiply both sides of the Taylor expansion (3.28) by Σ^{-1} to obtain the relation

$$\Delta = -\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \nabla^2 f_1(\theta'))\Delta + \Sigma^{-1}(\nabla^2 f_0(\theta^*) - \nabla^2 f_1(\theta^*))\Delta. \tag{3.29}$$

Thus, if we define the matrices $P = \nabla^2 f_0(\theta^*) - \nabla^2 f_1(\theta^*)$ and $Q = \nabla^2 f_1(\theta^*) - \nabla^2 f_1(\theta')$, equality (3.29) can be re-written as

$$\theta_1 - \theta^* = -\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)(\theta_1 - \theta^*). \tag{3.30}$$

Note that equation (3.30) holds when the conditions of Lemma 3 hold, and otherwise we may simply assert only that $\|\theta_1 - \theta^*\|_2 \leq R$. Roughly, we expect the final two terms in the error

expansion (3.30) to be of smaller order than the first term, since we hope that $\theta_1 - \theta^* \to 0$ and additionally that the Hessian differences decrease to zero at a sufficiently fast rate. We now formalize this intuition.

Inspecting the Taylor expansion (3.30), we see that there are several terms of a form similar to $(\nabla^2 f_0(\theta^*) - \nabla^2 f_1(\theta^*))(\theta_1 - \theta^*)$; using the smoothness Assumption C, we can convert these terms into higher order terms involving only $\theta_1 - \theta^*$. Thus, to effectively control the expansions (3.29) and (3.30), we must show that higher order terms of the form $\mathbb{E}[\|\theta_1 - \theta^*\|_2^k]$, for $k \geq 2$, decrease quickly enough in n.

Control of $\mathbb{E}[\|\theta_1 - \theta^*\|_2^k]$: Recalling the events (3.27), we define $\mathcal{E} := \mathcal{E}_0 \cap \mathcal{E}_1 \cap \mathcal{E}_2$ and then observe that

$$\begin{aligned}
\mathbb{E}[\|\theta_1 - \theta^*\|_2^k] &= \mathbb{E}[1_{(\mathcal{E})} \|\theta_1 - \theta^*\|_2^k] + \mathbb{E}[1_{(\mathcal{E}^c)} \|\theta_1 - \theta^*\|_2^k] \\
&\leq \frac{2^k \mathbb{E}[1_{(\mathcal{E})} \|\nabla f_1(\theta^*)\|_2^k]}{(1-\rho)^k \lambda^k} + \mathbb{P}(\mathcal{E}^c) R^k \\
&\leq \frac{2^k \mathbb{E}[\|\nabla f_1(\theta^*)\|_2^k]}{(1-\rho)^k \lambda^k} + \mathbb{P}(\mathcal{E}^c) R^k,
\end{aligned}$$

where we have used the bound $\|\theta - \theta^*\|_2 \leq R$ for all $\theta \in \Theta$, from Assumption A. Our goal is to prove that $\mathbb{E}[\|\nabla f_1(\theta^*)\|_2^k] = \mathcal{O}(n^{-k/2})$ and that $\mathbb{P}(\mathcal{E}^c) = \mathcal{O}(n^{-k/2})$. We move forward with a two lemmas that lay the groundwork for proving these two facts:

Lemma 4. *Under Assumption C, there exist constants C and C' (dependent only on the moments k_0 and k_1 respectively) such that*

$$\mathbb{E}[\|\nabla f_1(\theta^*)\|_2^{k_0}] \leq C \frac{G^{k_0}}{n^{k_0/2}}, \quad \text{and} \tag{3.31}$$

$$\mathbb{E}[\|\nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*)\|_2^{k_1}] \leq C' \frac{\log^{k_1/2}(2d) H^{k_1}}{n^{k_1/2}}. \tag{3.32}$$

See Appendix 3.5.6 for the proof of this claim.

As an immediate consequence of Lemma 4, we see that the events $\mathcal{E}_1$ and $\mathcal{E}_2$ defined by (3.27) occur with reasonably high probability. Indeed, recalling that $\mathcal{E} = \mathcal{E}_0 \cap \mathcal{E}_1 \cap \mathcal{E}_2$, Boole's law and the union bound imply

$$\begin{aligned}
&\mathbb{P}(\mathcal{E}^c) = \mathbb{P}(\mathcal{E}_0^c \cup \mathcal{E}_1^c \cup \mathcal{E}_2^c) \\
&\leq \mathbb{P}(\mathcal{E}_0^c) + \mathbb{P}(\mathcal{E}_1^c) + \mathbb{P}(\mathcal{E}_2^c) \\
&\leq \frac{\mathbb{E}[|\frac{1}{n}\sum_{i=1}^n L(X_i) - \mathbb{E}[L(X)]|^{k_2}]}{L^{k_2}} + \frac{2^{k_1}\mathbb{E}[\|\nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*)\|_2^{k_1}]}{\rho^{k_1}\lambda^{k_1}} + \frac{2^{k_0}\mathbb{E}[\|\nabla f_1(\theta^*)\|_2^{k_0}]}{(1-\rho)^{k_0}\lambda^{k_0}\delta_\rho^{k_0}} \\
&\leq C_2 \frac{1}{n^{k_2/2}} + C_1 \frac{\log^{k_1/2}(2d) H^{k_1}}{n^{k_1/2}} + C_0 \frac{G^{k_0}}{n^{k_0/2}}
\end{aligned} \tag{3.33}$$

for some universal constants C_0, C_1, C_2, where in the second-to-last line we have invoked the moment bound in Assumption C. Consequently, we find that

$$\mathbb{P}(\mathcal{E}^c)R^k = \mathcal{O}(R^k(n^{-k_1/2} + n^{-k_2/2} + n^{-k_0/2}) \quad \text{for any } k \in \mathbb{N}.$$

In summary, we have proved the following lemma:

Lemma 5. *Let Assumptions B and C hold. For any $k \in \mathbb{N}$ with $k \leq \min\{k_0, k_1, k_2\}$, we have*

$$\mathbb{E}[\|\theta_1 - \theta^*\|_2^k] = \mathcal{O}\left(n^{-k/2} \cdot \frac{G^k}{(1-\rho)^k\lambda^k} + n^{-k_0/2} + n^{-k_1/2} + n^{-k_2/2}\right) = \mathcal{O}\left(n^{-k/2}\right),$$

where the order statements hold as $n \to +\infty$.

Now recall the matrix $Q = \nabla^2 f_1(\theta^*) - \nabla^2 f_1(\theta')$ defined following equation (3.29). The following result controls the moments of its operator norm:

Lemma 6. *For $k \leq \min\{k_2, k_1, k_0\}/2$, we have $\mathbb{E}[|||Q|||_2^k] = \mathcal{O}(n^{-k/2})$.*

Proof We begin by using Jensen's inequality and Assumption C to see that

$$|||Q|||^k \leq \frac{1}{n}\sum_{i=1}^n \left|\left|\left|\nabla^2\phi(\theta'; X_i) - \nabla^2\phi(\theta^*; X_i)\right|\right|\right|^k \leq \frac{1}{n}\sum_{i=1}^n L(X_i)^k \|\theta' - \theta^*\|_2^k.$$

Now we apply the Cauchy-Schwarz inequality and Lemma 5, thereby obtaining

$$\mathbb{E}[|||Q|||_2^k] \leq \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^n L(X_i)^k\right)^2\right]^{\frac{1}{2}} \mathbb{E}\left[\|\theta_1 - \theta^*\|_2^{2k}\right]^{\frac{1}{2}} = \mathcal{O}\left(L^k \frac{G^k}{(1-\rho)^k\lambda^k} n^{-k/2}\right),$$

where we have used Assumption C again. □

Lemma 5 allows us to control the first term from our initial bound (3.26) almost immediately. Indeed, using our last Taylor expansion (3.30) and the definition of the event $\mathcal{E} = \mathcal{E}_0 \cap \mathcal{E}_1 \cap \mathcal{E}_2$, we have

$$\begin{aligned}
\mathbb{E}[\|\theta_1 - \theta^*\|_2^2] &= \mathbb{E}\left[1_{(\mathcal{E})}\left\|-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)(\theta_1 - \theta^*)\right\|_2^2\right] + \mathbb{E}[1_{(\mathcal{E}^c)}\|\theta_1 - \theta^*\|_2^2] \\
&\leq 2\mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^2\right] + 2\mathbb{E}\left[\left\|\Sigma^{-1}(P+Q)(\theta_1 - \theta^*)\right\|_2^2\right] + \mathbb{P}(\mathcal{E}^c)R^2,
\end{aligned}$$

where we have applied the inequality $(a+b)^2 \leq 2a^2 + 2b^2$. Again using this same inequality, then applying Cauchy-Schwarz and Lemmas 5 and 6, we see that

$$\begin{aligned}
\mathbb{E}\left[\left\|\Sigma^{-1}(P+Q)(\theta_1 - \theta^*)\right\|_2^2\right] &\leq 2\left|\left|\left|\Sigma^{-1}\right|\right|\right|_2^2 \left(\mathbb{E}[|||P|||_2^2\|\theta_1 - \theta^*\|_2^2] + \mathbb{E}[|||Q|||_2^2\|\theta_1 - \theta^*\|_2^2]\right) \\
&\leq 2\left|\left|\left|\Sigma^{-1}\right|\right|\right|_2^2 \left(\sqrt{\mathbb{E}[|||P|||_2^4]\mathbb{E}[\|\theta_1 - \theta^*\|_2^4]} + \sqrt{\mathbb{E}[|||Q|||_2^4]\mathbb{E}[\|\theta_1 - \theta^*\|_2^4]}\right) \\
&= \mathcal{O}(n^{-2}),
\end{aligned}$$

where we have used the fact that $\min\{k_0, k_1, k_2\} \geq 8$ to apply Lemma 6. Combining these results, we obtain the upper bound

$$\mathbb{E}[\|\theta_1 - \theta^*\|_2^2] \leq 2\mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^2\right] + \mathcal{O}(n^{-2}), \tag{3.34}$$

which completes the first part of our proof of Theorem 1.

Control of $\|\mathbb{E}[\theta_1 - \theta^*]\|_2^2$: It remains to consider the $\|\mathbb{E}[\theta_1 - \theta^*]\|_2^2$ term from our initial error inequality (3.26). When the events (3.27) occur, we know that all derivatives exist, so we may recursively apply our expansion (3.30) of $\theta_1 - \theta^*$ to find that

$$\begin{aligned}\theta_1 - \theta^* &= -\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)(\theta_1 - \theta^*) \\ &= \underbrace{-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)\left[-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)(\theta_1 - \theta^*)\right]}_{=:v}\end{aligned} \tag{3.35}$$

where we have introduced v as shorthand for the vector on the right hand side. Thus, with a bit of algebraic manipulation we obtain the relation

$$\theta_1 - \theta^* = 1_{(\mathcal{E})}v + 1_{(\mathcal{E}^c)}(\theta_1 - \theta^*) = v + 1_{(\mathcal{E}^c)}(\theta_1 - \theta^*) - 1_{(\mathcal{E}^c)}v = v + 1_{(\mathcal{E}^c)}(\theta_1 - \theta^* - v). \tag{3.36}$$

Now note that $\mathbb{E}[\nabla f_1(\theta^*)] = 0$ thus

$$\begin{aligned}\mathbb{E}[v] &= \mathbb{E}\left[-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)[-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)(\theta_1 - \theta^*)]\right] \\ &= \mathbb{E}\left[\Sigma^{-1}(P+Q)\Sigma^{-1}\left[(P+Q)(\theta_1 - \theta^*) - \nabla f_1(\theta^*)\right]\right].\end{aligned}$$

Thus, by re-substituting the appropriate quantities in (3.36) and applying the triangle inequality, we have

$$\begin{aligned}&\|\mathbb{E}[\theta_1 - \theta^*]\|_2 \\ &\leq \left\|\mathbb{E}[\Sigma^{-1}(P+Q)\Sigma^{-1}\left((P+Q)(\theta_1 - \theta^*) - \nabla f_1(\theta^*)\right)]\right\|_2 + \left\|\mathbb{E}[1_{(\mathcal{E}^c)}(\theta_1 - \theta^* - v)]\right\|_2 \\ &\leq \left\|\mathbb{E}[\Sigma^{-1}(P+Q)\Sigma^{-1}\left((P+Q)(\theta_1 - \theta^*) - \nabla f_1(\theta^*)\right)]\right\|_2 + \mathbb{E}[1_{(\mathcal{E}^c)}\|\theta_1 - \theta^*\|_2] \\ &\quad + \mathbb{E}\left[1_{(\mathcal{E}^c)}\left\|-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)\Sigma^{-1}\left[-\nabla f_1(\theta^*) + (P+Q)(\theta_1 - \theta^*)\right]\right\|_2\right].\end{aligned} \tag{3.37}$$

Since $\|\theta_1 - \theta^*\|_2 \leq R$ by assumption, we have

$$\mathbb{E}[1_{(\mathcal{E}^c)}\|\theta_1 - \theta^*\|_2] \leq \mathbb{P}(\mathcal{E}^c)R \overset{(i)}{=} \mathcal{O}(Rn^{-k/2})$$

for any $k \leq \min\{k_2, k_1, k_0\}$, where step (i) follows from the inequality (3.33). Hölder's inequality also yields that

$$\begin{aligned}\mathbb{E}\left[1_{(\mathcal{E}^c)}\left\|\Sigma^{-1}(P+Q)\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2\right] &\leq \mathbb{E}\left[1_{(\mathcal{E}^c)}\left\|\Sigma^{-1}(P+Q)\right\|_2\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2\right] \\ &\leq \sqrt{\mathbb{P}(\mathcal{E}^c)}\mathbb{E}\left[\left\|\Sigma^{-1}(P+Q)\right\|_2^4\right]^{1/4}\mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^4\right]^{1/4}.\end{aligned}$$

Recalling Lemmas 4 and 6, we have $\mathbb{E}[\|\Sigma^{-1}(P+Q)\|_2^4] = \mathcal{O}(\log^2(d)n^{-2})$, and we similarly have $\mathbb{E}[\|\Sigma^{-1}\nabla f_1(\theta^*)\|_2^4] = \mathcal{O}(n^{-2})$. Lastly, we have $\mathbb{P}(\mathcal{E}^c) = \mathcal{O}(n^{-k/2})$ for $k \leq \min\{k_0, k_1, k_2\}$, whence we find that for any such k,

$$\mathbb{E}\left[1_{(\mathcal{E}^c)} \left\|\Sigma^{-1}(P+Q)\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2\right] = \mathcal{O}\left(\sqrt{\log(d)}n^{-k/4-1}\right).$$

We can similarly apply Lemma 5 to the last remaining term in the inequality (3.37) to obtain that for any $k \leq \min\{k_2, k_1, k_0\}$,

$$\mathbb{E}\left[1_{(\mathcal{E}^c)} \left\|-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)\left[-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(P+Q)(\theta_1-\theta^*)\right]\right\|_2\right] = \mathcal{O}(n^{-k/2} + n^{-k/4-1}).$$

Applying these two bounds, we find that

$$\|\mathbb{E}[\theta_1 - \theta^*]\|_2 \leq \left\|\mathbb{E}\left[\Sigma^{-1}(P+Q)\Sigma^{-1}\left((P+Q)(\theta_1-\theta^*) - \nabla f_1(\theta^*)\right)\right]\right\|_2 + \mathcal{O}(n^{-k}) \tag{3.38}$$

for any k such that $k \leq \min\{k_0, k_1, k_2\}/2$ and $k \leq \min\{k_0, k_1, k_2\}/4 + 1$.

In the remainder of the proof, we show that part of the bound (3.38) still consists only of higher-order terms, leaving us with an expression not involving $\theta_1 - \theta^*$. To that end, note that

$$\mathbb{E}\left[\left\|\Sigma^{-1}(P+Q)\Sigma^{-1}(P+Q)(\theta_1-\theta^*)\right\|_2^2\right] = \mathcal{O}(n^{-3})$$

by three applications of Hölder's inequality, the fact that $\|Ax\|_2 \leq \|A\|_2 \|x\|_2$, and Lemmas 4, 5 and 6. Coupled with our bound (3.38), we use the fact that $(a+b)^2 \leq 2a^2 + 2b^2$ to obtain

$$\|\mathbb{E}[\theta_1 - \theta^*]\|_2^2 \leq 2\left\|\mathbb{E}[\Sigma^{-1}(P+Q)\Sigma^{-1}\nabla f_1(\theta^*)]\right\|_2^2 + \mathcal{O}(n^{-3}). \tag{3.39}$$

We focus on bounding the remaining expectation. We have the following series of inequalities:

$$\begin{aligned}
\left\|\mathbb{E}[\Sigma^{-1}(P+Q)\Sigma^{-1}\nabla f_1(\theta^*)]\right\|_2 &\overset{(i)}{\leq} \mathbb{E}\left[\|\Sigma^{-1}(P+Q)\|_2 \left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2\right] \\
&\overset{(ii)}{\leq} \left(\mathbb{E}\left[\|\Sigma^{-1}(P+Q)\|_2^2\right] \mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^2\right]\right)^{\frac{1}{2}} \\
&\overset{(iii)}{\leq} \left(2\mathbb{E}\left[\|\Sigma^{-1}P\|_2^2 + \|\Sigma^{-1}Q\|_2^2\right] \mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^2\right]\right)^{\frac{1}{2}}.
\end{aligned}$$

Here step (i) follows from Jensen's inequality and the fact that $\|Ax\|_2 \leq \|A\|_2 \|x\|_2$; step (ii) uses the Cauchy-Schwarz inequality; and step (iii) follows from the fact that $(a+b)^2 \leq 2a^2 + 2b^2$. We have already bounded the first two terms in the product in our proofs; in particular, Lemma 4 guarantees that $\mathbb{E}[\|P\|_2^2] \leq CH\log d/n$, while

$$\mathbb{E}[\|Q\|_2^2] \leq \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n L(X_i)^4\right]^{\frac{1}{2}} \mathbb{E}[\|\theta_1 - \theta^*\|_2^4]^{\frac{1}{2}} \leq C\frac{L^2G^2}{(1-\rho)^2\lambda^2} \cdot n^{-1}$$

for some numerical constant C (recall Lemma 6). Summarizing our bounds on $\|\!|P|\!\|_2$ and $\|\!|Q|\!\|_2$, we have

$$\begin{aligned}&\left\|\mathbb{E}\left[\Sigma^{-1}(P+Q)\Sigma^{-1}\nabla f_1(\theta^*)\right]\right\|_2^2 \\ &\leq 2\left\|\!\left|\Sigma^{-1}\right|\!\right\|_2^2\left(\frac{2H^2(\log d+1)}{n}+2C\frac{L^2G^2}{(1-\rho)^2\lambda^2 n}+\mathcal{O}(n^{-2})\right)\mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^2\right]. \quad (3.40)\end{aligned}$$

From Assumption C we know that $\mathbb{E}[\|\nabla f_1(\theta^*)\|_2^2] \leq G^2/n$ and $\|\!|\Sigma^{-1}|\!\|_2 \leq 1/\lambda$, and hence we can further simplify the bound (3.40) to obtain

$$\begin{aligned}\|\mathbb{E}[\theta_1-\theta^*]\|_2^2 &\leq \frac{C}{\lambda^2}\left(\frac{H^2\log d+L^2G^2/\lambda^2(1-\rho)^2}{n}\right)\mathbb{E}\left[\left\|\Sigma^{-1}\nabla f_1(\theta^*)\right\|_2^2\right]+\mathcal{O}(n^{-3}) \\ &= \frac{C}{\lambda^2}\left(\frac{H^2\log d+L^2G^2/\lambda^2(1-\rho)^2}{n^2}\right)\mathbb{E}\left[\left\|\Sigma^{-1}\nabla\phi(\theta^*;X)\right\|_2^2\right]+\mathcal{O}(n^{-3})\end{aligned}$$

for some numerical constant C, where we have applied our earlier inequality (3.39). Noting that we may (without loss of generality) take $\rho < \frac{1}{2}$, then applying this inequality with the bound (3.34) on $\mathbb{E}[\|\theta_1-\theta^*\|_2^2]$ we previously proved to our decomposition (3.26) completes the proof.

3.5.3 Proof of Theorem 2

Our proof of Theorem 2 begins with a simple inequality that mimics our first inequality (3.26) in the proof of Theorem 1. Recall the definitions of the averaged vector $\overline{\theta}_1$ and subsampled averaged vector $\overline{\theta}_2$. Let θ_1 denote the minimizer of the (an arbitrary) empirical risk f_1, and θ_2 denote the minimizer of the resampled empirical risk f_2 (from the same samples as θ_1). Then we have

$$\mathbb{E}\left[\left\|\frac{\overline{\theta}_1-r\overline{\theta}_2}{1-r}-\theta^*\right\|_2^2\right] \leq \left\|\mathbb{E}\left[\frac{\theta_1-r\theta_2}{1-r}-\theta^*\right]\right\|_2^2+\frac{1}{m}\mathbb{E}\left[\left\|\frac{\theta_1-r\theta_2}{1-r}-\theta^*\right\|_2^2\right]. \quad (3.41)$$

Thus, parallel to our proof of Theorem 1, it suffices to bound the two terms in the decomposition (3.41) separately. Specifically, we prove the following two lemmas.

Lemma 7. *Under the conditions of Theorem 2,*

$$\left\|\mathbb{E}\left[\frac{\theta_1-r\theta_2}{1-r}-\theta^*\right]\right\|_2^2 \leq \mathcal{O}(1)\frac{1}{r(1-r)^2}\left(\frac{M^2G^6}{\lambda^6}+\frac{G^4L^2}{\lambda^4}d\log d\right)\frac{1}{n^3}. \quad (3.42)$$

Lemma 8. *Under the conditions of Theorem 2,*

$$\mathbb{E}\left[\|\theta_1-\theta^*-r(\theta_2-\theta^*)\|_2^2\right] \leq (2+3r)\mathbb{E}\left[\left\|\nabla^2 f_0(\theta^*)^{-1}\nabla f_1(\theta^*)\right\|_2^2\right]+\mathcal{O}(n^{-2}) \quad (3.43)$$

In conjunction, Lemmas 7 and 8 coupled with the decomposition (3.41) yield the desired claim. Indeed, applying each of the lemmas to the decomposition (3.41), we see that

$$\mathbb{E}\left[\left\|\frac{\bar{\theta}_1 - r\bar{\theta}_2}{1-r} - \theta^*\right\|_2^2\right] \leq \frac{2+3r}{(1-r)^2 m}\mathbb{E}\left[\left\|\nabla^2 f_0(\theta^*)^{-1}\nabla f_1(\theta^*)\right\|_2^2\right] + \mathcal{O}\left(\frac{1}{(1-r)^2}m^{-1}n^{-2}\right) + \mathcal{O}\left(\frac{1}{r(1-r)^2}n^{-3}\right),$$

which is the statement of Theorem 2.

The remainder of our argument is devoted to establishing Lemmas 7 and 8. Before providing their proofs (in Section 3.5.3.3 and 3.5.3.4 respectively), we require some further set-up and auxiliary results. Throughout the rest of the proof, we use the notation

$$Y = Y' + \mathcal{R}_k$$

for some random variables Y and Y' to mean that there exists a random variable Z such that $Y = Y' + Z$ and $\mathbb{E}[\|Z\|_2^2] = \mathcal{O}(n^{-k})$.[1] The symbol $\mathcal{R}_k$ may indicate different random variables throughout a proof and is notational shorthand for a moment-based big-O notation. We also remark that if we have $\mathbb{E}[\|Z\|_2^2] = \mathcal{O}(a^k n^{-k})$, we have $Z = a^{k/2}\mathcal{R}_k$, since $(a^{k/2})^2 = a^k$. For shorthand, we also say that $\mathbb{E}[Z] = \mathcal{O}(h(n))$ if $\|\mathbb{E}[Z]\|_2 = \mathcal{O}(h(n))$, which implies that if $Z = \mathcal{R}_k$ then $\mathbb{E}[Z] = \mathcal{O}(n^{-k/2})$, since

$$\|\mathbb{E}[Z]\|_2 \leq \sqrt{\mathbb{E}[\|Z\|_2^2]} = \mathcal{O}(n^{-k/2}).$$

3.5.3.1 Optimization Error Expansion

In this section, we derive a sharper asymptotic expansion of the optimization errors $\theta_1 - \theta^*$. Recall our definition of the Kronecker product $\otimes$, where for vectors u, v we have $u \otimes v = uv^\top$. With this notation, we have the following expansion of $\theta_1 - \theta^*$. In these lemmas, $\mathcal{R}_3$ denotes a vector Z for which $\mathbb{E}[\|Z\|_2^2] \leq cn^{-3}$ for a numerical constant c.

Lemma 9. *Under the conditions of Theorem 2, we have*

$$\begin{aligned}\theta_1 - \theta^* = &-\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*) \\ &- \Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_1(\theta^*)) \otimes (\Sigma^{-1}\nabla f_1(\theta^*))\right) \\ &+ \left(M^2 G^6/\lambda^6 + G^4 L^2 d\log(d)/\lambda^4\right)\mathcal{R}_3.\end{aligned} \tag{3.44}$$

We prove Lemma 9 in Appendix 3.5.7. The lemma requires careful moment control over the expansion $\theta_1 - \theta^*$, leading to some technical difficulty, but is similar in spirit to the results leading to Theorem 1.

An immediately analogous result to Lemma 9 follows for our sub-sampled estimators. Since we use $\lceil rn \rceil$ samples to compute θ_2, the second level estimator, we find

[1] Formally, in our proof this will mean that there exist random vectors Y, Y', and Z that are measurable with respect to the σ-field $\sigma(X_1, \ldots, X_n)$, where $Y = Y' + Z$ and $\mathbb{E}[\|Z\|_2^2] = \mathcal{O}(n^{-k})$.

Lemma 10. *Under the conditions of Theorem 2, we have*

$$\begin{aligned}\theta_2 - \theta^* = &-\Sigma^{-1}\nabla f_2(\theta^*) + \Sigma^{-1}(\nabla^2 f_2(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_2(\theta^*) \\ &- \Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_2(\theta^*)) \otimes (\Sigma^{-1}\nabla f_2(\theta^*))\right) \\ &+ r^{-\frac{3}{2}}\left(M^2G^6/\lambda^6 + G^4L^2 d\log(d)/\lambda^4\right)\mathcal{R}_3.\end{aligned}$$

3.5.3.2 Bias Correction

Now that we have given Taylor expansions that describe the behavior of $\theta_1 - \theta^*$ and $\theta_2 - \theta^*$, we can prove Lemmas 7 and 8 (though, as noted earlier, we defer the proof of Lemma 8 to Appendix 3.5.3.4). The key insight is that expectations of terms involving $\nabla f_2(\theta^*)$ are nearly the same as expectations of terms involving $\nabla f_1(\theta^*)$, except that some corrections for the sampling ratio r are necessary.

We begin by noting that

$$\frac{\theta_1 - r\theta_2}{1-r} - \theta^* = \frac{\theta_1 - \theta^*}{1-r} - r\frac{\theta_2 - \theta^*}{1-r}. \tag{3.45}$$

In Lemmas 9 and 10, we derived expansions for each of the right hand side terms, and since

$$\mathbb{E}[\Sigma^{-1}\nabla f_1(\theta^*)] = 0 \quad \text{and} \quad \mathbb{E}[\Sigma^{-1}\nabla f_2(\theta^*)] = 0,$$

Lemmas 9 and 10 coupled with the rewritten correction (3.45) yield

$$\begin{aligned}\mathbb{E}[\theta_1 - \theta^* - r(\theta_2 - \theta^*)] = &-r\mathbb{E}[\Sigma^{-1}(\nabla^2 f_2(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_2(\theta^*)] \\ &+ \mathbb{E}[\Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*)] \\ &+ r\mathbb{E}[\Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_2(\theta^*)) \otimes (\Sigma^{-1}\nabla f_2(\theta^*))\right)] \\ &- \mathbb{E}[\Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_1(\theta^*)) \otimes (\Sigma^{-1}\nabla f_1(\theta^*))\right)] \\ &+ \mathcal{O}(1)r^{-1/2}\left(M^2G^6/\lambda^6 + G^4L^2 d\log(d)/\lambda^4\right)n^{-3/2}.\end{aligned} \tag{3.46}$$

Here the remainder terms follow because of the $r^{-3/2}\mathcal{R}_3$ term on $\theta_2 - \theta^*$.

3.5.3.3 Proof of Lemma 7

To prove the claim in the lemma, it suffices to show that

$$r\mathbb{E}[\Sigma^{-1}(\nabla^2 f_2(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_2(\theta^*)] = \mathbb{E}[\Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*)] \tag{3.47}$$

and

$$\begin{aligned}&r\mathbb{E}[\Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_2(\theta^*)) \otimes (\Sigma^{-1}\nabla f_2(\theta^*))\right)] \\ &\quad = \mathbb{E}[\Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_1(\theta^*)) \otimes (\Sigma^{-1}\nabla f_1(\theta^*))\right)]\end{aligned} \tag{3.48}$$

Indeed, these two claims combined with the expansion (3.46) yield the bound (3.42) in Lemma 7 immediately.

We first consider the difference (3.47). To make things notationally simpler, we define functions $A : \mathcal{X} \to \mathbb{R}^{d \times d}$ and $B : \mathcal{X} \to \mathbb{R}^d$ via $A(x) := \Sigma^{-1}(\nabla^2 \phi(\theta^*; x) - \Sigma)$ and $B(x) := \Sigma^{-1} \nabla \phi(\theta^*; x)$. If we let $S_1 = \{X_1, \ldots, X_n\}$ be the original samples and $S_2 = \{Y_1, \ldots, Y_{rn}\}$ be the subsampled dataset, we must show

$$r\mathbb{E}\left[\frac{1}{(rn)^2}\sum_{i,j}^{rn} A(Y_i)B(Y_j)\right] = \mathbb{E}\left[\frac{1}{n^2}\sum_{i,j}^{n} A(X_i)B(X_j)\right].$$

Since the Y_i are sampled without replacement (i.e., from P directly), and $\mathbb{E}[A(X_i)] = 0$ and $\mathbb{E}[B(X_i)] = 0$, we find that $\mathbb{E}[A(Y_i)B(Y_j)] = 0$ for $i \neq j$, and thus

$$\sum_{i,j}^{rn} \mathbb{E}[A(Y_i)B(Y_j)] = \sum_{i=1}^{rn} \mathbb{E}[A(Y_i)B(Y_i)] = rn\mathbb{E}[A(Y_1)B(Y_1)].$$

In particular, we see that the equality (3.47) holds:

$$\begin{aligned}\frac{r}{(rn)^2}\sum_{i,j}^{rn} \mathbb{E}[A(Y_i)B(Y_j)] = \frac{r}{rn}\mathbb{E}[A(Y_1)B(Y_1)] &= \frac{1}{n}\mathbb{E}[A(X_1)B(X_1)] \\ &= \frac{1}{n^2}\sum_{i,j}^{n} \mathbb{E}[A(X_i)B(X_j)].\end{aligned}$$

The statement (3.48) follows from analogous arguments.

3.5.3.4 Proof of Lemma 8

The proof of Lemma 8 follows from that of Lemmas 9 and 10. We first claim that

$$\theta_1 - \theta^* = -\Sigma^{-1}\nabla f_1(\theta^*) + \mathcal{R}_2 \quad \text{and} \quad \theta_2 - \theta^* = -\Sigma^{-1}\nabla f_2(\theta^*) + r^{-1}\mathcal{R}_2. \tag{3.49}$$

The proofs of both claims similar, so we focus on proving the second statement. Using the inequality $(a+b+c)^2 \leq 3(a^2+b^2+c^2)$ and Lemma 10, we see that

$$\begin{aligned}\mathbb{E}\left[\left\|\theta_2 - \theta^* + \Sigma^{-1}\nabla f_2(\theta^*)\right\|_2^2\right] &\leq 3\mathbb{E}\left[\left\|\Sigma^{-1}(\nabla^2 f_2(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_2(\theta^*)\right\|_2^2\right] \\ &\quad + 3\mathbb{E}\left[\left\|\Sigma^{-1}\nabla^3 f_0(\theta^*)\left((\Sigma^{-1}\nabla f_2(\theta^*)) \otimes (\Sigma^{-1}\nabla f_2(\theta^*))\right)\right\|_2^2\right] \\ &\quad + 3r^{-3}\mathcal{O}(n^{-3}).\end{aligned} \tag{3.50}$$

We now bound the first two terms in inequality (3.50). Applying the Cauchy-Schwarz inequality and Lemma 4, the first term can be upper bounded as

$$\begin{aligned}
&\mathbb{E}\left[\left|\left|\Sigma^{-1}(\nabla^2 f_2(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_2(\theta^*)\right|\right|_2^2\right] \\
&\leq \left(\mathbb{E}\left[\left|\left|\left|\Sigma^{-1}(\nabla^2 f_2(\theta^*) - \Sigma)\right|\right|\right|_2^4\right]\mathbb{E}\left[\left|\left|\Sigma^{-1}\nabla f_2(\theta^*)\right|\right|_2^4\right]\right)^{1/2} \\
&= \left(r^{-2}\right)\mathcal{O}(\log^2(d)n^{-2}) \cdot r^{-2}\mathcal{O}(n^{-2})\big)^{1/2} \;=\; r^{-2}\mathcal{O}(n^{-2}),
\end{aligned}$$

where the order notation subsumes the logarithmic factor in the dimension. Since $\nabla^3 f_0(\theta^*) : \mathbb{R}^{d^2} \to \mathbb{R}^d$ is linear, the second term in the inequality (3.50) may be bounded completely analogously as it involves the outer product $\Sigma^{-1}\nabla f_2(\theta^*) \otimes \Sigma^{-1}\nabla f_2(\theta^*)$. Recalling the bound (3.50), we have thus shown that

$$\mathbb{E}\left[\left|\left|\theta_2 - \theta^* + \Sigma^{-1}\nabla f_2(\theta^*)\right|\right|_2^2\right] = r^{-2}\mathcal{O}(n^{-2}),$$

or $\theta_2 - \theta^* = -\Sigma^{-1}\nabla f_2(\theta^*) + r^{-1}\mathcal{R}_2$. The proof of the first equality in equation (3.49) is entirely analogous.

We now apply the equalities (3.49) to obtain the result of the lemma. We have

$$\mathbb{E}\left[\|\theta_1 - \theta^* - r(\theta_2 - \theta^*)\|_2^2\right] = \mathbb{E}\left[\left|\left|-\Sigma^{-1}\nabla f_1(\theta^*) + r\Sigma^{-1}\nabla f_2(\theta^*) + \mathcal{R}_2\right|\right|_2^2\right].$$

Using the inequality $(a+b)^2 \leq (1+\eta)a^2 + (1+1/\eta)b^2$ for any $\eta \geq 0$, we have

$$\begin{aligned}
(a+b+c)^2 &\leq (1+\eta)a^2 + (1+1/\eta)(b+c)^2 \\
&\leq (1+\eta)a^2 + (1+1/\eta)(1+\alpha)b^2 + (1+1/\eta)(1+1/\alpha)c^2
\end{aligned}$$

for any $\eta, \alpha \geq 0$. Taking $\eta = 1$ and $\alpha = 1/2$, we obtain $(a+b+c)^2 \leq 2a^2 + 3b^2 + 6c^2$, so applying the triangle inequality, we have

$$\begin{aligned}
\mathbb{E}\left[\|\theta_1 - \theta^* - r(\theta_2 - \theta^*)\|_2^2\right] &= \mathbb{E}\left[\left|\left|-\Sigma^{-1}\nabla f_1(\theta^*) + r\Sigma^{-1}\nabla f_2(\theta^*) + \mathcal{R}_2\right|\right|_2^2\right] \qquad (3.51) \\
&\leq 2\mathbb{E}\left[\left|\left|\Sigma^{-1}\nabla f_1(\theta^*)\right|\right|_2^2\right] + 3r^2\mathbb{E}\left[\left|\left|\Sigma^{-1}\nabla f_2(\theta^*)\right|\right|_2^2\right] + \mathcal{O}(n^{-2}).
\end{aligned}$$

Since f_2 is a sub-sampled version of f_1, algebraic manipulations yield

$$\mathbb{E}\left[\left|\left|\Sigma^{-1}\nabla f_2(\theta^*)\right|\right|_2^2\right] = \frac{n}{rn}\mathbb{E}\left[\left|\left|\Sigma^{-1}\nabla f_1(\theta^*)\right|\right|_2^2\right] = \frac{1}{r}\mathbb{E}\left[\left|\left|\Sigma^{-1}\nabla f_1(\theta^*)\right|\right|_2^2\right]. \qquad (3.52)$$

Combining equations (3.51) and (3.52), we obtain the desired bound (3.43).

3.5.4 Proof of Theorem 3

We begin by recalling that if θ^n denotes the output of performing stochastic gradient on one machine, then from the inequality (3.26) we have the upper bound

$$\mathbb{E}[\left\|\overline{\theta}^n - \theta^*\right\|_2^2] \leq \frac{1}{m}\mathbb{E}[\|\theta^n - \theta^*\|_2^2] + \|\mathbb{E}[\theta^n - \theta^*]\|_2^2 .$$

To prove the error bound (3.21), it thus suffices to prove the inequalities

$$\mathbb{E}[\|\theta^n - \theta^*\|_2^2] \leq \frac{\alpha G^2}{\lambda^2 n}, \quad \text{and} \tag{3.53}$$

$$\|\mathbb{E}[\theta^n - \theta^*]\|_2^2 \leq \frac{\beta^2}{n^{3/2}}. \tag{3.54}$$

Before proving the theorem, we introduce some notation and a few preliminary results. Let $g_t = \nabla\phi(\theta^t; X_t)$ be the gradient of the t^{th} sample in stochastic gradient descent, where we consider running SGD on a single machine. We also let

$$\Pi(v) := \operatorname*{argmin}_{\theta\in\Theta} \left\{\|\theta - v\|_2^2\right\}$$

denote the projection of the point v onto the domain Θ.

We now state a known result, which gives sharp rates on the convergence of the iterates $\{\theta^t\}$ in stochastic gradient descent.

Lemma 11 (Rakhlin et al., 2011)**.** *Assume that $\mathbb{E}[\|g_t\|_2^2] \leq G^2$ for all t. Choosing $\eta_t = \frac{c}{\lambda t}$ for some $c \geq 1$, for any $t \in \mathbb{N}$ we have*

$$\mathbb{E}\left[\left\|\theta^t - \theta^*\right\|_2^2\right] \leq \frac{\alpha G^2}{\lambda^2 t} \quad \textit{where} \quad \alpha = 4c^2.$$

With these ingredients, we can now turn to the proof of Theorem 3. Lemma 11 gives the inequality (3.53), so it remains to prove that $\overline{\theta}^n$ has the smaller bound (3.54) on its bias. To that end, recall the neighborhood $U_\rho \subset \Theta$ in Assumption E, and note that

$$\begin{aligned}\theta^{t+1} - \theta^* &= \Pi(\theta^t - \eta_t g_t - \theta^*) \\ &= \theta^t - \eta_t g_t - \theta^* + 1_{(\theta^{t+1}\notin U_\rho)}\left(\Pi(\theta^t - \eta_t g_t) - (\theta^t - \eta_t g_t)\right)\end{aligned}$$

since when $\theta \in U_\rho$, we have $\Pi(\theta) = \theta$. Consequently, an application of the triangle inequality gives

$$\left\|\mathbb{E}[\theta^{t+1} - \theta^*]\right\|_2 \leq \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \mathbb{E}[\left\|(\Pi(\theta^t - \eta_t g_t) - (\theta^t - \eta_t g_t))1(\theta^{t+1} \notin U_\rho)\right\|_2].$$

By the definition of the projection and the fact that $\theta^t \in \Theta$, we additionally have

$$\left\|\Pi(\theta^t - \eta_t g_t) - (\theta^t - \eta_t g_t)\right\|_2 \leq \left\|\theta^t - (\theta^t - \eta_t g_t))\right\|_2 \leq \eta_t \left\|g_t\right\|_2 .$$

Thus, by applying Hölder's inequality (with the conjugate choices $(p, q) = (4, \frac{4}{3})$) and Assumption E, we have

$$\begin{aligned}
\left\|\mathbb{E}[\theta^{t+1} - \theta^*]\right\|_2 &\leq \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \eta_t \mathbb{E}[\|g_t\|_2 \, 1_{(\theta^{t+1} \notin U_\rho)}] \\
&\leq \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \eta_t \sqrt[4]{\mathbb{E}[\|g_t\|_2^4]} \left(\mathbb{E}[1^{4/3}_{(\theta^t \notin U_\rho)}]\right)^{3/4} \\
&\leq \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \eta_t G \left(\mathbb{P}(\theta^t \notin U_\rho)\right)^{3/4} \\
&\leq \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \eta_t G \left(\frac{\mathbb{E}\left\|\theta^{t+1} - \theta^*\right\|_2^2}{\rho^2}\right)^{3/4}, \qquad (3.55)
\end{aligned}$$

the inequality (3.55) following from an application of Markov's inequality. By applying Lemma 11, we finally obtain

$$\begin{aligned}
\left\|\mathbb{E}[\theta^{t+1} - \theta^*]\right\|_2 &\leq \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \eta_t G \left(\frac{\alpha G^2}{\lambda^2 \rho^2 t}\right)^{3/4} \\
&= \left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 + \frac{c\alpha^{3/4} G^{5/2}}{\lambda^{5/2} \rho^{3/2}} \cdot \frac{1}{t^{7/4}}. \qquad (3.56)
\end{aligned}$$

Now we turn to controlling the rate at which $\theta^t - \eta_t g_t$ goes to zero. Let $\phi_t(\cdot) = \phi(\cdot; X_t)$ be shorthand for the loss evaluated on the t^{th} data point. By defining

$$r_t = g_t - \nabla \phi_t(\theta^*) - \nabla^2 \phi_t(\theta^*)(\theta^t - \theta^*),$$

a bit of algebra yields

$$g_t = \nabla \phi_t(\theta^*) + \nabla^2 \phi_t(\theta^*)(\theta^t - \theta^*) + r_t.$$

Since θ^t belongs to the σ-field of $X_1, \ldots, X_{t-1}$, the Hessian $\nabla^2 \phi_t(\theta^*)$ is (conditionally) independent of θ^t and

$$\mathbb{E}[g_t] = \nabla^2 f_0(\theta^*) \mathbb{E}[\theta^t - \theta^*] + \mathbb{E}[r_t 1_{(\theta^t \in U_\rho)}] + \mathbb{E}[r_t 1_{(\theta^t \notin U_\rho)}]. \qquad (3.57)$$

If $\theta^t \in U_\rho$, then Taylor's theorem implies that r_t is the Lagrange remainder

$$r_t = (\nabla^2 \phi_t(\theta') - \nabla^2 \phi_t(\theta^*))(\theta' - \theta^*),$$

where $\theta' = \kappa \theta^t + (1 - \kappa)\theta^*$ for some $\kappa \in [0, 1]$. Applying Assumption E and Hölder's inequality, we find that since θ^t is conditionally independent of X_t,

$$\begin{aligned}
\mathbb{E}\left[\left\|r_t 1_{(\theta^t \in U_\rho)}\right\|_2\right] &\leq \mathbb{E}\left[\left\|\left\|\nabla^2 \phi(\theta'; X_t) - \nabla^2 \phi(\theta^*; X_t)\right\|\right\| \left\|\theta^t - \theta^*\right\|_2 1_{(\theta^t \in U_\rho)}\right] \\
&\leq \mathbb{E}\left[L(X_t) \left\|\theta^t - \theta^*\right\|_2^2\right] = \mathbb{E}[L(X_t)] \mathbb{E}[\left\|\theta^t - \theta^*\right\|_2^2] \\
&\leq L\mathbb{E}\left[\left\|\theta^t - \theta^*\right\|_2^2\right] \leq \frac{\alpha L G^2}{\lambda^2 t}.
\end{aligned}$$

On the other hand, when $\theta^t \notin U_\rho$, we have the following sequence of inequalities:

$$\begin{aligned}
\mathbb{E}\left[\left\|r_t 1_{(\theta^t \notin U_\rho)}\right\|_2\right] &\overset{(i)}{\leq} \sqrt[4]{\mathbb{E}[\|r_t\|_2^4]}\left(\mathbb{P}(\theta^t \notin U_\rho)\right)^{3/4} \\
&\overset{(ii)}{\leq} \sqrt[4]{3^3\left(\mathbb{E}[\|g_t\|_2^4] + \mathbb{E}[\|\nabla\phi_t(\theta^*)\|_2^4] + \mathbb{E}[\|\nabla^2\phi_t(\theta^*)(\theta^t-\theta^*)\|_2^4]\right)}\left(\mathbb{P}(\theta^t \notin U_\rho)\right)^{3/4} \\
&\leq 3^{3/4}\sqrt[4]{G^4+G^4+H^4R^4}\left(\mathbb{P}(\theta^t \notin U_\rho)\right)^{3/4} \\
&\overset{(iii)}{\leq} 3(G+HR)\left(\frac{\alpha G^2}{\lambda^2\rho^2 t}\right)^{3/4}.
\end{aligned}$$

Here step (i) follows from Hölder's inequality (again applied with the conjugates $(p,q) = (4, \frac{4}{3})$); step (ii) follows from Jensen's inequality, since $(a+b+c)^4 \leq 3^3(a^4+b^4+c^4)$; and step (iii) follows from Markov's inequality, as in the bounds (3.55) and (3.56). Combining our two bounds on r_t, we find that

$$\mathbb{E}[\|r_t\|_2] \leq \frac{\alpha L G^2}{\lambda^2 t} + \frac{3\alpha^{3/4}G^{3/2}(G+HR)}{\lambda^{3/2}\rho^{3/2}} \cdot \frac{1}{t^{3/4}}. \tag{3.58}$$

By combining the expansion (3.57) with the bound (3.58), we find that

$$\begin{aligned}
\left\|\mathbb{E}[\theta^t - \eta_t g_t - \theta^*]\right\|_2 &= \left\|\mathbb{E}[(I-\eta_t\nabla^2 f_0(\theta^*))(\theta^t-\theta^*) + \eta_t r_t]\right\|_2 \\
&\leq \left\|\mathbb{E}[(I-\eta_t\nabla^2 f_0(\theta^*))(\theta^t-\theta^*)]\right\|_2 + \frac{c\alpha L G^2}{\lambda^3 t^2} + \frac{3c\alpha^{3/4}G^{3/2}(G+HR)}{\lambda^{5/2}\rho^{3/2}} \cdot \frac{1}{t^{7/4}}.
\end{aligned}$$

Using the earlier bound (3.56), this inequality then yields

$$\left\|\mathbb{E}[\theta^{t+1}-\theta^*]\right\|_2 \leq \left|\!\left|\!\left|I-\eta_t\nabla^t f_0(\theta^*)\right|\!\right|\!\right|_2 \left\|\mathbb{E}[\theta^t-\theta^*]\right\|_2 + \frac{c\alpha^{3/4}G^{3/2}}{\lambda^{5/2}t^{7/4}}\left(\frac{\alpha^{1/4}LG^{1/2}}{\lambda^{1/2}t^{1/4}} + \frac{4G+HR}{\rho^{3/2}}\right).$$

We now complete the proof via an inductive argument using our immediately preceding bounds. Our reasoning follows a similar induction given by Rakhlin et al. [167]. First, note that by strong convexity and our condition that $|||\nabla^2 f_0(\theta^*)||| \leq H$, we have

$$\left|\!\left|\!\left|I-\eta_t\nabla^2 f_0(\theta^*)\right|\!\right|\!\right| = 1 - \eta_t\lambda_{\min}(\nabla^2 f_0(\theta^*) \leq 1 - \eta_t\lambda$$

whenever $1 - \eta_t H \geq 0$. Define $\tau_0 = \lceil cH/\lambda \rceil$; then for $t \geq t_0$ we obtain

$$\left\|\mathbb{E}[\theta^{t+1}-\theta^*]\right\|_2 \leq (1-c/t)\left\|\mathbb{E}[\theta^t-\theta^*]\right\|_2 + \frac{1}{t^{7/4}} \cdot \frac{c\alpha^{3/4}G^{3/2}}{\lambda^{5/2}}\left(\frac{\alpha^{1/4}LG^{1/2}}{\lambda^{1/2}t^{1/4}} + \frac{4G+HR}{\rho^{3/2}}\right). \tag{3.59}$$

For shorthand, we define two intermediate variables

$$a_t = \left\|\mathbb{E}(\theta^t-\theta^*)\right\|_2 \quad \text{and} \quad b = \frac{c\alpha^{3/4}G^{3/2}}{\lambda^{5/2}}\left(\frac{\alpha^{1/4}LG^{1/2}}{\lambda^{1/2}} + \frac{4G+HR}{\rho^{3/2}}\right).$$

Inequality (3.59) then implies the inductive relation $a_{t+1} \leq (1 - c/t)a_t + b/t^{7/4}$. Now we show that by defining $\beta = \max\{\tau_0 R, b/(c-1)\}$, we have $a_t \leq \beta/t^{3/4}$. Indeed, it is clear that $a_1 \leq \tau_0 R$. Using the inductive hypothesis, we then have

$$a_{t+1} \leq \frac{(1-c/t)\beta}{t^{3/4}} + \frac{b}{t^{7/4}} = \frac{\beta(t-1)}{t^{7/4}} - \frac{\beta(c-1)-b}{t^2} \leq \frac{\beta(t-1)}{t^{7/4}} \leq \frac{\beta}{(t+1)^{3/4}}.$$

This completes the proof of the inequality (3.54). □

Remark If we assume kth moment bounds instead of 4th, i.e., $\mathbb{E}[|||\nabla^2\phi(\theta^*; X)|||_2^k] \leq H^k$ and $\mathbb{E}[\|g_t\|_2^k] \leq G^k$, we find the following analogue of the bound (3.59):

$$\left\|\mathbb{E}[\theta^{t+1} - \theta^*]\right\|_2 \leq (1 - c/t)\left\|\mathbb{E}[\theta^t - \theta^*]\right\|_2 + \frac{1}{t^{\frac{2k-1}{k}}} \cdot \frac{c\alpha^{\frac{k-1}{k}} G^{\frac{2k-2}{k}}}{\lambda^{\frac{3k-2}{k}}} \left[\frac{\left(54^{1/k}+1\right)G + 54^{1/k}HR}{\rho^{\frac{2k-2}{k}}} + \frac{\alpha^{1/k} L G^{2/k}}{\lambda^{2/k} t^{1/k}}\right].$$

In this case, if we define

$$b = \frac{c\alpha^{\frac{k-1}{k}} G^{\frac{2k-2}{k}}}{\lambda^{\frac{3k-2}{k}}} \left[\frac{\left(54^{1/k}+1\right)G + 54^{1/k}HR}{\rho^{\frac{2k-2}{k}}} + \frac{\alpha^{1/k} L G^{2/k}}{\lambda^{2/k}}\right] \quad \text{and} \quad \beta = \max\left\{\tau_0 R, \frac{b}{c-1}\right\},$$

we have the same result except we obtain the bound $\|\mathbb{E}[\theta^n - \theta^*]\|_2^2 \leq \beta^2/n^{\frac{2k-2}{k}}$.

3.5.5 Proof of Lemma 3

We first prove that under the conditions given in the lemma statement, the function f_1 is $(1-\rho)\lambda$-strongly convex over the ball $U := \left\{\theta \in \mathbb{R}^d : \|\theta - \theta^*\|_2 < \delta_\rho\right\}$ around θ^*. Indeed, fix $\gamma \in U$, then use the triangle inequality to conclude that

$$\begin{aligned}\left|\left|\left|\nabla^2 f_1(\gamma) - \nabla^2 f_0(\theta^*)\right|\right|\right|_2 &\leq \left|\left|\left|\nabla^2 f_1(\gamma) - \nabla^2 f_1(\theta^*)\right|\right|\right|_2 + \left|\left|\left|\nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*)\right|\right|\right|_2 \\ &\leq L\|\gamma - \theta^*\|_2 + \frac{\rho\lambda}{2}.\end{aligned}$$

Here we used Assumption C on the first term and the fact that the event $\mathcal{E}_1$ holds on the second. By our choice of $\delta_\rho \leq \rho\lambda/4L$, this final term is bounded by $\lambda\rho$. In particular, we have

$$\nabla^2 f_0(\theta^*) \succeq \lambda I \quad \text{so} \quad \nabla^2 f_1(\gamma) \succeq \lambda I - \rho\lambda I = (1-\rho)\lambda I,$$

which proves that f_1 is $(1-\rho)\lambda$-strongly convex on the ball U.

In order to prove the conclusion of the lemma, we argue that since f_1 is (locally) strongly convex, if the function f_1 has small gradient at the point θ^*, it must be the case that the minimizer θ_1 of f_1 is near θ^*. Then we can employ reasoning similar to standard analyses of

optimality for globally strongly convex functions [e.g. 33, Chapter 9]. By definition of (the local) strong convexity on the set U, for any $\theta' \in \Theta$, we have

$$f_1(\theta') \geq f_1(\theta^*) + \langle \nabla f_1(\theta^*), \theta' - \theta^* \rangle + \frac{(1-\rho)\lambda}{2} \min\left\{ \|\theta^* - \theta'\|_2^2, \delta_\rho^2 \right\}.$$

Rewriting this inequality, we find that

$$\begin{aligned}\min\left\{ \|\theta^* - \theta'\|_2^2, \delta_\rho^2 \right\} &\leq \frac{2}{(1-\rho)\lambda} \left[f_1(\theta') - f_1(\theta^*) + \langle \nabla f_1(\theta^*), \theta' - \theta^* \rangle \right] \\ &\leq \frac{2}{(1-\rho)\lambda} \left[f_1(\theta') - f_1(\theta^*) + \|\nabla f_1(\theta^*)\|_2 \|\theta' - \theta^*\|_2 \right].\end{aligned}$$

Dividing each side by $\|\theta' - \theta^*\|_2$, then noting that we may set $\theta' = \kappa\theta_1 + (1-\kappa)\theta^*$ for any $\kappa \in [0,1]$, we have

$$\min\left\{ \kappa \|\theta_1 - \theta^*\|_2, \frac{\delta_\rho^2}{\kappa \|\theta_1 - \theta^*\|_2} \right\} \leq \frac{2\left[f_1(\kappa\theta_1 + (1-\kappa)\theta^*) - f_1(\theta^*) \right]}{\kappa(1-\rho)\lambda \|\theta_1 - \theta^*\|_2} + \frac{2\|\nabla f_1(\theta^*)\|_2}{(1-\rho)\lambda}.$$

Of course, $f_1(\theta_1) < f_1(\theta^*)$ by assumption, so we find that for any $\kappa \in (0,1)$ we have the strict inequality

$$\min\left\{ \kappa \|\theta_1 - \theta^*\|_2, \frac{\delta_\rho^2}{\kappa \|\theta_1 - \theta^*\|_2} \right\} < \frac{2\|\nabla f_1(\theta^*)\|_2}{(1-\rho)\lambda} \leq \delta_\rho,$$

the last inequality following from the definition of $\mathcal{E}_2$. Since this holds for any $\kappa \in (0,1)$, if $\|\theta_1 - \theta^*\|_2 > \delta_\rho$, we may set $\kappa = \delta_\rho / \|\theta_1 - \theta^*\|_2$, which would yield a contradiction. Thus, we have $\|\theta_1 - \theta^*\|_2 \leq \delta_\rho$, and by our earlier inequalities,

$$\|\theta_1 - \theta^*\|_2^2 \leq \frac{2}{(1-\rho)\lambda} \left[f_1(\theta_1) - f_1(\theta^*) + \|\nabla f_1(\theta^*)\|_2 \|\theta_1 - \theta^*\|_2 \right] \leq \frac{2\|\nabla f_1(\theta^*)\|_2}{(1-\rho)\lambda} \|\theta_1 - \theta^*\|_2.$$

Dividing by $\|\theta_1 - \theta^*\|_2$ completes the proof. □

3.5.6 Moment bounds

In this appendix, we state two useful moment bounds, showing how they combine to provide a proof of Lemma 4. The two lemmas are a vector and a non-commutative matrix variant of the classical Rosenthal inequalities. We begin with the case of independent random vectors:

Lemma 12 (de Acosta, 1981, Theorem 2.1)**.** *Let $k \geq 2$ and X_i be a sequence of independent random vectors in a separable Banach space with norm $\|\cdot\|$ and $\mathbb{E}[\|X_i\|^k] < \infty$. There exists a finite constant C_k such that*

$$\mathbb{E}\left[\left| \left\| \sum_{i=1}^n X_i \right\| - \mathbb{E}\left[\left\| \sum_{i=1}^n X_i \right\| \right] \right|^k \right] \leq C_k \left[\left(\sum_{i=1}^n \mathbb{E}[\|X_i\|^2] \right)^{k/2} + \sum_{i=1}^n \mathbb{E}[\|X_i\|^k] \right].$$

We say that a random matrix X is symmetrically distributed if X and $-X$ have the same distribution. For such matrices, we have:

Lemma 13 (Chen et al., 2012, Theorem A.1(2)). *Let $X_i \in \mathbb{R}^{d\times d}$ be independent and symmetrically distributed Hermitian matrices. Then*

$$\mathbb{E}\left[\left|\left|\left|\sum_{i=1}^n X_i\right|\right|\right|^k\right]^{1/k} \leq \sqrt{2e\log d}\left|\left|\left|\left(\sum_{i=1}^n \mathbb{E}\left[X_i^2\right]\right)^{1/2}\right|\right|\right| + 2e\log d\left(\mathbb{E}[\max_i |||X_i|||^k]\right)^{1/k}.$$

Equipped with these two auxiliary results, we turn to our proof Lemma 4. To prove the first bound (3.31), let $2 \leq k \leq k_0$ and note that by Jensen's inequality, we have

$$\mathbb{E}[\|\nabla f_1(\theta^*)\|_2^k] \leq 2^{k-1}\mathbb{E}\left[\left|\,\|\nabla f_1(\theta^*)\|_2 - \mathbb{E}[\|\nabla f_1(\theta^*)\|_2]\right|^k\right] + 2^{k-1}\mathbb{E}\left[\|\nabla f_1(\theta^*)\|_2\right]^k.$$

Again applying Jensen's inequality, $\mathbb{E}[\|\nabla\phi(\theta^*;X)\|_2^2] \leq G^2$. Thus by recalling the definition $\nabla f_1(\theta^*) = \frac{1}{n}\sum_{i=1}^n \nabla\phi(\theta^*;X_i)$ and applying the inequality

$$\mathbb{E}[\|\nabla f_1(\theta^*)\|_2] \leq \mathbb{E}[\|\nabla f_1(\theta^*)\|_2^2]^{1/2} \leq n^{-1/2}G,$$

we see that Lemma 12 implies $\mathbb{E}\left[\|\nabla f_1(\theta^*)\|_2^k\right]$ is upper bounded by

$$2^{k-1}C_k\left[\left(\frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[\|\nabla\phi(\theta;X_i)\|_2^2]\right)^{k/2} + \frac{1}{n^k}\sum_{i=1}^n \mathbb{E}[\|\nabla\phi(\theta^*;X_i)\|_2^k]\right] + 2^{k-1}\mathbb{E}[\|\nabla f_1(\theta^*)\|_2]^k$$
$$\leq 2^{k-1}\frac{C_k}{n^{k/2}}\left[\left(\frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|\nabla\phi(\theta^*;X_i)\|_2^2]\right)^{k/2} + \frac{1}{n^{k/2}}\sum_{i=1}^n \mathbb{E}[\|\nabla\phi(\theta^*;X_i)\|_2^k]\right] + \frac{2^{k-1}G^k}{n^{k/2}}.$$

Applying Jensen's inequality yields

$$\left(\frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|\nabla\phi(\theta^*;X_i)\|_2^2]\right)^{k/2} \leq \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|\nabla\phi(\theta^*;X_i)\|_2^2]^{k/2} \leq G^k,$$

completes the proof of the inequality (3.31).

The proof of the bound (3.32) requires a very slightly more delicate argument involving symmetrization step. Define matrices $Z_i = \frac{1}{n}\left(\nabla^2\phi(\theta^*;X_i) - \nabla^2 f_0(\theta^*)\right)$. If $\varepsilon_i \in \{\pm 1\}$ are i.i.d. Rademacher variables independent of Z_i, then for any integer k in the interval $[2,k_2]$, a standard symmetrization argument [e.g. 117, Lemma 6.3] implies that

$$\mathbb{E}\left[\left|\left|\left|\sum_{i=1}^n Z_i\right|\right|\right|^k\right]^{1/k} \leq 2\mathbb{E}\left[\left|\left|\left|\sum_{i=1}^n \varepsilon_i Z_i\right|\right|\right|^k\right]^{1/k}. \tag{3.60}$$

Now we may apply Lemma 13, since the matrices $\varepsilon_i Z_i$ are Hermitian and symmetrically distributed; by expanding the definition of the Z_i, we find that

$$\mathbb{E}\left[\left|\left|\left|\nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*)\right|\right|\right|^k\right]^{1/k} \leq 5\sqrt{\log d}\left|\left|\left|\left(\frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[(\nabla^2\phi(\theta; X_i) - \nabla^2 f_0(\theta^*))^2]\right)^{1/2}\right|\right|\right| + 4e\log d\left(n^{-k}\mathbb{E}[\max_i \left|\left|\left|\nabla^2\phi(\theta^*; X_i) - \nabla^2 f_0(\theta^*)\right|\right|\right|^k]\right)^{1/k}.$$

Since the X_i are i.i.d., we have

$$\left|\left|\left|\left(\frac{1}{n^2}\sum_{i=1}^n \mathbb{E}[(\nabla^2\phi(\theta; X_i) - \nabla^2 f_0(\theta^*))^2]\right)^{1/2}\right|\right|\right| = \left|\left|\left|n^{-1/2}\mathbb{E}\left[\left(\nabla^2\phi(\theta^*; X) - \nabla^2 f_0(\theta^*)\right)^2\right]^{1/2}\right|\right|\right| \leq n^{-1/2}\mathbb{E}\left[\left|\left|\left|\nabla^2\phi(\theta^*; X) - \nabla^2 f_0(\theta^*)\right|\right|\right|^2\right]^{1/2}$$

by Jensen's inequality, since $\left|\left|\left|A^{1/2}\right|\right|\right| = |||A|||^{1/2}$ for semidefinite A. Finally, noting that

$$\frac{1}{n^k}\mathbb{E}\left[\max_i \left|\left|\left|\nabla^2\phi(\theta^*; X_i) - \nabla^2 f_0(\theta^*)\right|\right|\right|^k\right] \leq \frac{n}{n^k}\mathbb{E}\left[\left|\left|\left|\nabla^2\phi(\theta^*; X) - \nabla^2 f_0(\theta^*)\right|\right|\right|^k\right] \leq n^{1-k}H^k$$

completes the proof of the second bound (3.32).

3.5.7 Proof of Lemma 9

The proof follows from a slightly more careful application of the Taylor expansion (3.28). The starting point in our proof is to recall the success events (3.27) and the joint event $\mathcal{E} := \mathcal{E}_0 \cap \mathcal{E}_1 \cap \mathcal{E}_2$. We begin by arguing that we may focus on the case where $\mathcal{E}$ holds. Let C denote the right hand side of the equality (3.44) except for the remainder $\mathcal{R}_3$ term. By Assumption C, we follow the bound (3.33) (with $\min\{k_0, k_1, k_2\} \geq 8$) to find that

$$\mathbb{E}\left[1_{(\mathcal{E}^c)}\|\theta_1 - \theta^*\|_2^2\right] = \mathcal{O}\left(R^2 n^{-4}\right),$$

so we can focus on the case where the joint event $\mathcal{E} = \mathcal{E}_0 \cap \mathcal{E}_1 \cap \mathcal{E}_2$ does occur.

Defining $\Delta = \theta_1 - \theta^*$ for notational convenience, on $\mathcal{E}$ we have that for some $\kappa \in [0, 1]$, with $\theta' = (1-\kappa)\theta_1 + \kappa\theta^*$,

$$\begin{aligned} 0 &= \nabla f_1(\theta^*) + \nabla^2 f_1(\theta^*)\Delta + \nabla^3 f_1(\theta')(\Delta \otimes \Delta) \\ &= \nabla f_1(\theta^*) + \nabla^2 f_0(\theta^*)\Delta + \nabla^3 f_0(\theta^*)(\Delta \otimes \Delta) \\ &\quad + (\nabla^2 f_1(\theta^*) - \nabla^2 f_0(\theta^*))\Delta + (\nabla^3 f_1(\theta') - \nabla^3 f_0(\theta^*))(\Delta \otimes \Delta). \end{aligned}$$

Now, we recall the definition $\Sigma = \nabla^2 f_0(\theta^*)$, the Hessian of the risk at the optimal point, and solve for the error Δ to see that

$$\begin{aligned} \Delta = &-\Sigma^{-1}\nabla f_1(\theta^*) - \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Delta - \Sigma^{-1}\nabla^3 f_1(\theta^*)(\Delta \otimes \Delta) \\ &+ \Sigma^{-1}(\nabla^3 f_0(\theta^*) - \nabla^3 f_1(\theta'))(\Delta \otimes \Delta) \end{aligned} \tag{3.61}$$

on the event $\mathcal{E}$. As we did in the proof of Theorem 1, specifically in deriving the recursive equality (3.35), we may apply the expansion (3.30) of $\Delta = \theta_1 - \theta^*$ to obtain a clean asymptotic expansion of Δ using (3.61). Recall the definition $P = \nabla^2 f_0(\theta^*) - \nabla^2 f_1(\theta^*)$ for shorthand here (as in the expansion (3.30), though we no longer require Q).

First, we claim that

$$1_{(\mathcal{E})}(\nabla^3 f_0(\theta^*) - \nabla^3 f_1(\theta'))(\Delta \otimes \Delta) = \left(M^2 G^6/\lambda^6 + G^4 L^2 d \log(d)/\lambda^4\right) \mathcal{R}_3. \tag{3.62}$$

To prove the above expression, we add and subtract $\nabla^3 f_1(\theta^*)$ (and drop $1_{(\mathcal{E})}$ for simplicity). We must control

$$(\nabla^3 f_0(\theta^*) - \nabla^3 f_1(\theta^*))(\Delta \otimes \Delta) + (\nabla^3 f_1(\theta^*) - \nabla^3 f_1(\theta'))(\Delta \otimes \Delta).$$

To begin, recall that $\|u \otimes v\|_2 = \left\|uv^\top\right\|_2 = \|u\|_2 \|v\|_2$. By Assumption D, on the event $\mathcal{E}$ we have that $\nabla^3 f_1$ is $(1/n)\sum_{i=1}^n M(X_i)$-Lipschitz, so defining $M_n = (1/n)\sum_{i=1}^n M(X_i)$, we have

$$\begin{aligned}
\mathbb{E}\left[1_{(\mathcal{E})} \left\|\left(\nabla^3 f_1(\theta^*) - \nabla^3 f_1(\theta')\right)(\Delta \otimes \Delta)\right\|_2^2\right] &\leq \mathbb{E}\left[M_n^2 \|\theta^* - \theta'\|_2^2 \|\Delta\|_2^4\right] \\
&\leq \mathbb{E}\left[M_n^8\right]^{1/4} \mathbb{E}\left[\|\theta_1 - \theta^*\|_2^8\right]^{3/4} \leq \mathcal{O}(1) M^2 \frac{G^6}{\lambda^6 n^3}
\end{aligned}$$

by Hölder's inequality and Lemma 5. The remaining term we must control is the derivative difference $\mathbb{E}[\|(\nabla^3 f_1(\theta^*) - \nabla^3 f_0(\theta^*))(\Delta \otimes \Delta)\|_2^2]$. Define the random vector-valued function $\mathsf{G} = \nabla(f_1 - f_0)$, and let G_j denote its jth coordinate. Then by definition we have

$$(\nabla^3 f_1(\theta^*) - \nabla^3 f_0(\theta^*))(\Delta \otimes \Delta) = \left[\Delta^\top(\nabla^2 \mathsf{G}_1(\theta^*))\Delta \quad \cdots \quad \Delta^\top(\nabla^2 \mathsf{G}_d(\theta^*))\Delta\right]^\top \in \mathbb{R}^d.$$

Therefore, by the Cauchy-Schwarz inequality and the fact that $x^\top A x \leq \|A\|_2 \|x\|_2^2$,

$$\begin{aligned}
\mathbb{E}\left[\left\|(\nabla^3 f_1(\theta^*) - \nabla^3 f_0(\theta^*))(\Delta \otimes \Delta)\right\|_2^2\right] &= \sum_{j=1}^d \mathbb{E}\left[\left(\Delta^\top(\nabla^2 \mathsf{G}_j(\theta^*))\Delta\right)^2\right] \\
&\leq \sum_{j=1}^d \left(\mathbb{E}\left[\|\Delta\|_2^8\right] \mathbb{E}\left[\left\|\nabla^2 \mathsf{G}_j(\theta^*)\right\|_2^4\right]\right)^{1/2}.
\end{aligned}$$

Applying Lemma 5 yields that $\mathbb{E}[\|\Delta\|_2^8] = \mathcal{O}(G^8/(\lambda^2 n)^4)$. Introducing the shorthand notation $\mathsf{g}(\cdot; x) := \nabla\phi(\cdot; x) - \nabla f_0(\cdot)$, we can write

$$\nabla^2 \mathsf{G}_j(\theta^*) = \frac{1}{n}\sum_{i=1}^n \nabla^2 \mathsf{g}_j(\theta^*; X_i)$$

For every coordinate j, the random matrices $\nabla^2 \mathsf{g}_j(\theta^*; X_i)$ $(i = 1, \ldots, n)$ are i.i.d. and mean zero. By Assumption C, we have $\left\|\nabla^2 \mathsf{g}_j(\theta^*; X_i)\right\|_2 \leq 2L(X_i)$, whence we have

$$\mathbb{E}[\left\|\nabla^2 \mathsf{g}_j(\theta^*; X_i)\right\|_2^8] \leq 2^8 L^8.$$

Applying Lemma 13, we obtain

$$\mathbb{E}\left[\left|\left|\left|\nabla^2 \mathsf{G}_j(\theta^*)\right|\right|\right|_2^4\right] \leq \mathcal{O}(1) L^4 n^{-2} \log^2(d),$$

and hence

$$\mathbb{E}\left[\left\|(\nabla^3 f_1(\theta^*) - \nabla^3 f_0(\theta^*))(\Delta \otimes \Delta)\right\|_2^2\right] \leq \mathcal{O}(1) \frac{G^4 L^2}{\lambda^4} d \log(d) n^{-3},$$

which implies the desired result (3.62). From now on, terms of the form $\mathcal{R}_3$ will have no larger constants than those in the equality (3.62), so we ignore them.

Now we claim that

$$1_{(\mathcal{E})} \nabla^3 f_1(\theta^*)(\Delta \otimes \Delta) = \nabla^3 f_1(\theta^*)((\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} \nabla f_1(\theta^*))) + \mathcal{R}_3. \tag{3.63}$$

Indeed, applying the expansion (3.30) to the difference $\Delta = \theta_1 - \theta^*$, we have on $\mathcal{E}$ that

$$\begin{aligned}\Delta \otimes \Delta = (\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} \nabla f_1(\theta^*)) + (\Sigma^{-1} P \Delta) \otimes (\Sigma^{-1} P \Delta) \\ - (\Sigma^{-1} P \Delta) \otimes (\Sigma^{-1} \nabla f_1(\theta^*)) - (\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} P \Delta).\end{aligned}$$

We can bound each of the second three outer products in the equality above similarly; we focus on the last for simplicity. Applying the Cauchy-Schwarz inequality, we have

$$\mathbb{E}\left[\left|\left|\left|(\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} P \Delta)\right|\right|\right|_2^2\right] \leq \left(\mathbb{E}\left[\left\|\Sigma^{-1} \nabla f_1(\theta^*)\right\|_2^4\right] \mathbb{E}\left[\left\|\Sigma^{-1} P(\theta_1 - \theta^*)\right\|_2^4\right]\right)^{\frac{1}{2}}.$$

From Lemmas 5 and 6, we obtain that

$$\mathbb{E}\left[\left\|\Sigma^{-1} \nabla f_1(\theta^*)\right\|_2^4\right] = \mathcal{O}(n^{-2}) \quad \text{and} \quad \mathbb{E}\left[\left\|\Sigma^{-1} P(\theta_1 - \theta^*)\right\|_2^4\right] = \mathcal{O}(n^{-4})$$

after an additional application of Cauchy-Schwarz for the second expectation. This shows that

$$(\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} P \Delta) = \mathcal{R}_3,$$

and a similar proof applies to the other three terms in the outer product $\Delta \otimes \Delta$. Using the linearity of $\nabla^3 f_1(\theta^*)$, we see that to prove the equality (3.63), all that is required is that

$$1_{(\mathcal{E}^c)} \nabla^3 f_1(\theta^*) \left((\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} \nabla f_1(\theta^*))\right) = \mathcal{R}_3. \tag{3.64}$$

For this, we apply Hölder's inequality several times. Indeed, we have

$$\begin{aligned}&\mathbb{E}\left[\left\|1_{(\mathcal{E}^c)} \nabla^3 f_1(\theta^*) \left((\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} \nabla f_1(\theta^*))\right)\right\|_2^2\right] \\ &\leq \mathbb{E}[1_{(\mathcal{E}^c)}]^{1/4} \mathbb{E}\left[\left\|\nabla^3 f_1(\theta^*) \left((\Sigma^{-1} \nabla f_1(\theta^*)) \otimes (\Sigma^{-1} \nabla f_1(\theta^*))\right)\right\|_2^{8/3}\right]^{3/4} \\ &\leq \mathbb{E}[1_{(\mathcal{E}^c)}]^{1/4} \mathbb{E}\left[\left|\left|\left|\nabla^3 f_1(\theta^*)\right|\right|\right|^{8/3} \left\|\Sigma^{-1} \nabla f_1(\theta^*)\right\|_2^{16/3}\right]^{3/4} \\ &\leq \mathbb{E}[1_{(\mathcal{E}^c)}]^{1/4} \mathbb{E}\left[\left|\left|\left|\nabla^3 f_1(\theta^*)\right|\right|\right|^8\right]^{1/4} \mathbb{E}\left[\left\|\Sigma^{-1} \nabla f_1(\theta^*)\right\|_2^8\right]^{2/4} = \mathcal{O}(n^{-1} \cdot L^2 \cdot n^{-2}).\end{aligned}$$

For the final asymptotic bound, we used equation (3.33) to bound $\mathbb{E}[1_{(\mathcal{E}^c)}]$, used the fact (from Assumption C) that $\mathbb{E}[L(X)^8] \leq L^8$ to bound the term involving $\nabla^3 f_1(\theta^*)$, and applied Lemma 4 to control $\mathbb{E}[\|\Sigma^{-1}\nabla f_1(\theta^*)\|_2^8]$. Thus the equality (3.64) holds, and this completes the proof of the equality (3.63).

For the final step in the lemma, we claim that

$$-1_{(\mathcal{E})}\Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Delta = \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*) + \mathcal{R}_3. \tag{3.65}$$

To prove (3.65) requires an argument completely parallel to that for our claim (3.63). As before, we use the expansion (3.30) of the difference Δ to obtain that on $\mathcal{E}$,

$$\begin{aligned}
&-\Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Delta \\
&= \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*) - \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}P\Delta.
\end{aligned}$$

Now apply Lemmas 5 and 6 to the final term after a few applications of Hölder's inequality. To finish the equality (3.65), we argue that $1_{(\mathcal{E}^c)}\Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*) = \mathcal{R}_3$, which follows exactly the line of reasoning used to prove the remainder (3.64).

Applying equalities (3.62), (3.63), and (3.65) to our earlier expansion (3.61) yields that

$$\begin{aligned}
\Delta &= 1_{(\mathcal{E})}\big[- \Sigma^{-1}\nabla f_1(\theta^*) - \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Delta - \Sigma^{-1}\nabla^3 f_1(\theta^*)(\Delta \otimes \Delta) \\
&\qquad + \Sigma^{-1}(\nabla^3 f_0(\theta^*) - \nabla^3 f_1(\theta'))(\Delta \otimes \Delta)\big] + 1_{(\mathcal{E}^c)}\Delta \\
&= -\Sigma^{-1}\nabla f_1(\theta^*) + \Sigma^{-1}(\nabla^2 f_1(\theta^*) - \Sigma)\Sigma^{-1}\nabla f_1(\theta^*) \\
&\qquad - \Sigma^{-1}\nabla^3 f_1(\theta^*)\left((\Sigma^{-1}\nabla f_1(\theta^*)) \otimes (\Sigma^{-1}\nabla f_1(\theta^*))\right) + \mathcal{R}_3 + 1_{(\mathcal{E}^c)}\Delta.
\end{aligned}$$

Finally, the bound (3.33) implies that $\mathbb{E}[1_{(\mathcal{E}^c)}\|\Delta\|_2^2] \leq \mathbb{P}(\mathcal{E}^c)R^2 = \mathcal{O}(n^{-4})$, which yields the claim.

Chapter 4

Divide-and-conquer methods for kernel ridge regression

This chapter devotes to generalizing the divide-and-conquer method to non-parametric regression. In non-parametric regression, the statistician receives N samples of the form $\{(x_i, y_i)\}_{i=1}^N$, where each $x_i \in \mathcal{X}$ is a covariate and $y_i \in \mathbb{R}$ is a real-valued response, and the samples are drawn i.i.d. from some unknown joint distribution $\mathbb{P}$ over $\mathcal{X} \times \mathbb{R}$. The goal is to estimate a function $\widehat{f} : \mathcal{X} \to \mathbb{R}$ that can be used to predict future responses based on observing only the covariates. Kernel ridge regression is a classical non-parametric regression method and is widely used in practice [see e.g. 87, 192]. Past work has established bounds on the estimation error for RKHS-based methods [106, 144, 74, 227], which have been refined and extended in more recent work [e.g., 197].

Although the statistical aspects of kernel ridge regression (KRR) are well-understood, the computation of the KRR estimate can be challenging for large datasets. In a standard implementation [179], the kernel matrix must be inverted, which requires $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory. Such scalings are prohibitive when the sample size N is large. As a consequence, approximations have been designed to avoid the expense of finding an exact minimizer. One family of approaches is based on low-rank approximation of the kernel matrix; examples include kernel PCA [183], the incomplete Cholesky decomposition [71], or Nyström sampling [213]. These methods reduce the time complexity to $\mathcal{O}(dN^2)$ or $\mathcal{O}(d^2N)$, where $d \ll N$ is the preserved rank. The associated prediction error has only been studied very recently. A second line of research has considered early-stopping of iterative optimization algorithms for KRR, including gradient descent [222, 169] and conjugate gradient methods [29], where early-stopping provides regularization against over-fitting and improves run-time. If the algorithm stops after t iterations, the aggregate time complexity is $\mathcal{O}(tN^2)$.

In this chapter, we study a different decomposition-based approach. The algorithm is appealing in its simplicity: we partition the dataset of size N randomly into m equal sized subsets, and we compute the kernel ridge regression estimate $\widehat{f}_i$ for each of the $i = 1, \ldots, m$ subsets independently, with a *careful choice* of the regularization parameter. The estimates are then averaged via $\bar{f} = (1/m) \sum_{i=1}^m \widehat{f}_i$. Our main theoretical result gives conditions under

which the average $\bar{f}$ achieves the minimax rate of convergence over the underlying Hilbert space. Even using naive implementations of KRR, this decomposition gives time and memory complexity scaling as $\mathcal{O}(N^3/m^2)$ and $\mathcal{O}(N^2/m^2)$, respectively. Moreover, our approach dovetails naturally with parallel and distributed computation: we are guaranteed superlinear speedup with m parallel processors (though we must still communicate the function estimates from each processor). It demonstrates the potential benefits of divide-and-conquer approaches for nonparametric and infinite-dimensional regression problems.

One difficulty in solving each of the sub-problems independently is how to choose the regularization parameter. Due to the infinite-dimensional nature of non-parametric problems, the choice of regularization parameter must be made with care [e.g., 87]. An interesting consequence of our theoretical analysis is in demonstrating that, even though each partitioned sub-problem is based only on the fraction N/m of samples, it is nonetheless *essential to regularize the partitioned sub-problems as though they had all N samples*. Consequently, from a local point of view, each sub-problem is under-regularized. This "under-regularization" allows the bias of each local estimate to be very small, but it causes a detrimental blow-up in the variance. However, as we prove, the m-fold averaging underlying the method reduces variance enough that the resulting estimator $\bar{f}$ still attains optimal convergence rate.

4.1 Problem set-up

We begin with the background and notation required for a precise statement of our problem.

4.1.1 Kernel ridge regression

Suppose that we are given a data set $\{(x_i, y_i)\}_{i=1}^N$ consisting of N i.i.d. samples drawn from an unknown distribution $\mathbb{P}$ over $\mathcal{X} \times \mathbb{R}$, and our goal is to estimate the function that minimizes the mean-squared error $\mathbb{E}[(f(X) - Y)^2]$, where the expectation is taken jointly over (X, Y) pairs. It is well-known that the optimal function is the conditional mean $f^*(x) := \mathbb{E}[Y \mid X = x]$. In order to estimate the unknown function f^*, we consider an M-estimator that is based on minimizing a combination of the least-squares loss defined over the dataset with a weighted penalty based on the squared Hilbert norm,

$$\widehat{f} := \operatorname*{argmin}_{f \in \mathcal{H}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(f(x_i) - y_i\right)^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\}, \tag{4.1}$$

where $\lambda > 0$ is a regularization parameter. When $\mathcal{H}$ is a reproducing kernel Hilbert space, then the estimator (4.1) is known as the *kernel ridge regression estimate*, or KRR for short. It is a natural generalization of the ordinary ridge regression estimate [91] to the non-parametric setting.

By the representer theorem for reproducing kernel Hilbert spaces [210], any solution to the KRR program (4.1) must belong to the linear span of the kernel functions $\{K(\cdot, x_i),\ i =$

$1, \ldots, N\}$. This fact allows the computation of the KRR estimate to be reduced to an N-dimensional quadratic program, involving the N^2 entries of the kernel matrix $\{K(x_i, x_j),\ i, j = 1, \ldots, n\}$. On the statistical side, a line of past work [74, 227, 39, 197, 95] has provided bounds on the estimation error of $\widehat{f}$ as a function of N and λ.

4.2 Main results and their consequences

We now turn to the description of our algorithm, followed by the statements of our main results, namely Theorems 4. The theorem provides an upper bound on the mean-squared prediction error for any trace class kernel. As we illustrate, the theorem provides concrete results when applied to specific classes of kernels. Indeed, as a corollary, we establish that our distributed KRR algorithm achieves minimax-optimal rates for three different kernel classes, namely finite-rank, Gaussian, and Sobolev.

4.2.1 Algorithm and assumptions

The divide-and-conquer algorithm Fast-KRR is easy to describe. Rather than solving the kernel ridge regression problem (4.1) on all N samples, the Fast-KRR method executes the following three steps:

1. Divide the set of samples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ evenly and uniformly at random into the m disjoint subsets $S_1, \ldots, S_m \subset \mathcal{X} \times \mathbb{R}$, such that every subset contains N/m samples.

2. For each $i = 1, 2, \ldots, m$, compute the *local KRR estimate*

$$\widehat{f}_i := \operatorname*{argmin}_{f \in \mathcal{H}} \left\{ \frac{1}{|S_i|} \sum_{(x,y) \in S_i} (f(x) - y)^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\}. \tag{4.2}$$

3. Average together the local estimates and output $\bar{f} = \frac{1}{m} \sum_{i=1}^{m} \widehat{f}_i$.

This description actually provides a family of estimators, one for each choice of the regularization parameter $\lambda > 0$. Our main result applies to any choice of λ, while our corollaries for specific kernel classes optimize λ as a function of the kernel.

We now describe our main assumptions. Our first assumption, for which we have two variants, deals with the tail behavior of the basis functions $\{\phi_j\}_{j=1}^{\infty}$.

Assumption F. *For some $k \geq 2$, there is a constant $\rho < \infty$ such that $\mathbb{E}[\phi_j(X)^{2k}] \leq \rho^{2k}$ for all $j \in \mathbb{N}$.*

In certain cases, we show that sharper error guarantees can be obtained by enforcing a stronger condition of uniform boundedness.

Assumption F′. *There is a constant $\rho < \infty$ such that $\sup_{x \in \mathcal{X}} |\phi_j(x)| \leq \rho$ for all $j \in \mathbb{N}$.*

Assumption F′ holds, for example, when the input x is drawn from a closed interval and the kernel is translation invariant, i.e. $K(x, x') = \psi(x - x')$ for some even function ψ. Given input space $\mathcal{X}$ and kernel K, the assumption is verifiable without the data.

Recalling that $f^*(x) := \mathbb{E}[Y \mid X = x]$, our second assumption involves the deviations of the zero-mean noise variables $Y - f^*(x)$. In the simplest case, when $f^* \in \mathcal{H}$, we require only a bounded variance condition:

Assumption G. *The function $f^* \in \mathcal{H}$, and for $x \in \mathcal{X}$, we have $\mathbb{E}[(Y - f^*(x))^2 \mid x] \leq \sigma^2$.*

When the function $f^* \notin \mathcal{H}$, we require a slightly stronger variant of this assumption. For each $\lambda \geq 0$, define

$$f^*_\lambda = \operatorname*{argmin}_{f \in \mathcal{H}} \left\{ \mathbb{E}\left[(f(X) - Y)^2\right] + \lambda \|f\|^2_{\mathcal{H}} \right\}. \tag{4.3}$$

Note that $f^* = f^*_0$ corresponds to the usual regression function. As $f^* \in L^2(\mathbb{P})$, for each $\lambda \geq 0$, the associated mean-squared error $\sigma^2_\lambda(x) := \mathbb{E}[(Y - f^*_\lambda(x))^2 \mid x]$ is finite for almost every x. In this more general setting, the following assumption replaces Assumption G:

Assumption G′. *For any $\lambda \geq 0$, there exists a constant $\tau_\lambda < \infty$ such that $\tau^4_\lambda = \mathbb{E}[\sigma^4_\lambda(X)]$.*

4.2.2 Statement of main results

With these assumptions in place, we are now ready for the statements of our main results. All of our results give bounds on the mean-squared estimation error $\mathbb{E}[\|\bar{f} - f^*\|_2^2]$ associated with the averaged estimate $\bar{f}$ based on an assigning $n = N/m$ samples to each of m machines. Both theorem statements involve the following three kernel-related quantities:

$$\operatorname{tr}(K) := \sum_{j=1}^{\infty} \mu_j, \quad \gamma(\lambda) := \sum_{j=1}^{\infty} \frac{1}{1 + \lambda/\mu_j}, \quad \text{and} \quad \beta_d = \sum_{j=d+1}^{\infty} \mu_j. \tag{4.4}$$

The first quantity is the kernel trace, which serves a crude estimate of the "size" of the kernel operator, and assumed to be finite. The second quantity $\gamma(\lambda)$, familiar from previous work on kernel regression [227], is the *effective dimensionality* of the kernel K with respect to $L^2(\mathbb{P})$. Finally, the quantity β_d is parameterized by a positive integer d that we may choose in applying the bounds, and it describes the tail decay of the eigenvalues of K. For $d = 0$, note that $\beta_0 = \operatorname{tr} K$. Finally, both theorems involve a quantity that depends on the number of moments k in Assumption F:

$$b(n, d, k) := \max\left\{ \sqrt{\max\{k, \log(d)\}}, \frac{\max\{k, \log(d)\}}{n^{1/2 - 1/k}} \right\}. \tag{4.5}$$

Here the integer $d \in \mathbb{N}$ is a free parameter that may be optimized to obtain the sharpest possible upper bound. (The algorithm's execution is independent of d.)

Theorem 4. *With $f^* \in \mathcal{H}$ and under Assumptions F and G, the mean-squared error of the averaged estimate $\bar{f}$ is upper bounded as*

$$\mathbb{E}\left[\left\|\bar{f} - f^*\right\|_2^2\right] \leq \left(8 + \frac{12}{m}\right)\lambda \left\|f^*\right\|_{\mathcal{H}}^2 + \frac{12\sigma^2\gamma(\lambda)}{N} + \inf_{d\in\mathbb{N}}\left\{T_1(d) + T_2(d) + T_3(d)\right\}, \tag{4.6}$$

where

$$T_1(d) = \frac{8\rho^4 \left\|f^*\right\|_{\mathcal{H}}^2 \operatorname{tr}(K)\beta_d}{\lambda}, \quad T_2(d) = \frac{4\left\|f^*\right\|_{\mathcal{H}}^2 + 2\sigma^2/\lambda}{m}\left(\mu_{d+1} + \frac{12\rho^4 \operatorname{tr}(K)\beta_d}{\lambda}\right), \quad \textit{and}$$

$$T_3(d) = \left(Cb(n,d,k)\frac{\rho^2\gamma(\lambda)}{\sqrt{n}}\right)^k \mu_0 \left\|f^*\right\|_{\mathcal{H}}^2 \left(1 + \frac{2\sigma^2}{m\lambda} + \frac{4\left\|f^*\right\|_{\mathcal{H}}^2}{m}\right),$$

and C denotes a universal (numerical) constant.

Theorem 4 is a general result that applies to any trace-class kernel. Although the statement appears somewhat complicated at first sight, it yields concrete and interpretable guarantees on the error when specialized to particular kernels, as we illustrate in Section 4.2.3.

Before doing so, let us make a few heuristic arguments in order to provide intuition. In typical settings, the term $T_3(d)$ goes to zero quickly: if the number of moments k is suitably large and number of partitions m is small—say enough to guarantee that $(b(n,d,k)\gamma(\lambda)/\sqrt{n})^k = \mathcal{O}(1/N)$—it will be of lower order. As for the remaining terms, at a high level, we show that an appropriate choice of the free parameter d leaves the first two terms in the upper bound (4.6) dominant. Note that the terms μ_{d+1} and β_d are decreasing in d while the term $b(n,d,k)$ increases with d. However, the increasing term $b(n,d,k)$ grows only logarithmically in d, which allows us to choose a fairly large value without a significant penalty. As we show in our corollaries, for many kernels of interest, as long as the number of machines m is not "too large," this tradeoff is such that $T_1(d)$ and $T_2(d)$ are also of lower order compared to the two first terms in the bound (4.6). In such settings, Theorem 4 guarantees an upper bound of the form

$$\mathbb{E}\left[\left\|\bar{f} - f^*\right\|_2^2\right] = \mathcal{O}(1) \cdot \Big[\underbrace{\lambda \left\|f^*\right\|_{\mathcal{H}}^2}_{\text{Squared bias}} + \underbrace{\frac{\sigma^2\gamma(\lambda)}{N}}_{\text{Variance}}\Big]. \tag{4.7}$$

This inequality reveals the usual bias-variance trade-off in non-parametric regression; choosing a smaller value of $\lambda > 0$ reduces the first squared bias term, but increases the second variance term. Consequently, the setting of λ that minimizes the sum of these two terms is defined by the relationship

$$\lambda \left\|f^*\right\|_{\mathcal{H}}^2 \simeq \sigma^2 \frac{\gamma(\lambda)}{N}. \tag{4.8}$$

This type of fixed point equation is familiar from work on oracle inequalities and local complexity measures in empirical process theory [15, 106, 74, 227], and when λ is chosen so

that the fixed point equation (4.8) holds this (typically) yields minimax optimal convergence rates [15, 106, 227, 39]. In Section 4.2.3, we provide detailed examples in which the choice λ^* specified by equation (4.8), followed by application of Theorem 4, yields minimax-optimal prediction error (for the Fast-KRR algorithm) for many kernel classes.

4.2.3 Some consequences

We now turn to deriving some explicit consequences of our main theorems for specific classes of reproducing kernel Hilbert spaces. In each case, our derivation follows the broad outline given the the remarks following Theorem 4: we first choose the regularization parameter λ to balance the bias and variance terms, and then show, by comparison to known minimax lower bounds, that the resulting upper bound is optimal. Finally, we derive an upper bound on the number of subsampled data sets m for which the minimax optimal convergence rate can still be achieved. Throughout this section, we assume that $f^* \in \mathcal{H}$.

4.2.3.1 Finite-rank Kernels

Our first corollary applies to problems for which the kernel has finite rank r, meaning that its eigenvalues satisfy $\mu_j = 0$ for all $j > r$. Examples of such finite rank kernels include the linear kernel $K(x, x') = \langle x, x' \rangle_{\mathbb{R}^d}$, which has rank at most $r = d$; and the kernel $K(x, x) = (1 + x\, x')^m$ generating polynomials of degree m, which has rank at most $r = m + 1$.

Corollary 3. *For a kernel with rank r, consider the output of the Fast-KRR algorithm with $\lambda = r/N$. Suppose that Assumption G and Assumptions F (or F′) hold, and that the number of processors m satisfy the bound*

$$m \leq c \frac{N^{\frac{k-4}{k-2}}}{r^{2\frac{k-1}{k-2}} \rho^{\frac{4k}{k-2}} \log^{\frac{k}{k-2}} r} \quad \textit{(Assumption F)} \qquad \textit{or} \qquad m \leq c \frac{N}{r^2 \rho^4 \log N} \quad \textit{(Assumption F′)},$$

where c is a universal (numerical) constant. For suitably large N, the mean-squared error is bounded as

$$\mathbb{E}\left[\left\|\bar{f} - f^*\right\|_2^2\right] = \mathcal{O}(1) \frac{\sigma^2 r}{N}. \tag{4.9}$$

For finite-rank kernels, the rate (4.9) is known to be minimax-optimal, meaning that there is a universal constant $c' > 0$ such that

$$\inf_{\widetilde{f}} \sup_{\|f^*\|_{\mathcal{H}} \leq 1} \mathbb{E}[\|\widetilde{f} - f^*\|_2^2] \geq c' \frac{r}{N}, \tag{4.10}$$

where the infimum ranges over all estimators $\widetilde{f}$ based on observing all N samples (and with no constraints on memory and/or computation). This lower bound follows from Theorem 2(a) of Raskutti et al. [171] with $s = d = 1$.

4.2.3.2 Polynomially Decaying Eigenvalues

Our next corollary applies to kernel operators with eigenvalues that obey a bound of the form

$$\mu_j \leq C\, j^{-2\nu} \quad \text{for all } j = 1, 2, \ldots, \tag{4.11}$$

where C is a universal constant, and $\nu > 1/2$ parameterizes the decay rate. We note that equation (4.4) assumes a finite kernel trace $\operatorname{tr}(K) := \sum_{j=1}^{\infty} \mu_j$. Since $\operatorname{tr}(K)$ appears in Theorem 4, it is natural to use $\sum_{j=1}^{\infty} C j^{-2\nu}$ as an upper bound on $\operatorname{tr}(K)$. This upper bound is finite if and only if $\nu > 1/2$.

Kernels with polynomial decaying eigenvalues include those that underlie for the Sobolev spaces with different orders of smoothness [e.g. 26, 80]. As a concrete example, the first-order Sobolev kernel $K(x, x') = 1 + \min\{x, x'\}$ generates an RKHS of Lipschitz functions with smoothness $\nu = 1$. Other higher-order Sobolev kernels also exhibit polynomial eigendecay with larger values of the parameter ν.

Corollary 4. *For any kernel with ν-polynomial eigendecay (4.11), consider the output of the Fast-KRR algorithm with $\lambda = (1/N)^{\frac{2\nu}{2\nu+1}}$. Suppose that Assumption G and Assumption F (or F′) hold, and that the number of processors satisfy the bound*

$$m \leq c\left(\frac{N^{\frac{2(k-4)\nu - k}{(2\nu+1)}}}{\rho^{4k}\log^k N}\right)^{\frac{1}{k-2}} \quad \textit{(Assumption F)} \quad \textit{or} \quad m \leq c\frac{N^{\frac{2\nu-1}{2\nu+1}}}{\rho^4 \log N} \quad \textit{(Assumption F′)},$$

where c is a constant only depending on ν. Then the mean-squared error is bounded as

$$\mathbb{E}\left[\left\|\bar{f} - f^*\right\|_2^2\right] = \mathcal{O}\left(\left(\frac{\sigma^2}{N}\right)^{\frac{2\nu}{2\nu+1}}\right). \tag{4.12}$$

The upper bound (4.12) is unimprovable up to constant factors, as shown by known minimax bounds on estimation error in Sobolev spaces [198, 205]; see also Theorem 2(b) of Raskutti et al. [171].

4.2.3.3 Exponentially Decaying Eigenvalues

Our final corollary applies to kernel operators with eigenvalues that obey a bound of the form

$$\mu_j \leq c_1 \exp(-c_2 j^2) \quad \text{for all } j = 1, 2, \ldots, \tag{4.13}$$

for strictly positive constants (c_1, c_2). Such classes include the RKHS generated by the Gaussian kernel $K(x, x') = \exp(-\|x - x'\|_2^2)$.

Corollary 5. *For a kernel with sub-Gaussian eigendecay (4.13), consider the output of the Fast-KRR algorithm with $\lambda = 1/N$. Suppose that Assumption G and Assumption F (or F′) hold, and that the number of processors satisfy the bound*

$$m \leq c \frac{N^{\frac{k-4}{k-2}}}{\rho^{\frac{4k}{k-2}} \log^{\frac{2k-1}{k-2}} N} \quad \textit{(Assumption F)} \qquad \textit{or} \qquad m \leq c \frac{N}{\rho^4 \log^2 N} \quad \textit{(Assumption F′)},$$

where c is a constant only depending on c_2. Then the mean-squared error is bounded as

$$\mathbb{E}\left[\left\|\bar{f} - f^*\right\|_2^2\right] = \mathcal{O}\left(\sigma^2 \frac{\sqrt{\log N}}{N}\right). \tag{4.14}$$

The upper bound (4.14) is minimax optimal; see, for example, Theorem 1 and Example 2 of the recent paper by Yang et al. [220].

4.2.3.4 Summary

Each corollary gives a critical threshold for the number m of data partitions: as long as m is below this threshold, the decomposition-based Fast-KRR algorithm gives the optimal rate of convergence. It is interesting to note that the number of splits may be quite large: each grows asymptotically with N whenever the basis functions have more than four moments (viz. Assumption F). Moreover, the Fast-KRR method can attain these optimal convergence rates while using substantially less computation than standard kernel ridge regression methods, as it requires solving problems only of size N/m.

4.2.4 The choice of regularization parameter

In practice, the local sample size on each machine may be different and the optimal choice for the regularization λ may not be known *a priori*, so that an adaptive choice of the regularization parameter λ is desirable [e.g. 205, Chapters 3.5–3.7]. We recommend using cross-validation to choose the regularization parameter, and we now sketch a heuristic argument that an adaptive algorithm using cross-validation may achieve optimal rates of convergence. (We leave fuller analysis to future work.)

Let λ_n be the (oracle) optimal regularization parameter given knowledge of the sampling distribution $\mathbb{P}$ and eigen-structure of the kernel K. We assume (cf. Corollary 4) that there is a constant $\nu > 0$ such that $\lambda_n \asymp n^{-\nu}$ as $n \to \infty$. Let n_i be the local sample size for each machine i and N the global sample size; we assume that $n_i \gg \sqrt{N}$ (clearly, $N \geq n_i$). First, use local cross-validation to choose regularization parameters $\widehat{W}\lambda_{n_i}$ and $\widehat{W}\lambda_{n_i^2/N}$ corresponding to samples of size n_i and n_i^2/N, respectively. Heuristically, if cross validation is successful, we expect to have $\widehat{W}\lambda_{n_i} \simeq n_i^{-\nu}$ and $\widehat{W}\lambda_{n_i^2/N} \simeq N^{\nu} n_i^{-2\nu}$, yielding that $\widehat{W}\lambda_{n_i}{}^2 / \widehat{W}\lambda_{n_i^2/N} \simeq N^{-\nu}$. With this intuition, we then compute local estimates

$$\widehat{f}_i := \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n_i} \sum_{(x,y) \in S_i} (f(x) - y)^2 + \widehat{W}\lambda_{(i)} \|f\|_{\mathcal{H}}^2 \quad \text{where } \widehat{W}\lambda_{(i)} := \frac{\widehat{W}\lambda_{n_i}^2}{\widehat{W}\lambda_{n_i^2/N}} \tag{4.15}$$

and global average estimate $\bar{f} = \sum_{i=1}^m \frac{n_i}{N} \widehat{f}_i$ as usual. Notably, we have $\widehat{W}\lambda_{(i)} \simeq \lambda_N$ in this heuristic setting. Using formula (4.15) and the average $\bar{f}$, we have

$$\mathbb{E}\Big[\big\|\bar{f} - f^*\big\|_2^2 \Big] = \mathbb{E}\Big[\Big\| \sum_{i=1}^m \frac{n_i}{N} \Big(\widehat{f}_i - \mathbb{E}[\widehat{f}_i] \Big) \Big\|_2^2 \Big] + \Big\| \sum_{i=1}^m \frac{n_i}{N} \Big(\mathbb{E}[\widehat{f}_i] - f^* \Big) \Big\|_2^2$$
$$\leq \sum_{i=1}^m \frac{n_i^2}{N^2} \mathbb{E}\left[\big\| \widehat{f}_i - \mathbb{E}[\widehat{f}_i] \big\|_2^2 \right] + \max_{i \in [m]} \left\{ \big\| \mathbb{E}[\widehat{f}_i] - f^* \big\|_2^2 \right\}. \tag{4.16}$$

Using Lemmas 14 and 15 from the proof of Theorem 4 to come and assuming that $\widehat{W}\lambda_n$ is concentrated tightly enough around λ_n, we obtain $\|\mathbb{E}[\widehat{f}_i] - f^*\|_2^2 = \mathcal{O}(\lambda_N \|f^*\|_{\mathcal{H}}^2)$ by Lemma 14 and that $\mathbb{E}[\|\widehat{f}_i - \mathbb{E}[\widehat{f}_i]\|_2^2] = \mathcal{O}(\frac{\gamma(\lambda_N)}{n_i})$ by Lemma 15. Substituting these bounds into inequality (4.16) and noting that $\sum_i n_i = N$, we may upper bound the overall estimation error as

$$\mathbb{E}\Big[\big\|\bar{f} - f^*\big\|_2^2 \Big] \leq \mathcal{O}(1) \cdot \left(\lambda_N \|f^*\|_{\mathcal{H}}^2 + \frac{\gamma(\lambda_N)}{N} \right).$$

While the derivation of this upper bound was non-rigorous, we believe that it is roughly accurate, and in comparison with the previous upper bound (4.7), it provides optimal rates of convergence.

4.3 Proofs of the main theorem and related results

We now turn to the proofs of Theorem 4 and Corollaries 3 through 5. This section contains only a high-level view of proof of Theorem 4; we defer more technical aspects to Section 4.5.

4.3.1 Proof of Theorem 4

Using the definition of the averaged estimate $\bar{f} = \frac{1}{m} \sum_{i=1}^m \widehat{f}_i$, a bit of algebra yields

$$\begin{aligned}
\mathbb{E}[\big\|\bar{f} - f^*\big\|_2^2] &= \mathbb{E}[\big\|(\bar{f} - \mathbb{E}[\bar{f}]) + (\mathbb{E}[\bar{f}] - f^*)\big\|_2^2] \\
&= \mathbb{E}[\big\|\bar{f} - \mathbb{E}[\bar{f}]\big\|_2^2] + \big\|\mathbb{E}[\bar{f}] - f^*\big\|_2^2 + 2\mathbb{E}[\langle \bar{f} - \mathbb{E}[\bar{f}], \mathbb{E}[\bar{f}] - f^* \rangle_{L^2(\mathbb{P})}] \\
&= \mathbb{E}\left[\Big\| \frac{1}{m} \sum_{i=1}^m (\widehat{f}_i - \mathbb{E}[\widehat{f}_i]) \Big\|_2^2 \right] + \big\|\mathbb{E}[\bar{f}] - f^*\big\|_2^2,
\end{aligned}$$

where we used the fact that $\mathbb{E}[\widehat{f}_i] = \mathbb{E}[\bar{f}]$ for each $i \in [m]$. Using this unbiasedness once more, we bound the variance of the terms $\widehat{f}_i - \mathbb{E}[\bar{f}]$ to see that

$$\begin{aligned}
\mathbb{E}\left[\big\|\bar{f} - f^*\big\|_2^2 \right] &= \frac{1}{m} \mathbb{E}\left[\|\widehat{f}_1 - \mathbb{E}[\widehat{f}_1]\|_2^2 \right] + \|\mathbb{E}[\widehat{f}_1] - f^*\|_2^2 \\
&\leq \frac{1}{m} \mathbb{E}\left[\|\widehat{f}_1 - f^*\|_2^2 \right] + \|\mathbb{E}[\widehat{f}_1] - f^*\|_2^2,
\end{aligned} \tag{4.17}$$

where we have used the fact that $\mathbb{E}[\widehat{f}_i]$ minimizes $\mathbb{E}[\|\widehat{f}_i - f\|_2^2]$ over $f \in \mathcal{H}$.

The error bound (4.17) suggests our strategy: we upper bound $\mathbb{E}[\|\widehat{f}_1 - f^*\|_2^2]$ and $\|\mathbb{E}[\widehat{f}_1] - f^*\|_2^2$ respectively. Based on equation (4.2), the estimate $\widehat{f}_1$ is obtained from a standard kernel ridge regression with sample size $n = N/m$ and ridge parameter λ. Accordingly, the following two auxiliary results provide bounds on these two terms, where the reader should recall the definitions of $b(n, d, k)$ and β_d from equation (4.4). In each lemma, C represents a universal (numerical) constant.

Lemma 14 (Bias bound)**.** *Under Assumptions F and G, for each $d = 1, 2, \ldots$, we have*

$$\|\mathbb{E}[\widehat{f}] - f^*\|_2^2 \leq 8\lambda \|f^*\|_{\mathcal{H}}^2 + \frac{8\rho^4 \|f^*\|_{\mathcal{H}}^2 \operatorname{tr}(K)\beta_d}{\lambda} + \left(Cb(n, d, k)\frac{\rho^2\gamma(\lambda)}{\sqrt{n}}\right)^k \mu_0 \|f^*\|_{\mathcal{H}}^2. \quad (4.18)$$

Lemma 15 (Variance bound)**.** *Under Assumptions F and G, for each $d = 1, 2, \ldots$, we have*

$$\begin{aligned}\mathbb{E}[\|\widehat{f} - f^*\|_2^2] &\leq 12\lambda \|f^*\|_{\mathcal{H}}^2 + \frac{12\sigma^2\gamma(\lambda)}{n} \\ &+ \left(\frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2\right)\left(\mu_{d+1} + \frac{12\rho^4 \operatorname{tr}(K)\beta_d}{\lambda} + \left(Cb(n, d, k)\frac{\rho^2\gamma(\lambda)}{\sqrt{n}}\right)^k \|f^*\|_2^2\right). \quad (4.19)\end{aligned}$$

The proofs of these lemmas, contained in Section 4.5.1 and 4.5.2 respectively, constitute one main technical contribution of this chapter. Given these two lemmas, the remainder of the theorem proof is straightforward. Combining the inequality (4.17) with Lemmas 14 and 15 yields the claim of Theorem 4.

Remarks: The proofs of Lemmas 14 and 15 are somewhat complex, but to the best of our knowledge, existing literature does not yield significantly simpler proofs. We now discuss this claim to better situate our technical contributions. Define the regularized population minimizer $f_\lambda^* := \operatorname{argmin}_{f\in\mathcal{H}}\{\mathbb{E}[(f(X) - Y)^2] + \lambda\|f\|_{\mathcal{H}}^2\}$. Expanding the decomposition (4.17) of the $L^2(\mathbb{P})$-risk into bias and variance terms, we obtain the further bound

$$\begin{aligned}\mathbb{E}\left[\left\|\bar{f} - f^*\right\|_2^2\right] &\leq \|\mathbb{E}[\widehat{f}_1] - f^*\|_2^2 + \frac{1}{m}\mathbb{E}\left[\|\widehat{f}_1 - f^*\|_2^2\right] \\ &= \underbrace{\|\mathbb{E}[\widehat{f}_1] - f^*\|_2^2}_{:=T_1} + \frac{1}{m}\Bigg(\underbrace{\|f_\lambda^* - f^*\|_2^2}_{:=T_2} + \underbrace{\mathbb{E}\left[\|\widehat{f}_1 - f^*\|_2^2\right] - \|f_\lambda^* - f^*\|_2^2}_{:=T_3}\Bigg) = T_1 + \frac{1}{m}(T_2 + T_3).\end{aligned}$$

In this decomposition, T_1 and T_2 are bias and approximation error terms induced by the regularization parameter λ, while T_3 is an excess risk (variance) term incurred by minimizing the empirical loss.

This upper bound illustrates three trade-offs in our subsampled and averaged kernel regression procedure:

- The trade-off between T_2 and T_3: when the regularization parameter λ grows, the bias term T_2 increases while the variance term T_3 converges to zero.
- The trade-off between T_1 and T_3: when the regularization parameter λ grows, the bias term T_1 increases while the variance term T_3 converges to zero.
- The trade-off between T_1 and the computation time: when the number of machines m grows, the bias term T_1 increases (as the local sample size $n = N/m$ shrinks), while the computation time N^3/m^2 decreases.

Theoretical results in the KRR literature focus on the trade-off between T_2 and T_3, but in the current context, we also need an upper bound on the bias term T_1, which is not relevant for classical (centralized) analyses.

With this setting in mind, Lemma 14 tightly upper bounds the bias T_1 as a function of λ and n. An essential part of the proof is to characterize the properties of $\mathbb{E}[\widehat{f}_1]$, which is the expectation of a nonparametric empirical loss minimizer. We are not aware of existing literature on this problem, and the proof of Lemma 14 introduces novel techniques for this purpose.

On the other hand, Lemma 15 upper bounds $\mathbb{E}[\|\widehat{f}_1 - f^*\|_2^2]$ as a function of λ and n. Past work has focused on bounding a quantity of this form, but for technical reasons, most work [e.g. 74, 145, 15, 227] focuses on analyzing the constrained form

$$\widehat{f}_i := \operatorname*{argmin}_{\|f\|_{\mathcal{H}} \leq C} \frac{1}{|S_i|} \sum_{(x,y) \in S_i} (f(x) - y)^2, \tag{4.20}$$

of kernel ridge regression. While this problem traces out the same set of solutions as that of the regularized kernel ridge regression estimator (4.2), it is non-trivial to determine a matched setting of λ for a given C. Zhang [225] provides one of the few analyses of the regularized ridge regression estimator (4.2) (or (4.1)), providing an upper bound of the form $\mathbb{E}[\|\widehat{f} - f^*\|_2^2] = \mathcal{O}(\lambda + \frac{1/\lambda}{n})$, which is at best $\mathcal{O}(\frac{1}{\sqrt{n}})$. In contrast, Lemma 15 gives upper bound $\mathcal{O}(\lambda + \frac{\gamma(\lambda)}{n})$; the effective dimension $\gamma(\lambda)$ is often much smaller than $1/\lambda$, yielding a stronger convergence guarantee.

4.3.2 Proof of Corollary 3

We first present a general inequality bounding the size of m for which optimal convergence rates are possible. We assume that d is chosen large enough such that we have $\log(d) \geq k$ and $d \geq N$. In the rest of the proof, our assignment to d will satisfy these inequalities. In this case, inspection of Theorem 4 shows that if m is small enough that

$$\left(\sqrt{\frac{\log d}{N/m}}\rho^2\gamma(\lambda)\right)^k \frac{1}{m\lambda} \leq \frac{\gamma(\lambda)}{N},$$

then the term $T_3(d)$ provides a convergence rate given by $\gamma(\lambda)/N$. Thus, solving the expression above for m, we find

$$\frac{m \log d}{N} \rho^4 \gamma(\lambda)^2 = \frac{\lambda^{2/k} m^{2/k} \gamma(\lambda)^{2/k}}{N^{2/k}} \quad \text{or} \quad m^{\frac{k-2}{k}} = \frac{\lambda^{\frac{2}{k}} N^{\frac{k-2}{k}}}{\gamma(\lambda)^{2\frac{k-1}{k}} \rho^4 \log d}.$$

Taking $(k-2)/k$-th roots of both sides, we obtain that if

$$m \leq \frac{\lambda^{\frac{2}{k-2}} N}{\gamma(\lambda)^{2\frac{k-1}{k-2}} \rho^{\frac{4k}{k-2}} \log^{\frac{k}{k-2}} d}, \tag{4.21}$$

then the term $T_3(d)$ of the bound (4.6) is $\mathcal{O}(\gamma(\lambda)/N)$.

Now we apply the bound (4.21) in the case in the corollary. Let us take $d = \max\{r, N\}$. Notice that $\beta_d = \beta_r = \mu_{r+1} = 0$. We find that $\gamma(\lambda) \leq r$ since each of its terms is bounded by 1, and we take $\lambda = r/N$. Evaluating the expression (4.21) with this value, we arrive at

$$m \leq \frac{N^{\frac{k-4}{k-2}}}{r^{2\frac{k-1}{k-2}} \rho^{\frac{4k}{k-2}} \log^{\frac{k}{k-2}} d}.$$

If we have sufficiently many moments that $k \geq \log N$, and $N \geq r$ (for example, if the basis functions ϕ_j have a uniform bound ρ, then k can be chosen arbitrarily large), then we may take $k = \log N$, which implies that $N^{\frac{k-4}{k-2}} = \Omega(N)$, $r^{2\frac{k-1}{k-2}} = \mathcal{O}(r^2)$ and $\rho^{\frac{4k}{k-2}} = \mathcal{O}(\rho^4)$; and we replace $\log d$ with $\log N$. Then so long as

$$m \leq c \frac{N}{r^2 \rho^4 \log N}$$

for some constant $c > 0$, we obtain an identical result.

4.3.3 Proof of Corollary 4

We follow the program outlined in our remarks following Theorem 4. We must first choose λ on the order of $\gamma(\lambda)/N$. To that end, we note that setting $\lambda = N^{-\frac{2\nu}{2\nu+1}}$ gives

$$\begin{aligned}\gamma(\lambda) = \sum_{j=1}^{\infty} \frac{1}{1 + j^{2\nu} N^{-\frac{2\nu}{2\nu+1}}} &\leq N^{\frac{1}{2\nu+1}} + \sum_{j > N^{\frac{1}{2\nu+1}}} \frac{1}{1 + j^{2\nu} N^{-\frac{2\nu}{2\nu+1}}} \\ &\leq N^{\frac{1}{2\nu+1}} + N^{\frac{2\nu}{2\nu+1}} \int_{N^{\frac{1}{2\nu+1}}} \frac{1}{u^{2\nu}} du = N^{\frac{1}{2\nu+1}} + \frac{1}{2\nu - 1} N^{\frac{1}{2\nu+1}}.\end{aligned}$$

Dividing by N, we find that $\lambda \approx \gamma(\lambda)/N$, as desired. Now we choose the truncation parameter d. By choosing $d = N^t$ for some $t \in \mathbb{R}_+$, then we find that $\mu_{d+1} \lesssim N^{-2\nu t}$ and an integration yields $\beta_d \lesssim N^{-(2\nu-1)t}$. Setting $t = 3/(2\nu - 1)$ guarantees that $\mu_{d+1} \lesssim N^{-3}$ and

$\beta_d \lesssim N^{-3}$; the corresponding terms in the bound (4.6) are thus negligible. Moreover, we have for any finite k that $\log d \gtrsim k$.

Applying the general bound (4.21) on m, we arrive at the inequality

$$m \leq c \frac{N^{-\frac{4\nu}{(2\nu+1)(k-2)}} N}{N^{\frac{2(k-1)}{(2\nu+1)(k-2)}} \rho^{\frac{4k}{k-2}} \log^{\frac{k}{k-2}} N} = c \frac{N^{\frac{2(k-4)\nu-k}{(2\nu+1)(k-2)}}}{\rho^{\frac{4k}{k-2}} \log^{\frac{k}{k-2}} N}.$$

Whenever this holds, we have convergence rate $\lambda = N^{-\frac{2\nu}{2\nu+1}}$. Now, let Assumption F′ hold. Then taking $k = \log N$, the above bound becomes (to a multiplicative constant factor) $N^{\frac{2\nu-1}{2\nu+1}}/\rho^4 \log N$ as claimed.

4.3.4 Proof of Corollary 5

First, we set $\lambda = 1/N$. Considering the sum $\gamma(\lambda) = \sum_{j=1}^{\infty} \mu_j/(\mu_j + \lambda)$, we see that for $j \leq \sqrt{(\log N)/c_2}$, the elements of the sum are bounded by 1. For $j > \sqrt{(\log N)/c_2}$, we make the approximation

$$\sum_{j \geq \sqrt{(\log N)/c_2}} \frac{\mu_j}{\mu_j + \lambda} \leq \frac{1}{\lambda} \sum_{j \geq \sqrt{(\log N)/c_2}} \mu_j \lesssim N \int_{\sqrt{(\log N)/c_2}}^{\infty} \exp(-c_2 t^2) dt = \mathcal{O}(1).$$

Thus we find that $\gamma(\lambda)+1 \leq c\sqrt{\log N}$ for some constant c. By choosing $d = N^2$, we have that the tail sum and $(d+1)$-th eigenvalue both satisfy $\mu_{d+1} \leq \beta_d \lesssim c_2^{-1} N^{-4}$. As a consequence, all the terms involving β_d or μ_{d+1} in the bound (4.6) are negligible.

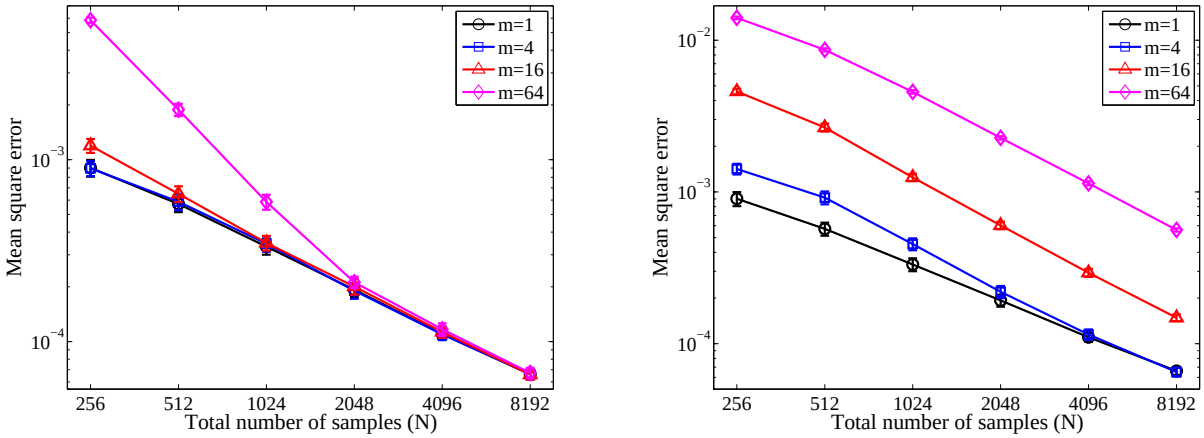
Recalling our inequality (4.21), we thus find that (under Assumption F), as long as the number of partitions m satisfies

$$m \leq c \frac{N^{\frac{k-4}{k-2}}}{\rho^{\frac{4k}{k-2}} \log^{\frac{2k-1}{k-2}} N},$$

the convergence rate of $\bar{f}$ to f^* is given by $\gamma(\lambda)/N \simeq \sqrt{\log N}/N$. Under the boundedness assumption F′, as we did in the proof of Corollary 3, we take $k = \log N$ in Theorem 4. By inspection, this yields the second statement of the corollary.

4.4 Experimental results

In this section, we report the results of experiments on both simulated and real-world data designed to test the sharpness of our theoretical predictions.



(a) With under-regularization (b) Without under-regularization

Figure 4.1. The squared $L^2(\mathbb{P})$-norm between between the averaged estimate $\bar{f}$ and the optimal solution f^*. (a) These plots correspond to the output of the Fast-KRR algorithm: each sub-problem is under-regularized by using $\lambda \asymp N^{-2/3}$. (b) Analogous plots when each sub-problem is *not* under-regularized—that is, with $\lambda = n^{-2/3} = (N/m)^{-2/3}$ chosen as if there were only a single dataset of size n.

4.4.1 Simulation studies

We begin by exploring the empirical performance of our subsample-and-average methods for a non-parametric regression problem on simulated datasets. For all experiments in this section, we simulate data from the regression model $y = f^*(x) + \varepsilon$ for $x \in [0,1]$, where $f^*(x) := \min(x, 1-x)$ is 1-Lipschitz, the noise variables $\varepsilon \sim N(0, \sigma^2)$ are normally distributed with variance $\sigma^2 = 1/5$, and the samples $x_i \sim \mathsf{Uni}[0,1]$. The Sobolev space of Lipschitz functions on $[0,1]$ has reproducing kernel $K(x,x') = 1 + \min\{x, x'\}$ and norm $\|f\|_{\mathcal{H}}^2 = f^2(0) + \int_0^1 (f'(z))^2 dz$. By construction, the function $f^*(x) = \min(x, 1-x)$ satisfies $\|f^*\|_{\mathcal{H}} = 1$. The kernel ridge regression estimator $\widehat{f}$ takes the form

$$\widehat{f} = \sum_{i=1}^{N} \alpha_i K(x_i, \cdot), \quad \text{where} \quad \alpha = (K + \lambda N I)^{-1} y, \tag{4.22}$$

and K is the $N \times N$ Gram matrix and I is the $N \times N$ identity matrix. Since the first-order Sobolev kernel has eigenvalues [80] that scale as $\mu_j \asymp (1/j)^2$, the minimax convergence rate in terms of squared $L^2(\mathbb{P})$-error is $N^{-2/3}$ (see e.g. [205, 198, 39]).

By Corollary 4 with $\nu = 1$, this optimal rate of convergence can be achieved by Fast-KRR with regularization parameter $\lambda \asymp N^{-2/3}$ as long as the number of partitions m satisfies $m \lesssim N^{1/3}$. In each of our experiments, we begin with a dataset of size $N = mn$, which we partition uniformly at random into m disjoint subsets. We compute the local estimator $\widehat{f}_i$ for each of the m subsets using n samples via (4.22), where the Gram matrix is constructed using the ith batch of samples (and n replaces N). We then compute $\bar{f} = (1/m) \sum_{i=1}^{m} \widehat{f}_i$.

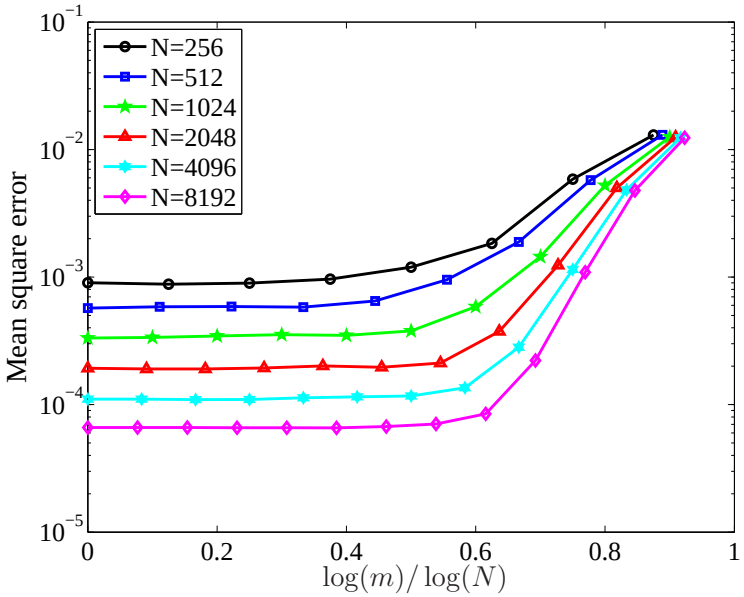


Figure 4.2. The mean-square error curves for fixed sample size but varied number of partitions. We are interested in the threshold of partitioning number m under which the optimal rate of convergence is achieved.

Our experiments compare the error of $\bar{f}$ as a function of sample size N, the number of partitions m, and the regularization λ.

In Figure 4.4.1(a), we plot the error $\|\bar{f} - f^*\|_2^2$ versus the total number of samples N, where $N \in \{2^8, 2^9, \ldots, 2^{13}\}$, using four different data partitions $m \in \{1, 4, 16, 64\}$. We execute each simulation 20 times to obtain standard errors for the plot. The black circled curve ($m = 1$) gives the baseline KRR error; if the number of partitions $m \leq 16$, Fast-KRR has accuracy comparable to the baseline algorithm. Even with $m = 64$, Fast-KRR's performance closely matches the full estimator for larger sample sizes ($N \geq 2^{11}$). In the right plot Figure 4.4.1(b), we perform an identical experiment, but we over-regularize by choosing $\lambda = n^{-2/3}$ rather than $\lambda = N^{-2/3}$ in each of the m sub-problems, combining the local estimates by averaging as usual. In contrast to Figure 4.4.1(a), there is an obvious gap between the performance of the algorithms when $m = 1$ and $m > 1$, as our theory predicts.

It is also interesting to understand the number of partitions m into which a dataset of size N may be divided while maintaining good statistical performance. According to Corollary 4 with $\nu = 1$, for the first-order Sobolev kernel, performance degradation should be limited as long as $m \lesssim N^{1/3}$. In order to test this prediction, Figure 4.2 plots the mean-square error $\|\bar{f} - f^*\|_2^2$ versus the ratio $\log(m)/\log(N)$. Our theory predicts that even as the number of partitions m may grow polynomially in N, the error should grow only above some constant value of $\log(m)/\log(N)$. As Figure 4.2 shows, the point that $\|\bar{f} - f^*\|_2$ begins to increase appears to be around $\log(m) \approx 0.45 \log(N)$ for reasonably large N. This empirical performance is somewhat better than the (1/3) thresholded predicted by Corollary 4, but it does confirm that the number of partitions m can scale polynomially with N while retaining

N		$m = 1$	$m = 16$	$m = 64$	$m = 256$	$m = 1024$
2^{12}	Error	$1.26 \cdot 10^{-4}$	$1.33 \cdot 10^{-4}$	$1.38 \cdot 10^{-4}$	N/A	N/A
	Time	1.12 (0.03)	0.03 (0.01)	0.02 (0.00)		
2^{13}	Error	$6.40 \cdot 10^{-5}$	$6.29 \cdot 10^{-5}$	$6.72 \cdot 10^{-5}$	N/A	N/A
	Time	5.47 (0.22)	0.12 (0.03)	0.04 (0.00)		
2^{14}	Error	$3.95 \cdot 10^{-5}$	$4.06 \cdot 10^{-5}$	$4.03 \cdot 10^{-5}$	$3.89 \cdot 10^{-5}$	N/A
	Time	30.16 (0.87)	0.59 (0.11)	0.11 (0.00)	0.06 (0.00)	
2^{15}	Error	Fail	$2.90 \cdot 10^{-5}$	$2.84 \cdot 10^{-5}$	$2.78 \cdot 10^{-5}$	N/A
	Time		2.65 (0.04)	0.43 (0.02)	0.15 (0.01)	
2^{16}	Error	Fail	$1.75 \cdot 10^{-5}$	$1.73 \cdot 10^{-5}$	$1.71 \cdot 10^{-5}$	$1.67 \cdot 10^{-5}$
	Time		16.65 (0.30)	2.21 (0.06)	0.41 (0.01)	0.23 (0.01)
2^{17}	Error	Fail	$1.19 \cdot 10^{-5}$	$1.21 \cdot 10^{-5}$	$1.25 \cdot 10^{-5}$	$1.24 \cdot 10^{-5}$
	Time		90.80 (3.71)	10.87 (0.19)	1.88 (0.08)	0.60 (0.02)

Table 4.1. Timing experiment giving $\|\bar{f} - f^*\|_2^2$ as a function of number of partitions m and data size N, providing mean run-time (measured in second) for each number m of partitions and data size N.

minimax optimality.

Our final experiment gives evidence for the improved time complexity partitioning provides. Here we compare the amount of time required to solve the KRR problem using the naive matrix inversion (4.22) for different partition sizes m and provide the resulting squared errors $\|\bar{f} - f^*\|_2^2$. Although there are more sophisticated solution strategies, we believe this is a reasonable proxy to exhibit Fast-KRR's potential. In Table 4.1, we present the results of this simulation, which we performed in Matlab using a Windows machine with 16GB of memory and a single-threaded 3.4Ghz processor. In each entry of the table, we give the mean error of Fast-KRR and the mean amount of time it took to run (with standard deviation over 10 simulations in parentheses; the error rate standard deviations are an order of magnitude smaller than the errors, so we do not report them). The entries "Fail" correspond to out-of-memory failures because of the large matrix inversion, while entries "N/A" indicate that $\|\bar{f} - f^*\|_2$ was significantly larger than the optimal value (rendering time improvements meaningless). The table shows that without sacrificing accuracy, decomposition via Fast-KRR can yield substantial computational improvements.

4.4.2 Real data experiments

We now turn to the results of experiments studying the performance of Fast-KRR on the task of predicting the year in which a song was released based on audio features associated with the song. We use the Million Song Dataset [21], which consists of 463,715 training examples and a second set of 51,630 testing examples. Each example is a song (track) released between 1922 and 2011, and the song is represented as a vector of timbre information computed about

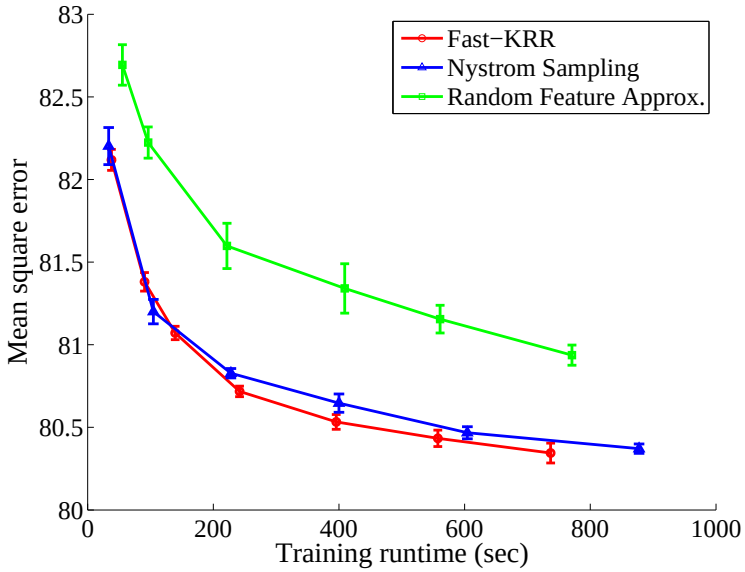


Figure 4.3. Results on year prediction on held-out test songs for Fast-KRR, Nyström sampling, and random feature approximation. Error bars indicate standard deviations over ten experiments.

the song. Each sample consists of the pair $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, where $x_i \in \mathbb{R}^d$ is a $d = 90$-dimensional vector and $y_i \in [1922, 2011]$ is the year in which the song was released. (For further details, see Bertin-Mahieux et al. [21]).

Our experiments with this dataset use the Gaussian radial basis kernel

$$K(x, x') = \exp\left(-\frac{\|x - x'\|_2^2}{2\sigma^2}\right). \tag{4.23}$$

We normalize the feature vectors x so that the timbre signals have standard deviation 1, and select the bandwidth parameter $\sigma = 6$ via cross-validation. For regularization, we set $\lambda = N^{-1}$; since the Gaussian kernel has exponentially decaying eigenvalues (for typical distributions on X), Corollary 5 shows that this regularization achieves the optimal rate of convergence for the Hilbert space.

In Figure 4.3, we compare the time-accuracy curve of Fast-KRR with two approximation-based methods, plotting the mean-squared error between the predicted release year and the actual year on test songs. The first baseline is Nyström subsampling [213], where the kernel matrix is approximated by a low-rank matrix of rank $r \in \{1, \ldots, 6\} \times 10^3$. The second baseline approach is an approximate form of kernel ridge regression using random features [166]. The algorithm approximates the Gaussian kernel (4.23) by the inner product of two random feature vectors of dimensions $D \in \{2, 3, 5, 7, 8.5, 10\} \times 10^3$, and then solves the resulting linear regression problem. For the Fast-KRR algorithm, we use seven partitions

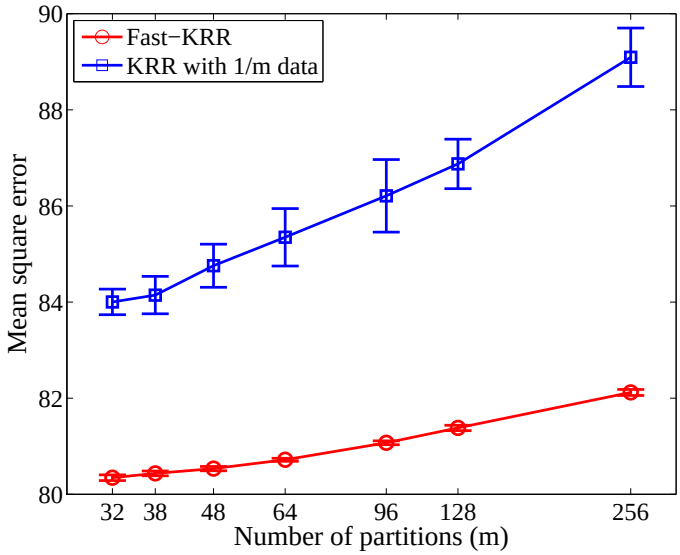


Figure 4.4. Comparison of the performance of Fast-KRR to a standard KRR estimator using a fraction $1/m$ of the data.

$m \in \{32, 38, 48, 64, 96, 128, 256\}$ to test the algorithm. Each algorithm is executed 10 times to obtain standard deviations (plotted as error-bars in Figure 4.3).

As we see in Figure 4.3, for a fixed time budget, Fast-KRR enjoys the best performance, though the margin between Fast-KRR and Nyström sampling is not substantial. In spite of this close performance between Nyström sampling and the divide-and-conquer Fast-KRR algorithm, it is worth noting that with parallel computation, it is trivial to accelerate Fast-KRR m times; parallelizing approximation-based methods appears to be a non-trivial task. Moreover, as our results in Section 4.2 indicate, Fast-KRR is minimax optimal in many regimes. We note in passing that standard linear regression with the original 90 features, while quite fast with runtime on the order of 1 second (ignoring data loading), has mean-squared-error 90.44, which is significantly worse than the kernel-based methods.

Our final experiment provides a sanity check: is the final averaging step in Fast-KRR even necessary? To this end, we compare Fast-KRR with standard KRR using a fraction $1/m$ of the data. For the latter approach, we employ the standard regularization $\lambda \approx (N/m)^{-1}$. As Figure 4.4 shows, Fast-KRR achieves much lower error rates than KRR using only a fraction of the data. Moreover, averaging stabilizes the estimators: the standard deviations of the performance of Fast-KRR are negligible compared to those for standard KRR.

$\widehat{f}$	Empirical KRR minimizer based on n samples
f^*	Optimal function generating data, where $y_i = f^*(x_i) + \varepsilon_i$
Δ	Error $\widehat{f} - f^*$
ξ_x	RKHS evaluator $\xi_x := K(x, \cdot)$, so $\langle f, \xi_x \rangle = \langle \xi_x, f \rangle = f(x)$
$\widehat{\Sigma}$	Operator mapping $\mathcal{H} \to \mathcal{H}$ defined as the outer product $\widehat{\Sigma} := \frac{1}{n} \sum_{i=1}^n \xi_{x_i} \otimes \xi_{x_i}$, so that $\widehat{\Sigma} f = \frac{1}{n} \sum_{i=1}^n \langle \xi_{x_i}, f \rangle \, \xi_{x_i}$
ϕ_j	jth orthonormal basis vector for $L^2(\mathbb{P})$
δ_j	Basis coefficients of Δ or $\mathbb{E}[\Delta \mid X]$ (depending on context), i.e. $\Delta = \sum_{j=1}^\infty \delta_j \phi_j$
θ_j	Basis coefficients of f^*, i.e. $f^* = \sum_{j=1}^\infty \theta_j \phi_j$
d	Integer-valued truncation point
M	Diagonal matrix with $M = \operatorname{diag}(\mu_1, \ldots, \mu_d)$
Q	Diagonal matrix with $Q = (I_{d \times d} + \lambda M^{-1})^{\frac{1}{2}}$
Φ	$n \times d$ matrix with coordinates $\Phi_{ij} = \phi_j(x_i)$
$v^{\downarrow}$	Truncation of vector v. For $v = \sum_j \nu_j \phi_j \in \mathcal{H}$, defined as $v^{\downarrow} = \sum_{j=1}^d \nu_j \phi_j$; for $v \in \ell_2(\mathbb{N})$ defined as $v^{\downarrow} = (v_1, \ldots, v_d)$
$v^{\uparrow}$	Untruncated part of vector v, defined as $v^{\uparrow} = (v_{d+1}, v_{d+1}, \ldots)$
β_d	The tail sum $\sum_{j>d} \mu_j$
$\gamma(\lambda)$	The sum $\sum_{j=1}^\infty 1/(1 + \lambda/\mu_j)$
$b(n, d, k)$	The maximum $\max\{\sqrt{\max\{k, \log(d)\}}, \max\{k, \log(d)\}/n^{1/2-1/k}\}$

Table 4.2: Notation used in proofs

4.5 Proofs of technical results

4.5.1 Proof of Lemma 14

This section is devoted to the bias bound stated in Lemma 14. Let $X = \{x_i\}_{i=1}^n$ be shorthand for the design matrix, and define the error vector $\Delta = \widehat{f} - f^*$. By Jensen's inequality, we have $\|\mathbb{E}[\Delta]\|_2 \leq \mathbb{E}[\|\mathbb{E}[\Delta \mid X]\|_2]$, so it suffices to provide a bound on $\|\mathbb{E}[\Delta \mid X]\|_2$. Throughout this proof and the remainder of the chapter, we represent the kernel evaluator by the function ξ_x, where $\xi_x := K(x, \cdot)$ and $f(x) = \langle \xi_x, f \rangle$ for any $f \in \mathcal{H}$. Using this notation, the estimate $\widehat{f}$ minimizes the empirical objective

$$\frac{1}{n} \sum_{i=1}^n \left(\langle \xi_{x_i}, f \rangle_{\mathcal{H}} - y_i \right)^2 + \lambda \|f\|_{\mathcal{H}}^2 . \tag{4.24}$$

This objective is Fréchet differentiable, and as a consequence, the necessary and sufficient conditions for optimality [132] of $\widehat{f}$ are that

$$\frac{1}{n} \sum_{i=1}^n \xi_{x_i} (\langle \xi_{x_i}, \widehat{f} - f^* \rangle_{\mathcal{H}} - \varepsilon_i) + \lambda \widehat{f} = \frac{1}{n} \sum_{i=1}^n \xi_{x_i} (\langle \xi_{x_i}, \widehat{f} \rangle_{\mathcal{H}} - y_i) + \lambda \widehat{f} = 0, \tag{4.25}$$

where the last equation uses the fact that $y_i = \langle \xi_{x_i}, f^* \rangle_{\mathcal{H}} + \varepsilon_i$. Taking conditional expectations over the noise variables $\{\varepsilon_i\}_{i=1}^n$ with the design $X = \{x_i\}_{i=1}^n$ fixed, we find that

$$\frac{1}{n} \sum_{i=1}^n \xi_{x_i} \langle \xi_{x_i}, \mathbb{E}[\Delta \mid X] \rangle + \lambda \mathbb{E}[\widehat{f} \mid X] = 0.$$

Define the sample covariance operator $\widehat{\Sigma} := \frac{1}{n} \sum_{i=1}^n \xi_{x_i} \otimes \xi_{x_i}$. Adding and subtracting λf^* from the above equation yields

$$(\widehat{\Sigma} + \lambda I) \mathbb{E}[\Delta \mid X] = -\lambda f^*. \tag{4.26}$$

Consequently, we see we have $\|\mathbb{E}[\Delta \mid X]\|_{\mathcal{H}} \leq \|f^*\|_{\mathcal{H}}$, since $\widehat{\Sigma} \succeq 0$.

We now use a truncation argument to reduce the problem to a finite dimensional problem. To do so, we let $\delta \in \ell_2(\mathbb{N})$ denote the coefficients of $\mathbb{E}[\Delta \mid X]$ when expanded in the basis $\{\phi_j\}_{j=1}^\infty$:

$$\mathbb{E}[\Delta \mid X] = \sum_{j=1}^\infty \delta_j \phi_j, \quad \text{with } \delta_j = \langle \mathbb{E}[\Delta \mid X], \phi_j \rangle_{L^2(\mathbb{P})}. \tag{4.27}$$

For a fixed $d \in \mathbb{N}$, define the vectors $\delta^{\downarrow} := (\delta_1, \ldots, \delta_d)$ and $\delta^{\uparrow} := (\delta_{d+1}, \delta_{d+2}, \ldots)$ (we suppress dependence on d for convenience). By the orthonormality of the collection $\{\phi_j\}$, we have

$$\|\mathbb{E}[\Delta \mid X]\|_2^2 = \|\delta\|_2^2 = \|\delta^{\downarrow}\|_2^2 + \|\delta^{\uparrow}\|_2^2. \tag{4.28}$$

We control each of the elements of the sum (4.28) in turn.

Control of the term $\|\delta^{\uparrow}\|_2^2$*:* By definition, we have

$$\|\delta^{\uparrow}\|_2^2 = \frac{\mu_{d+1}}{\mu_{d+1}} \sum_{j=d+1}^\infty \delta_j^2 \leq \mu_{d+1} \sum_{j=d+1}^\infty \frac{\delta_j^2}{\mu_j} \overset{(i)}{\leq} \mu_{d+1} \|\mathbb{E}[\Delta \mid X]\|_{\mathcal{H}}^2 \overset{(ii)}{\leq} \mu_{d+1} \|f^*\|_{\mathcal{H}}^2, \tag{4.29}$$

where inequality (i) follows since $\|\mathbb{E}[\Delta \mid X]\|_{\mathcal{H}}^2 = \sum_{j=1}^\infty \frac{\delta_j^2}{\mu_j}$; and inequality (ii) follows from the bound $\|\mathbb{E}[\Delta \mid X]\|_{\mathcal{H}} \leq \|f^*\|_{\mathcal{H}}$, which is a consequence of equality (4.26).

Control of the term $\|\delta^{\downarrow}\|_2^2$*:* Let $(\theta_1, \theta_2, \ldots)$ be the coefficients of f^* in the basis $\{\phi_j\}$. In addition, define the matrices $\Phi \in \mathbb{R}^{n \times d}$ by

$$\Phi_{ij} = \phi_j(x_i) \quad \text{for } i \in \{1, \ldots, n\}, \text{ and } j \in \{1, \ldots, d\}$$

and $M = \operatorname{diag}(\mu_1, \ldots, \mu_d) \succ 0 \in \mathbb{R}^{d \times d}$. Lastly, define the tail error vector $v \in \mathbb{R}^n$ by

$$v_i := \sum_{j > d} \delta_j \phi_j(x_i) = \mathbb{E}[\Delta^{\uparrow}(x_i) \mid X].$$

Let $l \in \mathbb{N}$ be arbitrary. Computing the (Hilbert) inner product of the terms in equation (4.26) with ϕ_l, we obtain

$$-\lambda\frac{\theta_l}{\mu_l} = \langle \phi_l, -\lambda f^* \rangle = \left\langle \phi_l, (\widehat{\Sigma} + \lambda)\mathbb{E}[\Delta \mid X] \right\rangle$$
$$= \frac{1}{n}\sum_{i=1}^{n} \langle \phi_l, \xi_{x_i} \rangle \langle \xi_{x_i}, \mathbb{E}[\Delta \mid X] \rangle + \lambda \langle \phi_l, \mathbb{E}[\Delta \mid X] \rangle = \frac{1}{n}\sum_{i=1}^{n} \phi_l(x_i)\mathbb{E}[\Delta(x_i) \mid X] + \lambda\frac{\delta_l}{\mu_l}.$$

We can rewrite the final sum above using the fact that $\Delta = \Delta^{\downarrow} + \Delta^{\uparrow}$, which implies

$$\frac{1}{n}\sum_{i=1}^{n} \phi_l(x_i)\mathbb{E}[\Delta(x_i) \mid X] = \frac{1}{n}\sum_{i=1}^{n} \phi_l(x_i)\left(\sum_{j=1}^{d} \phi_j(x_i)\delta_j + \sum_{j>d} \phi_j(x_i)\delta_j\right)$$

Applying this equality for $l = 1, 2, \ldots, d$ yields

$$\left(\frac{1}{n}\Phi^T\Phi + \lambda M^{-1}\right)\delta^{\downarrow} = -\lambda M^{-1}\theta^{\downarrow} - \frac{1}{n}\Phi^T v. \tag{4.30}$$

We now show how the expression (4.30) gives us the desired bound in the lemma. By defininining the shorthand matrix $Q = (I + \lambda M^{-1})^{1/2}$, we have

$$\frac{1}{n}\Phi^T\Phi + \lambda M^{-1} = I + \lambda M^{-1} + \frac{1}{n}\Phi^T\Phi - I = Q\left(I + Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}\right)Q.$$

As a consequence, we can rewrite expression (4.30) to

$$\left(I + Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}\right)Q\delta^{\downarrow} = -\lambda Q^{-1}M^{-1}\theta^{\downarrow} - \frac{1}{n}Q^{-1}\Phi^T v. \tag{4.31}$$

We now present a lemma bounding the terms in equality (4.31) to control $\delta^{\downarrow}$.

Lemma 16. *The following bounds hold:*

$$\left\|\lambda Q^{-1}M^{-1}\theta^{\downarrow}\right\|_2^2 \le \lambda \left\|f^*\right\|_{\mathcal{H}}^2, \quad \textit{and} \tag{4.32a}$$

$$\mathbb{E}\left[\left\|\frac{1}{n}Q^{-1}\Phi^T v\right\|_2^2\right] \le \frac{\rho^4 \left\|f^*\right\|_{\mathcal{H}}^2 \operatorname{tr}(K)\beta_d}{\lambda}. \tag{4.32b}$$

Define the event $\mathcal{E} := \left\{\left\|\left|Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}\right|\right\| \le 1/2\right\}$. *Under Assumption F with moment bound* $\mathbb{E}[\phi_j(X)^{2k}] \le \rho^{2k}$, *there exists a universal constant* C *such that*

$$\mathbb{P}(\mathcal{E}^c) \le \left(\max\left\{\sqrt{k \vee \log(d)}, \frac{k \vee \log(d)}{n^{1/2-1/k}}\right\}\frac{C\rho^2\gamma(\lambda)}{\sqrt{n}}\right)^k. \tag{4.33}$$

We defer the proof of this lemma to Appendix 4.5.1.1.

Based on this lemma, we can now complete the proof. Whenever the event $\mathcal{E}$ holds, we know that $I + Q^{-1}((1/n)\Phi^T\Phi - I)Q^{-1} \succeq (1/2)I$. In particular, we have

$$\|Q\delta^{\downarrow}\|_2^2 \leq 4\left\|\lambda Q^{-1}M^{-1}\theta^{\downarrow} + (1/n)Q^{-1}\Phi^T v\right\|_2^2$$

on $\mathcal{E}$, by Eq. (4.31). Since $\|Q\delta^{\downarrow}\|_2^2 \geq \|\delta^{\downarrow}\|_2^2$, the above inequality implies that

$$\|\delta^{\downarrow}\|_2^2 \leq 4\left\|\lambda Q^{-1}M^{-1}\theta^{\downarrow} + (1/n)Q^{-1}\Phi^T v\right\|_2^2$$

Since $\mathcal{E}$ is X-measureable, we thus obtain

$$\begin{aligned}\mathbb{E}\left[\|\delta^{\downarrow}\|_2^2\right] &= \mathbb{E}\left[1_{(\mathcal{E})}\|\delta^{\downarrow}\|_2^2\right] + \mathbb{E}\left[1_{(\mathcal{E}^c)}\|\delta^{\downarrow}\|_2^2\right] \\ &\leq 4\mathbb{E}\left[1_{(\mathcal{E})}\left\|\lambda Q^{-1}M^{-1}\theta^{\downarrow} + (1/n)Q^{-1}\Phi^T v\right\|_2^2\right] + \mathbb{E}\left[1_{(\mathcal{E}^c)}\|\delta^{\downarrow}\|_2^2\right].\end{aligned}$$

Applying the bounds (4.32a) and (4.32b), along with the elementary inequality $(a+b)^2 \leq 2a^2 + 2b^2$, we have

$$\mathbb{E}\left[\|\delta^{\downarrow}\|_2^2\right] \leq 8\lambda\|f^*\|_{\mathcal{H}}^2 + \frac{8\rho^4\|f^*\|_{\mathcal{H}}^2\operatorname{tr}(K)\beta_d}{\lambda} + \mathbb{E}\left[1_{(\mathcal{E}^c)}\|\delta^{\downarrow}\|_2^2\right]. \tag{4.34}$$

Now we use the fact that by the gradient optimality condition (4.26),

$$\|\mathbb{E}[\Delta \mid X]\|_2^2 \leq \mu_0\|\mathbb{E}[\Delta \mid X]\|_{\mathcal{H}}^2 \leq \mu_0\|f^*\|_{\mathcal{H}}^2$$

Recalling the shorthand (4.5) for $b(n,d,k)$, we apply the bound (4.33) to see

$$\mathbb{E}\left[1_{(\mathcal{E}^c)}\|\delta^{\downarrow}\|_2^2\right] \leq \mathbb{P}(\mathcal{E}^c)\mu_0\|f^*\|_{\mathcal{H}}^2 \leq \left(\frac{Cb(n,d,k)\rho^2\gamma(\lambda)}{\sqrt{n}}\right)^k \mu_0\|f^*\|_{\mathcal{H}}^2$$

Combining this with the inequality (4.34), we obtain the desired statement of Lemma 14.

4.5.1.1 Proof of Lemma 16

Proof of bound (4.32a)*:* Beginning with the proof of the bound (4.32a), we have

$$\begin{aligned}\left\|Q^{-1}M^{-1}\theta^{\downarrow}\right\|_2^2 &= (\theta^{\downarrow})^T(M^2 + \lambda M)^{-1}\theta^{\downarrow} \\ &\leq (\theta^{\downarrow})^T(\lambda M)^{-1}\theta^{\downarrow} = \frac{1}{\lambda}(\theta^{\downarrow})^T M^{-1}\theta^{\downarrow} \leq \frac{1}{\lambda}\|f^*\|_{\mathcal{H}}^2.\end{aligned}$$

Multiplying both sides by λ^2 gives the result.

Proof of bound (4.32b)*:* Next we turn to the proof of the bound (4.32b). We begin by re-writing $Q^{-1}\Phi^T v$ as the product of two components:

$$\frac{1}{n}Q^{-1}\Phi^T v = (M + \lambda I)^{-1/2}\left(\frac{1}{n}M^{1/2}\Phi^T v\right). \tag{4.35}$$

The first matrix is a diagonal matrix whose operator norm is bounded:

$$\left\|\left|(M+\lambda I)^{-1/2}\right|\right\| = \max_{j\in[d]} \frac{1}{\sqrt{\mu_j+\lambda}} \leq \frac{1}{\sqrt{\lambda}}. \tag{4.36}$$

For the second factor in the product (4.35), the analysis is a little more complicated. Let $\Phi_\ell = (\phi_l(x_1), \ldots, \phi_l(x_n))$ be the ℓth column of Φ. In this case,

$$\left\|M^{1/2}\Phi^T v\right\|_2^2 = \sum_{\ell=1}^d \mu_\ell (\Phi_\ell^T v)^2 \leq \sum_{\ell=1}^d \mu_\ell \left\|\Phi_\ell\right\|_2^2 \left\|v\right\|_2^2, \tag{4.37}$$

using the Cauchy-Schwarz inequality. Taking expectations with respect to the design $\{x_i\}_{i=1}^n$ and applying Hölder's inequality yields

$$\mathbb{E}[\left\|\Phi_\ell\right\|_2^2 \left\|v\right\|_2^2] \leq \sqrt{\mathbb{E}[\left\|\Phi_\ell\right\|_2^4]}\sqrt{\mathbb{E}[\left\|v\right\|_2^4]}.$$

We bound each of the terms in this product in turn. For the first, we have

$$\mathbb{E}[\left\|\Phi_\ell\right\|_2^4] = \mathbb{E}\left[\left(\sum_{i=1}^n \phi_\ell^2(X_i)\right)^2\right] = \mathbb{E}\left[\sum_{i,j=1}^n \phi_\ell^2(X_i)\phi_\ell^2(X_j)\right] \leq n^2\mathbb{E}[\phi_\ell^4(X_1)] \leq n^2\rho^4$$

since the X_i are i.i.d., $\mathbb{E}[\phi_\ell^2(X_1)] \leq \sqrt{\mathbb{E}[\phi_\ell^4(X_1)]}$, and $\mathbb{E}[\phi_\ell^4(X_1)] \leq \rho^4$ by assumption. Turning to the term involving v, we have

$$v_i^2 = \left(\sum_{j>d} \delta_j \phi_j(x_i)\right)^2 \leq \left(\sum_{j>d} \frac{\delta_j^2}{\mu_j}\right)\left(\sum_{j>d} \mu_j \phi_j^2(x_i)\right)$$

by Cauchy-Schwarz. As a consequence, we find

$$\begin{aligned}\mathbb{E}[\left\|v\right\|_2^4] = \mathbb{E}\left[\left(n\frac{1}{n}\sum_{i=1}^n v_i^2\right)^2\right] \leq n^2\frac{1}{n}\sum_{i=1}^n \mathbb{E}[v_i^4] &\leq n\sum_{i=1}^n \mathbb{E}\left[\left(\sum_{j>d}\frac{\delta_j^2}{\mu_j}\right)^2\left(\sum_{j>d}\mu_j\phi_j^2(X_i)\right)^2\right] \\ &\leq n^2\mathbb{E}\left[\left\|\mathbb{E}[\Delta \mid X]\right\|_{\mathcal{H}}^4\left(\sum_{j>d}\mu_j\phi_j^2(X_1)\right)^2\right],\end{aligned}$$

since the X_i are i.i.d. Using the fact that $\left\|\mathbb{E}[\Delta \mid X]\right\|_{\mathcal{H}} \leq \left\|f^*\right\|_{\mathcal{H}}$, we expand the second square to find

$$\frac{1}{n^2}\mathbb{E}[\left\|v\right\|_2^4] \leq \left\|f^*\right\|_{\mathcal{H}}^4 \sum_{j,k>d} \mathbb{E}\left[\mu_j\mu_k\phi_j^2(X_1)\phi_k^2(X_1)\right] \leq \left\|f^*\right\|_{\mathcal{H}}^4 \rho^4 \sum_{j,k>d}\mu_j\mu_k = \left\|f^*\right\|_{\mathcal{H}}^4\rho^4\left(\sum_{j>d}\mu_j\right)^2.$$

Combining our bounds on $\left\|\Phi_\ell\right\|_2$ and $\left\|v\right\|_2$ with our initial bound (4.37), we obtain the inequality

$$\mathbb{E}\left[\left\|M^{1/2}\Phi^T v\right\|_2^2\right] \leq \sum_{l=1}^d \mu_\ell\sqrt{n^2\rho^4}\sqrt{n^2\left\|f^*\right\|_{\mathcal{H}}^4\rho^4\left(\sum_{j>d}\mu_j\right)^2} = n^2\rho^4\left\|f^*\right\|_{\mathcal{H}}^2\left(\sum_{j>d}\mu_j\right)\sum_{l=1}^d\mu_\ell.$$

Dividing by n^2, recalling the definition of $\beta_d = \sum_{j>d} \mu_j$, and noting that $\operatorname{tr}(K) \geq \sum_{l=1}^{d} \mu_\ell$ shows that

$$\mathbb{E}\left[\left\|\frac{1}{n} M^{1/2} \Phi^T v\right\|_2^2\right] \leq \rho^4 \, \|f^*\|_{\mathcal{H}}^2 \, \beta_d \operatorname{tr}(K).$$

Combining this inequality with our expansion (4.35) and the bound (4.36) yields the claim (4.32b).

Proof of bound (4.33)*:* We consider the expectation of the norm of $Q^{-1}(\frac{1}{n}\Phi^T\Phi - I)Q^{-1}$. For each $i \in [n]$, $\pi_i := (\phi_1(x_i), \ldots, \phi_d(x_i))^T \in \mathbb{R}^d$, then π_i^T is the i-th row of the matrix $\Phi \in \mathbb{R}^{n \times d}$. Then we know that

$$Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1} = \frac{1}{n}\sum_{i=1}^{n} Q^{-1}(\pi_i \pi_i^T - I)Q^{-1}.$$

Define the sequence of matrices

$$A_i := Q^{-1}(\pi_i \pi_i^T - I)Q^{-1}$$

Then the matrices $A_i = A_i^T \in \mathbb{R}^{d \times d}$. Note that $\mathbb{E}[A_i] = 0$ and let ε_i be i.i.d. $\{-1, 1\}$-valued Rademacher random variables. Applying a standard symmetrization argument [116], we find that for any $k \geq 1$, we have

$$\mathbb{E}\left[\left|\!\left|\!\left|Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}\right|\!\right|\!\right|^k\right] = \mathbb{E}\left[\left|\!\left|\!\left|\frac{1}{n}\sum_{i=1}^{n} A_i\right|\!\right|\!\right|^k\right] \leq 2^k \mathbb{E}\left[\left|\!\left|\!\left|\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i A_i\right|\!\right|\!\right|^k\right]. \tag{4.38}$$

Lemma 17. *The quantity* $\mathbb{E}\left[\left|\!\left|\!\left|\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i A_i\right|\!\right|\!\right|^k\right]^{1/k}$ *is upper bounded by*

$$\sqrt{e(k \vee 2\log(d))}\frac{\rho^2 \sum_{j=1}^{d} \frac{1}{1+\lambda/\mu_j}}{\sqrt{n}} + \frac{4e(k \vee 2\log(d))}{n^{1-1/k}}\left(\sum_{j=1}^{d} \frac{\rho^2}{1+\lambda/\mu_j}\right). \tag{4.39}$$

We take this lemma as given for the moment, returning to prove it shortly. Recall the definition of the constant $\gamma(\lambda) = \sum_{j=1}^{\infty} 1/(1+\lambda/\mu_j) \geq \sum_{j=1}^{d} 1/(1+\lambda/\mu_j)$. Then using our symmetrization inequality (4.38), we have

$$\begin{aligned}
&\mathbb{E}\left[\left|\!\left|\!\left|Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}\right|\!\right|\!\right|^k\right] \\
&\leq 2^k \left(\sqrt{e(k \vee \log(d))}\frac{\rho^2\gamma(\lambda)}{\sqrt{n}} + \frac{4e(k \vee 2\log(d))}{n^{1-1/k}}\rho^2\gamma(\lambda)\right)^k \\
&\leq \max\left\{\sqrt{k \vee \log(d)}, \frac{k \vee \log(d)}{n^{1/2-1/k}}\right\}^k \left(\frac{C\rho^2\gamma(\lambda)}{\sqrt{n}}\right)^k,
\end{aligned} \tag{4.40}$$

where C is a numerical constant. By definition of the event $\mathcal{E}$, we see by Markov's inequality that for any $k \in \mathbb{R}, k \geq 1$,

$$\mathbb{P}(\mathcal{E}^c) \leq \frac{\mathbb{E}\left[\left\|\left\|Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)\right\|\right\|^k\right]}{2^{-k}} \leq \max\left\{\sqrt{k \vee \log(d)}, \frac{k \vee \log(d)}{n^{1/2-1/k}}\right\}^k \left(\frac{2C\rho^2\gamma(\lambda))}{\sqrt{n}}\right)^k.$$

This completes the proof of the bound (4.33).

It remains to prove Lemma 17, for which we make use of the following result, due to Chen et al. [45, Theorem A.1(2)].

Lemma 18. *Let $X_i \in \mathbb{R}^{d\times d}$ be independent symmetrically distributed Hermitian matrices. Then*

$$\mathbb{E}\left[\left\|\sum_{i=1}^n X_i\right\|^k\right]^{1/k} \leq \sqrt{e(k \vee 2\log d)}\left\|\sum_{i=1}^n \mathbb{E}[X_i^2]\right\|^{1/2} + 2e(k \vee 2\log d)\left(\mathbb{E}[\max_i \|X_i\|^k]\right)^{1/k}. \tag{4.41}$$

The proof of Lemma 17 is based on applying this inequality with $X_i = \varepsilon_i A_i/n$, and then bounding the two terms on the right-hand side of inequality (4.41).

We begin with the first term. Note that for any symmetric matrix Z, we have the matrix inequalities $0 \preceq \mathbb{E}[(Z - \mathbb{E}[Z])^2] = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 \preceq \mathbb{E}[Z^2]$, so

$$\mathbb{E}[A_i^2] = \mathbb{E}[Q^{-1}(\pi_i\pi_i^T - I)Q^{-2}(\pi_i\pi_i^T - I)Q^{-1}] \preceq \mathbb{E}[Q^{-1}\pi_i\pi_i^T Q^{-2}\pi_i\pi_i^T Q^{-1}].$$

Instead of computing these moments directly, we provide bounds on their norms. Since $\pi_i\pi_i^T$ is rank one and Q is diagonal, we have

$$\left\|Q^{-1}\pi_i\pi_i^T Q^{-1}\right\| = \pi_i^T(I + \lambda M^{-1})^{-1}\pi_i = \sum_{j=1}^d \frac{\phi_j(x_i)^2}{1 + \lambda/\mu_j}.$$

We also note that, for any $k \in \mathbb{R}, k \geq 1$, convexity implies that

$$\begin{aligned}\left(\sum_{j=1}^d \frac{\phi_j(x_i)^2}{1+\lambda/\mu_j}\right)^k &= \left(\frac{\sum_{l=1}^d 1/(1+\lambda/\mu_\ell)}{\sum_{l=1}^d 1/(1+\lambda/\mu_\ell)}\sum_{j=1}^d \frac{\phi_j(x_i)^2}{1+\lambda/\mu_j}\right)^k \\ &\leq \left(\sum_{l=1}^d \frac{1}{1+\lambda/\mu_\ell}\right)^k \frac{1}{\sum_{l=1}^d 1/(1+\lambda/\mu_\ell)}\sum_{j=1}^d \frac{\phi_j(x_i)^{2k}}{1+\lambda/\mu_j},\end{aligned}$$

so if $\mathbb{E}[\phi_j(X_i)^{2k}] \leq \rho^{2k}$, we obtain

$$\mathbb{E}\left[\left(\sum_{j=1}^d \frac{\phi_j(x_i)^2}{1+\lambda/\mu_j}\right)^k\right] \leq \left(\sum_{j=1}^d \frac{1}{1+\lambda/\mu_j}\right)^k \rho^{2k}. \tag{4.42}$$

The sub-multiplicativity of matrix norms implies $\left|\left|\left|(Q^{-1}\pi_i\pi_i^T Q^{-1})^2\right|\right|\right| \leq \left|\left|\left|Q^{-1}\pi_i\pi_i^T Q^{-1}\right|\right|\right|^2$, and consequently we have

$$\mathbb{E}\left[\left|\left|\left|(Q^{-1}\pi_i\pi_i^T Q^{-1})^2\right|\right|\right|\right] \leq \mathbb{E}\left[\left(\pi_i^T(I+\lambda M^{-1})^{-1}\pi_i\right)^2\right] \leq \rho^4\left(\sum_{j=1}^{d}\frac{1}{1+\lambda/\mu_j}\right)^2,$$

where the final step follows from inequality (4.42). Combined with first term on the right-hand side of Lemma 18, we have thus obtained the first term on the right-hand side of expression (4.39).

We now turn to the second term in expression (4.39). For real $k \geq 1$, we have

$$\mathbb{E}[\max_i \left|\left|\left|\varepsilon_i A_i/n\right|\right|\right|^k] = \frac{1}{n^k}\mathbb{E}[\max_i \left|\left|\left|A_i\right|\right|\right|^k] \leq \frac{1}{n^k}\sum_{i=1}^{n}\mathbb{E}[\left|\left|\left|A_i\right|\right|\right|^k]$$

Since norms are sub-additive, we find that

$$\left|\left|\left|A_i\right|\right|\right|^k \leq 2^{k-1}\left(\sum_{j=1}^{d}\frac{\phi_j(x_i)^2}{1+\lambda/\mu_j}\right)^k + 2^{k-1}\left|\left|\left|Q^{-2}\right|\right|\right|^k = 2^{k-1}\left(\sum_{j=1}^{d}\frac{\phi_j(x_i)^2}{1+\lambda/\mu_j}\right)^k + 2^{k-1}\left(\frac{1}{1+\lambda/\mu_1}\right)^k.$$

Since $\rho \geq 1$ (recall that the ϕ_j are an orthonormal basis), we apply inequality (4.42), to find that

$$\mathbb{E}[\max_i \left|\left|\left|\varepsilon_i A_i/n\right|\right|\right|^k] \leq \frac{1}{n^{k-1}}\left[2^{k-1}\left(\sum_{j=1}^{d}\frac{1}{1+\lambda/\mu_j}\right)^k\rho^{2k} + 2^{k-1}\left(\frac{1}{1+\lambda/\mu_1}\right)^k\rho^{2k}\right].$$

Taking kth roots yields the second term in the expression (4.39).

4.5.2 Proof of Lemma 15

This proof follows an outline similar to Lemma 14. We begin with a simple bound on $\|\Delta\|_{\mathcal{H}}$:

Lemma 19. *Under Assumption G, we have* $\mathbb{E}[\|\Delta\|_{\mathcal{H}}^2 \mid X] \leq 2\sigma^2/\lambda + 4\|f^*\|_{\mathcal{H}}^2$.

Proof We have

$$\begin{aligned}\lambda\,\mathbb{E}[\,\|\widehat{f}\|_{\mathcal{H}}^2 \mid \{x_i\}_{i=1}^n] &\leq \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(\widehat{f}(x_i)-f^*(x_i)-\varepsilon_i)^2 + \lambda\|\widehat{f}\|_{\mathcal{H}}^2 \mid \{x_i\}_{i=1}^n\right]\\ &\overset{(i)}{\leq} \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\varepsilon_i^2 \mid x_i] + \lambda\,\|f^*\|_{\mathcal{H}}^2\\ &\overset{(ii)}{\leq} \sigma^2 + \lambda\,\|f^*\|_{\mathcal{H}}^2,\end{aligned}$$

where inequality (i) follows since $\widehat{f}$ minimizes the objective function (4.1); and inequality (ii) uses the fact that $\mathbb{E}[\varepsilon_i^2 \mid x_i] \leq \sigma^2$. Applying the triangle inequality to $\|\Delta\|_{\mathcal{H}}$ along with the elementary inequality $(a+b)^2 \leq 2a^2 + 2b^2$, we find that

$$\mathbb{E}[\|\Delta\|_{\mathcal{H}}^2 \mid \{x_i\}_{i=1}^n] \leq 2\|f^*\|_{\mathcal{H}}^2 + 2\mathbb{E}[\|\widehat{f}\|_{\mathcal{H}}^2 \mid \{x_i\}_{i=1}^n] \;\leq\; \frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2,$$

which completes the proof. □

With Lemma 19 in place, we now proceed to the proof of the theorem proper. Recall from Lemma 14 the optimality condition

$$\frac{1}{n}\sum_{i=1}^n \xi_{x_i}(\langle \xi_{x_i}, \widehat{f} - f^* \rangle - \varepsilon_i) + \lambda \widehat{f} = 0. \tag{4.43}$$

Now, let $\delta \in \ell_2(\mathbb{N})$ be the expansion of the error Δ in the basis $\{\phi_j\}$, so that $\Delta = \sum_{j=1}^\infty \delta_j \phi_j$, and (again, as in Lemma 14), we choose $d \in \mathbb{N}$ and truncate Δ via

$$\Delta^{\downarrow} := \sum_{j=1}^d \delta_j \phi_j \quad \text{and} \quad \Delta^{\uparrow} := \Delta - \Delta^{\downarrow} = \sum_{j>d} \delta_j \phi_j.$$

Let $\delta^{\downarrow} \in \mathbb{R}^d$ and $\delta^{\uparrow}$ denote the corresponding vectors for the above. As a consequence of the orthonormality of the basis functions, we have

$$\mathbb{E}[\|\Delta\|_2^2] = \mathbb{E}[\|\Delta^{\downarrow}\|_2^2] + \mathbb{E}[\|\Delta^{\uparrow}\|_2^2] = \mathbb{E}[\|\delta^{\downarrow}\|_2^2] + \mathbb{E}[\|\delta^{\uparrow}\|_2^2]. \tag{4.44}$$

We bound each of the terms (4.44) in turn.

By Lemma 19, the second term is upper bounded as

$$\mathbb{E}[\|\Delta^{\uparrow}\|_2^2] = \sum_{j>d} \mathbb{E}[\delta_j^2] \leq \sum_{j>d} \frac{\mu_{d+1}}{\mu_j} \mathbb{E}[\delta_j^2] = \mu_{d+1}\mathbb{E}[\|\Delta^{\uparrow}\|_{\mathcal{H}}^2] \leq \mu_{d+1}\left(\frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2\right). \tag{4.45}$$

The remainder of the proof is devoted the bounding the term $\mathbb{E}[\|\Delta^{\downarrow}\|_2^2]$ in the decomposition (4.44). By taking the Hilbert inner product of ϕ_k with the optimality condition (4.43), we find as in our derivation of the matrix equation (4.30) that for each $k \in \{1, \ldots, d\}$

$$\frac{1}{n}\sum_{i=1}^n \sum_{j=1}^d \phi_k(x_i)\phi_j(x_i)\delta_j + \frac{1}{n}\sum_{i=1}^n \phi_k(x_i)(\Delta^{\uparrow}(x_i) - \varepsilon_i) + \lambda\frac{\delta_k}{\mu_k} = 0.$$

Given the expansion $f^* = \sum_{j=1}^\infty \theta_j \phi_j$, define the tail error vector $v \in \mathbb{R}^n$ by $v_i = \sum_{j>d} \delta_j \phi_j(x_i)$, and recall the definition of the eigenvalue matrix $M = \operatorname{diag}(\mu_1, \ldots, \mu_d) \in \mathbb{R}^{d \times d}$. Given the matrix Φ defined by its coordinates $\Phi_{ij} = \phi_j(x_i)$, we have

$$\left(\frac{1}{n}\Phi^T\Phi + \lambda M^{-1}\right)\delta^{\downarrow} = -\lambda M^{-1}\theta^{\downarrow} - \frac{1}{n}\Phi^T v + \frac{1}{n}\Phi^T\varepsilon. \tag{4.46}$$

As in the proof of Lemma 14, we find that

$$\left(I + Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}\right)Q\delta^{\downarrow} = -\lambda Q^{-1}M^{-1}\theta^{\downarrow} - \frac{1}{n}Q^{-1}\Phi^T v + \frac{1}{n}Q^{-1}\Phi^T\varepsilon, \tag{4.47}$$

where we recall that $Q = (I + \lambda M^{-1})^{1/2}$.

We now recall the bounds (4.32a) and (4.33) from Lemma 16, as well as the previously defined event $\mathcal{E} := \{|||Q^{-1}\left(\frac{1}{n}\Phi^T\Phi - I\right)Q^{-1}||| \leq 1/2\}$. When $\mathcal{E}$ occurs, the expression (4.47) implies the inequality

$$\|\Delta^{\downarrow}\|_2^2 \leq \|Q\delta^{\downarrow}\|_2^2 \leq 4\left\|-\lambda Q^{-1}M^{-1}\theta^{\downarrow} - (1/n)Q^{-1}\Phi^T v + (1/n)Q^{-1}\Phi^T\varepsilon\right\|_2^2.$$

When $\mathcal{E}$ fails to hold, Lemma 19 may still be applied since $\mathcal{E}$ is measureable with respect to $\{x_i\}_{i=1}^n$. Doing so yields

$$\begin{aligned}
&\mathbb{E}[\|\Delta^{\downarrow}\|_2^2] = \mathbb{E}[1_{(\mathcal{E})}\|\Delta^{\downarrow}\|_2^2] + \mathbb{E}[1_{(\mathcal{E}^c)}\|\Delta^{\downarrow}\|_2^2] \\
&\leq 4\mathbb{E}\left[\left\|-\lambda Q^{-1}M^{-1}\theta^{\downarrow} - (1/n)Q^{-1}\Phi^T v + (1/n)Q^{-1}\Phi^T\varepsilon\right\|_2^2\right] + \mathbb{E}\left[1_{(\mathcal{E}^c)}\mathbb{E}[\|\Delta^{\downarrow}\|_2^2 \mid \{x_i\}_{i=1}^n]\right] \\
&\leq 4\mathbb{E}\left[\left\|\lambda Q^{-1}M^{-1}\theta^{\downarrow} + \frac{1}{n}Q^{-1}\Phi^T v - \frac{1}{n}Q^{-1}\Phi^T\varepsilon\right\|_2^2\right] + \mathbb{P}(\mathcal{E}^c)\left(\frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2\right).
\end{aligned} \tag{4.48}$$

Since the bound (4.33) still holds, it remains to provide a bound on the first term in the expression (4.48).

As in the proof of Lemma 14, we have $\|\lambda Q^{-1}M^{-1}\theta^{\downarrow}\|_2^2 \leq \lambda\|f^*\|_{\mathcal{H}}^2$ via the bound (4.32a). Turning to the second term inside the norm, we claim that, under the conditions of Lemma 15, the following bound holds:

$$\mathbb{E}\left[\left\|(1/n)Q^{-1}\Phi^T v\right\|_2^2\right] \leq \frac{\rho^4\operatorname{tr}(K)\beta_d(2\sigma^2/\lambda + 4\|f^*\|_{\mathcal{H}}^2)}{\lambda}. \tag{4.49}$$

This claim is an analogue of our earlier bound (4.32b), and we prove it shortly. Lastly, we bound the norm of $Q^{-1}\Phi^T\varepsilon/n$. Noting that the diagonal entries of Q^{-1} are $1/\sqrt{1+\lambda/\mu_j}$, we have

$$\mathbb{E}\left[\left\|Q^{-1}\Phi^T\varepsilon\right\|_2^2\right] = \sum_{j=1}^{d}\sum_{i=1}^{n}\frac{1}{1+\lambda/\mu_j}\mathbb{E}[\phi_j^2(X_i)\varepsilon_i^2]$$

Since $\mathbb{E}[\phi_j^2(X_i)\varepsilon_i^2] = \mathbb{E}[\phi_j^2(X_i)\mathbb{E}[\varepsilon_i^2 \mid X_i]] \leq \sigma^2$ by assumption, we have the inequality

$$\mathbb{E}\left[\left\|(1/n)Q^{-1}\Phi^T\varepsilon\right\|_2^2\right] \leq \frac{\sigma^2}{n}\sum_{j=1}^{d}\frac{1}{1+\lambda/\mu_j}.$$

The last sum is bounded by $(\sigma^2/n)\gamma(\lambda)$. Applying the inequality $(a+b+c)^2 \leq 3a^2+3b^2+3c^2$ to inequality (4.48), we obtain

$$\mathbb{E}\left[\|\Delta^{\downarrow}\|_2^2\right] \leq 12\lambda\|f^*\|_{\mathcal{H}}^2 + \frac{12\sigma^2\gamma(\lambda)}{n} + \left(\frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2\right)\left(\frac{12\rho^4\operatorname{tr}(K)\beta_d}{\lambda} + \mathbb{P}(\mathcal{E}^c)\right).$$

Applying the bound (4.33) to control $\mathbb{P}(\mathcal{E}^c)$ and bounding $\mathbb{E}[\|\Delta^{\uparrow}\|_2^2]$ using inequality (4.45) completes the proof of the lemma.

It remains to prove bound (4.49). Recalling the inequality (4.36), we see that

$$\left\|(1/n)Q^{-1}\Phi^T v\right\|_2^2 \leq \left|\left|\left|Q^{-1}M^{-1/2}\right|\right|\right|^2 \left\|(1/n)M^{1/2}\Phi^T v\right\|_2^2 \leq \frac{1}{\lambda}\left\|(1/n)M^{1/2}\Phi^T v\right\|_2^2. \tag{4.50}$$

Let Φ_ℓ denote the ℓth column of the matrix Φ. Taking expectations yields

$$\mathbb{E}\left[\left\|M^{1/2}\Phi^T v\right\|_2^2\right] = \sum_{l=1}^{d}\mu_\ell \mathbb{E}[\langle \Phi_\ell, v\rangle^2] \leq \sum_{l=1}^{d}\mu_\ell \mathbb{E}\left[\|\Phi_\ell\|_2^2\,\|v\|_2^2\right] = \sum_{l=1}^{d}\mu_\ell \mathbb{E}\left[\|\Phi_\ell\|_2^2\,\mathbb{E}\left[\|v\|_2^2 \mid X\right]\right].$$

Now consider the inner expectation. Applying the Cauchy-Schwarz inequality as in the proof of the bound (4.32b), we have

$$\|v\|_2^2 = \sum_{i=1}^{n} v_i^2 \leq \sum_{i=1}^{n}\left(\sum_{j>d}\frac{\delta_j^2}{\mu_j}\right)\left(\sum_{j>d}\mu_j\phi_j^2(X_i)\right).$$

Notably, the second term is X-measureable, and the first is bounded by $\|\Delta^{\uparrow}\|_{\mathcal{H}}^2 \leq \|\Delta\|_{\mathcal{H}}^2$. We thus obtain

$$\mathbb{E}\left[\left\|M^{1/2}\Phi^T v\right\|_2^2\right] \leq \sum_{i=1}^{n}\sum_{l=1}^{d}\mu_\ell \mathbb{E}\left[\|\Phi_\ell\|_2^2\left(\sum_{j>d}\mu_j\phi_j^2(X_i)\right)\mathbb{E}[\|\Delta\|_{\mathcal{H}}^2 \mid X]\right]. \tag{4.51}$$

Lemma 19 provides the bound $2\sigma^2/\lambda + 4\|f^*\|_{\mathcal{H}}^2$ on the final (inner) expectation.

The remainder of the argument proceeds precisely as in the bound (4.32b). We have

$$\mathbb{E}[\|\Phi_\ell\|_2^2\,\phi_j(X_i)^2] \leq n\rho^4$$

by the moment assumptions on ϕ_j, and thus

$$\mathbb{E}\left[\left\|M^{1/2}\Phi^T v\right\|_2^2\right] \leq \sum_{l=1}^{d}\sum_{j>d}\mu_\ell\mu_j n^2\rho^4\left(\frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2\right) \leq n^2\rho^4\beta_d\,\mathrm{tr}(K)\left(\frac{2\sigma^2}{\lambda} + 4\|f^*\|_{\mathcal{H}}^2\right).$$

Dividing by λn^2 completes the proof.

Chapter 5

Distributed optimization of self-concordant loss

In this chapter, we study a more general setup of empirical risk minimization — when the loss function is not necessarily strongly convex, but regularized by a squared ℓ_2-norm. The regularization parameter may diminish to zero as a function of the data size. Recall the notations introduced in Chapter 2. Our distributed computing system consists of m machines, and each has access to n samples $z_{i,1}, \ldots, z_{i,n}$, for $i = 1, \ldots, m$. Then each machine can evaluate a local empirical loss function

$$f_i(w) =: \frac{1}{n} \sum_{j=1}^{n} \phi(w, z_{i,j}), \qquad i = 1, \ldots, m.$$

Our goal is to minimize the overall empirical loss defined with all mn samples:

$$f(w) =: \frac{1}{m} \sum_{i=1}^{m} f_i(w) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \phi(w, z_{i,j}). \tag{5.1}$$

For stability and generalization purposes, we often add a regularization term $(\lambda/2)\|w\|_2^2$ to make the empirical loss function strongly convex. More specifically, we modify the definition of $f_i(w)$ as

$$f_i(w) =: \frac{1}{n} \sum_{j=1}^{n} \phi(w, z_{i,j}) + \frac{\lambda}{2}\|w\|_2^2, \qquad i = 1, \ldots, m. \tag{5.2}$$

Our goal is to develop *communication-efficient* distributed algorithms, which try to use a minimal number of communication rounds to reach certain precision in minimizing $f(w)$.

5.1 Communication efficiency of distributed convex optimization algorithms

We assume that each communication round requires only simple map-reduce type of operations, such as broadcasting a vector in $\mathbb{R}^d$ to the m machines and computing the sum or average of m vectors in $\mathbb{R}^d$. Typically, if a distributed iterative algorithm takes T iterations to converge, then it communicates at least T rounds (usually one or two communication rounds per iteration). Therefore, we can measure the communication efficiency of a distributed algorithm by its iteration complexity $T(\epsilon)$, which is the number of iterations required by the algorithm to find a solution w_T such that $f(w_T) - f(w_\star) \leq \epsilon$.

For a concrete discussion, we make the following assumption:

Assumption H. *The function $f : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable, and there exist constants $L \geq \lambda > 0$ such that*

$$\lambda I \preceq f''(w) \preceq LI, \qquad \forall\, w \in \mathbb{R}^d,$$

where $f''(w)$ denotes the Hessian of f at w, and I is the $d \times d$ identity matrix.

Functions that satisfy Assumption H are often called L-smooth and λ-strongly convex. The value $\kappa = L/\lambda \geq 1$ is called the *condition number* of f, which is a key quantity in characterizing the complexity of iterative algorithms. We focus on ill-conditioned cases where $\kappa \gg 1$.

A straightforward approach for minimizing $f(w)$ is distributed implementation of the classical gradient descent method. More specifically, at each iteration k, each machine computes the local gradient $f_i'(w_k) \in \mathbb{R}^d$ and sends it to a master node to compute $f'(w_k) = (1/m)\sum_{i=1}^m f_i'(w_k)$. The master node takes a gradient step to compute w_{k+1}, and broadcasts it to each machine for the next iteration. The iteration complexity of this method is the same as the classical gradient method: $\mathcal{O}(\kappa \log(1/\epsilon))$, which is linear in the condition number κ (*e.g.*, [154]). If we use accelerated gradient methods [154, 155, 126], then the iteration complexity can be improved to $\mathcal{O}(\sqrt{\kappa}\log(1/\epsilon))$.

Another popular technique for distributed optimization is to use the alternating direction method of multipliers (ADMM); see, e.g., [34, Section 8]. Under the assumption that each local function f_i is L-smooth and λ-strongly convex, the ADMM approach can achieve linear convergence, and the best known complexity is $\mathcal{O}(\sqrt{\kappa}\log(1/\epsilon))$ [60]. This turns out to be the same order as for accelerated gradient methods. In this case, ADMM can actually be considered as an accelerated primal-dual first-order method; see the discussions in [42, Section 4].

The polynomial dependence of the iteration complexity on the condition number can be unsatifactory. For machine learning applications, both the precision ϵ and the regularization parameter λ should decrease while the overall sample size mn increases, typically on the order of $\Theta(1/\sqrt{mn})$ (*e.g.*, [32, 187]). This translates into the condition number κ being $\Theta(\sqrt{mn})$. In this case, the iteration complexity, and thus the number of communication

rounds, scales as $(mn)^{1/4}$ for both accelerated gradient methods and ADMM (with careful tuning of the penalty parameter). This suggests that the number of communication rounds grows with the total sample size.

Despite the rich literature on distributed optimization (e.g., [22, 168, 34, 4, 64, 59, 172, 231, 188]), most algorithms involve high communication cost. In particular, their iteration complexity have similar or worse dependency on the condition number as the methods discussed above. It can be argued that the iteration complexity $\mathcal{O}(\sqrt{\kappa}\log(1/\epsilon))$ cannot be improved in general for distributed first-order methods — after all, it is optimal for centralized first-order methods under the same assumption that $f(w)$ is L-smooth and λ-strongly convex [153, 154]. Thus in order to obtain better communication efficiency, we need to look into further problem structure and/or alternative optimization methods. And we need both in this chapter.

First, we note that the above discussion on iteration complexity does not exploit the fact that each function f_i is generated by, or can be considered as, SAA of a stochastic optimization problem. Since the data $z_{i,j}$ are i.i.d. samples from a common distribution, the local empirical loss functions $f_i(w) = (1/n)\sum_{j=1}^{n}\phi(w, z_{i,j})$ will be similar to each other if the local sample size n is large. Under this assumption, Zhang et al. [231] studied a one-shot averaging scheme that approximates the minimizer of function f by simply averaging the minimizers of f_i. For a fixed condition number, the one-shot approach is communication efficient because it achieves optimal dependence on the overall sample size mn (in the sense of statistical lower bounds). But their conclusion doesn't allow the regularization parameter λ to decrease to zero as n goes to infinity (see discussions in [190]).

Exploiting the stochastic nature alone seems not enough to overcome ill-conditioning in the regime of first-order methods. This motivates the development of distributed second-order methods. Recently, Shamir et al. [190] proposed a distributed approximate Newton-type (DANE) method. Their method takes advantage of the fact that, under the stochastic assumptions of SAA, the Hessians $f_1'', f_2'', \ldots, f_m''$ are similar to each other. For quadratic loss functions, DANE is shown to converge in $\widetilde{\mathcal{O}}\big((L/\lambda)^2 n^{-1}\log(1/\epsilon)\big)$ iterations with high probability, where the notation $\widetilde{\mathcal{O}}(\cdot)$ hides additional logarithmic factors involving m and d. If $\lambda \sim 1/\sqrt{mn}$ as in machine learning applications, then the iteration complexity becomes $\widetilde{\mathcal{O}}(m\log(1/\epsilon))$, which scales linearly with the number of machines m, not the total sample size mn. However, the analysis in [190] does not guarantee that DANE has the same convergence rate on non-quadratic functions.

5.2 Outline of our approach

We propose a communication-efficient distributed second-order method for minimizing the overall empirical loss $f(w)$ defined in (5.1). Our method is based on an *inexact* damped Newton method. Assume $f(w)$ is strongly convex and has continuous second derivatives. In the *exact* damped Newton method (e.g., [154, Section 4.1.5]), we first choose an initial point

$w_0 \in \mathbb{R}^d$, and then repeat

$$w_{k+1} = w_k - \frac{1}{1+\delta(w_k)}\Delta w_k, \qquad k = 0, 1, 2, \ldots, \tag{5.3}$$

where Δw_k and $\delta(w_k)$ are the Newton step and the Newton decrement, respectively, defined as

$$\begin{aligned} \Delta w_k &= [f''(w_k)]^{-1} f'(w_k) \ , \\ \delta(w_k) &= \sqrt{f'(w_k)^T [f''(w_k)]^{-1} f'(w_k)} = \sqrt{(\Delta w_k)^T f''(w_k) \Delta w_k} \ . \end{aligned} \tag{5.4}$$

Since f is the average of $f_1, \ldots, f_m$, its gradient and Hessian can be written as

$$f'(w_k) = \frac{1}{m}\sum_{i=1}^{m} f_i'(w_k), \qquad f''(w_k) = \frac{1}{m}\sum_{i=1}^{m} f_i''(w_k). \tag{5.5}$$

In order to compute Δw_k in a distributed setting, the naive approach would require all the machines to send their gradients and Hessians to a master node (say machine 1). However, the task of transmitting the Hessians (which are $d \times d$ matrices) can be prohibitive for large dimensions d. A better alternative is to use the conjugate gradient (CG) method to compute Δw_k as the solution to a linear system $f''(w_k)\Delta w_k = f'(w_k)$. Each iteration of the CG method requires a matrix-vector product of the form

$$f''(w_k)v = \frac{1}{m}\sum_{i=1}^{m} f_i''(w_k)v,$$

where v is some vector in $\mathbb{R}^d$. More specifically, the master node can broadcast the vector v to each machine, each machine computes $f_i''(w_k)v \in \mathbb{R}^d$ locally and sends it back to the master node, which then forms the average $f''(w_k)v$ and performs the CG update. Due to the iterative nature of the CG method, we can only compute the Newton direction and Newton decrement approximately, especially with limited number of communication rounds.

The overall method has two levels of loops: the outer-loop of the damped Newton method, and the inner loop of the CG method for computing the inexact Newton steps. A similar approach (using a distributed truncated Newton method) was proposed in [237, 125] for ERM of linear predictors, and it was reported to perform very well in practice. However, the total number of CG iterations (each takes a round of communication) may still be high.

First, consider the outer loop complexity. It is well-known that Newton-type methods have asymptotic superlinear convergence. However, in classical analysis of Newton's method (e.g., [33, Section 9.5.3]), the number of steps needed to reach the superlinear convergence zone still depends on the condition number; more specifically, it scales quadratically in κ. To solve this problem, we resort to the machinery of self-concordant functions [157, 154]. For self-concordant empirical losses, we show that the iteration complexity of the inexact damped Newton method has a much weaker dependence on the condition number.

Second, consider the inner loop complexity. The convergence rate of the CG method also depends on the condition number κ: it takes $\mathcal{O}(\sqrt{\kappa}\log(1/\varepsilon))$ CG iterations to compute an ε-precise Newton step. Thus we arrive at the dilemma that the overall complexity of the CG-powered inexact Newton method is no better than accelerated gradient methods or ADMM. To overcome this difficulty, we exploit the stochastic nature of the problem and propose to use a preconditioned CG (PCG) method for solving the Newton system. Roughly speaking, if the local Hessians $f_1''(w_k), \ldots, f_m''(w_k)$ are "similar" to each other, then we can use any local Hessian $f_i''(w_k)$ as a preconditioner. Without loss of generality, let $P = f_1''(w_k) + \mu I$, where μ is an estimate of the spectral norm $\|f_1''(w_k) - f''(w_k)\|_2$. Then we use CG to solve the pre-conditioned linear system

$$P^{-1}f''(w_k)\Delta w_k = P^{-1}f'(w_k),$$

where the preconditioning (multiplication by P^{-1}) can be computed locally at machine 1 (the master node). The convergence rate of PCG depends on the condition number of the matrix $P^{-1}f''(w_k)$, which is close to 1 if the spectral norm $\|f_1''(w_k) - f''(w_k)\|_2$ is small.

To exactly characterize the similarity between $f_1''(w_k)$ and $f''(w_k)$, we rely on stochastic analysis in the framework of SAA or ERM. We show that with high probability, $\|f_1''(w_k) - f''(w_k)\|_2$ decreases as $\widetilde{\mathcal{O}}(\sqrt{d/n})$ in general, and $\widetilde{\mathcal{O}}(\sqrt{1/n})$ for quadratic loss. Therefore, when n is large, the preconditioning is very effective and the PCG method converges to sufficient precision within a small number of iterations. The stochastic assumption is also critical for obtaining an initial point w_0 which further brings down the overall iteration complexity.

Combining the above ideas, we propose and analyze an algorithm for Distributed Self-Concordant Optimization (DiSCO, which also stands for Distributed SeCond-Order method, or Distributed Stochastic Convex Optimization). We show that several popular empirical loss functions in machine learning, including ridge regression, regularized logistic regression and a (new) smoothed hinge loss, are actually self-concordant. For ERM with these loss functions, Table 5.1 lists the number of communication rounds required by DiSCO and several other algorithms to find an ϵ-optimal solution. As the table shows, the communication cost of DiSCO weakly depends on the number of machines m and on the feature dimension d, and is independent of the local sample size n (excluding logarithmic factors). Comparing to DANE [190], DiSCO not only improves the communication efficiency on quadratic loss, but also handles non-quadratic classification tasks.

5.3 Inexact damped Newton method

In this section, we propose and analyze an inexact damped Newton method for minimizing self-concordant functions. Without loss of generality, we assume the objective function $f : \mathbb{R}^d \to \mathbb{R}$ is standard self-concordant (see background in Chapter 2). In addition, we assume that Assumption H holds. Our method is described in Algorithm 1. If we let $\epsilon_k = 0$ for all $k \geq 0$, then $v_k = [f''(w_k)]^{-1}f'(w_k)$ is the exact Newton step and δ_k is the Newton

Algorithm	Number of Communication Rounds $\widetilde{\mathcal{O}}(\cdot)$	
	Ridge Regression (quadratic loss)	Binary Classification (logistic loss, smoothed hinge loss)
Accelerated Gradient	$(mn)^{1/4}\log(1/\epsilon)$	$(mn)^{1/4}\log(1/\epsilon)$
ADMM	$(mn)^{1/4}\log(1/\epsilon)$	$(mn)^{1/4}\log(1/\epsilon)$
DANE [190]	$m\log(1/\epsilon)$	$(mn)^{1/2}\log(1/\epsilon)$
DiSCO (our algorithm)	$m^{1/4}\log(1/\epsilon)$	$m^{3/4}d^{1/4} + m^{1/4}d^{1/4}\log(1/\epsilon)$

Table 5.1. Communication efficiency of several distributed algorithms for ERM of linear predictors, when the regularization parameter λ in (5.2) is on the order of $1/\sqrt{mn}$. All results are deterministic or high probability upper bounds, except that the last one, DiSCO for binary classification, is a bound in expectation (with respect to the randomness in generating the i.i.d. samples). For DiSCO, the dependence on ϵ can be improved to $\log\log(1/\epsilon)$ with superlinear convergence.

Algorithm 1: Inexact damped Newton method

input: initial point w_0 and specification of a nonnegative sequence $\{\epsilon_k\}$.

repeat for $k = 0, 1, 2, \ldots$

1. Find a vector v_k such that $\|f''(w_k)v_k - f'(w_k)\|_2 \le \epsilon_k$.
2. Compute $\delta_k = \sqrt{v_k^T f''(w_k)v_k}$ and update $w_{k+1} = w_k - \frac{1}{1+\delta_k}v_k$.

until a stopping criterion is satisfied.

decrement defined in (5.4), so the algorithm reduces to the exact damped Newton method given in (5.3). But here we allow the computation of the Newton step (hence also the Newton decrement) to be inexact and contain approximation errors.

The explicit account of approximation errors is essential for distributed optimization. In particular, if $f(w) = (1/m)\sum_{i=1}^m f_i(w)$ and the components f_i locate on separate machines, then we can only perform Newton updates approximately with limited communication budget. Even in a centralized setting on a single machine, analysis of approximation errors can be important if the Newton system is solved by iterative algorithms such as the conjugate gradient method.

Before presenting the convergence analysis, we need to introduce two auxiliary functions

$$\begin{aligned} \omega(t) &= t - \log(1+t), \qquad t \ge 0, \\ \omega_*(t) &= -t - \log(1-t), \qquad 0 \le t < 1. \end{aligned}$$

These two functions are very useful for characterizing the properties of self-concordant functions; see [154, Section 4.1.4] for a detailed account. Here, we simply note that $\omega(0) = \omega_*(0) = 0$, both are strictly increasing for $t \ge 0$, and $\omega_*(t) \to \infty$ as $t \to 1$.

We also need to define two auxiliary vectors

$$\widetilde{u}_k = [f''(w_k)]^{-1/2} f'(w_k),$$
$$\widetilde{v}_k = [f''(w_k)]^{1/2} v_k.$$

The norm of the first vector, $\|\widetilde{u}_k\|_2 = \sqrt{f'(w_k)^T [f''(w_k)]^{-1} f'(w_k)}$, is the exact Newton decrement. The norm of the second one is $\|\widetilde{v}_k\|_2 = \delta_k$, which is computed during each iteration of Algorithm 1. Note that we do *not* compute $\widetilde{u}_k$ or $\widetilde{v}_k$ in Algorithm 1. They are introduced solely for the purpose of convergence analysis. The following Theorem is proved in Section 5.7.1.

Theorem 5. *Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is a standard self-concordant function and Assumption H holds. If we choose the sequence $\{\epsilon_k\}_{k\geq 0}$ in Algorithm 1 as*

$$\epsilon_k = \beta(\rho/L)^{1/2} \|f'(w_k)\|_2 \quad \text{with } \beta = 1/20, \tag{5.6}$$

then:

(a) For any $k \geq 0$, we have $f(w_{k+1}) \leq f(w_k) - \frac{1}{2}\omega(\|\widetilde{u}_k\|_2)$.

(b) If $\|\widetilde{u}_k\|_2 \leq 1/6$, then we have $\omega(\|\widetilde{u}_{k+1}\|_2) \leq \frac{1}{2}\omega(\|\widetilde{u}_k\|_2)$.

As mentioned before, when $\epsilon_k = 0$, the vector $v_k = [f''(w_k)]^{-1} f'(w_k)$ becomes the exact Newton step. In this case, we have $\widetilde{v}_k = \widetilde{u}_k$, and it can be shown that $f(w_{k+1}) \leq f(w_k) - \omega(\|\widetilde{u}_k\|_2)$ for all $k \geq 0$ and the exact damped Newton method has quadratic convergence when $\|\widetilde{u}_k\|_2$ is small (see [154, Section 4.1.5]). With the approximation error ϵ_k specified in (5.6), we have

$$\begin{aligned}\|\widetilde{v}_k - \widetilde{u}_k\|_2 &\leq \|(f''(w_k))^{-1/2}\|_2 \|f''(w_k)v_k - f'(w_k)\|_2 \leq \rho^{-1/2}\epsilon_k \\ &= \beta L^{-1/2} \|f'(w_k)\|_2 \leq \beta \|\widetilde{u}_k\|_2,\end{aligned}$$

which implies

$$(1-\beta)\|\widetilde{u}_k\|_2 \leq \|\widetilde{v}_k\|_2 \leq (1+\beta)\|\widetilde{u}_k\|_2. \tag{5.7}$$

Section 5.7.1 shows that when β is sufficiently small, the above inequality leads to the conclusion in part (a). Compared with the exact damped Newton method, the guaranteed reduction of the objective value per iteration is cut by half.

Part (b) of Theorem 5 suggests a linear rate of convergence when $\|\widetilde{u}_k\|_2$ is small. This is slower than the quadratic convergence rate of the exact damped Newton method, due to the allowed approximation errors in computing the Newton step. However, when v_k is computed through a distributed iterative algorithm (like the distributed PCG algorithm in Section 5.4.2), a smaller ϵ_k would require more local computational effort and more rounds of inter-machine communication. The choice in equation (5.6) is a reasonable trade-off in practice.

Using Theorem 5, we can derive the iteration complexity of Algorithm 1 for obtaining an arbitrary accuracy. We present this result as a corollary.

Corollary 6. *Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is a standard self-concordant function and Assumption H holds. If we choose the sequence $\{\epsilon_k\}$ in Algorithm 1 as in (5.6), then for any $\epsilon > 0$, we have $f(w_k) - f(w_\star) \le \epsilon$ whenever $k \ge K$ where*

$$K = \left\lceil \frac{f(w_0) - f(w_\star)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2 \left(\frac{2\,\omega(1/6)}{\epsilon} \right) \right\rceil. \tag{5.8}$$

Here $\lceil t \rceil$ denotes the smallest nonnegative *integer that is larger or equal to t.*

Proof Since $\omega(t)$ is strictly increasing for $t \ge 0$, part (a) of Theorem 5 implies that if $\|\widetilde{u}_k\|_2 > 1/6$, one step of Algorithm 1 decreases the value of $f(w)$ by at least a constant $\frac{1}{2}\omega(1/6)$. So within at most $K_1 = \lceil \frac{f(w_0)-f(w_\star)}{\frac{1}{2}\omega(1/6)} \rceil$ iterations, we are guaranteed that $\|\widetilde{u}_k\|_2 \le 1/6$.

According to [154, Theorem 4.1.13], if $\|\widetilde{u}_k\|_2 < 1$, then we have

$$\omega(\|\widetilde{u}_k\|_2) \le f(w_k) - f(w_\star) \le \omega_*(\|\widetilde{u}_k\|_2). \tag{5.9}$$

Moreover, it is easy to check that $\omega_*(t) \le 2\,\omega(t)$ for $0 \le t \le 1/6$. Therefore, using part (b) of Theorem 5, we conclude that when $k \ge K_1$,

$$f(w_k) - f(w_\star) \;\le\; 2\,\omega(\|\widetilde{u}_k\|_2) \;\le\; 2(1/2)^{k-K_1}\omega(\|\widetilde{u}_{K_1}\|_2) \;\le\; 2(1/2)^{k-K_1}\omega(1/6).$$

Bounding the right-hand side of the above inequality by ϵ, we have $f(w_k) - f(w_\star) \le \epsilon$ whenever $k \ge K_1 + \left\lceil \log_2 \left(\frac{2\,\omega(1/6)}{\epsilon} \right) \right\rceil = K$, which is the desired result.

We note that when $\|\widetilde{u}_k\|_2 \le 1/6$ (as long as $k \ge K_1$), we have $f(w_k) - f(w_\star) \le 2\,\omega(1/6)$. Thus for $\epsilon > 2\,\omega(1/6)$, it suffices to have $k \ge K_1$. □

5.3.1 Stopping criteria

We discuss two stopping criteria for Algorithm 1. The first one is based on the strong convexity of f, which leads to the inequality (e.g., [154, Theorem 2.1.10])

$$f(w_k) - f(w_\star) \le \frac{1}{2\lambda}\|f'(w_k)\|_2^2.$$

Therefore, we can use the stopping criterion $\|f'(w_k)\|_2 \le \sqrt{2\lambda\epsilon}$, which implies $f(w_k) - f(w_\star) \le \epsilon$. However, this choice can be too conservative in practice (see discussions in [33, Section 9.1.2]).

Another choice for the stopping criterion is based on self-concordance. Using the fact that $\omega_*(t) \le t^2$ for $0 \le t \le 0.68$ (see [33, Section 9.6.3]), we have

$$f(w_k) - f(w_\star) \le \omega_*(\|\widetilde{u}_k\|_2) \le \|\widetilde{u}_k\|_2^2 \tag{5.10}$$

provided $\|\widetilde{u}_k\|_2 \leq 0.68$. Since we do not compute $\|\widetilde{u}_k\|_2$ (the exact Newton decrement) directly in Algorithm 1, we can use δ_k as an approximation. Using the inequality (5.7), and noticing that $\|\widetilde{v}_k\|_2 = \delta_k$, we conclude that

$$\delta_k \leq (1-\beta)\sqrt{\epsilon}$$

implies $f(w_k) - f(w_\star) \leq \epsilon$ when $\epsilon \leq 0.68^2$. Since δ_k is computed at each iteration of Algorithm 1, this can serve as a good stopping criterion.

5.3.2 Scaling for non-standard self-concordant functions

In many applications, we need to deal with empirical loss functions that are not standard self-concordant; see the examples in Section 2.3. Suppose a regularized loss function $\ell(w)$ is self-concordant with parameter $M_\ell > 2$. By Lemma 1, the scaled function $f = \eta\ell$ with $\eta = M_\ell^2/4$ is standard self-concordant. We can apply Algorithm 1 to minimize the scaled function f, and rewrite it in terms of the function ℓ and the scaling constant η.

Using the sequence $\{\epsilon_k\}$ defined in (5.6), the condition for computing v_k in Step 1 is

$$\|f''(w_k)v_k - f'(w_k)\|_2 \leq \beta(\lambda/L)^{1/2}\|f'(w_k)\|_2.$$

Let λ_ℓ and L_ℓ be the strong convexity and smoothness parameters of the function ℓ. With the scaling, we have $\lambda = \eta\lambda_\ell$ and $L = \eta L_\ell$, thus their ratio (the condition number) does not change. Therefore the above condition is equivalent to

$$\|\ell''(w_k)v_k - \ell'(w_k)\|_2 \leq \beta(\lambda_\ell/L_\ell)^{1/2}\|\ell'(w_k)\|_2. \tag{5.11}$$

In other words, the precision requirement in Step 1 is *scaling invariant*.

Step 2 of Algorithm 1 can be rewritten as

$$w_{k+1} = w_k - \frac{v_k}{1+\sqrt{\eta}\cdot\sqrt{v_k^T\ell''(w_k)v_k}}. \tag{5.12}$$

Here, the factor η explicitly appears in the formula. By choosing a larger scaling factor η, the algorithm chooses a smaller stepsize. This adjustment is intuitive because the convergence of Newton-type method relies on local smoothness conditions. By multiplying a large constant to ℓ, the function's Hessian becomes less smooth, so that the stepsize should shrink.

In terms of complexity analysis, if we target to obtain $\ell(w_k) - \ell(w_\star) \leq \epsilon$, then the iteration bound in (5.8) becomes

$$\left\lceil \frac{\eta\big(\ell(w_0) - \ell(w_\star)\big)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2\left(\frac{2\,\omega(1/6)}{\eta\epsilon}\right) \right\rceil. \tag{5.13}$$

For ERM problems in supervised learning, the self-concordant parameter M_ℓ, and hence the scaling factor $\eta = M_\ell^2/4$, can grow with the number of samples. For example, the

regularization parameter γ in (2.6) often scales as $1/\sqrt{N}$ where $N = mn$ is the total number of samples. Lemma 2 suggests that η grows on the order of $\sqrt{mn}$. A larger η will render the second term in (5.13) less relevant, but the first term grows with the sample size mn. In order to counter the effect of the growing scaling factor, we need to choose the initial point w_0 judiciously to guarantee a small initial gap. This will be explained further in the next sections.

5.4 The DiSCO algorithm

In this section, we adapt the inexact damped Newton method (Algorithm 1) to a distributed system, in order to minimize

$$f(w) = \frac{1}{m}\sum_{i=1}^{m} f_i(w), \tag{5.14}$$

where each function f_i can only be evaluated locally at machine i. This involves two questions: (1) how to set the initial point w_0 and (2) how to compute the inexact Newton step v_k in a distributed manner. After answering these two questions, we will present the overall DiSCO algorithm and analyze its communication complexity.

5.4.1 Initialization

In accordance with the averaging structure in (5.14), we choose the initial point based on averaging. More specifically, we let

$$w_0 = \frac{1}{m}\sum_{i=1}^{m} \widehat{W}_i, \tag{5.15}$$

where each $\widehat{W}_i$ is the solution to a local optimization problem at machine i:

$$\widehat{W}_i = \arg\min_{w\in\mathbb{R}^d} \left\{ f_i(w) + \frac{\rho}{2}\|w\|_2^2 \right\}, \qquad i = 1, \ldots, m. \tag{5.16}$$

Here $\rho \geq 0$ is a regularization parameter, which we will discuss in detail in the context of stochastic analysis in Section 5.5. Roughly speaking, if each f_i is constructed with n i.i.d. samples as in (5.2), then we can choose $\rho \sim 1/\sqrt{n}$ to make $\mathbb{E}[f(w_0) - f(w_\star)]$ decreasing as $\mathcal{O}(1/\sqrt{n})$. In this section, we simply regard it as an input parameter.

Here we comment on the computational cost of solving (5.16) locally at each machine. Suppose each $f_i(w)$ has the form in (5.2), then the local optimization problems in (5.16) become

$$\widehat{W}_i = \arg\min_{w\in\mathbb{R}^d} \left\{ \frac{1}{n}\sum_{j=1}^{n} \phi(w, z_{i,j}) + \frac{\lambda+\rho}{2}\|w\|_2^2 \right\}, \qquad i = 1, \ldots, m. \tag{5.17}$$

The finite average structure of the above objective function can be effectively exploited by the stochastic average gradient (SAG) method [175, 181] or its new variant SAGA [58]. Each step of these methods processes only one component function $\phi(w, z_{i,j})$, picked uniformly at random. Suppose $f_i(w)$ is L-smooth, then SAG returns an ϵ-optimal solution with $\mathcal{O}\big((n + \frac{L+\rho}{\lambda+\rho}) \log(1/\epsilon)\big)$ steps of stochastic updates. For ERM of linear predictors, we can also use the stochastic dual coordinate ascent (SDCA) method [186], which has the same complexity. We also mention some recent progress in accelerated stochastic coordinate gradient methods [184, 127, 229], which can be more efficient both in theory and practice.

5.4.2 Distributed computing of the inexact Newton step

In each iteration of Algorithm 1, we need to compute an inexact Newton step v_k such that $\|f''(w_k)v_k - f'(w_k)\|_2 \leq \epsilon_k$. This boils down to solving the Newton system $f''(w_k)v_k = f'(w_k)$ approximately. When the objective f has the averaging form (5.14), its Hessian and gradient are given in (5.5). In the setting of distributed optimization, we propose to use a preconditioned conjugate gradient (PCG) method to solve the Newton system.

To simplify notation, we use H to represent $f''(w_k)$ and use H_i to represent $f_i''(w_k)$. Without loss of generality, we define a preconditioning matrix using the local Hessian at the first machine (the master node):

$$P =: H_1 + \mu I,$$

where $\mu > 0$ is a small regularization parameter. Algorithm 2 describes our distributed PCG method for solving the preconditioned linear system

$$P^{-1}Hv_k = P^{-1}f'(w_k).$$

In particular, the master machine carries out the main steps of the classical PCG algorithm (e.g., [76, Section 10.3]), and all machines (including the master) compute the local gradients and Hessians and perform matrix-vector multiplications. Communication between the master and other machines are used to form the overall gradient $f'(w_k)$ and the matrix-vector products

$$Hu^{(t)} = \frac{1}{m}\sum_{i=1}^{m} f_i''(w_k)u^{(t)}, \qquad Hv^{(t)} = \frac{1}{m}\sum_{i=1}^{m} f_i''(w_k)v^{(t)}.$$

We note that the overall Hessian $H = f''(w_k)$ is never formed and the master machine only stores and updates the vectors $Hu^{(t)}$ and $Hv^{(t)}$.

As explained in Section 5.2, the motivation for preconditioning is that when H_1 is sufficiently close to H, the condition number of $P^{-1}H$ might be close to 1, which is much smaller than that of H itself. As a result, the PCG method may converge much faster than CG without preconditioning. The following lemma characterizes the extreme eigenvalues of $P^{-1}H$ based on the closeness between H_1 and H.

Algorithm 2: Distributed PCG algorithm (given w_k and μ, compute v_k and δ_k)

master machine $(i = 1)$ | *machines* $i = 1, \ldots, m$

input: $w_k \in \mathbb{R}^d$ and $\mu \geq 0$.
let $H = f''(w_k)$ and $P = f_1''(w_k) + \mu I$.

communication:
broadcasts w_k to other machines; $\longrightarrow$ compute $f_i'(w_k)$
aggregate $f_i'(w_k)$ to form $f'(w_k)$. $\longleftarrow$

initialization: compute ϵ_k given in (5.6) and set
$v^{(0)} = 0$, $s^{(0)} = P^{-1}r^{(0)}$,
$r^{(0)} = f'(w_k)$, $u^{(0)} = s^{(0)}$.

repeat for $t = 0, 1, 2 \ldots,$

1. **communication:**
broadcast $u^{(t)}$ and $v^{(t)}$; $\longrightarrow$ compute $f_i''(w_k)u^{(t)}$
aggregate to form $Hu^{(t)}$ and $Hv^{(t)}$. $\longleftarrow$ compute $f_i''(w_k)v^{(t)}$
2. compute $\alpha_t = \frac{\langle r^{(t)}, s^{(t)} \rangle}{\langle u^{(t)}, Hu^{(t)} \rangle}$ and update
$v^{(t+1)} = v^{(t)} + \alpha_t u^{(t)}$,
$r^{(t+1)} = r^{(t)} - \alpha_t H u^{(t)}$.
3. compute $\beta_t = \frac{\langle r^{(t+1)}, s^{(t+1)} \rangle}{\langle r^{(t)}, s^{(t)} \rangle}$ and update
$s^{(t+1)} = P^{-1}r^{(t+1)}$,
$u^{(t+1)} = s^{(t+1)} + \beta_t u^{(t)}$.

until $\|r^{(t+1)}\|_2 \leq \epsilon_k$

return $v_k = v^{(t+1)}$, $r_k = r^{(t+1)}$, and $\delta_k = \sqrt{v_k^T H v^{(t)} + \alpha^{(t)} v_k^T H u^{(t)}}$.

Lemma 20. *Suppose Assumption H holds. If* $\|H_1 - H\|_2 \leq \mu$, *then we have*

$$\sigma_{\max}(P^{-1}H) \leq 1, \tag{5.18}$$

$$\sigma_{\min}(P^{-1}H) \geq \frac{\rho}{\rho + 2\mu}. \tag{5.19}$$

Here $\|\cdot\|_2$ *denote the spectral norm of a matrix, and* $\sigma_{\max}(\cdot)$ *and* $\sigma_{\min}(\cdot)$ *denote the largest and smallest eigenvalues of a diagonalizable matrix, respectively.*

Proof Since both P and H are symmetric and positive definite, all eigenvalues of $P^{-1}H$ are positive real numbers (e.g., [93, Section 7.6]). The eigenvalues of $P^{-1}H$ are identical to

that of $P^{-1/2}HP^{-1/2}$. Thus, it suffices to prove inequalities (5.18) and (5.19) for the matrix $P^{-1/2}HP^{-1/2}$. To prove inequality (5.18), we need to show that $H \preceq P = H_1 + \mu I$. This is equivalent to $H - H_1 \preceq \mu I$, which is a direct consequence of the assumption $\|H_1 - H\|_2 \leq \mu I$.

Similarly, the second inequality (5.19) is equivalent to $H \succeq \frac{\rho}{\rho+2\mu}(H_1 + \mu I)$, which is the same as $\frac{2\mu}{\rho}H - \mu I \succeq H_1 - H$. Since $H \succeq \rho I$ (by Assumption H), we have $\frac{2\mu}{\rho}H - \mu I \succeq \mu I$. The additional assumption $\|H_1 - H\|_2 \leq \mu I$ implies $\mu I \succeq H_1 - H$, which complete the proof. □

By Assumption H, the condition number of the Hessian matrix is $\kappa(H) = L/\lambda$, which can be very large if λ is small. Lemma 20 establishes that the condition number of the preconditioned linear system is

$$\kappa(P^{-1}H) = \frac{\sigma_{\max}(P^{-1}H)}{\sigma_{\min}(P^{-1}H)} = 1 + \frac{2\mu}{\lambda}, \tag{5.20}$$

provided that $\|H_1 - H\|_2 \leq \mu$. When μ is small (comparable with λ), the condition number $\kappa(P^{-1}H)$ is close to one and can be much smaller than $\kappa(H)$. Based on classical convergence analysis of the CG method (e.g., [133, 8]), the following lemma shows that Algorithm 2 terminates in $\mathcal{O}(\sqrt{1+\mu/\rho})$ iterations. See Section 5.7.2 for the proof.

Lemma 21. *Suppose Assumption H holds and assume that* $\|H_1 - H\|_2 \leq \mu$. *Let*

$$T_\mu = \left\lceil \sqrt{1 + \frac{2\mu}{\rho}} \log\left(\frac{2\sqrt{L/\lambda}\|f'(w_k)\|_2}{\epsilon_k} \right) \right\rceil.$$

Then Algorithm 2 terminates in T_μ *iterations and the output* v_k *satisfies* $\|Hv_k - f'(w_k)\|_2 \leq \epsilon_k$.

When the tolerance ϵ_k is chosen as in (5.6), the iteration bound T_μ is independent of $f'(w_k)$, i.e.,

$$T_\mu = \left\lceil \sqrt{1 + \frac{2\mu}{\rho}} \log\left(\frac{2L}{\beta\rho} \right) \right\rceil. \tag{5.21}$$

Under Assumption H, we always have $\|H_1 - H\|_2 \leq L$. If we choose $\mu = L$, then Lemma 21 implies that Algorithm 2 terminates in $\widetilde{\mathcal{O}}(\sqrt{L/\rho})$ iterations. where the notation $\widetilde{\mathcal{O}}(\cdot)$ hides logarithmic factors. In practice, however, the matrix norm $\|H_1 - H\|_2$ is usually much smaller than L due to the stochastic nature of f_i. Thus, we can choose μ to be a tight upper bound on $\|H_1 - H\|_2$, and expect the algorithm terminating in $\widetilde{\mathcal{O}}(\sqrt{\mu/\rho})$ iterations. In Section 5.5, we show that if the local empirical losses f_i are constructed with n i.i.d. samples from the same distribution, then $\|H_1 - H\|_2 \sim 1/\sqrt{n}$ with high probability. As a consequence, the iteration complexity of Algorithm 2 is upper bounded by $\widetilde{\mathcal{O}}(1+\lambda^{-1/2}n^{-1/4})$.

We wrap up this section by discussing the computation and communication complexities of Algorithm 2. The bulk of computation is at the master machine, especially computing

Algorithm 3: DiSCO

input: parameters $\rho, \mu \geq 0$ and precision $\epsilon > 0$.
initialize: compute w_0 according to (5.15) and (5.16).
repeat for $k = 0, 1, 2, \ldots$

1. Run Algorithm 2: given w_k and μ, compute v_k and δ_k.
2. Update $w_{k+1} = w_k - \frac{1}{1+\delta_k} v_k$.

until $\delta_k \leq (1-\beta)\sqrt{\epsilon}$.
output: $\widehat{W} = w_{k+1}$.

the vector $s^{(t)} = P^{-1} r^{(t)}$ in Step 3, which is equivalent to minimize the quadratic function $(1/2) s^T P s - s^T r^{(t)}$. Using $P = f_1''(w_k) + \mu I$ and the form of $f_1(w)$ in (5.2), this is equivalent to

$$s^{(t)} = \arg\min_{s \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{j=1}^{n} \frac{s^T \phi''(w_k, z_{i,j}) s}{2} + \langle r^{(t)}, s \rangle + \frac{\lambda + \mu}{2} \|s\|_2^2 \right\}. \tag{5.22}$$

This problem has the same structure as (5.17), and an ϵ-optimal solution can be obtained with $\mathcal{O}\big((n + \frac{L+\mu}{\lambda+\mu}) \log(1/\epsilon)\big)$ stochastic-gradient type of steps (see discussions at the end of Section 5.4.1).

As for the communication complexity, we need one round of communication at the beginning of Algorithm 2 to compute $f'(w_k)$. Then, each iteration takes one round of communication to compute $Hu^{(t)}$ and $Hv^{(t)}$. Thus, the total rounds of communication is bounded by $T_\mu + 1$.

5.4.3 Communication efficiency of DiSCO

Putting everything together, we present the DiSCO algorithm in Algorithm 3. Here we study its communication efficiency. Recall that by one round of communication, the master machine broadcasts a message of $\mathcal{O}(d)$ bits to all machines, and every machine processes the aggregated message and sends a message of $\mathcal{O}(d)$ bits back to the master. The following proposition gives an upper bound on the number of communication rounds taken by the DiSCO algorithm.

Theorem 6. *Assume that f is a standard self-concordant function and it satisfies Assumption H. Suppose the input parameter μ in Algorithm 3 is an upper bound on $\|f_1''(w_k) - f''(w_k)\|_2$ for all $k \geq 0$. Then for any $\epsilon > 0$, in order to find a solution $\widehat{W}$ satisfying $f(\widehat{W}) - f(w_\star) < \epsilon$, the total number of communication rounds T is bounded by*

$$T \leq 1 + \left(\left\lceil \frac{f(w_0) - f(w_\star)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2 \left(\frac{2\omega(1/6)}{\epsilon} \right) \right\rceil \right) \left(2 + \sqrt{1 + \frac{2\mu}{\rho}} \log\left(\frac{2L}{\beta\rho} \right) \right). \tag{5.23}$$

Algorithm 4: Adaptive DiSCO

input: parameters $\rho \geq 0$ and $\mu_0 > 0$, and precision $\epsilon > 0$.
initialize: compute w_0 according to (5.15) and (5.16).
repeat for $k = 0, 1, 2, \ldots$

1. Run Algorithm 2 up to T_{μ_k} PCG iterations, with output v_k, δ_k, r_k and ϵ_k.
2. **if** $\|r_k\|_2 > \epsilon_k$ **then**
   set $\mu_k := 2\mu_k$ and go to Step 1;
   else
   set $\mu_{k+1} := \mu_k/2$ and go to Step 3.
3. Update $w_{k+1} = w_k - \frac{1}{1+\delta_k} v_k$.

until $\delta_k \leq (1-\beta)\sqrt{\epsilon}$.
output: $\widehat{W} = w_{k+1}$.

Ignoring logarithmic terms and universal constants, the rounds of communication T is bounded by

$$\widetilde{\mathcal{O}}\left(\left(f(w_0) - f(w_\star) + \log(1/\epsilon)\right)\sqrt{1 + 2\mu/\lambda}\right).$$

Proof First we notice that the number of communication rounds in each call of Algorithm 2 is no more than $1 + T_\mu$, where T_μ is given in (5.21), and the extra 1 accounts for the communication round to form $f'(w_k)$. Corollary 6 states that in order to guarantee $f(w_k) - f(w_\star) \leq \epsilon$, the total number of calls of Algorithm 2 in DiSCO is bounded by K given in (5.8). Thus the total number of communication rounds is bounded by $1 + K(1 + T_\mu)$, where the extra one count is for computing the initial point w_0 defined in (5.15). □

It can be hard to give a good *a priori* estimate of μ that satisfies the condition in Theorem 6. In practice, we can adjust the value of μ adaptively while running the algorithm. Inspired by a line search procedure studied in [155], we propose an adaptive DiSCO method, described in Algorithm 4. The following proposition bounds the rounds of communication required by this algorithm.

Theorem 7. *Assume that f is a standard self-concordant function and it satisfies Assumption H. Let $\mu_{\max}$ be the largest value of μ_k generated by Algorithm 4,* i.e., *$\mu_{\max} = \max\{\mu_0, \mu_1, \ldots, \mu_K\}$ where K is the number of outer iterations. Then for any $\epsilon > 0$, in order to find a solution $\widehat{W}$ satisfying $f(\widehat{W}) - f(w_\star) < \epsilon$, the total number of communication rounds T is bounded by*

$$T \leq 1 + \left(2\left\lceil \frac{f(w_0) - f(w_\star)}{\omega(1/6)} \right\rceil + 2\left\lceil \log_2\left(\frac{2\omega(1/6)}{\epsilon}\right)\right\rceil + \log_2\left(\frac{\mu_{\max}}{\mu_0}\right)\right) \times \left(2 + \sqrt{1 + \frac{2\mu_{\max}}{\rho}}\,\log\left(\frac{2L}{\beta\rho}\right)\right).$$

Proof Let n_k be the number of calls to Algorithm 2 during the kth iteration of Algorithm 4. We have

$$\mu_{k+1} = \frac{1}{2}\mu_k 2^{n_k-1} = \mu_k 2^{n_k-2},$$

which implies

$$n_k = 2 + \log_2 \frac{\mu_{k+1}}{\mu_k}.$$

The total number of calls to Algorithm 2 is

$$N_K = \sum_{k=0}^{K-1} n_k = \sum_{k=0}^{K-1} \left(1 + \log_2 \frac{\mu_{k+1}}{\mu_k}\right) = 2K + \log_2 \frac{\mu_K}{\mu_0} \leq 2K + \log_2 \frac{\mu_{\max}}{\mu_0}.$$

Since each call of Algorithm 2 involves no more than $T_{\mu_{\max}} + 1$ communication rounds, we have

$$T \leq 1 + N_K(T_{\mu_{\max}} + 1).$$

Plugging in the expression of K in (5.8) and $T_{\mu_{\max}}$ in (5.21), we obtain the desired result. □

From the above proof, we see that the average number of calls to Algorithm 2 at each iteration is $2 + \frac{1}{K}\log_2\left(\frac{\mu_K}{\mu_0}\right)$, roughly twice as the non-adaptive Algorithm 3. Ignoring logarithmic terms and universal constants, the number of communication round T used by Algorithm 4 is bounded by

$$\widetilde{\mathcal{O}}\left(\left(f(w_0) - f(w_\star) + \log_2(1/\epsilon)\right)\sqrt{1 + 2\mu_{\max}/\lambda}\right).$$

In general, we can update μ_k in Algorithm 4 as follows:

$$\mu_k := \begin{cases} \theta_{\text{inc}}\mu_k & \text{if } \|r_k\|_2 > \epsilon_k, \\ \mu_k/\theta_{\text{dec}} & \text{if } \|r_k\|_2 \leq \epsilon_k, \end{cases}$$

with any $\theta_{\text{inc}} > 1$ and $\theta_{\text{dec}} \geq 1$ (see [155]). We have used $\theta_{\text{inc}} = \theta_{\text{dec}} = 2$ to simplify presentation.

5.5 Stochastic analysis

From Theorems 6 and 7 of the previous section, we see that the communication complexity of the DiSCO algorithm mainly depends on two quantities: the initial objective gap $f(w_0) - f(w_\star)$ and the upper bound μ on the spectral norms $\|f_1''(w_k) - f''(w_k)\|_2$ for all $k \geq 0$. As we discussed in Section 5.3.2, the initial gap $f(w_0) - f(w_\star)$ may grow with the number of samples due to the scaling used to make the objective function standard self-concordant. On the other hand, the upper bound μ may decrease as the number of samples increases based on the intuition that the local Hessians and the global Hessian become similar to each

other. In this section, we show how to exploit the stochastic origin of the problem to mitigate the effect of objective scaling and quantify the notion of similarity between local and global Hessians. These lead to improved complexity results.

We focus on the setting of distributed optimization of *regularized* empirical loss. That is, our goal is to minimize $f(w) = (1/m)\sum_{i=1}^m f_i(w)$, where

$$f_i(w) = \frac{1}{n}\sum_{j=1}^{n} \phi(w, z_{i,j}) + \frac{\lambda}{2}\|w\|_2^2, \qquad i = 1, \ldots, m. \tag{5.24}$$

We assume that $z_{i,j}$ are i.i.d. samples from a common distribution. Our theoretical analysis relies on refined assumptions on the smoothness of the loss function ϕ. In particular, we assume that for any z in the sampling space $\mathcal{Z}$, the function $\phi(\cdot, z)$ has bounded first derivative in a compact set, and its second derivatives are bounded and Lipschitz continuous. We formalize these statements in the following assumption.

Assumption I. *There are finite constants (V_0, G, L, M), such that for any $z \in \mathcal{Z}$:*

(i) $\phi(w, z) \geq 0$ *for all* $w \in \mathbb{R}^d$, *and* $\phi(0, z) \leq V_0$;

(ii) $\|\phi'(w, z)\|_2 \leq G$ *for any* $\|w\|_2 \leq \sqrt{2V_0/\rho}$;

(iii) $\|\phi''(w, z)\|_2 \leq L - \rho$ *for any* $w \in \mathbb{R}^d$;

(iv) $\|\phi''(u, z) - \phi''(w, z)\|_2 \leq M\|u - w\|_2$ *for any* $u, w \in \mathbb{R}^d$.

For the regularized empirical loss in (5.24), condition *(iii)* in the above assumption implies $\rho I \preceq f_i''(w) \preceq LI$ for $i = 1, \ldots, m$, which in turn implies Assumption H.

Recall that the initial point w_0 is obtained as the average of the solutions to m regularized local optimization problems; see equations (5.15) and (5.16). The following lemma shows that expected value of the initial gap $f(w_0) - f(w_\star)$ decreases with order $1/\sqrt{n}$ as the local sample size n increases. The proof uses the notion and techniques of *uniform stability* for analyzing the generalization performance of ERM [32]. See Section 5.7.3 for the proof.

Lemma 22. *Suppose that Assumption I holds and* $\mathbb{E}[\|w_\star\|_2^2] \leq D^2$ *for some constant* $D > 0$. *If we choose* $\rho = \frac{\sqrt{6}G}{\sqrt{n}D}$ *in* (5.16) *to compute* $\widehat{W}_i$, *then the initial point* $w_0 = \frac{1}{m}\sum_{i=1}^m \widehat{W}_i$ *satisfies*

$$\max\{\|w_\star\|_2, \|w_0\|_2\} \leq \sqrt{\frac{2V_0}{\rho}} \tag{5.25}$$

and

$$\mathbb{E}[f(w_0) - f(w_\star)] \leq \frac{\sqrt{6}GD}{\sqrt{n}}. \tag{5.26}$$

Here the expectation is taken with respect to the randomness in generating the i.i.d. data.

Next, we show that with high probability, $\|f_i''(w) - f''(w)\|_2 \sim \sqrt{d/n}$ for any $i \in \{1, \ldots, m\}$ and for any vector w in an ℓ_2-ball. Thus, if the number of samples n is large, the Hessian matrix of f can be approximated well by that of f_i. The proof uses random matrix concentration theories [137]. We defer the proof to Section 5.7.4.

Lemma 23. *Suppose Assumption I holds. For any $r > 0$ and any $i \in \{1, \ldots, m\}$, we have with probability at least $1 - \delta$,*

$$\sup_{\|w\|_2 \le r} \|f_i''(w) - f''(w)\|_2 \le \mu_{r,\delta},$$

where

$$\mu_{r,\delta} =: \min\left\{L, \sqrt{\frac{32L^2 d}{n}} \cdot \sqrt{\log\left(1 + \frac{rM\sqrt{2n}}{L}\right) + \frac{\log(md/\delta)}{d}}\right\}. \tag{5.27}$$

If $\phi(w, z_{i,j})$ are quadratic functions in w, then we have $M = 0$ in Assumption I. In this case, Lemma 23 implies $\|f_i''(w) - f''(w)\|_2 \sim \sqrt{1/n}$. For general non-quadratic loss, Lemma 23 implies $\|f_i''(w) - f''(w)\|_2 \sim \sqrt{d/n}$. We use this lemma to obtain an upper bound on the spectral norm of the Hessian distances $\|f_1''(w_k) - f''(w_k)\|_2$, where the vectors w_k are generated by Algorithm 1.

Corollary 7. *Suppose Assumption I holds and the sequence $\{w_k\}_{k \ge 0}$ is generated by Algorithm 1. Let $r = \left(\frac{2V_0}{\rho} + \frac{2G}{\rho}\sqrt{\frac{2V_0}{\rho}}\right)^{1/2}$. Then with probability at least $1 - \delta$, we have for all $k \ge 0$,*

$$\|f_1''(w_k) - f''(w_k)\|_2 \le \min\left\{L, \sqrt{\frac{32L^2 d}{n}} \cdot \sqrt{\log\left(1 + \frac{rM\sqrt{2n}}{L}\right) + \frac{\log(md/\delta)}{d}}\right\}. \tag{5.28}$$

Proof We begin by upper bounding the ℓ_2-norm of w_k, for $k = 0, 1, 2 \ldots$, generated by Algorithm 1. By Theorem 5, we have $f(w_k) \le f(w_0)$ for all $k \ge 0$. By Assumption I (i), we have $\phi(w, z) \ge 0$ for all $w \in \mathbb{R}^d$ and $z \in \mathcal{Z}$. As a consequence,

$$\frac{\rho}{2}\|w_k\|_2^2 \le f(w_k) \le f(w_0) \le f(0) + G\|w_0\|_2 \le V_0 + G\|w_0\|_2.$$

Substituting $\|w_0\|_2 \le \sqrt{2V_0/\rho}$ (see Lemma 22) into the above inequality yields

$$\|w_k\|_2 \le \left(\frac{2V_0}{\rho} + \frac{2G}{\rho}\sqrt{\frac{2V_0}{\rho}}\right)^{1/2} = r.$$

Thus, we have $\|w_k\|_2 \le r$ for all $k \ge 0$. Applying Lemma 23 establishes the corollary. □

Here we remark that the dependence on d of the upper bound in (5.28) comes from Lemma 23, where the bound needs to hold for all point in a d-dimensional ball with radius r. However, for the analysis of the DiSCO algorithm, we only need the matrix concentration bound to hold for a finite number of vectors $w_0, w_1, \ldots, w_K$, instead of for all vectors satisfying $\|w\|_2 \leq r$. Thus we conjecture that the bound in (5.28), especially its dependence on the dimension d, is too conservative and very likely can be tightened.

We are now ready to present the main results of our stochastic analysis. The following theorem provides an upper bound on the expected number of communication rounds required by the DiSCO algorithm to find an ϵ-optimal solution. Here the expectation is taken with respect to the randomness in generating the i.i.d. data set $\{z_{i,j}\}$.

Theorem 8. *Let Assumption I hold. Assume that the regularized empirical loss function f is standard self-concordant, and its minimizer $w_\star = \arg\min_w f(w)$ satisfies $\mathbb{E}[\|w_\star\|_2^2] \leq D^2$ for some constant $D > 0$. Let the input parameters to Algorithm 3 be $\rho = \frac{\sqrt{6}G}{\sqrt{n}D}$ and $\mu = \mu_{r,\delta}$ in (5.27) with*

$$r = \left(\frac{2V_0}{\rho} + \frac{2G}{\rho}\sqrt{\frac{2V_0}{\rho}}\right)^{1/2}, \qquad \delta = \frac{GD}{\sqrt{n}} \cdot \frac{\sqrt{\lambda/(4L)}}{4V_0 + 2G^2/\lambda}. \tag{5.29}$$

Then for any $\epsilon > 0$, the total number of communication rounds T required to reach $f(\widehat{W}) - f(w_\star) \leq \epsilon$ is bounded by

$$\mathbb{E}[T] \leq 1 + \left(C_1 + \frac{6}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)\left(2 + C_2\left(1 + 2\sqrt{\frac{32L^2 d\, C_3}{\rho^2 n}}\right)^{1/2}\right),$$

where C_1, C_2, C_3 are $\widetilde{\mathcal{O}}(1)$ or logarithmic terms:

$$\begin{aligned}
C_1 &= \left(1 + \left\lceil \log_2\left(\frac{2\omega(1/6)}{\epsilon}\right)\right\rceil\right)\left(1 + \frac{1}{\sqrt{n}} \cdot \frac{GD}{4V_0 + 2G^2/\lambda}\right), \\
C_2 &= \log\left(\frac{2L}{\beta\rho}\right), \\
C_3 &= \log\left(1 + \frac{rM\sqrt{2n}}{L}\right) + \frac{\log(dm/\delta)}{d}.
\end{aligned}$$

In particular, ignoring numerical constants and logarithmic terms, we have

$$\mathbb{E}[T] = \widetilde{\mathcal{O}}\left(\left(\log(1/\epsilon) + \frac{GD}{n^{1/2}}\right)\left(1 + \frac{L^{1/2}d^{1/4}}{\rho^{1/2}n^{1/4}}\right)\right).$$

Proof Suppose Algorithm 3 terminates in K iterations, and let t_k be the number of conjugate gradient steps in each call of Algorithm 2, for $k = 0, 1, \ldots, K-1$. For any given $\mu > 0$, we define T_μ as in (5.21). Let $\mathcal{A}$ denotes the event that $t_k \leq T_\mu$ for all

$k \in \{0, \ldots, K-1\}$. Let $\bar{\mathcal{A}}$ be the complement of $\mathcal{A}$, *i.e.*, the event that $t_k > T_\mu$ for some $k \in \{0, \ldots, K-1\}$. In addition, let the probabilities of the events $\mathcal{A}$ and $\bar{\mathcal{A}}$ be $1-\delta$ and δ respectively. By the law of total expectation, we have

$$\mathbb{E}[T] = \mathbb{E}[T|\mathcal{A}]\mathbb{P}(\mathcal{A}) + \mathbb{E}[T|\bar{\mathcal{A}}]\mathbb{P}(\bar{\mathcal{A}}) = (1-\delta)\mathbb{E}[T|\mathcal{A}] + \delta\,\mathbb{E}[T|\bar{\mathcal{A}}].$$

When the event $\mathcal{A}$ happens, we have $T \le 1 + K(T_\mu + 1)$ where T_μ is given in (5.21); otherwise we have $T \le 1 + K(T_L + 1)$, where

$$T_L = \sqrt{2 + \frac{2L}{\lambda}} \log\left(\frac{2L}{\beta\lambda}\right) \tag{5.30}$$

is an upper bound on the number of PCG iterations in Algorithm 2 when the event $\bar{\mathcal{A}}$ happens (see the analysis in Section 5.7.5). Since Algorithm 2 always return a v_k such that $\|f''(w_k)v_k - f'(w_k)\|_2 \le \epsilon_k$, the outer iteration count K share the same bound (5.8), which depends on the random variable $f(w_0) - f(w_\star)$. However, T_μ and T_L are deterministic constants. So we have

$$\begin{aligned}\mathbb{E}[T] &\le 1 + (1-\delta)\mathbb{E}[K(T_\mu+1)|\mathcal{A}] + \delta\,\mathbb{E}[K(T_L+1)|\bar{\mathcal{A}}] \\ &= 1 + (1-\delta)(T_\mu+1)\mathbb{E}[K|\mathcal{A}] + \delta(T_L+1)\mathbb{E}[K|\bar{\mathcal{A}}].\end{aligned} \tag{5.31}$$

Next we bound $\mathbb{E}[K|\mathcal{A}]$ and $\mathbb{E}[K|\bar{\mathcal{A}}]$ separately. To bound $\mathbb{E}[K|\mathcal{A}]$, we use

$$\mathbb{E}[K] = (1-\delta)\mathbb{E}[K|\mathcal{A}] + \delta\,\mathbb{E}[K|\bar{\mathcal{A}}] \ge (1-\delta)\mathbb{E}[K|\mathcal{A}]$$

to obtain

$$\mathbb{E}[K|\mathcal{A}] \le \mathbb{E}[K]/(1-\delta). \tag{5.32}$$

In order to bound $\mathbb{E}[K|\bar{\mathcal{A}}]$, we derive a deterministic bound on $f(w_0) - f(w_\star)$. By Lemma 22, we have $\|w_0\|_2 \le \sqrt{2V_0/\lambda}$, which together with Assumption I (ii) yields

$$\|f'(w)\|_2 \le G + \lambda\|w\|_2 \le G + \sqrt{2\lambda V_0}.$$

Combining with the strong convexity of f, we obtain

$$f(w_0) - f(w_\star) \le \frac{1}{2\lambda}\|f'(w_0)\|_2^2 \le \frac{1}{2\lambda}\left(G + \sqrt{2\lambda V_0}\right)^2 \le 2V + \frac{G^2}{\lambda}.$$

Therefore by Corollary 6,

$$K \le K_{\max} =: 1 + \frac{4V_0 + 2G^2/\lambda}{\omega(1/6)} + \left\lceil \log_2\left(\frac{2\omega(1/6)}{\epsilon}\right)\right\rceil, \tag{5.33}$$

where the additional 1 counts compensate for removing one $\lceil\cdot\rceil$ operator in (5.8).

Using inequality (5.31), the bound on $\mathbb{E}[K|\mathcal{A}]$ in (5.32) and the bound on $\mathbb{E}[K|\bar{\mathcal{A}}]$ in (5.33), we obtain

$$\mathbb{E}[T] \leq 1 + (T_\mu + 1)\mathbb{E}[K] + \delta(T_L + 1)K_{\max}.$$

Now we can bound $\mathbb{E}[K]$ by Corollary 6 and Lemma 22. More specifically,

$$\mathbb{E}[K] \leq \frac{\mathbb{E}[f(w_0) - f(w_\star)]}{\frac{1}{2}\omega(1/6)} + \left\lceil \log_2\left(\frac{2\omega(1/6)}{\epsilon}\right)\right\rceil + 1 = C_0 + \frac{2\sqrt{6}}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}, \tag{5.34}$$

where $C_0 = 1 + \lceil \log_2(2\omega(1/6)/\epsilon)\rceil$. With the choice of δ in (5.29) and the definition of T_L in (5.30), we have

$$\begin{aligned}
\delta(T_L + 1)K_{\max} &= \frac{GD}{\sqrt{n}} \cdot \frac{\sqrt{\lambda/(4L)}}{4V_0 + 2G^2/\lambda}\left(2 + \sqrt{2 + \frac{2L}{\lambda}}\log\left(\frac{2L}{\beta\lambda}\right)\right)\left(C_0 + \frac{4V_0 + 2G^2/\lambda}{\omega(1/6)}\right) \\
&= \left(\frac{C_0}{\sqrt{n}} \cdot \frac{GD}{4V_0 + 2G^2/\lambda} + \frac{1}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)\left(\sqrt{\frac{\lambda}{L}} + C_2\sqrt{\frac{\lambda}{2L} + \frac{1}{2}}\right) \\
&\leq \left(\frac{C_0}{\sqrt{n}} \cdot \frac{GD}{4V_0 + 2G^2/\lambda} + \frac{1}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)\left(2 + C_2\sqrt{1 + \frac{2\mu}{\lambda}}\right) \\
&= \left(\frac{C_0}{\sqrt{n}} \cdot \frac{GD}{4V_0 + 2G^2/\lambda} + \frac{1}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)(T_\mu + 1)
\end{aligned}$$

Putting everything together, we have

$$\begin{aligned}
\mathbb{E}[T] &\leq 1 + \left(C_0 + \frac{C_0}{\sqrt{n}} \cdot \frac{GD}{4V_0 + 2G^2/\lambda} + \frac{2\sqrt{6} + 1}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)(T_\mu + 1) \\
&\leq 1 + \left(C_1 + \frac{6}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)(T_\mu + 1).
\end{aligned}$$

Replacing T_μ by its expression in (5.21) and applying Corollary 7, we obtain the desired result. □

According to Theorem 8, we need to set the two input parameters ρ and μ in Algorithm 3 appropriately to obtain the desired communication efficiency. Using the adaptive DiSCO method given in Algorithm 4, we can avoid the explicit specification of $\mu = \mu_{r,\delta}$ defined in (5.27) and (5.29). This is formalized in the following theorem.

Theorem 9. *Let Assumption I hold. Assume that the regularized empirical loss function f is standard self-concordant, and its minimizer $w_\star = \arg\min_w f(w)$ satisfies $\mathbb{E}[\|w_\star\|_2^2] \leq D^2$ for some constant $D > 0$. Let the input parameters to Algorithm 4 be $\rho = \frac{\sqrt{6}G}{\sqrt{n}D}$ and any $\mu_0 > 0$. Then the total number of communication rounds T required to reach $f(\widehat{W}) - f(w_\star) \leq \epsilon$ is bounded by*

$$\mathbb{E}[T] = \widetilde{\mathcal{O}}\left(\left(\log(1/\epsilon) + \frac{GD}{n^{1/2}}\right)\left(1 + \frac{L^{1/2}d^{1/4}}{\rho^{1/2}n^{1/4}}\right)\right).$$

Proof In Algorithm 4, the parameter μ_k is automatically tuned such that the number of PCG iterations in Algorithm 2 is no more than T_{μ_k}. By Corollary 7, with probability at least $1-\delta$, we have

$$\max\{\mu_0, \ldots, \mu_K\} \leq 2\mu_{r,\delta}$$

where $\mu_{r,\delta}$ is defined in (5.27), and r and δ are given in (5.29). Therefore we can use the same arguments in the proof of Theorem 8 to show that

$$\mathbb{E}[T] \leq 1 + \left(\tilde{C}_1 + \frac{6}{\omega(1/6)} \cdot \frac{GD}{\sqrt{n}}\right)\left(2 + C_2\left(1 + 4\sqrt{\frac{32L^2 d\, C_3}{\rho^2 n}}\right)^{1/2}\right)$$

where

$$\tilde{C}_1 = \left(2 + 2\left\lceil \log_2\left(\frac{2\omega(1/6)}{\epsilon}\right)\right\rceil + \log_2\left(\frac{L}{\mu_0}\right)\right)\left(1 + \frac{1}{\sqrt{n}} \cdot \frac{GD}{4V_0 + 2G^2/\lambda}\right),$$

and C_2 and C_3 are the same as given in Theorem 8. Ignoring constants and logarithmic terms, we obtain the desired result. □

Remarks The expectation bounds on the rounds of communication given in Theorems 8 and 9 are obtained by combining two consequences of averaging over a large number of i.i.d. local samples. One is the expected reduction of the initial gap $f(w_0) - f(w_\star)$ (Lemma 22), which helps to mitigate the effect of objective scaling used to make f standard self-concordant. The other is a high-probability bound that characterizes the similarity between the local and global Hessians (Corollary 7). If the empirical loss f is standard self-concordant without scaling, then we can regard $f(w_0) - f(w_\star)$ as a constant, and only need to use Corollary 7 to obtain a high-probability bound. This is demonstrated for the case of linear regression in Section 5.5.1.

For applications where the loss function needs to be rescaled to be standard self-concordant, the convexity parameter λ as well as the "constants" (V_0, G, L, M) in Assumption I also need to be rescaled. If the scaling factor grows with n, then we need to rely on Lemma 22 to balance the effects of scaling. As a result, we only obtain bounds on the expected number of communication rounds. These are demonstrated in Section 5.5.2 for binary classification with logistic regression and a smoothed hinge loss.

5.5.1 Application to linear regression

We consider linear regression with quadratic regularization (ridge regression). More specifically, we minimize the overall empirical loss function

$$f(w) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{i,j} - w^T x_{i,j})^2 + \frac{\lambda}{2}\|w\|_2^2, \tag{5.35}$$

where the i.i.d. instances $(x_{i,j}, y_{i,j})$ are sampled from $\mathcal{X} \times \mathcal{Y}$. We assume that $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$ are bounded: there exist constants B_x and B_y such that $\|x\|_2 \leq B_x$ and $|y| \leq B_y$ for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$. It can be shown that the least-squares loss $\phi(w, (x, y)) = (y - w^T x)^2$ satisfies Assumption I with

$$V_0 = B_y^2, \qquad G = 2B_x \left(B_y + B_x B_y \sqrt{2/\lambda} \right), \qquad L = \lambda + 2B_x^2, \qquad M = 0.$$

Thus we can apply Theorems 8 and 9 to obtain an expectation bound on the number of communication rounds for DiSCO. For linear regression, however, we can obtain a stronger result.

Since f is a quadratic function, it is self-concordant with parameter 0, and by definition also standard self-concordant (with parameter 2). In this case, we do not need to rescale the objective function, and can regard the initial gap $f(w_0) - f(w_\star)$ as a constant. As a consequence, we can directly apply Theorem 6 and Corollary 7 to obtain a high probability bound on the communication complexity, which is stronger than the expectation bounds in Theorems 8 and 9. In particular, Theorem 6 states that if

$$\left\| f_1''(w_k) - f''(w_k) \right\|_2 \leq \mu, \quad \text{for all} \quad k = 0, 1, 2, \ldots, \tag{5.36}$$

then the number of communication rounds T is bounded as

$$T \leq 1 + \left(\left\lceil \frac{f(w_0) - f(w_\star)}{\omega(1/6)} \right\rceil + \left\lceil \log_2 \left(\frac{2\omega(1/6)}{\epsilon} \right) \right\rceil \right) \left(2 + \sqrt{1 + \frac{2\mu}{\lambda}} \log \left(\frac{2L}{\beta\lambda} \right) \right).$$

Since there is no scaling, the initial gap $f(w_0) - f(w_\star)$ can be considered as a constant. For example, we can simply pick $w_0 = 0$ and have

$$f(0) - f(w_\star) \leq f(0) = \frac{1}{N} \sum_{i=1}^{N} y_i^2 \leq B_y^2.$$

By Corollary 7 and the fact that $M = 0$ for quadratic functions, the condition (5.36) holds with probability at least $1 - \delta$ if we choose

$$\mu = \sqrt{\frac{32L^2 d}{n}} \sqrt{\frac{\log(md/\delta)}{d}} = \frac{8L}{\sqrt{n}} \sqrt{2 \log(md/\delta)}. \tag{5.37}$$

Further using $L \leq \lambda + 2B_x^2$, we obtain the following corollary.

Corollary 8. *Suppose we apply DiSCO (Algorithm 3) to minimize $f(w)$ defined in (5.35) with the input parameter μ in (5.37), and let T be the total number of communication rounds required to find an ϵ-optimal solution. With probability at least $1 - \delta$, we have*

$$T = \widetilde{\mathcal{O}}\left(\left(1 + \frac{B_x}{\lambda^{1/2} n^{1/4}} \right) \log(1/\epsilon) \log(md/\delta) \right). \tag{5.38}$$

We note that the same conclusion also holds for the adaptive DiSCO algorithm (Algorithm 4), where we do not need to specify the input parameter μ based on (5.37). For the adaptive DiSCO algorithm, the bound in (5.38) holds for any $\delta \in (0, 1)$.

The communication complexity guaranteed by Corollary 8 is strictly better than that of distributed implementation of the accelerated gradient method and ADMM (*cf.* Table 5.1). If we choose $\lambda = \Theta(1/\sqrt{mn})$, then Corollary 8 implies

$$T = \widetilde{\mathcal{O}}\left(m^{1/4} \log(1/\epsilon)\right)$$

with high probability. The DANE algorithm [190], under the same setting, converges in $\widetilde{\mathcal{O}}(m \log(1/\epsilon))$ iterations with high probability (and each iteration requires two rounds of communication). Thus DiSCO enjoys a better communication efficiency.

5.5.2 Application to binary classification

For binary classification, we consider the following regularized empirical loss function

$$\ell(w) =: \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \varphi(y_{i,j} w^T x_{i,j}) + \frac{\gamma}{2} \|w\|_2^2, \tag{5.39}$$

where $x_{i,j} \in \mathcal{X} \subset \mathbb{R}^d$, $y_{i,j} \in \{-1, 1\}$, and $\varphi : \mathbb{R} \to \mathbb{R}$ is a convex surrogate function for the binary loss. We further assume that the elements of $\mathcal{X}$ are bounded, *i.e.*, we have $\sup_{x \in \mathcal{X}} \|x\|_2 \leq B$ for some finite B.

Under the above assumptions, Lemma 2 gives conditions on φ for ℓ to be self-concordant. As we have seen in Section 2.3, the function ℓ usually needs to be scaled by a large factor to become standard self-concordant. Let the scaling factor be η, we can use DiSCO to minimize the scaled function $f(w) = \eta \ell(w)$. Next we discuss the theoretical implications for logistic regression and the smoothed hinge loss constructed in Section 2.3. These results are summarized in Table 5.1.

Logistic Regression For logistic regression, we have $\varphi(t) = \log(1 + e^{-t})$. In Section 2.3, we have shown that the logistic loss satisfies the condition of Lemma 2 with $Q = 1$ and $\alpha = 0$. Consequently, with the factor $\eta = \frac{B^2}{4\gamma}$, the rescaled function $f(w) = \eta \ell(w)$ is standard self-concordant. If we express f in the standard form

$$f(w) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \phi(y_{i,j} w^T x_{i,j}) + \frac{\lambda}{2} \|w\|_2^2, \tag{5.40}$$

then we have $\phi(w, (x, y)) = \eta \varphi(y w^T x)$ and $\lambda = \eta\gamma$. It is easy to check that Assumption I holds with

$$V_0 = \eta \log(2), \qquad G = \eta B, \qquad L = \eta(B^2/4 + \gamma), \qquad M = \eta B^3/10,$$

which all containing the scaling factor η. Plugging these scaled constants into Theorems 8 and 9, we have the following corollary.

Corollary 9. *For logistic regression, the number of communication rounds required by DiSCO to find an ϵ-optimal solution is bounded by*

$$\mathbb{E}[T] = \widetilde{\mathcal{O}}\Big(\Big(\log(1/\epsilon) + \frac{B^3 D}{\gamma n^{1/2}}\Big)\Big(1 + \frac{Bd^{1/4}}{\gamma^{1/2} n^{1/4}}\Big)\Big).$$

In the specific case when $\gamma = \Theta(1/\sqrt{mn})$, Corollary 9 implies

$$\mathbb{E}[T] = \widetilde{\mathcal{O}}\left(m^{3/4} d^{1/4} + m^{1/4} d^{1/4} \log(1/\epsilon)\right).$$

If we ignore logarithmic terms, then the expected number of communication rounds is independent of the sample size n, and only grows slowly with the number of machines m.

Smoothed Hinge Loss We consider minimizing $\ell(w)$ in (5.39) where the loss function φ is the smoothed hinge loss defined in (2.7), which depends on a parameter $p \geq 3$. Using Lemma 2, we have shown in Section 2.3 that $\ell(w)$ is self-concordant with parameter M_p given in (2.8). As a consequence, by choosing

$$\eta = \frac{M_p^2}{4} = \frac{(p-2)^2 B^{2+\frac{4}{p-2}}}{4\gamma^{1+\frac{2}{p-2}}},$$

the function $f(w) = \eta\ell(w)$ is standard self-concordant. If we express f in the form of (5.40), then $\phi(w, (x, y)) = \eta\varphi_p(yw^T x)$ and $\lambda = \eta\gamma$. It is easy to verify that Assumption I holds with

$$V_0 = \eta, \qquad G = \eta B, \qquad L = \eta(B^2 + \lambda), \qquad M = \eta(p-2)B^3.$$

If we choose $p = 2 + \log(1/\gamma)$, then applying Theorems 8 and 9 yields the following result.

Corollary 10. *For the smoothed hinge loss φ_p defined in (2.7) with $p = 2 + \log(1/\gamma)$, the total number of communication rounds required by DiSCO to find an ϵ-optimal solution is bounded by*

$$\mathbb{E}[T] = \widetilde{\mathcal{O}}\Big(\Big(\log(1/\epsilon) + \frac{B^3 D}{\gamma n^{1/2}}\Big)\Big(1 + \frac{Bd^{1/4}}{\gamma^{1/2} n^{1/4}}\Big)\Big).$$

Thus, the smoothed hinge loss enjoys the same communication efficiency as the logistic loss.

5.6 Numerical experiments

In this section, we conduct numerical experiments to compare the DiSCO algorithm with several state-of-the-art distributed optimization algorithms: the ADMM algorithm (*e.g.*, [34]), the accelerated full gradient method (AFG) [154, Section 2.2], the L-BFGS quasi-Newton method (*e.g.*, [159, Section 7.2]), and the DANE algorithm [190].

Dataset name	number of samples	number of features	sparsity
Covtype	581,012	54	22%
RCV1	20,242	47,236	0.16%
News20	19,996	1,355,191	0.04%

Table 5.2: Summary of three binary classification datasets.

The algorithms ADMM, AFG and L-BFGS are well known and each has a rich literature. In particular, using ADMM for empirical risk minimization in a distributed setting is straightforward; see [34, Section 8]. For AFG and L-BFGS, we use the simple distributed implementation discussed in Section 5.1: at each iteration k, each machine computes the local gradients $f_i'(w_k)$ and sends it to the master machine to form $f'(w_k) = (1/m)\sum_{i=1}^m f_i'(w_k)$, and the master machine executes the main steps of the algorithm to compute w_{k+1}. The iteration complexities of these algorithms stay the same as their classical analysis for a centralized implementation, and each iteration usually involves one or two rounds of communication.

Here we briefly describe the DANE (Distributed Approximate NEwton) algorithm proposed by Shamir et al. [190]. Each iteration of DANE takes two rounds of communication to compute w_{k+1} from w_k. The first round of communication is used to compute the gradient $f'(w_k) = (1/m)\sum_{i=1}^m f_i'(w_k)$. Then each machine solves the local minimization problem

$$v_{k+1,i} = \arg\min_{w\in\mathbb{R}^d} \left\{ f_i(w) - \langle f_i'(w_k) - f'(w_k), w\rangle + \frac{\mu}{2}\|w - w_k\|_2^2 \right\},$$

and take a second round of communication to compute $w_{k+1} = (1/m)\sum_{i=1}^m v_{k+1,i}$. Here $\mu \geq 0$ is a regularization parameter with a similar role as in DiSCO. For minimizing the quadratic loss in (5.35), the iteration complexity of DANE is $\widetilde{\mathcal{O}}((L/\lambda)^2 n^{-1}\log(1/\epsilon))$. As summarized in Table 5.1, if the condition number L/λ grows as $\sqrt{mn}$, then DANE is more efficient than AFG and ADMM when n is large. However, the same complexity cannot be guaranteed for minimizing non-quadratic loss functions. According to the analysis in [190], the convergence rate of DANE on non-quadratic functions might be as slow as the ordinary full gradient descent method.

5.6.1 Experiment setup

For comparison, we solve three binary classification tasks using logistic regression. The datasets are obtained from the LIBSVM datasets [43] and summarized in Table 5.2. These datasets are selected to cover different relations between the sample size $N = mn$ and the feature dimensionality d: $N \gg d$ (Covtype [27]), $N \approx d$ (RCV1 [120]) and $N \ll d$ (News20 [103, 113]). For each dataset, our goal is to minimize the regularized empirical loss function:

$$\ell(w) = \frac{1}{N}\sum_{i=1}^N \log(1 + \exp(-y_i(w^T x_i))) + \frac{\gamma}{2}\|w\|_2^2$$

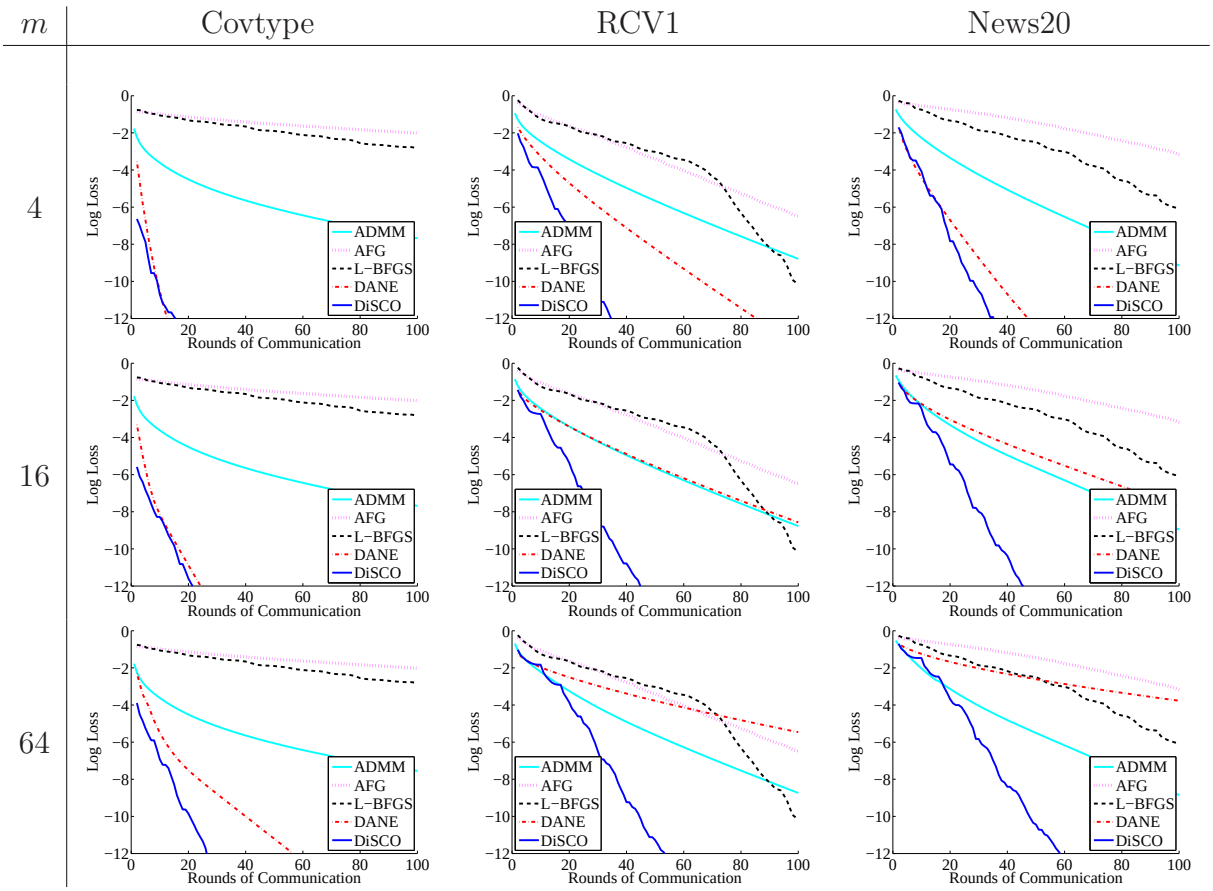
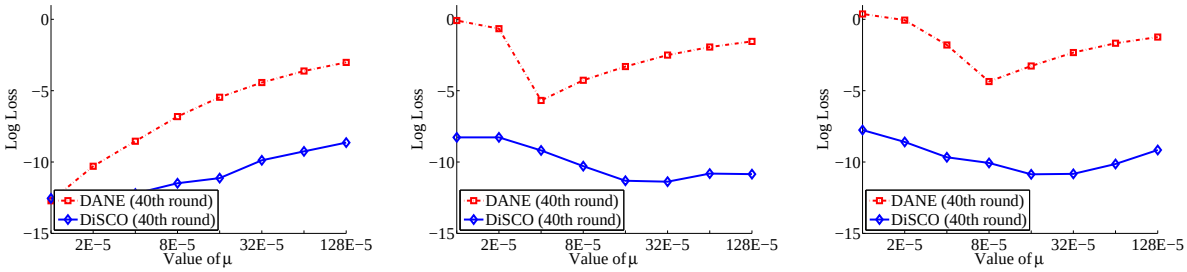


Figure 5.1. Comparing DiSCO with other distributed optimization algorithms. We splits each dataset evenly to m machines, with $m \in \{4, 16, 64\}$. Each plot above shows the reduction of the logarithmic gap $\log_{10}(\ell(\widehat{W}) - \ell(w_\star))$ (the vertical axis) versus the number of communication rounds (the horizontal axis) taken by each algorithm.

where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. The data have been normalized so that $\|x_i\| = 1$ for all $i = 1, \ldots, N$. The regularization parameter is set to be $\gamma = 10^{-5}$.

We describe some implementation details. In Section 5.5.2, the theoretical analysis suggests that we scale the function $\ell(w)$ by a factor $\eta = B^2/(4\gamma)$. Here we have $B = 1$ due to the normalization of the data. In practice, we find that DiSCO converges faster without rescaling. Thus, we use $\eta = 1$ for all experiments. For Algorithm 3, we choose the input parameters $\mu = m^{1/2}\mu_0$, where μ_0 is chosen manually. In particular, we used $\mu_0 = 0$ for Covtype, $\mu_0 = 4 \times 10^{-4}$ for RCV1, and $\mu_0 = 2 \times 10^{-4}$ for News20. For the distributed PCG method (Algorithm 2), we choose the stopping precision $\epsilon_k = \|f'(w_k)\|_2/10$.

Among other methods in comparison, we manually tune the penalty parameter of ADMM and the regularization parameter μ for DANE to optimize their performance. For AFG, we used an adaptive line search scheme [155, 126] to speed up its convergence. For L-BFGS, we adopted the memory size 30 (number of most recent iterates and gradients stored) as a



Covtype RCV1 News20

Figure 5.2. Comparing the sensitivity of DiSCO and DANE with respect to the regularization parameter μ, when the datasets are split on $m = 16$ machines. We varied μ from 10^{-5} to 128×10^{-5}. The vertical axis is the logarithmic gap $\log_{10}(\ell(\widehat{W}) - \ell(w_\star))$ after 40 rounds of communications.

general rule of thumb suggested in [159],

We want to evaluate DiSCO not only on w_k, but also in the middle of calculating v_k, to show its progress after each round of communication. To this end, we follow equation (5.12) to define an intermediate solution $\widehat{W}_k^t$ for each iteration t of the distributed PCG method (Algorithm 2):

$$\widehat{W}_k^t = w_k - \frac{v^{(t)}}{1 + \sqrt{\eta}\big(v^{(t)}\big)^T \ell''(w_k) v^{(t)}\big)^{1/2}},$$

and evaluate the associated objective function $\ell(\widehat{W}_k^t)$. This function value is treated as a measure of progress after each round of communication.

5.6.2 Performance evaluation

It is important to note that different algorithms take different number of communication rounds per iteration. ADMM requires one round of communication per iteration. For AFG and L-BFGS, each iteration consists of at least two rounds of communications: one for finding the descent direction, and another one or more for searching the stepsize. For DANE, there are also two rounds of communications per iteration, for computing the gradient and for aggregating the local solutions. For DiSCO, each iteration in the inner loop takes one round of communication, and there is an additional round of communication at the beginning of each inner loop. Since we are interested in the communication efficiency of the algorithms, we plot their progress in reducing the objective value with respect to the number of communication rounds taken.

We plot the performance of ADMM, AFG, L-BFGS, DANE and DiSCO in Figure 5.1. According to the plots, DiSCO converges substantially faster than ADMM and AFG. It is also notably faster than L-BFGS and DANE. In particular, the convergence speed (and the communication efficiency) of DiSCO is more robust to the number of machines in the distributed system. For $m = 4$, the performance of DiSCO is somewhat comparable to that

of DANE. As m grows to 16 and 64, the convergence of DANE becomes significantly slower, while the performance of DiSCO only degrades slightly. This coincides with the theoretical analysis: the iteration complexity of DANE is proportional to m, but the iteration complexity of DiSCO is proportional to $m^{1/4}$.

Since both DANE and DiSCO take a regularization parameter μ, we study their sensitivity to the choices of this parameter. Figure 5.2 shows the performance of DANE and DiSCO with the value of μ varying from 10^{-5} to 128×10^{-5}. We observe that the curves of DiSCO are relatively smooth and stable. In contrast, the curves of DANE exhibit sharp valley at particular values of μ. This suggests that DiSCO is more robust to the non-optimal choice of parameters.

5.7 Proofs of technical results

5.7.1 Proof of Theorem 5

First, we recall the definitions of the two auxiliary functions

$$\begin{aligned} \omega(t) &= t - \log(1+t), \qquad t \geq 0, \\ \omega_*(t) &= -t - \log(1-t), \qquad 0 \leq t < 1, \end{aligned}$$

which form a pair of convex conjugate functions.

We notice that Step 2 of Algorithm 1 is equivalent to

$$w_{k+1} - w_k = \frac{v_k}{1+\delta_k} = \frac{v_k}{1+\|\widetilde{v}_k\|_2},$$

which implies

$$\|[f''(w_k)]^{1/2}(w_{k+1} - w_k)\|_2 = \frac{\|\widetilde{v}_k\|_2}{1+\|\widetilde{v}_k\|_2} < 1. \tag{5.41}$$

When inequality (5.41) holds, Nesterov [154, Theorem 4.1.8] has shown that

$$f(w_{k+1}) \leq f(w_k) + \langle f'(w_k), w_{k+1} - w_k \rangle + \omega_*\big(\|[f''(w_k)]^{1/2}(w_{k+1} - w_k)\|_2\big).$$

Using the definition of functions ω and ω_*, and with some algebraic operations, we obtain

$$\begin{aligned} f(w_{k+1}) &\leq f(w_k) - \frac{\langle \widetilde{u}_k, \widetilde{v}_k \rangle}{1+\|\widetilde{v}_k\|_2} - \frac{\|\widetilde{v}_k\|_2}{1+\|\widetilde{v}_k\|_2} + \log(1+\|\widetilde{v}_k\|_2) \\ &= f(w_k) - \omega(\|\widetilde{u}_k\|_2) + \big(\omega(\|\widetilde{u}_k\|_2) - \omega(\|\widetilde{v}_k\|_2)\big) + \frac{\langle \widetilde{v}_k - \widetilde{u}_k, \widetilde{v}_k \rangle}{1+\|\widetilde{v}_k\|_2}. \end{aligned} \tag{5.42}$$

By the second-order mean-value theorem, we have

$$\omega(\|\widetilde{u}_k\|_2) - \omega(\|\widetilde{v}_k\|_2) = \omega'(\|\widetilde{v}_k\|_2)(\|\widetilde{u}_k\|_2 - \|\widetilde{v}_k\|_2) + \frac{1}{2}\omega''(t)\,(\|\widetilde{u}_k\|_2 - \|\widetilde{v}_k\|_2)^2$$

for some t satisfying

$$\min\{\|\widetilde{u}_k\|_2, \|\widetilde{v}_k\|_2\} \leq t \leq \max\{\|\widetilde{u}_k\|_2, \|\widetilde{v}_k\|_2\}.$$

Using the inequality (5.7), we can upper bound the second derivative $\omega''(t)$ as

$$\omega''(t) = \frac{1}{(1+t)^2} \leq \frac{1}{1+t} \leq \frac{1}{1+\min\{\|\widetilde{u}_k\|_2, \|\widetilde{v}_k\|_2\}} \leq \frac{1}{1+(1-\beta)\|\widetilde{u}_k\|_2}.$$

Therefore,

$$\begin{aligned}
\omega(\|\widetilde{u}_k\|_2) - \omega(\|\widetilde{v}_k\|_2) &= \frac{(\|\widetilde{u}_k\|_2 - \|\widetilde{v}_k\|_2)\|\widetilde{v}_k\|_2}{1+\|\widetilde{v}_k\|_2} + \frac{1}{2}\omega''(t)\,(\|\widetilde{u}_k\|_2 - \|\widetilde{v}_k\|_2)^2 \\
&\leq \frac{\|\widetilde{u}_k - \widetilde{v}_k\|_2\|\widetilde{v}_k\|_2}{1+(1-\beta)\|\widetilde{u}_k\|_2} + \frac{(1/2)\|\widetilde{u}_k - \widetilde{v}_k\|_2^2}{1+(1-\beta)\|\widetilde{u}_k\|_2} \\
&\leq \frac{\beta(1+\beta)\|\widetilde{u}_k\|_2^2 + (1/2)\beta^2\|\widetilde{u}_k\|_2^2}{1+(1-\beta)\|\widetilde{u}_k\|_2}
\end{aligned}$$

In addition, we have

$$\frac{\langle \widetilde{v}_k - \widetilde{u}_k, \widetilde{v}_k\rangle}{1+\|\widetilde{v}_k\|_2} \leq \frac{\|\widetilde{u}_k - \widetilde{v}_k\|_2\|\widetilde{v}_k\|_2}{1+\|\widetilde{v}_k\|_2} \leq \frac{\beta(1+\beta)\|\widetilde{u}_k\|_2^2}{1+(1-\beta)\|\widetilde{u}_k\|_2}.$$

Combining the two inequalities above, and using the relation $t^2/(1+t) \leq 2\omega(t)$ for all $t \geq 0$, we obtain

$$\begin{aligned}
\omega(\|\widetilde{u}_k\|_2) - \omega(\|\widetilde{v}_k\|_2) + \frac{\langle \widetilde{v}_k - \widetilde{u}_k, \widetilde{v}_k\rangle}{1+\|\widetilde{v}_k\|_2} &\leq \big(2\beta(1+\beta) + (1/2)\beta^2\big)\,\frac{\|\widetilde{u}_k\|_2^2}{1+(1-\beta)\|\widetilde{u}_k\|_2} \\
&= \left(\frac{2\beta + (5/2)\beta^2}{(1-\beta)^2}\right)\frac{(1-\beta)^2\|\widetilde{u}_k\|_2^2}{1+(1-\beta)\|\widetilde{u}_k\|_2} \\
&\leq \left(\frac{2\beta + (5/2)\beta^2}{(1-\beta)^2}\right) 2\omega\big((1-\beta)\|\widetilde{u}_k\|_2\big) \\
&\leq \left(\frac{4\beta + 5\beta^2}{1-\beta}\right)\omega\big(\|\widetilde{u}_k\|_2\big).
\end{aligned}$$

In the last inequality above, we used the fact that for any $t \geq 0$ we have $\omega((1-\beta)t) \leq (1-\beta)\omega(t)$, which is the result of convexity of $\omega(t)$ and $\omega(0) = 0$; more specifically,

$$\omega((1-\beta)t) = \omega(\beta\cdot 0 + (1-\beta)t) \leq \beta\omega(0) + (1-\beta)\omega(t) = (1-\beta)\omega(t).$$

Substituting the above upper bound into inequality (5.42) yields

$$f(w_{k+1}) \leq f(w_k) - \left(1 - \frac{4\beta + 5\beta^2}{1-\beta}\right)\omega(\|\widetilde{u}_k\|_2). \tag{5.43}$$

With inequality (5.43), we are ready to prove the statements of the lemma. In particular, Part (a) of the Lemma holds for any $0 \leq \beta \leq 1/10$.

For part (b), we assume that $\|\widetilde{u}_k\|_2 \leq 1/6$. According to [154, Theorem 4.1.13], when $\|\widetilde{u}_k\|_2 < 1$, it holds that for every $k \geq 0$,

$$\omega(\|\widetilde{u}_k\|_2) \leq f(w_k) - f(w_\star) \leq \omega_*(\|\widetilde{u}_k\|_2). \tag{5.44}$$

Combining this sandwich inequality with inequality (5.43), we have

$$\begin{aligned}
\omega(\|\widetilde{u}_{k+1}\|_2) &\leq f(w_{k+1}) - f(w_\star) \\
&\leq f(w_k) - f(w_\star) - \omega(\|\widetilde{u}_k\|_2) + \frac{4\beta + 5\beta^2}{1-\beta}\omega(\|\widetilde{u}_k\|_2) \\
&\leq \omega_*(\|\widetilde{u}_k\|_2) - \omega(\|\widetilde{u}_k\|_2) + \frac{4\beta + 5\beta^2}{1-\beta}\omega(\|\widetilde{u}_k\|_2).
\end{aligned} \tag{5.45}$$

It is easy to verify that $\omega_*(t) - \omega(t) \leq 0.26\,\omega(t)$ for all $t \leq 1/6$, and $(4\beta + 5\beta^2)/(1-\beta) \leq 0.23$ if $\beta \leq 1/20$. Applying these two inequalities to inequality (5.45) completes the proof.

It should be clear that other combinations of the value of β and bound on $\|\widetilde{u}_k\|_2$ are also possible. For example, for $\beta = 1/10$ and $\|\widetilde{u}_k\|_2 \leq 1/10$, we have $\omega(\|\widetilde{u}_{k+1}\|_2) \leq 0.65\,\omega(\|\widetilde{u}_k\|_2)$.

5.7.2 Proof of Lemma 21

It suffices to show that the algorithm terminates at iteration $t \leq T_\mu - 1$, because when the algorithm terminates, it outputs a vector v_k which satisfies $\|Hv_k - f'(w_k)\|_2 = \|r^{(t+1)}\|_2 \leq \epsilon_k$. Denote by $v^* = H^{-1}f'(w_k)$ the solution of the linear system $Hv_k = f'(w_k)$. By the classical analysis on the preconditioned conjugate gradient method (e.g., [133, 8]), Algorithm 2 has the convergence rate

$$(v^{(t)} - v^*)^T H(v^{(t)} - v^*) \leq 4\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2t} (v^*)^T H v^*, \tag{5.46}$$

where $\kappa = 1 + 2\mu/\rho$ is the condition number of $P^{-1}H$ given in (5.20). For the left-hand side of inequality (5.46), we have

$$(v^{(t)} - v^*)^T H(v^{(t)} - v^*) = (r^{(t)})^T H^{-1} r^{(t)} \geq \frac{\|r^{(t)}\|_2^2}{L}.$$

For the right-hand side of inequality (5.46), we have

$$(v^*)^T H v^* = (f'(w_k))^T H^{-1} f'(w_k) \leq \frac{\|f'(w_k)\|_2^2}{\rho}.$$

Combining the above two inequalities with inequality (5.46), we obtain

$$\|r^{(t)}\|_2 \leq 2\sqrt{\frac{L}{\lambda}}\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^t \|f'(w_k)\|_2 \leq 2\sqrt{\frac{L}{\lambda}}\left(1 - \sqrt{\frac{\rho}{\rho + 2\mu}}\right)^t \|f'(w_k)\|_2.$$

To guarantee that $\|r^{(t)}\|_2 \le \epsilon_k$, it suffices to have

$$t \;\ge\; \frac{\log\left(\frac{2\sqrt{L/\lambda}\|f'(w_k)\|_2}{\epsilon_k}\right)}{-\log\left(1-\sqrt{\frac{\rho}{\rho+2\mu}}\right)} \;\ge\; \sqrt{1+\frac{2\mu}{\lambda}}\,\log\left(\frac{2\sqrt{L/\lambda}\|f'(w_k)\|_2}{\epsilon_k}\right),$$

where in the last inequality we used $-\log(1-x) \ge x$ for $0 < x < 1$. Comparing with the definition of T_μ, this is the desired result.

5.7.3 Proof of Lemma 22

First, we prove inequality (5.25). Recall that $w_\star$ and $\widehat{W}_i$ minimizes $f(w)$ and $f_i(w) + \frac{\rho}{2}\|w\|_2^2$. Since both function are ρ-strongly convex, we have

$$\begin{aligned}
&\frac{\rho}{2}\|w_\star\|_2^2 \le f(w_\star) \le f(0) \le V_0,\\
&\frac{\rho}{2}\|\widehat{W}_i\|_2^2 \le f_i(\widehat{W}_i) + \frac{\rho}{2}\|\widehat{W}_i\|_2^2 \le f_i(0) \le V_0,
\end{aligned}$$

which implies $\|w_\star\|_2 \le \sqrt{\frac{2V_0}{\rho}}$ and $\|\widehat{W}_i\|_2 \le \sqrt{\frac{2V_0}{\rho}}$. Then inequality (5.25) follows since w_0 is the average over $\{\widehat{W}_i\}_{i=1}^m$.

In the rest of Section 5.7.3, we prove inequality (5.26). Let z be a random variable in $\mathcal{Z} \subset \mathbb{R}^p$ with an unknown probability distribution. We define a regularized population risk:

$$R(w) = \mathbb{E}_z[\phi(w,z)] + \frac{\lambda+\rho}{2}\|w\|_2^2.$$

Let S be a set of n i.i.d. samples in $\mathcal{Z}$ from the same distribution. We define a regularized empirical risk

$$r_S(w) = \frac{1}{n}\sum_{z\in S}\phi(w,z) + \frac{\lambda+\rho}{2}\|w\|_2^2,$$

and its minimizer

$$\widehat{W}_S = \arg\min_w \; r_S(w).$$

The following lemma states that the population risk of $\widehat{W}_S$ is very close to its empirical risk. The proof is based on the notion of *stability* of regularized empirical risk minimization [32].

Lemma 24. *Suppose Assumption I holds and S is a set of n i.i.d. samples in $\mathcal{Z}$. Then we have*

$$\mathbb{E}_S\big[R(\widehat{W}_S) - r_S(\widehat{W}_S)\big] \le \frac{2G^2}{\rho n}.$$

Proof Let $S = \{z_1, \ldots, z_n\}$. For any $k \in \{1, \ldots, n\}$, we define a modified training set $S^{(k)}$ by replacing z_k with another sample $\widetilde{z}_k$, which is drawn from the same distribution and is independent of S. The empirical risk on $S^{(k)}$ is defined as

$$r_S^{(k)}(w) = \frac{1}{n} \sum_{z \in S^{(k)}} \phi(w, z) + \frac{\lambda + \rho}{2} \|w\|_2^2.$$

and let $\widehat{W}_S^{(k)} = \arg\min_w r_S^{(k)}(w)$. Since both r_S and $r_S^{(k)}$ are ρ-strongly convex, we have

$$r_S(\widehat{W}_S^{(k)}) - r_S(\widehat{W}_S) \geq \frac{\rho}{2} \|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2^2$$
$$r_S^{(k)}(\widehat{W}_S) - r_S^{(k)}(\widehat{W}_S^{(k)}) \geq \frac{\rho}{2} \|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2^2.$$

Summing the above two inequalities, and noticing that

$$r_S(w) - r_S^{(k)}(w) = \frac{1}{n}(\phi(w, z_k) - \phi(w, \widetilde{z}_k)),$$

we have

$$\|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2^2 \leq \frac{1}{\rho n} \left(\phi(\widehat{W}_S^{(k)}, z_k) - \phi(\widehat{W}_S^{(k)}, \widetilde{z}_k) - \phi(\widehat{W}_S, z_k) + \phi(\widehat{W}_S, \widetilde{z}_k) \right). \tag{5.47}$$

By Assumption I (ii) and the facts $\|\widehat{W}_S\|_2 \leq \sqrt{2V_0/\lambda}$ and $\|\widehat{W}_S^{(k)}\|_2 \leq \sqrt{2V_0/\lambda}$, we have

$$\left|\phi(\widehat{W}_S^{(k)}, z) - \phi(\widehat{W}_S, z)\right| \leq G \|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2, \qquad \forall\, z \in \mathcal{Z}.$$

Combining the above Lipschitz condition with (5.47), we obtain

$$\|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2^2 \leq \frac{2G}{\rho n} \|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2.$$

As a consequence, we have $\|\widehat{W}_S^{(k)} - \widehat{W}_S\|_2 \leq \frac{2G}{\rho n}$, and therefore

$$\left|\phi(\widehat{W}_S^{(k)}, z) - \phi(\widehat{W}_S, z)\right| \leq \frac{2G^2}{\rho n}, \qquad \forall\, z \in \mathcal{Z}. \tag{5.48}$$

In the terminology of learning theory, this means that empirical minimization over the regularized loss $r_S(w)$ has *uniform stability* $2G^2/(\rho n)$ with respect to the loss function ϕ; see [32].

For any fixed $k \in \{1, \ldots, n\}$, since $\widetilde{z}_k$ is independent of S, we have

$$\begin{aligned}
\mathbb{E}_S\big[R(\widehat{W}_S) - r_S(\widehat{W}_S)\big] &= \mathbb{E}_S\left[\mathbb{E}_{\widetilde{z}_k}[\phi(\widehat{W}_S, \widetilde{z}_k)] - \frac{1}{n}\sum_{j=1}^{n} \phi(\widehat{W}_S, z_j)\right] \\
&= \mathbb{E}_{S,\widetilde{z}_k}\big[\phi(\widehat{W}_S, \widetilde{z}_k) - \phi(\widehat{W}_S, z_k)\big] \\
&= \mathbb{E}_{S,\widetilde{z}_k}\big[\phi(\widehat{W}_S, \widetilde{z}_k) - \phi(\widehat{W}_S^{(k)}, \widetilde{z}_k)\big],
\end{aligned}$$

where the second equality used the fact that $\mathbb{E}_S[\phi(\widehat{W}_S, z_j)$ has the same value for all $j = 1, \ldots, n$, and the third equality used the symmetry between the pairs (S, z_k) and $(S^{(k)}, \widetilde{z}_k)$ (also known as the *renaming* trick; see [32, Lemma 7]). Combining the above equality with (5.48) yields the desired result. □

Next, we consider a distributed system with m machines, where each machine has a local dataset S_i of size n, for $i = 1, \ldots, m$. To simplify notation, we denote the local regularized empirical loss function and its minimizer by $r_i(w)$ and $\widehat{W}_i$, respectively. We would like to bound the excessive error when applying $\widehat{W}_i$ to a different dataset S_j. Notice that

$$\begin{aligned}&\mathbb{E}_{S_i,S_j}\big[r_j(\widehat{W}_i) - r_j(\widehat{W}_j)\big] \\ &= \underbrace{\mathbb{E}_{S_i,S_j}\big[r_j(\widehat{W}_i) - r_i(\widehat{W}_i)\big]}_{v_1} + \underbrace{\mathbb{E}_{S_i,S_j}\big[r_i(\widehat{W}_i) - r_j(\widehat{W}_R)\big]}_{v_2} + \underbrace{\mathbb{E}_{S_j}\big[r_j(\widehat{W}_R) - r_j(\widehat{W}_j)\big]}_{v_3}\end{aligned} \tag{5.49}$$

where $\widehat{W}_R$ is the constant vector minimizing $R(w)$. Since S_i and S_j are independent, we have

$$v_1 = \mathbb{E}_{S_i}\big[\mathbb{E}_{S_j}[r_j(\widehat{W}_i)] - r_i(\widehat{W}_i)\big] = \mathbb{E}_{S_i}\big[R(\widehat{W}_i) - r_i(\widehat{W}_i)\big] \le \frac{2G^2}{\rho n},$$

where the inequality is due to Lemma 24. For the second term, we have

$$v_2 = \mathbb{E}_{S_i}\big[r_i(\widehat{W}_i) - \mathbb{E}_{S_j}[r_j(\widehat{W}_R)]\big] = \mathbb{E}_{S_i}\big[r_i(\widehat{W}_i) - r_i(\widehat{W}_R)\big] \le 0.$$

It remains to bound the third term v_3. We first use the strong convexity of r_j to obtain (e.g., [154, Theorem 2.1.10])

$$r_j(\widehat{W}_R) - r_j(\widehat{W}_j) \le \frac{\|r_j'(\widehat{W}_R)\|_2^2}{2\rho}, \tag{5.50}$$

where $r_j'(\widehat{W}_R)$ denotes the gradient of r_j at $\widehat{W}_R$. If we index the elements of S_j by $z_1, \ldots, z_n$, then

$$r_j'(\widehat{W}_R) = \frac{1}{n}\sum_{k=1}^{n}\Big(\phi'(\widehat{W}_R, z_k) + (\rho + \rho)\widehat{W}_R\Big). \tag{5.51}$$

By the optimality condition of $\widehat{W}_R = \arg\min_w R(w)$, we have for any $k \in \{1, \ldots, n\}$,

$$\mathbb{E}_{z_k}\big[\phi'(\widehat{W}_R, z_k) + (\lambda + \rho)\widehat{W}_R\big] = 0.$$

Therefore, according to (5.51), the gradient $r_j(\widehat{W}_R)$ is the average of n independent and zero-mean random vectors. Combining (5.50) and (5.51) with the definition of v_3 in (5.49),

we have

$$\begin{aligned} v_3 &\le \frac{\mathbb{E}_{S_j}\left[\sum_{k=1}^n \|\phi'(\widehat{W}_R, z_k) + (\rho+\rho)\widehat{W}_R\|_2^2\right]}{2\rho n^2} \\ &= \frac{\sum_{k=1}^n \mathbb{E}_{S_j}\left[\|\phi'(\widehat{W}_R, z_k) + (\rho+\rho)\widehat{W}_R\|_2^2\right]}{2\rho n^2} \\ &\le \frac{\sum_{k=1}^n \mathbb{E}[\|\phi'(\widehat{W}_R, z_k)\|_2^2]}{2\rho n^2} \\ &\le \frac{G^2}{2\rho n}. \end{aligned}$$

In the equality above, we used the fact that $\phi'(\widehat{W}_R, z_k)+(\lambda+\rho)\widehat{W}_R$ are i.i.d. zero-mean random variables; so the variance of their sum equals the sum of their variances. The last inequality above is due to Assumption I (ii) and the fact that $\|\widehat{W}_R\|_2 \le \sqrt{2V_0/(\lambda+\rho)} \le \sqrt{2V_0/\lambda}$. Combining the upper bounds for v_1, v_2 and v_3, we have

$$\mathbb{E}_{S_i,S_j}\left[r_j(\widehat{W}_i) - r_j(\widehat{W}_j)\right] \le \frac{3G^2}{\rho n}. \tag{5.52}$$

Recall the definition of $f(w)$ as

$$f(w) = \frac{1}{mn}\sum_{i=1}^m\sum_{k=1}^n \phi(w, z_{i,k}) + \frac{\lambda}{2}\|w\|_2^2,$$

where $z_{i,k}$ denotes the kth sample at machine i. Let $r(w) = \frac{1}{m}\sum_{j=1}^m r_j(w)$; then we have

$$r(w) = f(w) + \frac{\rho}{2}\|w\|_2^2. \tag{5.53}$$

We compare the value $r(\widehat{W}_i)$, for any $i \in \{1, \ldots, m\}$, with the minimum of $r(w)$:

$$\begin{aligned} r(\widehat{W}_i) - \min_w r(w) &= \frac{1}{m}\sum_{j=1}^m r_j(\widehat{W}_i) - \min_w \frac{1}{m}\sum_{j=1}^m r_j(w) \\ &\le \frac{1}{m}\sum_{j=1}^m r_j(\widehat{W}_i) - \frac{1}{m}\sum_{j=1}^m \min_w r_j(w) \\ &= \frac{1}{m}\sum_{j=1}^m \left(r_j(\widehat{W}_i) - r_j(\widehat{W}_j)\right). \end{aligned}$$

Taking expectation with respect to all the random data sets $S_1, \ldots, S_m$ and using (5.52), we obtain

$$\mathbb{E}[r(\widehat{W}_i) - \min_w r(w)] \le \frac{1}{m}\sum_{j=1}^n \mathbb{E}[r_j(\widehat{W}_i) - r_j(\widehat{W}_j)] \le \frac{3G^2}{\rho n}. \tag{5.54}$$

Finally, we bound the expected value of $f(\widehat{W}_i)$:

$$\begin{aligned}\mathbb{E}[f(\widehat{W}_i)] &\leq \mathbb{E}[r(\widehat{W}_i)] \leq \mathbb{E}\left[\min_w r(w)\right] + \frac{3G^2}{\rho n} \\ &\leq \mathbb{E}\left[f(w_\star) + \frac{\rho}{2}\|w_\star\|_2^2\right] + \frac{3G^2}{\rho n} \\ &\leq \mathbb{E}\left[f(w_\star)\right] + \frac{\rho D^2}{2} + \frac{3G^2}{\rho n},\end{aligned}$$

where the first inequality holds because of (5.53), the second inequality is due to (5.54), and the last inequality follows from the assumption that $\mathbb{E}[\|w_\star\|_2] \leq D^2$. Choosing $\rho = \sqrt{\frac{6G^2}{nD^2}}$ results in $\mathbb{E}[f(\widehat{W}_i) - f(w_\star)] \leq \frac{\sqrt{6}GD}{\sqrt{n}}$ for every $i \in \{1, \ldots, m\}$. Since $w_0 = \frac{1}{m}\sum_{i=1}^m \widehat{W}_i$, using the convexity of function f yields $\mathbb{E}[f(w_0) - f(w_\star)] \leq \frac{\sqrt{6}GD}{\sqrt{n}}$, which is the desired result.

5.7.4 Proof of Lemma 23

We consider the regularized empirical loss functions $f_i(w)$ defined in (5.24). For any two vectors $u, w \in \mathbb{R}^d$ satisfying $\|u - w\|_2 \leq \varepsilon$, Assumption I (iv) implies

$$\|f_i''(u) - f_i''(w)\|_2 \leq M\varepsilon.$$

Let $B(0, r)$ be the ball in $\mathbb{R}^d$ with radius r, centered at the origin. Let $N_\varepsilon^{\text{cov}}(B(0, r))$ be the *covering number* of $B(0, r)$ by balls of radius ε, *i.e.*, the minimum number of balls of radiusr ε required to cover $B(0, r)$. We also define $N_\varepsilon^{\text{pac}}(B(0, r))$ as the *packing number* of $B(0, r)$, *i.e.*, the maximum number of disjoint balls whose centers belong to $B(0, r)$. It is easy to verify that

$$N_\varepsilon^{\text{cov}}(B(0, r)) \leq N_{\varepsilon/2}^{\text{pac}}(B(0, r)) \leq (1 + 2r/\varepsilon)^d.$$

Therefore, there exist a set of points $U \subseteq \mathbb{R}^d$ with cardinality at most $(1 + 2r/\varepsilon)^d$, such that for any vector $w \in B(0, r)$, we have

$$\min_{u \in U} \|f_i''(w) - f_i''(u)\|_2 \leq M\varepsilon. \tag{5.55}$$

We consider an arbitrary point $u \in U$ and the associated Hessian matrices for the functions $f_i(w)$ defined in (5.24). We have

$$f_i''(u) = \frac{1}{n}\sum_{j=1}^n \left(\phi''(u, z_{i,j}) + \rho I\right), \qquad i = 1, \ldots, m.$$

The components of the above sum are i.i.d. matrices which are upper bounded by LI. By the matrix Hoeffding's inequality [137, Corollary 4.2], we have

$$\mathbb{P}\left[\|f_i''(u) - \mathbb{E}[f_i''(u)]\|_2 > t\right] \leq d \cdot e^{-\frac{nt^2}{2L^2}}.$$

Note that $\mathbb{E}[f_1''(w)] = \mathbb{E}[f''(w)]$ for any $w \in B(0,r)$. Using the triangular inequality and inequality (5.55), we obtain

$$\begin{aligned}\|f_1''(w) - f''(w)]\|_2 &\le \|f_1''(w) - \mathbb{E}[f_1''(w)]\|_2 + \|f''(w) - \mathbb{E}[f''(w)]\|_2 \\ &\le 2 \max_{i\in\{1,\dots,m\}} \|f_i''(w) - \mathbb{E}[f_i''(w)]\|_2 \\ &\le 2 \max_{i\in\{1,\dots,m\}} \Big(\max_{u\in U} \|f_i''(u) - \mathbb{E}[f_i''(u)]\|_2 + M\varepsilon \Big). \end{aligned} \tag{5.56}$$

Applying the union bound, we have with probability at least

$$1 - md(1 + 2r/\varepsilon)^d \cdot e^{-\frac{nt^2}{2L^2}},$$

the inequality $\|f_i''(u) - \mathbb{E}[f_i''(u)]\|_2 \le t$ holds for every $i \in \{1,\dots,m\}$ and every $u \in U$. Combining this probability bound with inequality (5.56), we have

$$\mathbb{P}\Big[\sup_{w\in B(0,r)} \|f_1''(w) - f''(w)\|_2 > 2t + 2M\varepsilon \Big] \le md\,(1 + 2r/\varepsilon)^d \cdot e^{-\frac{nt^2}{2L^2}}. \tag{5.57}$$

As the final step, we choose $\varepsilon = \frac{\sqrt{2}L}{\sqrt{n}M}$ and then choose t to make the right-hand side of inequality (5.57) equal to δ. This yields the desired result.

5.7.5 More analysis on the number of PCG iterations

Here we analyze the number of iterations of the distributed PCG method (Algorithm 2) when μ is misspecified, *i.e.*, when μ used in $P = H_1 + \mu I$ is not an upper bound on $\|H_1 - H\|_2$. For simplicity of discussion, we assume that Assumption H holds, $\|H_1 - H\|_2 \le L$ and $\mu \le L$. In this case, we can show (using similar arguments for proving Lemma 20):

$$\begin{aligned}\sigma_{\max}((H_1 + \mu I)^{-1} H) &\le \frac{2L}{L + \mu}, \\ \sigma_{\min}((H_1 + \mu I)^{-1} H) &\ge \frac{\lambda}{L + \mu + \lambda}. \end{aligned}$$

Hence the condition number of the preconditioned linear system is

$$\kappa_{\mu,L} = \frac{2L}{\lambda}\left(1 + \frac{\lambda}{L+\mu}\right) \le 2 + \frac{2L}{\lambda},$$

and the number of PCG iterations is bounded by (*cf.* Section 5.7.2)

$$\left\lceil \sqrt{\kappa_{\mu,L}} \log\left(\frac{2L}{\beta\lambda}\right) \right\rceil \le \sqrt{2 + \frac{2L}{\lambda}} \log\left(\frac{2L}{\beta\lambda}\right).$$

This gives the bound on number of PCG iterations in (5.30).

Part III

Theories of distributed computing

Chapter 6

Communication complexity of statistical estimation

In this chapter, we study the communication complexity of statistical estimation problems. Suppose we are interested in estimating some parameter $\theta(P)$ of an unknown distribution P, based on a dataset of N i.i.d. observations. In the distributed setting, there are m different machines, and each machine is assigned a subset of the sample of size $n = \lfloor \frac{N}{m} \rfloor$. Each machine may perform arbitrary operations on its own subset of data, and it then communicates results of these intermediate computations to the other processors or to a central fusion node. In this chapter, we try to answer the following question: what is the minimal number of bits that must be exchanged in order to achieve (up to constant factors) the optimal estimation error realized by a centralized scheme?

While, there is a very rich literature on statistical minimax (e.g. [96, 223, 219, 205]), little of it characterizes the effects of limiting communication. In other areas, ranging from theoretical computer science [221, 2, 110], decentralized detection and estimation (e.g., [204, 136]), to information theory (e.g., [85, 69]), there is of course a substantial literature on communication complexity. While related to these bodies of work, our problem formulation and results differ in several ways.

- In theoretical computer science [221, 2, 110], the prototypical problem is the distributed computation of a bivariate function $\theta : \mathcal{X} \times \mathcal{Y} \to \Theta$, defined on two discrete sets $\mathcal{X}$ and $\mathcal{Y}$, using a protocol that exchanges bits between processsors. The most classical problem is to find a protocol that computes $\theta(x, y)$ correctly for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and exchanges the smallest number of bits to do so. More recent work studies randomization and introduces information-theoretic measures for communication complexity [41, 13, 14], where the problem is to guarantee that $\theta(x, y)$ is computed correctly with high probability under a given (known) distribution P on x and y. In contrast, our goal is to recover characteristics an unknown distribution P based on observations drawn from P. Though this difference is somewhat subtle, it makes work on communication complexity difficult to apply in our settings. However, lower bounds

on the estimation of population quantities $\theta(P)$ based on communication-constrained observations—including those we present here—do imply lower bounds in classical communication complexity settings.

- Work in decentralized detection and estimation also studies limits of communication. For example, Tsitsiklis and Luo [203] provide lower bounds on the difficulty of distributed convex optimization, and Luo and Tsitsiklis [135] study limits on certain distributed algebraic computations. In these problems, as in other early work in communication complexity, data held by the distributed parties may be chosen adversarially, which precludes conclusions about statistical estimation. Other work in distributed control provides lower bounds on consensus and averaging, but in settings where messages sent are restricted to be of particular smooth forms [160]. Study of communication complexity has also given rise to interesting algorithmic schemes; for example, Luo [134] considers architectures in which machines may send only a single bit to a centralized processor; for certain problems, he shows that if each machine receives a single one-dimensional sample, it is possible to achieve the optimal centralized rate to within constant factors.

- Information theorists have also studied problems of studied estimation; for instance, see the paper [85] for an overview. In particular, this body of work focuses on the problem of testing a hypothesis or estimating a parameter from samples $\{(x_i, y_i)\}_{i=1}^n$ where $\{(x_i)\}_{i=1}^n$ and $\{(y_i)\}_{i=1}^n$ are correlated but stored separately in two machines. Han and Amari [85] study estimation error for encoding rates $R > 0$, or with sequences of rates R_n converging to zero as the sample size n increases. In contrast to these asymptotic formulations—which often allow more communication than is required to attain the centralized (unconstrained) minimax rates in our settings—our goal is to study fixed bounds on rates (say of the form $R_n \leq t/n$) for finite sample sizes n, and ask when it is possible to achieve the minimax statistical rate.

We formulate and study two decentralized variants of the centralized statistical minimax risk, one based on protocols that engage in only a single round of message-passing, and the other based on interactive protocols that can use multiple rounds of communication. The main question of interest is the following: how must the communication budget B scale as a function of the sample size n at each machine, the total number of machines m, and the problem dimension d so that the decentralized minimax risk matches the centralized version up to constant factors? For some problems, we exhibit an exponential gap between this communication requirement and the number of bits required to describe the problem solution (up to statistical precision); for instance, see Theorems 10 and 11 for results of this type. For example, we show that for problems such as location estimation in Gaussian and binomial families, the amount of communication must scale linearly in the product dm of the dimension number of machines m, which is exponentially larger than the $\mathcal{O}(d \log m)$ bits required to specify the problem or communicate its solution. To exhibit these gaps, we provide lower bounds using information-theoretic techniques, with the main novel ingredient

being certain forms of quantitative data processing inequalities. We also establish (nearly) sharp upper bounds, some of which are based on recent work by a subset of current authors on practical schemes for distributed estimation (see Zhang et al. [231]).

Notation: For a random variable X, we let P_X denote the probability measure on X, so that $P_X(S) = P(X \in S)$, and we abuse notation by writing p_X for the probability mass function or density of X, depending on the situation, so that $p_X(x) = P(X = x)$ in the discrete case and denotes the density of X at x when p_X is a density. We use log to denote log-base e and $\log_2$ for log in base 2. For discrete random variable X, we let $H(X) = -\sum_x p_X(x) \log p_X(x)$ denote the (Shannon) entropy (in ents), and for probability distributions P, Q on a set $\mathcal{X}$, with densities p, q with respect to a base measure μ, we write the KL-divergence as

$$D_{\rm kl}\left(P \| Q\right) := \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} d\mu(x).$$

The mutual information $I(X;Y)$ between random variables X and Y where Y has distribution P_Y is defined as

$$I(X;Y) := \mathbb{E}_{P_X}\left[D_{\rm kl}\left(P_Y(\cdot \mid X) \| P_Y(\cdot)\right)\right] = \int D_{\rm kl}\left(P_Y(\cdot \mid X = x) \| P_Y(\cdot)\right) dP_X(x).$$

We let $\vee$ and $\wedge$ denote maximum and minimum, respectively, so that $a \vee b = \max\{a, b\}$. For an integer $k \geq 1$, we use $[k]$ as shorthand for the set $\{1, \ldots, k\}$. We let $a_{1:n}$ be shorthand for a sequence $a_1, \ldots, a_n$, and the notation $a_n \gtrsim b_n$ means there is a numerical constant $c > 0$ such that $a_n \geq c b_n$ for all n. Given a set A, we let $\sigma(A)$ denote the Borel σ-field on A.

6.1 Background and problem set-up

In this section, we begin by giving background on the classical notion of minimax risk in statistics. We then introduce two distributed variants of the minimax risk based on the notions of independent and interactive protocols, respectively.

6.1.1 Classical minimax risk

For a family of probability distributions $\mathcal{P}$, consider a function $\theta : \mathcal{P} \to \Theta \subseteq \mathbb{R}^d$. A canonical example throughout the chapter is the mean function, namely $\theta(P) = \mathbb{E}_P[X]$. Another simple example is the median $\theta(P) = \text{med}_P(X)$, or more generally, quantiles of the distribution P. Now suppose that we are given a collection of N observations, say $X_{1:N} := \{X_1, \ldots X_N\}$, drawn i.i.d. from some unknown member P of $\mathcal{P}$. Based on the sample $X_{1:N}$, our goal is to estimate the parameter $\theta(P)$, and an estimator $\widehat{\theta}$ is a measurable function of the N-vector $X_{1:N} \in \mathcal{X}^N$ into Θ.

We assess the quality of an estimator $\widehat{\theta} = \widehat{\theta}(X_1^N)$ via its mean-squared error

$$R(\widehat{\theta}, \theta(P)) := \mathbb{E}_P\left[\|\widehat{\theta}(X_{1:N}) - \theta(P)\|_2^2\right],$$

where the expectation is taken over the sample $X_{1:N}$. For an estimator $\widehat{\theta}$, the function $P \mapsto R(\widehat{\theta}, \theta(P))$ defines the *risk function* of $\widehat{\theta}$ over the family $\mathcal{P}$. Taking the supremum all $P \in \mathcal{P}$ yields the worst-case risk of the estimator. The *minimax rate* for the family $\mathcal{P}$ is defined in terms of the best possible estimator for this worst-case criterion, namely via the saddle point criterion

$$\mathfrak{M}_N(\theta, \mathcal{P}) := \inf_{\widehat{\theta}} \sup_{P \in \mathcal{P}} R(\widehat{\theta}, \theta(P)), \tag{6.1}$$

where the infimum ranges over all measurable functions of the data $X_{1:N}$. Many papers in mathematical statistics study the classical minimax risk (6.1), and its behavior is precisely characterized for a range of problems [96, 223, 219, 205]. We consider a few instances of such problems in the sequel.

6.1.2 Distributed protocols

The classical minimax risk (6.1) imposes no constraints on the choice of estimator $\widehat{\theta}$. In this section, we introduce a refinement of the minimax risk that calibrates the effect of communication constraints. Suppose we have a collection of m distinct computers or processing units. Assuming for simplicity[1] that N is a multiple of m, we can then divide our full data set $X_{1:N}$ into a family of m subsets, each containing $n = \frac{N}{m}$ distinct observations, with $X^{(i)}$ denoting the subset assigned to machine $i \in [m] = \{1, \ldots, m\}$. With this set-up, our goal is to estimate $\theta(P)$ via local operations at each machine i on the data subset $X^{(i)}$ while performing a limited amount of communication between machines.

More precisely, our focus is a class of distributed protocols Π, in which at each round $t = 1, 2, \ldots$, machine i sends a message $Y_{t,i}$ that is a measurable function of the local data $X^{(i)}$ and potentially of past messages. It is convenient to model this message as being sent to a central fusion center. Let $\bar{Y}_t = \{Y_{t,i}\}_{i \in [m]}$ denote the collection of all messages sent at round t. Given a total of T rounds, the protocol Π collects the sequence $(\bar{Y}_1, \ldots, \bar{Y}_T)$, and constructs an estimator $\widehat{W}\theta := \widehat{W}\theta(\bar{Y}_1, \ldots, \bar{Y}_T)$. The length $L_{t,i}$ of message $Y_{t,i}$ is the minimal number of bits required to encode it, and the total $L = \sum_{t=1}^{T} \sum_{i=1}^{m} L_{t,i}$ of all messages sent corresponds to the *total communication cost* of the protocol. Note that the communication cost is a random variable, since the length of the messages may depend on the data, and the protocol may introduce auxiliary randomness.

It is useful to distinguish two different protocol classes, namely *independent* versus *interactive*. An independent protocol Π is based on a single round ($T = 1$) of communication in which machine i sends a single message $Y_{1,i}$ to the fusion center. Since there are no past messages, the message $Y_{1,i}$ can depend only on the local sample $X^{(i)}$. Given a family $\mathcal{P}$, the

[1] Although we assume in this chapter that every machine has the same amount of data, our techniques are sufficiently general to allow for different sized subsets for each machine.

class of independent protocols with budget $B \geq 0$ is

$$\mathcal{A}_{\text{ind}}(B, \mathcal{P}) = \left\{ \text{independent protocols } \Pi \text{ such that } \sup_{P \in \mathcal{P}} \mathbb{E}_P \left[\sum_{i=1}^{m} L_i \right] \leq B \right\}.$$

(For simplicity, we use Y_i to indicate the message sent from processor i and L_i to denote its length in the independent case.) It can be useful in some situations to have more granular control on the amount of communication, in particular by enforcing budgets on a per-machine basis. In such cases, we introduce the shorthand $B_{1:m} = (B_1, \ldots, B_m)$ and define

$$\mathcal{A}_{\text{ind}}(B_{1:m}, \mathcal{P}) = \left\{ \text{independent protocols } \Pi \text{ such that } \sup_{P \in \mathcal{P}} \mathbb{E}_P[L_i] \leq B_i \text{ for } i \in [m] \right\}.$$

In contrast to independent protocols, the class of interactive protocols allows for interaction at different stages of the message passing process. In particular, suppose that machine i sends message $Y_{t,i}$ to the fusion center at time t, which then posts it on a "public blackboard," where all machines may read $Y_{t,i}$ (this posting and reading incurs no communication cost). We think of this as a global broadcast system, which may be natural in settings in which processors have limited power or upstream capacity, but the centralized fusion center can send messages without limit. In the interactive setting, the message $Y_{t,i}$ is a measurable function of the local data $X^{(i)}$ and the past messages $\bar{Y}_{1:t-1}$. The family of interactive protocols with budget $B \geq 0$ is

$$\mathcal{A}_{\text{inter}}(B, \mathcal{P}) = \left\{ \text{interactive protocols } \Pi \text{ such that } \sup_{P \in \mathcal{P}} \mathbb{E}_P[L] \leq B \right\}.$$

6.1.3 Distributed minimax risks

We can now define the distributed minimax risks that are the central objects of study in this chapter. Our goal is to characterize the best achievable performance of estimators $\widehat{\theta}$ that are functions of the vector of messages $\bar{Y}_1^T := (\bar{Y}_1, \ldots, \bar{Y}_T)$. As in the classical minimax setting (6.1), we measure the quality of a protocol Π and estimator $\widehat{\theta}$ by the mean-squared error

$$R(\widehat{\theta}, \theta(P)) := \mathbb{E}_{P,\Pi} \left[\|\widehat{\theta}(\bar{Y}_1^T) - \theta(P)\|_2^2 \right],$$

where the expectation is now taken over the randomness in the messages, which is due to both their dependence on the underlying data as well as possible randomness in the protocol. Given a communication budget B, the *minimax risk for independent protocols* is

$$\mathfrak{M}_{n,m}^{\text{ind}}(\theta, \mathcal{P}, B) := \inf_{\Pi \in \mathcal{A}_{\text{ind}}(B, \mathcal{P})} \inf_{\widehat{W}\theta} \sup_{P \in \mathcal{P}} R\big(\widehat{\theta}, \theta(P)\big). \tag{6.2}$$

Here, the infimum is taken jointly over all independent procotols Π that satisfy the budget constraint B, and over all estimators $\widehat{W}\theta$ that are measurable functions of the messages in the

protocol. The minimax risk (6.2) should also be understood to depend on both the number of machines m and the individual sample size n (we leave this implicit on the right hand side of definition (6.2)). We define the *minimax risk for interactive protocols*, denoted by $\mathfrak{M}^{\mathsf{inter}}_{n,m}$, analogously, where we instead take the infimum over the class of interactive protocols. These communication-dependent minimax risks are the central objects in this chapter: they provide a sharp characterization of the optimal estimation rate as a function of the communication budget B.

6.2 Main results and their consequences

We now turn to the statement of our main results, along with some discussion of their consequences. We begin with a rather simple bound based on the metric entropy of the parameter space; it confirms the natural intuition that any procedure must communicate at least as many bits as are required to describe a problem solution. We show that this bound is tight for certain problems, but our subsequent more refined techniques allow substantially sharper guarantees.

6.2.1 Lower bound based on metric entropy

We begin with a general but relatively naive lower bound that depends only on the geometric structure of the parameter space, as captured by its metric entropy. In particular, given a subset $\Theta \subset \mathbb{R}^d$, we say $\{\theta^1, \ldots, \theta^K\}$ are δ-separated if $\|\theta^i - \theta^j\|_2 \geq \delta$ for $i \neq j$. We then define the *packing entropy* of Θ as

$$\mathcal{E}_\Theta(\delta) := \log_2 \max \big\{ M \in \mathbb{N} \mid \{\theta_1, \ldots, \theta_M\} \subset \Theta \text{ are } \delta\text{-separated} \big\}.$$

It is straightforward to see that the packing entropy continuous from the right and non-increasing in δ, so that the inverse function $\mathcal{E}_\Theta^{-1}(B) := \sup\{\delta \mid \mathcal{E}_\Theta(\delta) \geq B\}$ is well-defined. With this definition, we have the following claim:

Proposition 1. *For any family of distributions $\mathcal{P}$ and parameter set $\Theta = \theta(\mathcal{P})$, the interactive minimax risk is lower bounded as*

$$\mathfrak{M}^{\mathsf{inter}}_{n,m}(\theta, \mathcal{P}, B) \geq \frac{1}{8}\big(\mathcal{E}_\Theta^{-1}(2B+2)\big)^2.$$

We prove this proposition in Section 6.3.1. The same lower bound trivially holds for $\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}, B)$, as any independent protocol is a special case of an interactive protocol. Although Proposition 1 is a relatively generic statement, not exploiting any particular structure of the problem, it is in general unimprovable by more than constant factors, as the following example illustrates.

Example 2 (Bounded mean estimation)**.** *Suppose our goal is to estimate the mean $\theta = \theta(P)$ of a class of distributions $\mathcal{P}$ supported on the interval $[0,1]$, so that $\Theta = \theta(\mathcal{P}) = [0,1]$.*

Suppose that a single machine ($m = 1$) receives n i.i.d. observations X_i according to P. The packing entropy has lower bound $\mathcal{E}_\Theta(\delta) \geq \log_2(1/\delta)$, and consequently, Proposition 1 implies that the distributed minimax risk is lower bounded as

$$\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}, B) \geq \mathfrak{M}^{\mathsf{inter}}_{n,m}(\theta, \mathcal{P}, B) \;\geq\; \frac{1}{8}\big(2^{-2B-2}\big)^2.$$

Setting $B = \frac{1}{4}\log_2 n$ yields the lower bound $\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}([0,1]), B) \geq \frac{1}{128n}$.

This lower bound is sharp up to the constant pre-factor; it can be achieved by the following simple method. Given its n observations, the single machine computes the sample mean $\overline{X}_n := \frac{1}{n}\sum_{i=1}^n X_i$. The sample mean must lie in the interval $[0,1]$, and so can be quantized to accuracy $\frac{1}{n}$ using $\log_2 n$ bits, and this quantized version $\widehat{\theta}$ can be transmitted. A straightforward calculation shows that $\mathbb{E}[(\widehat{\theta} - \theta)^2] \leq \frac{2}{n}$, and Proposition 1 yields an order-optimal bound. ■

6.2.2 Independent protocols in multi-machine settings

We would like to study how the *budget* B—the number of bits required to achieve the minimax rate—scales with the number of machines m. For our first set of results in this setting, we consider the non-interactive case, where each machine i sends messages Y_i independently of all the other machines. These results serve as pre-cursors to our later results on interactive protocols.

We first provide lower bounds for mean estimation in the d-dimensional normal location family model:

$$\mathcal{N}_d := \{N(\theta, \sigma^2 I_{d\times d}) \mid \theta \in \Theta = [-1,1]^d\}. \tag{6.3}$$

Here each machine receives an i.i.d. sample of size n from a normal distribution $N(\theta, \sigma^2 I_{d\times d})$ with unknown mean θ. The following result provides a lower bound on the distributed minimax risk with independent communication:

Theorem 10. *Given a communication budget B_i for each machine $i = 1, \ldots, m$, there exists a universal (numerical) constant c such that*

$$\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{N}_d, B_{1:m}) \geq c\frac{\sigma^2 d}{mn} \min\Big\{\frac{mn}{\sigma^2}, \frac{m}{\log m}, \frac{m}{(\sum_{i=1}^m \min\{1, \frac{B_i}{d}\})\log m} \vee 1\Big\}. \tag{6.4}$$

See Section 6.3.4 for the proof of this claim.

Given centralized access to the full mn-sized sample, the minimax rate for the mean-squared error is $\frac{\sigma^2 d}{mn}$ (e.g. Lehmann and Casella [119]). This optimal rate is achieved by the sample mean. Consequently, the lower bound (6.4) shows that each machine individually must communicate at least $\frac{d}{\log m}$ bits for a decentralized procedure to match the centralized rate. If we ignore logarithmic factors, this lower bound is achievable by the following simple procedure:

(i) First, each machine computes the sample mean of its local data, and truncates it to the interval $[-1 - \frac{\sigma}{\sqrt{n}}, 1 + \frac{\sigma}{\sqrt{n}}]$.

(ii) Next, each machine quantizes each coordinate of the resulting estimate to precision $\frac{\sigma^2}{mn}$, using $\mathcal{O}(1)\, d \log \frac{mn}{\sigma^2}$ bits to do so.

(iii) The machines send these quantized averages to the fusion center using $B = \mathcal{O}(1)\, dm \log \frac{n}{\sigma^2}$ total bits.

(iv) Finally, the fusion center averages them, obtaining an estimate with mean-squared error of the order $\frac{\sigma^2 d}{mn}$.

The techniques we develop also apply to other families of probability distributions, and we finish our discussion of independent communication protocols by presenting a result that gives lower bounds sharp to numerical constant prefactors. In particular, we consider mean estimation for the family $\mathcal{P}_d$ of distributions supported on the compact set $[-1, 1]^d$. One instance of such a distribution is the Bernoulli family taking values on the Boolean hypercube $\{-1, 1\}^d$.

Proposition 2. *Assume that each of m machines receives a single observation ($n = 1$) from a distribution in $\mathcal{P}_d$. There exists a universal constant $c > 0$ such that*

$$\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}_d, B_{1:m}) \geq c \frac{d}{m} \min \left\{ m, \frac{m}{\sum_{i=1}^m \min\{1, \frac{B_i}{d}\}} \right\},$$

where B_i is the budget for machine i.

See Section 6.3.3 for the proof.

The standard minimax rate for d-dimensional mean estimation on $\mathcal{P}_d$ scales as d/m, which is achieved by the sample mean. Proposition 2 shows that to achieve this scaling, we must have $\sum_{i=1}^m \min\{1, \frac{B_i}{d}\} \gtrsim m$, showing that each machine must send $B_i \gtrsim d$ bits. This lower bound is also achieved by a simple scheme:

(i) Each machine i receives an observation $X_i \in [-1, 1]^d$. Based on this observation. it generates a Bernoulli random vector $Z_i = (Z_{i1}, \ldots, Z_{id})$ with $Z_{ij} \in \{0, 1\}$ taking the value 1 with probability $(1 + X_{ij})/2$, independently across coordinates.

(ii) Machine i uses d bits to send the vector $Z_i \in \{0, 1\}^d$ to the fusion center.

(iii) The fusion center then computes the average $\widehat{\theta} = \frac{1}{m} \sum_{i=1}^m (2Z_i - 1)$. This average is unbiased, and its expected squared error is bounded by d/m.

Note that for both the normal location family of Theorem 10 and the simpler bounded single observation model in Proposition 2, there is an exponential gap between the information required to describe the problem to the minimax mean squared error of $\frac{d}{mn}$—which scales as as $\mathcal{O}(1) d \log(mn)$—and the number of bits that must be communicated, which scales nearly linearly in m. See also our discussion following Theorem 11.

6.2.3 Interactive protocols in multi-machine settings

Having provided results on mean estimation in the non-interactive setting, we now turn to the substantially harder setting of distributed statistical inference where feedback is permitted. As described in Section 6.1.2, in the interactive setting the fusion center may freely broadcast every message received to all other machines in the network. This freedom allows more powerful algorithms, rendering the task of proving lower bounds more challenging.

Let us begin by considering the uniform location family $\mathcal{U}_d = \{P_\theta,\ \theta \in [-1,1]^d\}$, where P_θ is the uniform distribution on the rectangle $[\theta_1 - 1, \theta_1 + 1] \times \cdots \times [\theta_d - 1, \theta_d + 1]$. For this problem, a direct application of Proposition 1 gives a nearly sharp result:

Proposition 3. *Consider the uniform location family $\mathcal{U}_d$ with n i.i.d. observations per machine:*

(a) There are universal (numerical) constants $c_1, c_2 > 0$ such that

$$\mathfrak{M}^{\mathsf{inter}}_{n,m}(\theta, \mathcal{U}, B) \geq c_1 \max\left\{\exp\left(-c_2 \frac{B}{d}\right), \frac{d}{(mn)^2}\right\}.$$

(b) Conversely, given a budget of $B = d\big[2\log_2(2mn) + \log(m)(\lceil \log_2 d \rceil + 2\log_2(2mn))\big]$ bits, there is a universal constant c such that

$$\mathfrak{M}^{\mathsf{inter}}_{n,m}(\theta, \mathcal{U}, B) \leq c \frac{d}{(mn)^2}.$$

See Section 6.3.5 for the proof of this claim.

If each of the m machines receives n observations, we have a total sample size of mn, so the minimax rate over all centralized procedures scales as $d/(mn)^2$ (for instance, see Lehmann and Casella [119]). Consequently, Proposition 3(b) shows that the number of bits required to achieve the centralized rate has only *logarithmic* dependence on the number m of machines. Part (a) shows that this logarithmic dependence on m is unavoidable: at least $B \gtrsim d \log(mn)$ bits are necessary to attain the optimal rate of $\frac{d}{(mn)^2}$.

It is natural to wonder whether such logarithmic dependence holds more generally. The following result shows that it does not: for some problems, the dependence on m must be (nearly) linear. In particular, we reconsider estimation in the normal location family model (6.3), showing a lower bound that is nearly identical to that of Theorem 10.

Theorem 11. *For $i = 1, \ldots, m$, assume that each machine receives an i.i.d. sample of size n from a normal location model (6.3) and that there is a total communication budget B. Then there exists a universal (numerical) constant c such that*

$$\mathfrak{M}^{\mathsf{inter}}_{n,m}(\theta, \mathcal{N}_d, B) \geq c \frac{\sigma^2 d}{mn} \ \min\Big\{\frac{mn}{\sigma^2},\ \frac{m}{(B/d+1)\ \log m} \vee 1\Big\}. \tag{6.5}$$

See Section 6.3.6 for the proof of this claim.

Theorem 11 is analogous to, but slightly weaker than, the corresponding lower bound from Theorem 10 for the non-interactive setting. In particular, the lower bound (6.5) shows that at least $B \gtrsim \frac{dm}{\log m}$ bits are required for any distributed procedure—even allowing fully interactive communication—to attain the centralized minimax rate. Thus, in order to achieve the minimax rate up to logarithmic factors, the total number of bits communicated must scale (nearly) linearly with the product of the dimension d and number of machines m.

Moreover, these two theorems show that there is an exponential gap between the number of bits required to communicate the problem solution and the number required to compute it in a distributed manner. More specifically, assuming (for simplicity) that $\sigma^2 = 1$, describing a solution of the normal mean estimation problem to accuracy $\frac{d}{mn}$ in squared ℓ_2-error requires at most $\mathcal{O}(1)d\log(mn)$ bits. On the other hand, these two theorems show that nearly dm bits *must* be communicated. This linear scaling in m is dramatically different from—exponentially worse than—the logarithmic scaling for the uniform family. Establishing sharp communication-based lower bounds thus requires careful study of the underlying family of distributions.

Note that in both Theorems 10 and 11, the upper and lower bounds differ by logarithmic factors in the sample size n and number of machines m. It would be interesting to close this minor gap. Another open question is whether the distributed minimax rates for the independent and interactive settings are the same up to constant factors, or whether their scaling actually differs in terms of these logarithmic factors.

6.2.4 Consequences for regression

The problems of mean estimation studied in the previous section, though simple in appearance, are closely related to other, more complex problems. In this section, we show how lower bounds on mean estimation can be used to establish lower bounds for distributed estimation in two standard but important generalized linear models [86]: linear regression and probit regression.

6.2.4.1 Linear regression

Let us begin with a distributed instantiation of linear regression with fixed design matrices. Concretely, suppose that each of m machines has stored a fixed design matrix $A^{(i)} \in \mathbb{R}^{n\times d}$ and then observes a response vector $b^{(i)} \in \mathbb{R}^d$ from the standard linear regression model

$$b^{(i)} = A^{(i)}\theta + \varepsilon^{(i)}, \tag{6.6}$$

where $\varepsilon^{(i)} \sim N(0, \sigma^2 I_{n\times n})$ are independent noise vectors. Our goal is to estimate the unknown regression vector $\theta \in \Theta = [-1, 1]^d$, identical for each machine. Our result involves the

smallest and largest eigenvalues of the rescaled design matrices via the quantities

$$\lambda_{\max}^2 := \max_{i \in \{1,\ldots,m\}} \frac{\lambda_{\max}(A^{(i)\top} A^{(i)})}{n}, \quad \text{and} \quad \lambda_{\min}^2 := \min_{i \in \{1,\ldots,m\}} \frac{\kappa_{\mathcal{L}}(A^{(i)\top} A^{(i)})}{n} > 0. \tag{6.7}$$

Corollary 11. *Given the linear regression model (6.6), there is a universal positive constant c such that*

$$\mathfrak{M}_{n,m}^{\mathsf{inter}}(\theta, \mathcal{P}, B) \geq c \frac{\sigma^2 d}{\lambda_{\max}^2 mn} \min\left\{ \frac{\lambda_{\max}^2 mn}{\sigma^2}, \frac{m}{(B/d+1)\log m} \vee 1 \right\}. \tag{6.8a}$$

Conversely, given a budgets $B_i \geq dm \log(mn)$, there is a universal constant c' such that

$$\mathfrak{M}_{n,m}^{\mathsf{ind}}(\theta, \mathcal{P}, B_{1:m}) \leq \frac{c'}{\lambda_{\min}^2} \frac{\sigma^2 d}{mn}. \tag{6.8b}$$

It is a classical fact (e.g. [119]) that the minimax rate for d-dimensional linear regression scales as $d\sigma^2/(nm)$. Part (a) of Corollary 11 shows this optimal rate is attainable only if the total budget B grows as $\frac{dm}{\log m}$. Part (b) of the corollary shows that the minimax rate is achievable—even using an independent protocol—with budgets that match the lower bound to within logarithmic factors.

Proof The upper bound (6.8b) follows from the results of Zhang et al. [231]. Their results imply that the upper bound can be achieved by solving each regression problem separately, quantizing the (local) solution vectors $\widehat{W}\theta^{(i)} \in [-1,1]^d$ to accuracy $\frac{1}{mn}$ using $B_i = \lceil d \log_2(mn) \rceil$ bits and performing a form of approximate averaging.

In order to prove the lower bound (6.8a), we show that solving an arbitrary Gaussian mean estimation problem can be reduced to solving a specially constructed linear regression problem. This reduction allows us to apply the lower bound from Theorem 11. Given $\theta \in \Theta$, consider the Gaussian mean model

$$X^{(i)} = \theta + w^{(i)}, \quad \text{where} \quad w^{(i)} \sim N\Big(0, \frac{\sigma^2}{\lambda_{\max}^2 n} I_{d \times d}\Big).$$

Each machine i has its own design matrix $A^{(i)}$, and we use it to construct a response vector $b^{(i)} \in \mathbb{R}^n$. Since $\lambda_{\max}(A^{(i)\top} A^{(i)}/n) \leq \lambda_{\max}^2$, the matrix $\Sigma^{(i)} := \sigma^2 I_{n \times n} - \frac{\sigma^2}{\lambda_{\max}^2 n} A^{(i)} (A^{(i)})^\top$ is positive semidefinite. Consequently, we may form a response vector via

$$b^{(i)} = A^{(i)} X^{(i)} + z^{(i)} = A^{(i)}\theta + A^{(i)} w^{(i)} + z^{(i)}, \quad z^{(i)} \sim N(0, \Sigma^{(i)}) \text{ independent of } w^{(i)}. \tag{6.9}$$

The independence of $w^{(i)}$ and $z^{(i)}$ guarantees that $b^{(i)} \sim N(A^{(i)}\theta, \sigma^2 I_{n \times n})$, so the pair $(b^{(i)}, A^{(i)})$ is faithful to the regression model (6.6).

Now consider a protocol $\Pi \in \mathcal{A}_{\text{inter}}(B, \mathcal{P})$ that can solve any regression problem to within accuracy δ, so that $\mathbb{E}[\|\widehat{W}\theta - \theta\|_2^2] \leq \delta^2$. By the previously described reduction, the protocol Π can also solve the mean estimation problem to accuracy δ, in particular via the pair $(A^{(i)}, b^{(i)})$ described in expression (6.9). Combined with this reduction, the corollary thus follows from Theorem 11. □

6.2.4.2 Probit regression

We now turn to the problem of binary classification, in particular considering the probit regression model. As in the previous section, each of m machines has a fixed design matrix $A^{(i)} \in \mathbb{R}^{n \times d}$, where $A^{(i,k)}$ denotes the kth row of $A^{(i)}$. Machine i receives n binary responses $Z^{(i)} = (Z^{(i,1)}, \ldots, Z^{(i,n)})$, drawn from the conditional distribution

$$\mathbb{P}\big(Z^{(i,k)} = 1 \mid A^{(i,k)}, \theta\big) = \Phi(A^{(i,k)}\theta) \quad \text{for some fixed } \theta \in \Theta = [-1, 1]^d, \tag{6.10}$$

where $\Phi(\cdot)$ denotes the standard normal CDF. The log-likelihood of the probit model (6.10) is concave (cf. [33, Exercise 3.54]). Under condition (6.7) on the design matrices, we have:

Corollary 12. *Given the probit model (6.10), there is a universal constant $c > 0$ such that*

$$\mathfrak{M}^{\mathsf{inter}}_{n,m}(\theta, \mathcal{P}, B_{1:m}) \geq c \frac{d}{\lambda^2_{\max} mn} \min\left\{\lambda^2_{\max} mn, \frac{m}{(B/d+1)\,\log m}\right\}. \tag{6.11a}$$

Conversely, given a budgets $B_i \geq d \log(mn)$, there is a universal constant c' such that

$$\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}, B_{1:m}) \leq \frac{c'}{\lambda^2_{\min}} \frac{d}{mn}. \tag{6.11b}$$

Proof As in Corollary 11, the upper bound (6.11b) follows from the results of Zhang et al. [231].

Turning to the lower bound (6.11a), our strategy is to show that probit regression is at least as hard as linear regression, in particular by demonstrating that any linear regression problem can be solved via estimation in a specially constructed probit model. Given an arbitrary regression vector $\theta \in \Theta$, consider a linear regresion problem (6.6) with noise variance $\sigma^2 = 1$. We construct the binary responses for our probit regression $(Z^{(i,1)}, \ldots, Z^{(i,n)})$ by

$$Z^{(i,k)} = \begin{cases} 1 & \text{if } b^{(i,k)} \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.12}$$

By construction, we have $\mathbb{P}(Z^{(i,k)} = 1 \mid A^{(i)}, \theta) = \Phi(A^{(i,k)}\theta)$ as desired for our model (6.10). By inspection, any protocol $\Pi \in \mathcal{A}_{\text{inter}}(B, \mathcal{P})$ solving the probit regression problem provides an estimator with the same mean-squared error as the original linear regression problem via the construction (6.12). Consequently, the lower bound (6.11a) follows from Corollary 11. □

6.3 Proofs of main results

We now turn to the proofs of our main results, deferring more technical results to Section 6.4.

6.3.1 Proof of Proposition 1

This result is based on the classical reduction from estimation to testing (e.g., [96]). For a given $\delta > 0$, introduce the shorthand $M = 2^{\mathcal{E}_\Theta(2\delta)}$ for the 2δ packing number, and form a collection of points $\{\theta_1, \ldots, \theta_M\}$ that form a maximal 2δ-packing of Θ. Now consider any family of conditional distributions $\{P(\cdot \mid \nu),\ \nu \in [M]\}$ such that $\theta\big(P(\cdot \mid \nu)\big) = \theta_\nu$.

Suppose that we sample an index V uniformly at random from $[M]$, and then draw a sample $X \sim P(\cdot \mid V)$. The associated testing problem is to determine the underlying instantiation of the randomly chosen index. Let $Y = (Y_1, \ldots, Y_T)$ denote the messages sent by the protocol Π, and let $\widehat{W}\theta(Y)$ denote any estimator of θ based on Y. Any such estimator defines a testing function via

$$\widehat{W}V := \operatorname*{argmin}_{\nu \in \mathcal{V}} \|\widehat{W}\theta(Y) - \theta_\nu\|_2.$$

Since $\{\theta_\nu\}_{\nu \in \mathcal{V}}$ is a 2δ-packing, we are guaranteed that $\|\widehat{W}\theta(Y) - \theta_\nu\|_2 \geq \delta$ whenever $\widehat{W}V \neq V$, whence

$$\begin{aligned}\max_{\nu \in \mathcal{V}} \mathbb{E}\big[\|\widehat{W}\theta(Y) - \theta_\nu\|_2^2\big] &\geq \sum_{\nu \in \mathcal{V}} \mathbb{P}(V = \nu)\mathbb{E}\big[\|\widehat{W}\theta(Y) - \theta_V\|_2^2 \mid V = \nu\big] \\ &\geq \sum_{\nu \in \mathcal{V}} \delta^2 \mathbb{P}(V = \nu)\mathbb{P}(\widehat{W}V \neq V \mid V = \nu) \;=\; \delta^2\, \mathbb{P}(\widehat{W}V \neq V). \quad (6.13)\end{aligned}$$

It remains to lower bound the testing error $\mathbb{P}(\widehat{W}V \neq V)$. Fano's inequality [53, Chapter 2] yields

$$\mathbb{P}(\widehat{W}V \neq V) \geq 1 - \frac{I(V;Y) + 1}{\mathcal{E}_\Theta(2\delta)}.$$

Finally, the mutual information can be upper bounded as

$$I(V;Y) \overset{(i)}{\leq} H(Y) \overset{(ii)}{\leq} B, \quad (6.14)$$

where inequality (i) is an immediate consequence of the definition of mutual information, and inequality (ii) follows from Shannon's source coding theorem [53]. Combining inequalities (6.13) and (6.14) yields

$$\mathfrak{M}_{n,m}^{\mathsf{inter}}(\theta, \mathcal{P}, B) \geq \delta^2 \Big\{1 - \frac{B+1}{\mathcal{E}_\Theta(2\delta)}\Big\} \quad \text{for any } \delta > 0.$$

Because $1 - \frac{B+1}{\mathcal{E}_\Theta(2\delta)} \geq \frac{1}{2}$ for any any choice of δ such that $2\delta \leq \mathcal{E}_\Theta^{-1}(2B+2)$, setting $\delta = \frac{1}{2}\mathcal{E}_\Theta^{-1}(2B+2)$ yields the claim.

6.3.2 A slight refinement

We now describe a slight refinement of the classical reduction from estimation to testing that underlies many of the remaining proofs. It is somewhat more general, since we no longer map the original estimation problem to a strict test, but rather a test that allows errors. We then leverage some variants of Fano's inequality developed by a subset of the current authors [63].

Defining $\mathcal{V} = \{-1, +1\}^d$, we consider an indexed family of probability distributions $\{P(\cdot \mid \nu)\}_{\nu \in \mathcal{V}} \subset \mathcal{P}$. Each member of this family defines the parameter $\theta_\nu := \theta(P(\cdot \mid \nu)) \in \Theta$. In particular, suppose that we construct the distributions such that $\theta_\nu = \delta\nu$, where $\delta > 0$ is a fixed quantity that we control. For any $\nu \neq \nu'$, we are then guaranteed that

$$\|\theta_\nu - \theta_{\nu'}\|_2 = 2\delta\sqrt{d_{\text{ham}}(\nu, \nu')} \;\geq\; 2\delta$$

where $d_{\text{ham}}(\nu, \nu')$ is the Hamming distance between $\nu, \nu' \in \mathcal{V}$. This lower bound shows that $\{\theta_\nu\}_{\nu \in \mathcal{V}}$ is a special type of 2δ-packing, in that the squared ℓ_2-distance grows proportionally to the Hamming distance between the indices ν and ν'.

Now suppose that we draw an index V from $\mathcal{V}$ uniformly at random, then drawing a sample X from the distribution $P(\cdot \mid V)$. Fixing $t \geq 0$, the following lemma [63] reduces the problem of estimating θ to finding a point $\nu \in \mathcal{V}$ within distance t of the random variable V.

Lemma 25. *Let V be uniformly sampled from $\mathcal{V}$. For any estimator $\widehat{W}\theta$ and any $t \geq 0$, we have*

$$\sup_{P \in \mathcal{P}} \mathbb{E}[\|\widehat{W}\theta - \theta(P)\|_2^2] \geq \delta^2 \left(\lfloor t \rfloor + 1\right) \inf_{\widehat{W}\nu} P\Big(d_{\text{ham}}(\widehat{W}\nu, V) > t\Big),$$

where the infimum ranges over all testing functions $\widehat{W}\nu$ mapping the observations X to $\mathcal{V}$.

Setting $t = 0$, we recover the standard reduction from estimation to testing as used in the proof of Proposition 1. The lemma allows for some additional flexibility in that it suffices to show that, for some $t > 0$ to be chosen, it is difficult to identify V within a Hamming radius of t. The following variant [63] of Fano's inequality controls this type of error probability:

Lemma 26. *Let $V \to X \to \widehat{W}V$ be a Markov chain, where V is uniform on $\mathcal{V}$. For any $t \geq 0$, we have*

$$\mathbb{P}(d_{\text{ham}}(\widehat{W}V, V) > t) \geq 1 - \frac{I(V; X) + \log 2}{\log \frac{|\mathcal{V}|}{N_t}},$$

where $N_t := \max_{\nu \in \mathcal{V}} |\{\nu' \in \mathcal{V} : d_{\text{ham}}(\nu, \nu') \leq t\}|$ is the size of the largest t-neighborhood in $\mathcal{V}$.

We thus have a clear avenue for obtaining lower bounds: constructing a large packing set $\mathcal{V}$ with (1) relatively small t-neighborhoods, and (2) such that the mutual information $I(V; X)$ can be controlled. Given this set-up, the remaining technical challenge is the development of *quantitative data processing inequalities*, which allow us to characterize the effect

of bit-constraints on the mutual information $I(V;X)$. In general, these bounds are significantly tighter than the trivial upper bound used in the proof of Proposition 1. Examples of such inequalities in the sequel include Lemmas 27, 30, and 33.

6.3.3 Proof of Proposition 2

Given an index $\nu \in \mathcal{V}$, suppose that each machine i receives a d-dimensional sample $X^{(i)}$ with coordinates independently sampled according to

$$P(X_j = \nu_j \mid \nu) = \frac{1+\delta\nu_j}{2} \quad \text{and} \quad P(X_j = -\nu_j \mid \nu) = \frac{1-\delta\nu_j}{2}.$$

Note that by construction, we have $\theta_\nu = \delta\nu = \mathbb{E}_\nu[X]$, as well as

$$\max_{x_j} \frac{P(x_j \mid \nu)}{P(x_j \mid \nu')} \leq \frac{1+\delta}{1-\delta} = e^\alpha \qquad \text{where } \alpha := \log \tfrac{1+\delta}{1-\delta}. \tag{6.15}$$

Moreover, note that for any pair (i,j), the sample $X_j^{(i)}$, when conditioned on V_j, is independent of the variables $\{X_{j'}^{(i)} : j' \neq j\} \cup \{V_{j'} : j' \neq j\}$.

Recalling that Y_i denotes the message sent by machine i, consider the Markov chain $V \to X^{(i)} \to Y_i$. By the usual data processing inequality [53], we have $I(V;Y_i) \leq I(X^{(i)};Y_i)$. The following result is a quantitative form of this statement, showing how the likelihood ratio bound (6.15) causes a contraction in the mutual information.

Lemma 27. *Under the preceding conditions, we have*

$$I(V;Y_i) \leq 2(e^{2\alpha}-1)^2 \, I(X^{(i)};Y_i).$$

See Section 6.4.2.1 for the proof of this result. It is similar in spirit to recent results of Duchi et al. [66, Theorems 1–3], who establish quantitative data processing inequalities in the context of privacy-preserving data analysis. Our proof, however, is different, as we have the Markov chain $V \to X \to Y$, and instead of a likelihood ratio bound on the channel $X \to Y$ as in the paper [66], we place a likelihood ratio bound on $V \to X$.

Next we require a certain tensorization property of the mutual information, valid in the case of independent protocols:

Lemma 28. *When Y_i is a function only of $X^{(i)}$, then*

$$I(V;Y_{1:m}) \leq \sum_{i=1}^m I(V;Y_i).$$

See Section 6.4.2.2 for a proof of this claim.

We can now complete the proof of the proposition. Using Lemma 27, we have

$$I(V;Y_i) \le 2\left(e^{2\log\frac{1+\delta}{1-\delta}}-1\right)^2 I(X^{(i)};Y_i) \;=\; 2\big(\frac{(1+\delta)^2}{(1-\delta)^2}-1\big)^2 \;\le\; 80\delta^2 I(X^{(i)};Y_i),$$

valid for $\delta \in [0, 1/5]$. Applying Lemma 28 yields

$$I(V;Y_{1:m}) \le \sum_{i=1}^m I(V;Y_i) \le 80\delta^2 \sum_{i=1}^m I(Y_i;X^{(i)}).$$

The remainder of the proof is broken into two cases, namely $d \ge 10$ and $d < 10$.

Case $d \ge 10$: By the definition of mutual information, we have

$$I(Y_i;X^{(i)}) \le \min\{H(Y_i), H(X^{(i)})\} \;\le\; \min\{B_i, d\},$$

where the final step follows since $H(X^{(i)}) \le d$ and $H(Y_i) \le B_i$, the latter inequality following from Shannon's source coding theorem [53]. Putting together the pieces, we have

$$I(V;Y_{1:m}) \le 80\delta^2 \sum_{i=1}^m \min\{B_i, d\}.$$

Combining this upper bound on mutual information with Lemmas 25 and 26 yields the lower bound

$$\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}, B_{1:m}) \ge \delta^2(\lfloor d/6 \rfloor + 1)\left(1 - \frac{80\delta^2 \sum_{i=1}^m \min\{B_i, d\} + \log 2}{d/6}\right).$$

The choice $\delta^2 = \min\{1/25, d/960 \sum_{i=1}^m \min\{B_i, d\}\}$ guarantees that the expression inside parentheses in the previous display is lower bounded by $2/25$, which completes the proof for $d \ge 10$.

Case $d < 10$: In this case, we make use of Le Cam's method instead of Fano's method. More precisely, by reducing to a smaller dimensional problem, we may assume without loss of generality that $d = 1$, and we set $\mathcal{V} = \{-1, 1\}$. Letting V be uniformly distributed on $\mathcal{V}$, the Bayes error for binary hypothesis testing is (e.g. [223, 205, Chapter 2])

$$\inf_{\widehat{W}\nu} \mathbb{P}(\widehat{W}\nu \ne V) = \frac{1}{2} - \frac{1}{2}\left\|P_1 - P_{-1}\right\|_{\mathrm{TV}}.$$

As $\theta_\nu = \delta\nu$ by construction, the reduction from estimation to testing in Lemma 25 implies

$$\inf_{\widehat{W}\theta} \max_{P \in \{P_1, P_{-1}\}} \mathbb{E}[\|\widehat{W}\theta - \theta(P)\|_2^2] \ge \delta^2\left(\frac{1}{2} - \frac{1}{2}\left\|P_1 - P_{-1}\right\|_{\mathrm{TV}}\right).$$

Finally, as we show in Section 6.4.2.3, we have the following consequence of Pinsker's inequality:

$$\|P_Y(\cdot \mid V = \nu) - P_Y(\cdot \mid V = \nu')\|_{\mathrm{TV}}^2 \le 2I(Y; V). \tag{6.16}$$

Thus

$$\mathfrak{M}_{n,m}^{\mathsf{ind}}(\theta, \mathcal{P}, B_{1:m}) \ge \delta^2 \left(\frac{1}{2} - \frac{1}{2} \sqrt{2I(V; Y_{1:m})} \right). \tag{6.17}$$

Arguing as in the previous case ($d \ge 10$), we have the upper bound $I(X^{(i)}; Y_i) \le \min\{B_i, 1\}$, and hence

$$\mathfrak{M}_{n,m}^{\mathsf{ind}}(\theta, \mathcal{P}, B_{1:m}) \ge \delta^2 \left[\frac{1}{2} - 7 \left(\delta^2 \sum_{i=1}^{m} \min\{B_i, 1\} \right)^{\frac{1}{2}} \right].$$

Setting $\delta^2 = \min\left\{ \frac{1}{25}, \frac{1}{400 \sum_{i=1}^{m} \min\{B_i, 1\}} \right\}$ completes the proof.

6.3.4 Proof of Theorem 10

This proof follows a similar outline to that of Proposition 2. We assume that the sample $X^{(i)}$ at machine i contains n_i independent observations from the multivariate normal distribution, and we will use the fact that $n_i \equiv n$ at the end of the proof, demonstrating that the proof technique is sufficiently general to allow for different sized subsets in each machine. We represent the ith as a $d \times n_i$ matrix $X^{(i)} \in \mathbb{R}^{d \times n_i}$. We use $X^{(i,k)}$ and $X_j^{(i)}$ to denote, respectively, the kth column and jth row of this matrix. Throughout this argument, we assume that $m \ge 5$; otherwise, Proposition 1 provides a stronger result.

As in the previous section, we consider a testing problem in which the index $V \in \{-1, +1\}^d$ is drawn uniformly at random. Our first step is to provide a quantitative data processing inequality analogous to Lemma 27, but which applies in somewhat more general settings. To that end, we abstract a bit from our current setting, and consider a model such that for any (i, j), we assume that given V_j, the jth row row $X_j^{(i)}$ is conditionally independent of all other rows $\{X_{j'}^{(i)} : j' \ne j\}$ and all other packing indices $\{V_{j'} : j' \ne j\}$. In addition, letting P_{X_j} denote the probability measure of $X_j^{(i)}$, we assume that there exist measurable sets $G_j \subset \mathrm{range}(X_j^{(i)})$ such that

$$\sup_{S \in \sigma(G_j)} \frac{P_{X_j}(S \mid V = \nu)}{P_{X_j}(S \mid V = \nu')} \le \exp(\alpha),$$

Let E_j be a $\{0, 1\}$-valued indicator variable for the event $X_j^{(i)} \in G_j$ (i.e. $E_j = 1$ iff $X_j^{(i)} \in G_j$, and we leave the indexing on i implicit). We have the following bound:

Lemma 29. *Under the conditions stated in the preceding paragraph, we have*

$$I(V; Y_i) \le 2\big(e^{4\alpha} - 1\big)^2 I(X^{(i)}; Y_i) + \sum_{j=1}^{d} H(E_j) + \sum_{j=1}^{d} P(E_j = 0).$$

See Section 6.4.3.1 for the proof of this claim.

Our next step is to bound the terms involving the indicator variables E_j. Fixing some $\delta > 0$, for each $\nu \in \{-1, 1\}^d$ define $\theta_\nu = \delta\nu$, and conditional on $V = \nu \in \{-1, 1\}^d$, let $X^{(i,k)}$, $k = 1, \ldots, n_i$, be drawn i.i.d. from a $N(\theta_\nu, \sigma^2 I_{d\times d})$ distribution. The following lemma applies to any pair of non-negative numbers (a, δ) such that

$$\max_{i\in[m]} \frac{\sqrt{n_i}a\delta}{\sigma^2} \leq \frac{1}{4} \quad \text{and} \quad a \geq \delta \max_{i\in[m]} \sqrt{n_i}. \tag{6.18}$$

It also involves the binary entropy function $h_2(p) := -p\log_2(p) - (1-p)\log_2(1-p)$.

Lemma 30. *For any pair (a, δ) satisfying condition (6.18), we have*

$$I(V; Y_i) \leq \frac{dn_i\delta^2}{\sigma^2}, \qquad \textit{and} \tag{6.19a}$$

$$I(V; Y_i) \leq 128\frac{\delta^2 a^2}{\sigma^4} n_i H(Y_i) + d\, h_2(p_i^*) + d\, p_i^*, \tag{6.19b}$$

where $p_i^* := \min\left\{2\exp\left(-\frac{(a-\sqrt{n_i}\delta)^2}{2\sigma^2}\right), \frac{1}{2}\right\}$.

With the bounds (6.19a) and (6.19b) on the mutual information $I(Y_i; V)$, we may now divide our proof into two cases: when $d < 10$ and $d \geq 10$.

Case $d \geq 10$: In this case, we require an additional auxiliary result, which we prove via Lemma 30. (See Section 6.4.3.3 for the proof of this claim.)

Lemma 31. *For all $\delta \in \left[0, \frac{\sigma}{16}(\log m \max_i n_i)^{-\frac{1}{2}}\right]$, we have*

$$\sum_{i=1}^m I(V; Y_i) \leq \delta^2 \sum_{i=1}^m \frac{n_i}{\sigma^2} \min\left\{128 \cdot 16 \log m \cdot H(Y_i), d\right\} + d\left(\frac{2}{49} + 2 \cdot 10^{-5}\right). \tag{6.20}$$

Combining the upper bound (6.20) on the mutual information with the minimax lower bounds in Lemmas 25 and 26, and noting that $6(2/49 + 2 \cdot 10^{-5}) + 6\log 2/d \leq 2/3$ when $d \geq 10$ yields the following minimax bound:

$$\mathfrak{M}_{n,m}^{\mathsf{ind}}(\theta, \mathcal{P}, B_{1:m}) \geq \delta^2\big(\lfloor d/6 \rfloor + 1\big)\left(\frac{1}{3} - \frac{6\delta^2 \sum_{i=1}^m n_i \min\{128 \cdot 16 \log m \cdot H(Y_i), d\}}{d\sigma^2}\right). \tag{6.21}$$

Using this result, we now complete the proof of the theorem By Shannon's source coding theorem, we have $H(Y_i) \leq B_i$, whence the minimax bound (6.21) becomes

$$\delta^2\big(\lfloor d/6 \rfloor + 1\big)\left(\frac{1}{3} - \frac{6\delta^2 \sum_{i=1}^m n_i \min\{128 \cdot 16 B_i \log m, d\}}{d\sigma^2}\right).$$

In particular, if we choose

$$\delta^2 = \min\left\{1, \frac{\sigma^2}{16^2 \max_i n_i \log m}, \frac{d\sigma^2}{36 \sum_{i=1}^m n_i \min\{128 \cdot 16 B_i \log m, d\}}\right\}, \tag{6.22}$$

we obtain

$$\frac{1}{3} - \delta^2 \frac{6 \sum_{i=1}^m n_i \min\{128 \cdot 16 B_i \log m, d\}}{d\sigma^2} \geq \frac{1}{6},$$

which yields the minimax lower bound

$$\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}, B_{1:m}) \geq \frac{1}{6}\big(\lfloor d/6 \rfloor + 1\big) \min\left\{1, \frac{\sigma^2}{16^2 \max_i n_i \log m}, \frac{d\sigma^2}{36 \sum_{i=1}^m n_i \min\{128 \cdot 16 B_i \log m, d\}}\right\}.$$

To obtain inequality (6.4), we simplify by assuming that $n_i \equiv n$ for all i and perform simple algebraic manipulations, noting that the minimax lower bound $d\sigma^2/(nm)$ holds independently of any communication budget.

Case $d < 10$: As in the proof of Proposition 2, we cover this case by reducing to dimension $d = 1$ and applying Le Cam's method, in particular via the lower bound (6.17). Substituting in the δ^2 assignment (6.22) and the relation $H(Y_i) \leq B_i$ into Lemmas 28 and 31, we find that

$$I(V; Y_{1:m}) \leq \sum_{i=1}^m I(V; Y_i) \leq \frac{1}{36} + \frac{2}{49} + 2 \cdot 10^{-5} < \frac{1}{8}.$$

Applying Le Cam's method to this upper bound implies the lower bound $\mathfrak{M}^{\mathsf{ind}}_{n,m}(\theta, \mathcal{P}, B_{1:m}) \geq \delta^2/4$, which completes the proof.

6.3.5 Proof of Proposition 3

Proposition 3 involves both a lower and upper bound. We prove the upper bound by exhibiting a specific interactive protocol Π^*, and the lower bound via an application of Proposition 1.

Proof of lower bound: Applying Proposition 1 requires a lower bound on the packing entropy of $\Theta = [-1, 1]^d$. By a standard volume argument [12], the 2δ-packing entropy has lower bound

$$\mathcal{E}_\Theta(2\delta) \geq \log_2 \frac{\text{Volume}(\Theta)}{\text{Volume}(\{x \in \mathbb{R}^d : \|x\|_2 \leq 2\delta\})} \geq d \log\left(\frac{1}{2\delta}\right).$$

Inverting the relation $B = \mathcal{E}_\Theta(\delta) = \mathcal{E}_\Theta(1/(mn))$ yields the lower bound.

Proof of upper bound: Consider the following communication protocol $\Pi^* \in \mathcal{A}_{\text{inter}}(B, \mathcal{P})$:

(i) Each machine $i \in [m]$ computes its local minimum $a_j^{(i)} = \min\{X_j^{(i,k)} : k \in [n]\}$ for each coordinate $j \in [d]$.

(ii) Machine 1 broadcasts the vector $a^{(1)}$, where each of its components is quantized to accuracy $(mn)^{-2}$ in $[-2, 2]$, rounding down, using $2d \log_2(2mn)$ bits. Upon receiving the broadcast, all machines initialize global minimum variables $s_j \leftarrow a_j^{(1)}$ for $j = 1, \ldots, d$.

(iii) In the order $i = 2, 3, \ldots, m$, machine i performs the following operations:

  (i) Find all indices j such that $a_j^{(i)} < s_j$, calling this set J_i. For each index $j \in J_i$, machine i updates $s_j \leftarrow a_j^{(i)}$, and then broadcasts the list of indices J_i (which requires $|J_i| \lceil \log_2 d \rceil$ bits) and the associated values s_j, using a total of $|J_i| \lceil \log_2 d \rceil + 2|J_i| \log(2mn)$ bits.

  (ii) All other machines update their local vectors s after receiving machine i's update.

(iv) One machine outputs $\widehat{W}\theta = s + 1$.

Using the protocol Π^* above, it is clear that for each $j \in [d]$ we have computed the global minimum

$$s_j^* = \min\left\{X_j^{(i,k)} \mid i \in [m], k \in [n]\right\}$$

to within accuracy $1/(mn)^2$ (because of quantization). As a consequence, classical convergence analyses (e.g. [119]) yield that the estimator $\widehat{W}\theta = s+1$ achieves the minimax optimal convergence rate $\mathbb{E}[\|\widehat{W}\theta - \theta\|_2^2] \leq c\frac{d}{(mn)^2}$, where $c > 0$ is a numerical constant.

It remains to understand the communication complexity of the protocol Π^*. To do so, we study steps 2 and 3. In Step 2, machine 1 sends a $2d \log_2(2mn)$-bit message as Y_1. In Step 3, machine i sends $|J_i|(\lceil \log_2 d \rceil + 2\log_2(2mn))$ bits, that is, at most

$$\sum_{j=1}^{d} 1_{\left(a_j^{(i)} < \min\{a_j^{(1)}, \ldots, a_j^{(i-1)}\}\right)} (\lceil \log_2 d \rceil + 2\log_2(2mn))$$

bits, as no message is sent for index j if $a_j^{(i)} \geq \min\{a_j^{(1)}, \ldots, a_j^{(i-1)}\}$. By inspection, this event happens with probability bounded by $1/i$, so we find that the expected length of message Y_i is

$$\mathbb{E}[L_i] \leq \frac{d(\lceil \log_2 d \rceil + 2\log_2(2mn))}{i}.$$

Putting all pieces together, we obtain that

$$\begin{aligned}\mathbb{E}[L] = \sum_{i=1}^{m} \mathbb{E}[L_i] &\leq 2d\log(2mn) + \sum_{i=2}^{m} \frac{d(\lceil \log_2 d \rceil + 2\log_2(2mn))}{i} \\ &\leq d\big[2\log_2(2mn) + \log(m)(\lceil \log d \rceil + 2\log_2(2mn))\big].\end{aligned}$$

6.3.6 Proof of Theorem 11

As in the proof of Theorem 10, we choose $V \in \{-1,1\}^d$ uniformly at random, and for some $\delta > 0$ to be chosen, we define the parameter vector $\theta := \delta V$. Suppose that machine i draws a sample $X^{(i)} \in \mathbb{R}^{d\times n}$ of size n i.i.d. according to a $N(\theta, \sigma^2 I_{d\times d})$ distribution. We denote the full sample—across all machines—along dimension j by X_j. In addition, for each $j \in [d]$, we let $V_{\setminus j}$ denote the coordinates of $V \in \{-1,1\}^d$ except the jth coordinate.

Although the local samples are independent, since we now allow for interactive protocols, the messages can be dependent: the sequence of random variables $Y = (Y_1, \ldots, Y_T)$ is generated in such a way that the distribution of Y_t is $(X^{(i_t)}, Y_{1:t-1})$-measurable, where $i_t \in \{1, \ldots, m\}$ is the machine index upon which Y_t is based (i.e. the machine sending message Y_t). We assume without loss of generality that the sequence $\{i_1, i_2, \ldots, \}$ is fixed in advance: if the choice of index i_t is not fixed but chosen based on $Y_{1:t-1}$ and X, we simply say there exists a default value (say no communication or $Y_t = \perp$) that indicates "nothing" and has no associated bit cost.

To prove our result, we require an analogue of Lemma 29 (cf. the proof of Theorem 10). Assuming temporarily that $d = 1$, we prove our analogue for one-dimensional interactive protocols, and in the sequel, we show how it is possible to we reduce multi-dimension problems to this statement. As in the proof of Theorem 10, we abstract a bit from our specific setting, instead assuming a likelihood ratio constraint, and provide a data processing inequality for our setting. Let V be a Bernoulli variable uniformly distributed on $\{-1,1\}$, and let $P_{X^{(i)}}$ denote the probability measure of the ith sample $X^{(i)} \in \mathbb{R}^n$. Suppose there is a (measurable) set G such that for any $\nu, \nu' \in \{-1,1\}$, we have

$$\sup_{S\in\sigma(G)} \frac{P_{X^{(i)}}(S \mid \nu)}{P_{X^{(i)}}(S \mid \nu')} \leq e^{\alpha}. \tag{6.23}$$

Finally, let E be a $\{0,1\}$-valued indicator variable for the event $\cap_{i=1}^m \{X^{(i)} \in G\}$.

Lemma 32. *Under the previously stated conditions, we have*

$$I(V;Y) \leq 2\big(e^{4\alpha} - 1\big)^2 I(X;Y) + H(E) + P(E = 0).$$

See Section 6.4.4.1 for the proof.

Using this lemma as a building block, we turn to the case that $X^{(i)}$ is d-dimensional. Making an explicit choice of the set G, we obtain the following concrete bound on the mutual information. The lemma applies to any pair (a, δ) of non-negative reals such that

$$\frac{\sqrt{n}a\delta}{\sigma^2} \leq \frac{1}{4} \quad \text{and} \quad a \geq \delta\sqrt{n},$$

and, as in Lemma 30, involves the binary entropy function $h_2(p) := -p\log(p) - (1-p)\log(1-p)$.

Lemma 33. *Under the preceding conditions, we have*

$$I(V_j; Y \mid V_{\setminus j}) \leq 128 \frac{\delta^2 n a^2}{\sigma^4} I(X_j; Y \mid V_{\setminus j}) + m h_2(p^*) + m p^*$$

where $p^* := \min \left\{ 2 \exp\left(- \frac{(a - \sqrt{n}\delta)^2}{2\sigma^2} \right), \frac{1}{2} \right\}$.

We prove the lemma in Section 6.4.4.2.

To apply Lemma 33, we require two further intermediate bounds on mutual information terms. By the chain rule for mutual information [53], we have

$$\begin{aligned} I(V; Y) = \sum_{j=1}^{d} I(V_j; Y \mid V_{1:j-1}) &= \sum_{j=1}^{d} [H(V_j \mid V_{1:j-1}) - H(V_j \mid Y, V_{1:j-1})] \\ &\overset{(i)}{=} \sum_{j=1}^{d} [H(V_j \mid V_{\setminus j}) - H(V_j \mid Y, V_{1:j-1})], \end{aligned}$$

where equality (i) follows since the variable V_j is independent of $V_{\setminus j}$. Since conditioning can only reduce entropy, we have $H(V_j \mid Y, V_{1:j-1}) \geq H(V_j \mid Y, V_{\setminus j})$, and hence

$$I(V; Y) \leq \sum_{j=1}^{d} [H(V_j \mid V_{\setminus j}) - H(V_j \mid Y, V_{\setminus j})] = \sum_{j=1}^{d} I(V_j; Y \mid V_{\setminus j}). \tag{6.24}$$

Turning to our second intermediate bound, by the definition of the conditional mutual information, we have

$$\begin{aligned} \sum_{j=1}^{d} I(X_j; Y \mid V_{-j}) &= \sum_{j=1}^{d} [H(X_j \mid V_{\setminus j}) - H(X_j \mid Y, V_{\setminus j})] \\ &\overset{(i)}{=} H(X) - \sum_{j=1}^{d} H(X_j \mid Y, V_{\setminus j}) \\ &\overset{(ii)}{\leq} H(X) - \sum_{j=1}^{d} H(X_j \mid Y, V) \\ &\overset{(iii)}{\leq} H(X) - H(X \mid Y, V) = I(X; Y, V), \end{aligned}$$

where equality (i) follows by the independence of X_j and $V_{\setminus j}$, inequality (ii) because conditioning reduces entropy, and inequality (iii) because $H(X \mid Y, V) \leq \sum_j H(X_j \mid Y, V)$. Noting that $I(X; V, Y) \leq H(V, Y) \leq H(Y) + d$, we conclude that

$$\sum_{j=1}^{d} I(X_j; Y \mid V_{\setminus j}) \leq I(X; V, Y) \leq H(Y) + d. \tag{6.25}$$

We can now complete the proof of the theorem. Combining inequalities (6.24) and (6.25) with Lemma 33 yields

$$I(V;Y) \leq 128\frac{\delta^2 n a^2}{\sigma^4}(H(Y)+d) + m\,d\,h_2(p^*) + mdp^*, \tag{6.26}$$

where we recall that $p^* = \{2\exp\big(-\frac{(a-\sqrt{n}\delta)^2}{2\sigma^2}\big), \frac{1}{2}\}$.

Inequality (6.26) is the analog of inequality (6.19b) in the proof of Theorem 10; accordingly, we may follow the same steps to complete the proof. The case $d < 10$ is entirely analogous; the case $d \geq 10$ involves a few minor differences that we describe here.

Setting $a = 4\sigma\sqrt{\log m}$, choosing some δ in the interval $[0, \frac{\sigma}{16\sqrt{n\log m}}]$, and then applying the bound (6.26), we find that

$$I(V;Y) \leq \delta^2\frac{128\cdot 16 n\log m}{\sigma^2}(H(Y)+d) + d\left(\frac{2}{49} + 2\cdot 10^{-5}\right).$$

Combining this upper bound on the mutual information with Lemmas 25 and 26, we find that

$$\begin{aligned}\mathfrak{M}_{n,m}^{\mathsf{inter}}(\theta,\mathcal{P},B) &\geq \delta^2(\lfloor d/6\rfloor+1)\left[\frac{1}{3} - (128\cdot 16\cdot 6)\delta^2\frac{(H(Y)+d)n\log m}{d\sigma^2}\right] \\ &\geq \delta^2(\lfloor d/6\rfloor+1)\left[\frac{1}{3} - (128\cdot 16\cdot 6)\delta^2\frac{(B+d)n\log m}{d\sigma^2}\right],\end{aligned}$$

where the second step follows since $H(Y) \leq B$, by the source coding theorem [53]. Setting

$$\delta^2 = \min\left\{1, \frac{\sigma^2}{256 n\log m}, \frac{d\sigma^2}{2048\cdot 36\cdot n(B+d)\log m}\right\} = \min\left\{1, \frac{d\sigma^2}{2048\cdot 36\cdot n(B+d)\log m}\right\},$$

we obtain

$$\mathfrak{M}_{n,m}^{\mathsf{inter}}(\theta,\mathcal{P},B) \geq \delta^2\frac{\lfloor d/6\rfloor+1}{6} = \min\left\{1, \frac{d\sigma^2}{2048\cdot 36\cdot n(B+d)\log m}\right\}\frac{\lfloor d/6\rfloor+1}{6}$$

Noting that $\mathfrak{M}_{n,m}^{\mathsf{inter}}(\theta,\mathcal{P},B) \geq \mathfrak{M}_{n,m}^{\mathsf{inter}}(\theta,\mathcal{P},\infty) \gtrsim \frac{\sigma^2 d}{nm}$ completes the proof.

6.4 Proofs of technical results

6.4.1 Contractions in total variation distance

As noted in the main body of the chapter, our results rely on certain quantitative data processing inequalities. They are inspired by results on information contraction under privacy constraints developed by a subset of the current authors (Duchi et al. [66]). In this section,

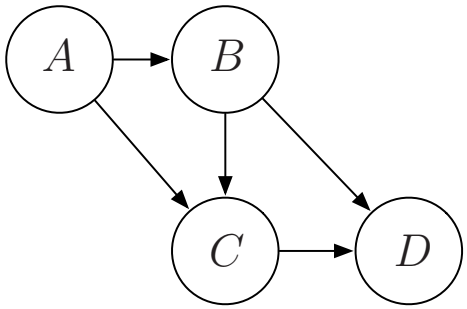


Figure 6.1: Graphical model for Lemma 34

we present a technical result—a contraction in total variation distance—that underlies many of our proofs of the data processing inequalities (Lemmas 27, 29, and 32).

Consider a random vector (A, B, C, D) with joint distribution $P_{A,B,C,D}$, where A, C and D take on discrete values. Denoting the conditional distribution of A given B by $P_{A|B}$, suppose that (A, B, C, D) respect the conditional independence properties defined by the directed graphical model in Figure 6.1. In analytical terms, we have

$$P_{A,B,C,D} = P_A P_{B|A} P_{C|A,B} P_{D|B,C}. \tag{6.27}$$

In addition, we assume that there exist functions $\Psi_1 : \mathcal{A}\times\sigma(\mathcal{C}) \to \mathbb{R}_+$ and $\Psi_2 : \mathcal{B}\times\sigma(\mathcal{C}) \to \mathbb{R}_+$ such that

$$P_C(S \mid A, B) = \Psi_1(A, S)\Psi_2(B, S) \tag{6.28}$$

for any (measureable) set S in the range $\mathcal{C}$ of C. Since C is assumed discrete, we abuse notation and write $P(C = c \mid A, B) = \Psi_1(A, c)\Psi_2(B, c)$. Lastly, suppose that

$$\sup_{S\in\sigma(\mathcal{B})} \frac{P_B(S \mid A = a)}{P_B(S \mid A = a')} \leq \exp(\alpha) \qquad \text{for all } a, a' \in \mathcal{A}. \tag{6.29}$$

The following lemma applies to the absolute difference

$$\Delta(a, C, D) := \big|P(A = a \mid C, D) - P(A = a \mid C)\big|.$$

Lemma 34. *Under conditions* (6.27), (6.28), *and* (6.29), *we have*

$$\Delta(a, C, D) \leq 2\big(e^{2\alpha} - 1\big) \min\big\{P(A = a \mid C), P(A = a \mid C, D)\big\} \left\|P_B(\cdot \mid C, D) - P_B(\cdot \mid C)\right\|_{\mathrm{TV}}.$$

Proof By assumption, A is independent of D given $\{B, C\}$. Thus we may write

$$\Delta(a, C, D) = \Big| \int P(A = a \mid B = b, C)\big(dP_B(b \mid C, D) - dP_B(b \mid C)\big)\Big|.$$

Combining this equation with the relation $\int_{\mathcal{B}} P(A = a \mid C)\big(dP_B(b \mid C, D) - dP_B(b \mid C)\big) = 0$, we find that

$$\Delta(a, C, D) = \Big| \int_{\mathcal{B}} \big(P(A = a \mid B = b, C) - P(A = a \mid C)\big)\big(dP_B(b \mid C, D) - dP_B(b \mid C)\big)\Big|.$$

Using the fact that $|\int f(b)d\mu(b)| \leq \sup_b\{|f(b)|\} \int |d\mu(b)|$ for any signed measure μ on $\mathcal{B}$, we conclude from the previous equality that for any version $P_A(\cdot \mid B, C)$ of the conditional probability of A given $\{B, C\}$ that since $\int |d\mu| = \|\mu\|_{\mathrm{TV}}$,

$$\Delta(a, C, D) \leq 2 \sup_{b \in \mathcal{B}}\{|P(A = a \mid B = b, C) - P(A = a \mid C)|\} \, \|P_B(\cdot \mid C, D) - P_B(\cdot \mid C)\|_{\mathrm{TV}} \, .$$

Thus, to prove the lemma, it is sufficient to show[2] that for any $b \in \mathcal{B}$

$$|P(A = a \mid B = b, C) - P(A = a \mid C)| \leq (e^{2\alpha} - 1) \min\{P(A = a \mid C), P(A = a \mid C, D)\}. \tag{6.30}$$

To prove this upper bound, we consider the joint distribution (6.27) and likelihood ratio bound (6.29). The distributions $\{P_B(\cdot \mid A = a)\}_{a \in \mathcal{A}}$ are all absolutely continuous with respect to one another by assumption (6.29), so it is no loss of generality to assume that there exists a density $p_B(\cdot \mid A = a)$ for which $P(B \in S \mid A = a) = \int p_B(b \mid A = a)d\mu(b)$, for some fixed measure μ and for which the ratio $p_B(b \mid A = a)/p_B(b \mid A = a') \in [e^{-\alpha}, e^{\alpha}]$ for all b. By elementary conditioning we have for any $S_{\mathcal{B}} \in \sigma(\mathcal{B})$ and $c \in \mathcal{C}$ that

$$\begin{aligned}
&P(A = a \mid B \in S_{\mathcal{B}}, C = c) \\
&\quad = \frac{P(A = a, B \in S_{\mathcal{B}}, C = c)}{P(B \in S_{\mathcal{B}}, C = c)} \\
&\quad = \frac{P(B \in S_{\mathcal{B}}, C = c \mid A = a)P(A = a)}{\sum_{a' \in \mathcal{A}} P(A = a')P(B \in S_{\mathcal{B}}, C = c \mid A = a')} \\
&\quad = \frac{P(A = a) \int_{S_{\mathcal{B}}} P(C = c \mid B = b, A = a)p_B(b \mid A = a)d\mu(b)}{\sum_{a' \in \mathcal{A}} P(A = a') \int_{S_{\mathcal{B}}} P(C = c \mid B = b, A = a')p_B(b \mid A = a')d\mu(b)},
\end{aligned}$$

where for the last equality we used the conditional independence assumptions (6.27). But now we recall the decomposition formula (6.28), and we can express the likelihood functions by

$$P(A = a \mid B \in S_{\mathcal{B}}, C = c) = \frac{P(A = a) \int_{S_{\mathcal{B}}} \Psi_1(a, c)\Psi_2(b, c)p_B(b \mid A = a)d\mu(b)}{\sum_{a'} P(A = a') \int_{S_{\mathcal{B}}} \Psi_1(a', c)\Psi_2(b, c)p_B(b \mid A = a')d\mu(b)}.$$

As a consequence, there is a version of the conditional distribution of A given B and C such that

$$P(A = a \mid B = b, C = c) = \frac{P(A = a)\Psi_1(a, c)p_B(b \mid A = a)}{\sum_{a'} P(A = a')\Psi_1(a', c)p_B(b \mid A = a')}. \tag{6.31}$$

[2] If $P(A = a \mid C)$ is undefined, we simply set it to have value 1 and assign $P(A = a \mid B, C) = 1$ as well.

Define the shorthand

$$\beta = \frac{P(A = a)\Psi_1(a, c)}{\sum_{a' \in \mathcal{A}} P(A = a')\Psi_1(a', c)}.$$

We claim that

$$e^{-\alpha}\beta \leq P(A = a \mid B = b, C = c) \leq e^{\alpha}\beta. \tag{6.32}$$

Assuming the correctness of bound (6.32), we establish inequality (6.30). Indeed, $P(A = a \mid C = c)$ is a weighted average of $P(A = a \mid B = b, C = c)$, so we also have the same upper and lower bound for $P(A = a \mid C)$, that is

$$e^{-\alpha}\beta \leq P(A = a \mid C) \leq e^{\alpha}\beta.$$

The conditional independence assumption that A is independent of D given B, C (recall Figure 6.1 and the product (6.27)) implies

$$\begin{aligned} P(A = a \mid C = c, D = d) &= \int_{\mathcal{B}} P(A = a \mid B = b, C = c, D = d) dP_B(b \mid C = c, D = d) \\ &= \int_{\mathcal{B}} P(A = a \mid B = b, C = c) dP_B(b \mid C = c, D = d), \end{aligned}$$

and the final integrand belongs to $\beta[e^{-\alpha}, e^{\alpha}]$. Combining the preceding three displayed expressions, we find that

$$\begin{aligned} |P(A = a \mid B = b, C) - P(A = a \mid C)| &\leq \big(e^{\alpha} - e^{-\alpha}\big)\beta \\ &\leq \big(e^{\alpha} - e^{-\alpha}\big)e^{\alpha} \min\big\{P(A = a \mid C), P(A = a \mid C, D)\big\}. \end{aligned}$$

This completes the proof of the upper bound (6.30).

It remains to prove inequality (6.32). We observe from expression (6.31) that

$$P(A = a \mid B = b, C) = \frac{P(A = a)\Psi_1(a, C)}{\sum_{a' \in \mathcal{A}} P(A = a')\Psi_1(a', C)\frac{p_B(b|A=a')}{p_B(b|A=a)}}.$$

By the likelihood ratio bound (6.29), we have $p_B(b \mid A = a')/p_B(b \mid A = a) \in [e^{-\alpha}, e^{\alpha}]$, and combining this with the above equation yields inequality (6.32). □

6.4.2 Auxiliary results for Proposition 2

In this appendix, we collect the proofs of auxiliary results involved in the proof of Proposition 2.

6.4.2.1 Proof of Lemma 27

Let $Y = Y_i$; throughout the proof we suppress the dependence on the index i (and similarly let $X = X^{(i)}$ denote a single fixed sample). We begin with the observation that by the chain rule for mutual information,

$$I(V; Y) = \sum_{j=1}^{d} I(V_j; Y \mid V_{1:j-1}).$$

Using the definition of mutual information and non-negativity of the KL-divergence, we have

$$\begin{aligned} I(V_j; Y \mid V_{1:j-1}) &= \mathbb{E}_{V_{1:j-1}}\left[\mathbb{E}_Y\left[D_{\rm kl}\left(P_{V_j}(\cdot \mid Y, V_{1:j-1}) \| P_{V_j}(\cdot \mid V_{1:j-1})\right) \mid V_{1:j-1}\right]\right] \\ &\le \mathbb{E}_{V_{1:j-1}}\big[\mathbb{E}_Y\big[D_{\rm kl}\left(P_{V_j}(\cdot \mid Y, V_{1:j-1}) \| P_{V_j}(\cdot \mid V_{1:j-1})\right) \\ &\qquad + D_{\rm kl}\left(P_{V_j}(\cdot \mid V_{1:j-1}) \| P_{V_j}(\cdot \mid Y, V_{1:j-1})\right) \mid V_{1:j-1}\big]\big]. \end{aligned}$$

Now, we require an argument that builds off of our technical Lemma 34. We claim that Lemma 34 implies that

$$\begin{aligned} &|P(V_j = \nu_j \mid V_{1:j-1}, Y) - P(V_j = \nu_j \mid V_{1:j-1})| \\ &\quad \le 2(e^{2\alpha} - 1) \min\left\{P(V_j = \nu_j \mid V_{1:j-1}, Y), P(V_j = \nu_j \mid V_{1:j-1})\right\} \\ &\qquad \times \left\|P_{X_j}(\cdot \mid V_{1:j-1}, Y) - P_{X_j}(\cdot \mid V_{1:j-1})\right\|_{\rm TV}. \end{aligned} \tag{6.33}$$

Indeed, making the identification

$$V_j \to A, \quad X_j \to B, \quad V_{1:j-1} \to C, \quad Y \to D,$$

the random variables satisfy the condition (6.27) clearly, condition (6.28) because $V_{1:j-1}$ is independent of V_j and X_j, and condition (6.29) by construction. This gives inequality (6.33) by our independence assumptions. Expanding our KL divergence bound, we have

$$\begin{aligned} &D_{\rm kl}\left(P_{V_j}(\cdot \mid Y, V_{1:j-1}) \| P_{V_j}(\cdot \mid V_{1:j-1})\right) + D_{\rm kl}\left(P_{V_j}(\cdot \mid V_{1:j-1}) \| P_{V_j}(\cdot \mid Y, V_{1:j-1})\right) \\ &\qquad = \sum_{\nu_j} \left(P_{V_j}(\nu_j \mid Y, V_{1:j-1}) - P_{V_j}(\nu_j \mid V_{1:j-1})\right) \log \frac{P_{V_j}(\nu_j \mid Y, V_{1:j-1})}{P_{V_j}(\nu_j \mid V_{1:j-1})}. \end{aligned}$$

Now, using the elementary inequality for $a, b \ge 0$ that

$$\left|\log \frac{a}{b}\right| \le \frac{|a - b|}{\min\{a, b\}},$$

inequality (6.33) implies that

$$\begin{aligned} &\left(P_{V_j}(\nu_j \mid Y, V_{1:j-1}) - P_{V_j}(\nu_j \mid V_{1:j-1})\right) \log \frac{P_{V_j}(\nu_j \mid Y, V_{1:j-1})}{P_{V_j}(\nu_j \mid V_{1:j-1})} \\ &\le \frac{\left(P_{V_j}(\nu_j \mid Y, V_{1:j-1}) - P_{V_j}(\nu_j \mid V_{1:j-1})\right)^2}{\min\{P_{V_j}(\nu_j \mid Y, V_{1:j-1}), P_{V_j}(\nu_j \mid V_{1:j-1})\}} \\ &\le 4(e^{2\alpha} - 1)^2 \min\left\{P_{V_j}(\nu_j \mid Y, V_{1:j-1}), P_{V_j}(\nu_j \mid V_{1:j-1})\right\} \left\|P_{X_j}(\cdot \mid V_{1:j-1}, Y) - P_{X_j}(\cdot \mid V_{1:j-1})\right\|_{\rm TV}^2. \end{aligned}$$

Substituting this into our bound on KL-divergence, we obtain

$$I(V_j; Y \mid V_{1:j-1}) = \mathbb{E}_{V_{1:j-1}}\left[\mathbb{E}_Y\left[D_{\mathrm{kl}}\left(P_{V_j}(\cdot \mid Y, V_{1:j-1}) \| P_{V_j}(\cdot \mid V_{1:j-1})\right) \mid V_{1:j-1}\right]\right]$$
$$\leq 4(e^{2\alpha}-1)^2 \mathbb{E}_{V_{1:j-1}}\left[\mathbb{E}_Y\left[\left\|P_{X_j}(\cdot \mid V_{1:j-1}, Y) - P_{X_j}(\cdot \mid V_{1:j-1})\right\|_{\mathrm{TV}}^2 \mid V_{1:j-1}\right]\right].$$

Using Pinsker's inequality, we then find that

$$\mathbb{E}_{V_{1:j-1}}\left[\mathbb{E}_Y\left[\left\|P_{X_j}(\cdot \mid V_{1:j-1}, Y) - P_{X_j}(\cdot \mid V_{1:j-1})\right\|_{\mathrm{TV}}^2 \mid V_{1:j-1}\right]\right]$$
$$\leq \frac{1}{2}\mathbb{E}_{V_{1:j-1}}\left[\mathbb{E}_Y\left[D_{\mathrm{kl}}\left(P_{X_j}(\cdot \mid Y, V_{1:j-1}) \| P_{X_j}(\cdot \mid V_{1:j-1})\right) \mid V_{1:j-1}\right]\right] = \frac{1}{2} I(X_j; Y \mid V_{1:j-1}).$$

In particular, we have

$$I(V_j; Y \mid V_{1:j-1}) \leq 2\left(e^{2\alpha}-1\right)^2 I(X_j; Y \mid V_{1:j-1}). \tag{6.34}$$

Lastly, we argue that $I(X_j; Y \mid V_{1:j-1}) \leq I(X_j; Y \mid X_{1:j-1})$. Indeed, we have by definition that

$$\begin{aligned} I(X_j; Y \mid V_{1:j-1}) &\overset{(i)}{=} H(X_j) - H(X_j \mid Y, V_{1:j-1}) \\ &\overset{(ii)}{\leq} H(X_j) - H(X_j \mid Y, V_{1:j-1}, X_{1:j-1}) \\ &\overset{(iii)}{=} H(X_j \mid X_{1:j-1}) - H(X_j \mid Y, X_{1:j-1}) = I(X_j; Y \mid X_{1:j-1}). \end{aligned}$$

Here, equality (i) follows since X_j is independent of $V_{1:j-1}$, inequality (ii) because conditioning reduces entropy, and equality (iii) because X_j is independent of $X_{1:j-1}$. Thus

$$I(V; Y) = \sum_{j=1}^{d} I(V_j; Y \mid V_{1:j-1}) \leq 2(e^{2\alpha}-1)^2 \sum_{j=1}^{d} I(X_j; Y \mid X_{1:j-1}) = 2(e^{2\alpha}-1)^2 I(X; Y),$$

which completes the proof.

6.4.2.2 Proof of Lemma 28

By assumption, the message Y_i is constructed based only on $X^{(i)}$. Therefore, we have

$$\begin{aligned} I(V; Y_{1:m}) = \sum_{i=1}^{m} I(V; Y_i \mid Y_{1:i-1}) &= \sum_{i=1}^{m} H(Y_i \mid Y_{1:i-1}) - H(Y_i \mid V, Y_{1:i-1}) \\ &\leq \sum_{i=1}^{m} H(Y_i) - H(Y_i \mid V, Y_{1:i-1}) \\ &= \sum_{i=1}^{m} H(Y_i) - H(Y_i \mid V) = \sum_{i=1}^{m} I(V; Y_i) \end{aligned}$$

where we have used that conditioning reduces entropy and Y_i is conditionally independent of $Y_{1:i-1}$ given V.

6.4.2.3 Proof of inequality (6.16)

Let P_ν be shorthand for $P_Y(\cdot \mid V = \nu)$. The triangle inequality implies that

$$\|P_\nu - P_{\nu'}\|_{\rm TV} \le \|P_\nu - (1/2)(P_\nu + P_{\nu'})\|_{\rm TV} + \frac{1}{2}\|P_\nu - P_{\nu'}\|_{\rm TV},$$

and similarly swapping the roles of ν' and ν, whence

$$\|P_\nu - P_{\nu'}\|_{\rm TV} \le 2\min\{\|P_\nu - (1/2)(P_{\nu'} + P_\nu)\|_{\rm TV}, \|P_{\nu'} - (1/2)(P_{\nu'} + P_\nu)\|_{\rm TV}\}.$$

By Pinsker's inequality, we thus have the upper bound

$$\begin{aligned}\|P_\nu - P_{\nu'}\|_{\rm TV}^2 &\le 2\min\{D_{\rm kl}\left(P_\nu \| (1/2)(P_\nu + P_{\nu'})\right), D_{\rm kl}\left(P_{\nu'} \| (1/2)(P_\nu + P_{\nu'})\right)\} \\ &\le D_{\rm kl}\left(P_\nu \| (1/2)(P_\nu + P_{\nu'})\right) + D_{\rm kl}\left(P_{\nu'} \| (1/2)(P_\nu + P_{\nu'})\right) = 2I(Y;V)\end{aligned}$$

by the definition of mutual information.

6.4.3 Auxiliary results for Theorem 10

In this appendix, we collect the proofs of auxiliary results involved in the proof of Theorem 10.

6.4.3.1 Proof of Lemma 29

This proof is similar to that Lemma 27, but we must be careful when conditioning on events of the form $X_j^{(i)} \in G_j$. For notational simplicity, we again suppress all dependence of X and Y on the machine index i. Our goal is to prove that

$$I(V_j; Y \mid V_{1:j-1}) \le H(E_j) + P(E_j = 0) + 2\left(e^{4\alpha} - 1\right)^2 I(X_j; Y \mid V_{1:j-1}). \tag{6.35}$$

Up to the additive terms, this is equivalent to the earlier bound (6.34) in the proof of Lemma 27, so that proceeding *mutatis mutandis* completes the proof. We now turn to proving inequality (6.35).

We begin by noting that $I(X;Y \mid Z) \le I(X, W; Y \mid Z)$ for any random variables W, X, Y, Z, because conditioning reduces entropy:

$$I(X;Y \mid Z) = H(Y \mid Z) - H(Y \mid X, Z) \le H(Y \mid Z) - H(Y \mid W, X, Z) = I(X, W; Y \mid Z). \tag{6.36}$$

As a consequence, recalling the random variable E_j (the indicator of $X_j \in G_j$), we have

$$\begin{aligned}I(V_j; Y \mid V_{1:j-1}) \le I(V_j; Y, E_j \mid V_{1:j-1}) &= I(V_j; Y \mid E_j, V_{1:j-1}) + I(V_j; E_j \mid V_{1:j-1}) \\ &\le I(V_j; Y \mid E_j, V_{1:j-1}) + H(E_j \mid V_{1:j-1}) \\ &= I(V_j; Y \mid E_j, V_{1:j-1}) + H(E_j),\end{aligned} \tag{6.37}$$

where the final equality follows because E_j is independent of $V_{1:j-1}$. Comparing to inequality (6.35), we need only control the first term in the bound (6.37).

To that end, note that given E_j, the variable V_j is independent of $V_{1:j-1}$, $X_{1:j-1}$, $V_{j+1:d}$, and $X_{j+1:d}$. Moreover, by the assumption in the lemma we have for any $S \in \sigma(G_j)$ that

$$\frac{P_{X_j}(S \mid V = \nu, E_j = 1)}{P_{X_j}(S \mid V = \nu', E_j = 1)} = \frac{P_{X_j}(S \mid V = \nu)}{P_{X_j}(X_j \in G_j \mid V = \nu)} \frac{P_{X_j}(X_j \in G_j \mid V = \nu')}{P_{X_j}(X_j \in S \mid V = \nu')} \leq \exp(2\alpha).$$

Applying Lemma 34 yields that the difference

$$\Delta_j := P(V_j = \nu_j \mid V_{1:j-1}, Y, E_j = 1) - P(V_j = \nu_j \mid V_{1:j-1}, E_j = 1)$$

is bounded as

$$\begin{aligned} |\Delta_j| \leq 2\big(e^{4\alpha} - 1\big) \big\| P_{X_j}(\cdot \mid V_{1:j-1}, Y, E_j = 1) - P_{X_j}(\cdot \mid V_{1:j-1}, E_j = 1) \big\|_{\mathrm{TV}} \\ \times \min\big\{ P(V_j = \nu_j \mid V_{1:j-1}, Y, E_j = 1), P(V_j = \nu_j \mid V_{1:j-1}, E_j = 1) \big\} \end{aligned}$$

(cf. the inequality (6.33) in the proof of Lemma 27). Proceeding as in the proof of Lemma 27, this expression leads to the bound

$$I(V_j; Y \mid V_{1:j-1}, E_j = 1) \leq 2\big(e^{4\alpha} - 1\big)^2 I(X_j; Y \mid V_{1:j-1}, E_j = 1). \tag{6.38}$$

By the definition of conditional mutual information,

$$\begin{aligned} I(V_j; Y \mid E_j, V_{1:j-1}) &= P(E_j = 1) I(V_j; Y \mid V_{1:j-1}, E_j = 1) + P(E_j = 0) I(V_j; Y \mid V_{1:j-1}, E_j = 0) \\ &\leq I(V_j; Y \mid V_{1:j-1}, E_j = 1) + P(E_j = 0) \log 2, \end{aligned}$$

where the inequality follows because $V_j \in \{-1, 1\}$. But combining this inequality with the bounds (6.38) and (6.37) gives the desired result (6.35).

6.4.3.2 Proof of Lemma 30

In order to prove inequality (6.19a), we note that $V \to X^{(i)} \to Y_i$ forms a Markov chain. Thus, the classical data-processing inequality [53] implies that

$$I(V; Y_i) \leq I(V; X^{(i)}) \leq \sum_{k=1}^{n_i} I(V; X^{(i,k)}).$$

Let P_ν denote the conditional distribution of $X^{(i,k)}$ given $V = \nu$. Then the convexity of the KL-divergence establishes inequality (6.19a) via

$$I(V; X^{(i,k)}) \leq \frac{1}{|\mathcal{V}|^2} \sum_{\nu, \nu' \in \mathcal{V}} D_{\mathrm{kl}}\left(P_\nu \| P_{\nu'}\right) = \frac{\delta^2}{2\sigma^2} \frac{1}{|\mathcal{V}|^2} \sum_{\nu, \nu' \in \mathcal{V}} \|\nu - \nu'\|_2^2 = \frac{d\delta^2}{\sigma^2}.$$

To prove inequality (6.19b), we apply Lemma 29. First, consider two one-dimensional normal distributions, each with n_i independent observations and variance σ^2, but where one has mean δ and the other mean $-\delta$. For fixed $a \geq 0$, the ratio of their densities is

$$\frac{\exp(-\frac{1}{2\sigma^2} \sum_{l=1}^{n_i} (x_l - \delta)^2)}{\exp(-\frac{1}{2\sigma^2} \sum_{l=1}^{n_i} (x_l + \delta)^2)} = \exp\left(\frac{\delta}{\sigma^2} \sum_{l=1}^{n_i} x_l\right) \leq \exp\left(\frac{\sqrt{n_i}\delta a}{\sigma^2}\right)$$

whenever $|\sum_l x_l| \le \sqrt{n_i}a$. As a consequence, we see that by taking the sets

$$G_j = \left\{x \in \mathbb{R}^{n_i} : \left|\sum_{l=1}^{n_i} x_l\right| \le \sqrt{n_i}a\right\},$$

we satisfy the conditions of Lemma 29 with the quantity α defined as $\alpha = \sqrt{n_i}\delta a/\sigma^2$. In addition, when $\alpha \le 1.2564$, we have $\exp(\alpha) - 1 \le 2\alpha$, so under the conditions of the lemma, $\exp(4\alpha) - 1 = \exp(4\sqrt{n_i}\delta a/\sigma^2) - 1 \le 8\sqrt{n_i}\delta a/\sigma^2$. Recalling the definition of the indicator random variable $E_j = 1\{X_j^{(i)} \in G_j\}$ from Lemma 29, we obtain

$$I(V; Y_i) \le 128\frac{\delta^2 a^2}{\sigma^4} n_i I(X^{(i)}; Y_i) + \sum_{j=1}^{d} H(E_j) + \sum_{j=1}^{d} P(E_j = 0). \tag{6.39}$$

Comparing this inequality with inequality (6.19b), we see that we must bound the probability of the event $E_j = 0$.

Bounding $P(E_j = 0)$ is not challenging, however. From standard Gaussian tail bounds, we have for Z_l distributed i.i.d. according to $N(\delta, \sigma^2)$ that

$$\begin{aligned} P(E_j = 0) &= P\left(\left|\sum_{l=1}^{n_i} Z_l\right| \ge \sqrt{n_i}a\right) \\ &= P\left(\sum_{l=1}^{n_i}(Z_l - \delta) \ge \sqrt{n_i}a - n\delta\right) + P\left(\sum_{l=1}^{n_i}(Z_l - \delta) \le -\sqrt{n_i}a - n\delta\right) \\ &\le 2\exp\left(-\frac{(a - \sqrt{n_i}\delta)^2}{2\sigma^2}\right). \end{aligned} \tag{6.40}$$

Since $h_2(p) \le h_2(\frac{1}{2})$ and $I(V; Y_i) \le d\log 2$ regardless, this provides the bounds on the entropy and probability terms in inequality (6.39) to yield the result (6.19b).

6.4.3.3 Proof of Lemma 31

Combining inequalities (6.19a) and (6.19b) yields

$$\begin{aligned} I(V; Y_i) \le \frac{n_i\delta^2}{\sigma^2} \min\left\{128\frac{a^2}{\sigma^2}H(Y_i), d\right\} + d\, h_2\left(\min\left\{2\exp\left(-\frac{(a - \sqrt{n_i}\delta)^2}{2\sigma^2}\right), \frac{1}{2}\right\}\right) \\ + 2d\exp\left(-\frac{(a - \sqrt{n_i}\delta)^2}{2\sigma^2}\right), \end{aligned} \tag{6.41}$$

true for all $a, \delta \ge 0$ and n_i, σ^2 such that $\sqrt{n_i}a\delta \le 1.2564\sigma^2/4$ and $a \ge \delta\sqrt{n_i}$.

Now, we consider each of the terms in the bound in inequality (6.41) in turn, finding settings of δ and a so that each term is small. Let us set $a = 4\sigma\sqrt{\log m}$. We begin with the third term in the bound (6.41), where we note that by defininining δ_3 as the positive root of

$$\delta_3^2 := \frac{\sigma^2}{16 \cdot 16\log(m)\max_i n_i}, \tag{6.42}$$

then for $0 \le \delta \le \delta_3$ the conditions $\frac{\sqrt{n_i}a\delta}{\sigma^2} \le \frac{1.2564}{4}$ and $\sqrt{n_i}\delta \le a$ in Lemma 30 are satisfied. In addition, we have $(a - \sqrt{n_i}\delta)^2 \ge (4 - 1/256)^2\sigma^2 \log m \ge 15\sigma^2 \log m$ for $0 \le \delta \le \delta_3$, so for such δ

$$\sum_{i=1}^{m} 2\exp\left(-\frac{(a-\sqrt{n_i}\delta)^2}{2\sigma^2}\right) \le 2m\exp(-(15/2)\log m) = \frac{2}{m^{15/2}} < 2 \cdot 10^{-5}.$$

Secondly, we have $h_2(q) \le (6/5)\sqrt{q}$ for $q \ge 0$. As a consequence, we see that for δ_2 chosen identically to the choice (6.42) for δ_3, we have

$$\sum_{i=1}^{m} 2h_2\left(2\exp\left(-\frac{(a-\sqrt{n_i}\delta_2)^2}{2\sigma^2}\right)\right) \le \frac{12m}{5}\sqrt{2}\exp(-(15/4)\log m) < \frac{2}{49}.$$

In particular, with the choice $a = 4\sigma\sqrt{\log m}$ and for all $0 \le \delta \le \delta_3$, inequality (6.41) implies the desired bound (6.20).

6.4.4 Auxiliary results for Theorem 11

In this appendix, we collect the proofs of auxiliary results for Theorem 11.

6.4.4.1 Proof of Lemma 32

We state an intermediate claim from which Lemma 32 follows quickly. Let us temporarily assume that the set G in the statement of the lemma is $G = \text{range}(X^{(i)})$, so that there is no restriction on the distributions $P_{X^{(i)}}$, that is, the likelihood ratio bound (6.23) holds for all measurable sets S. We claim that in this case,

$$I(V;Y) \le 2\left(e^{2\alpha} - 1\right)^2 I(X;Y). \tag{6.43}$$

Assuming that we have established inequality (6.43), the proof of Lemma 32 follows, *mutatis mutandis*, as in the proof of Lemma 29 from Lemma 27.

Let us now prove the claim (6.43). By the chain-rule for mutual information, we have

$$I(V;Y) = \sum_{t=1}^{T} I(V;Y_t \mid Y_{1:t-1}).$$

Let $P_{Y_t}(\cdot \mid Y_{1:t-1})$ denote the (marginal) distribution of Y_t given $Y_{1:t-1}$ and define $P_V(\cdot \mid Y_{1:t})$ to be the distribution of V conditional on $Y_{1:t}$. Then we have by marginalization that

$$P_V(\cdot \mid Y_{1:t-1}) = \int P_V(\cdot \mid Y_{1:t-1}, y_t) dP_{Y_t}(y_t \mid Y_{1:t-1})$$

and thus

$$I(V;Y_t \mid Y_{1:t-1}) = \mathbb{E}_{Y_{1:t-1}}\left[\mathbb{E}_{Y_t}\left[D_{\text{kl}}\left(P_V(\cdot \mid Y_{1:t}) \| P_V(\cdot \mid Y_{1:t-1})\right) \mid Y_{1:t-1}\right]\right].$$

We now bound the above KL divergence using the assumed likelihood ratio bound on P_X in the lemma (when $G = \mathcal{X}$, the entire sample space).

By the nonnegativity of the KL divergence, we have

$$\begin{aligned}
&D_{\mathrm{kl}}\left(P_V(\cdot \mid Y_{1:t}) \| P_V(\cdot \mid Y_{1:t-1})\right) \\
&\le D_{\mathrm{kl}}\left(P_V(\cdot \mid Y_{1:t}) \| P_V(\cdot \mid Y_{1:t-1})\right) + D_{\mathrm{kl}}\left(P_V(\cdot \mid Y_{1:t-1}) \| P_V(\cdot \mid Y_{1:t})\right) \\
&= \sum_{\nu \in \mathcal{V}} \left(p_V(\nu \mid Y_{1:t-1}) - p_V(\nu \mid Y_{1:t})\right) \log \frac{p_V(\nu \mid Y_{1:t-1})}{p_V(\nu \mid Y_{1:t})}
\end{aligned}$$

where p_V denotes the p.m.f. of V.

Next we claim that the difference $\Delta_t := \left|p_V(\nu \mid Y_{1:t-1}) - p_V(\nu \mid Y_{1:t})\right|$ is upper bounded as

$$|\Delta_t| \le 2\left(e^{2n\alpha} - 1\right) \min\left\{p_V(\nu \mid Y_{1:t-1}), p_V(\nu \mid Y_{1:t})\right\} \left\|P_{X^{(i_t)}}(\cdot \mid Y_{1:t}) - P_{X^{(i_t)}}(\cdot \mid Y_{1:t-1})\right\|_{\mathrm{TV}}. \tag{6.44}$$

Deferring the proof of this claim to the end of this section, we give the remainder of the proof. First, by a first-order convexity argument, we have

$$|\log a - \log b| \le \frac{|a-b|}{\min\{a,b\}} \quad \text{for any } a, b > 0.$$

Combining this bound with inequality (6.44) yields

$$\begin{aligned}
\Delta_t \log \frac{p_V(\nu \mid Y_{1:t-1})}{p_V(\nu \mid Y_{1:t})} &\le \frac{\Delta_t^2}{\min\{p_V(\nu \mid Y_{1:t-1}), p_V(\nu \mid Y_{1:t})\}} \\
&\le 4\left(e^{2n\alpha} - 1\right)^2 \min\left\{p_V(\nu \mid Y_{1:t-1}), p_V(\nu \mid Y_{1:t})\right\} \left\|P_{X^{(i_t)}}(\cdot \mid Y_{1:t}) - P_{X^{(i_t)}}(\cdot \mid Y_{1:t-1})\right\|_{\mathrm{TV}}^2.
\end{aligned}$$

Since p_V is a p.m.f., we have the following upper bound on the symmetrized KL divergence between $P_V(\cdot \mid Y_{1:t})$ and $P_V(\cdot \mid Y_{1:t-1})$:

$$\begin{aligned}
&D_{\mathrm{kl}}\left(P_V(\cdot \mid Y_{1:t}) \| P_V(\cdot \mid Y_{1:t-1})\right) + D_{\mathrm{kl}}\left(P_V(\cdot \mid Y_{1:t-1}) \| P_V(\cdot \mid Y_{1:t})\right) \\
&\le 4\left(e^{2n\alpha} - 1\right)^2 \left\|P_{X^{(i_t)}}(\cdot \mid Y_{1:t}) - P_{X^{(i_t)}}(\cdot \mid Y_{1:t-1})\right\|_{\mathrm{TV}}^2 \sum_{\nu \in \mathcal{V}} \min\left\{p_V(\nu \mid Y_{1:t-1}), p_V(\nu \mid Y_{1:t})\right\} \\
&\le 4\left(e^{2n\alpha} - 1\right)^2 \left\|P_{X^{(i_t)}}(\cdot \mid Y_{1:t}) - P_{X^{(i_t)}}(\cdot \mid Y_{1:t-1})\right\|_{\mathrm{TV}}^2 \\
&\le \frac{1}{2} D_{\mathrm{kl}}\left(P_{X^{(i_t)}}(\cdot \mid Y_{1:t}) \| P_{X^{(i_t)}}(\cdot \mid Y_{1:t-1})\right),
\end{aligned}$$

where the final step follows from Pinsker's inequality. Taking expectations, we have

$$\frac{1}{2}\mathbb{E}_{Y_{1:t-1}}\left[\mathbb{E}_{Y_t}\left[D_{\mathrm{kl}}\left(P_{X^{(i_t)}}(\cdot \mid Y_{1:t}) \| P_{X^{(i_t)}}(\cdot \mid Y_{1:t-1})\right) \mid Y_{1:t-1}\right]\right] = \frac{1}{2} I(X^{(i_t)}; Y_t \mid Y_{1:t-1}).$$

Finally, because conditioning reduces entropy (recall inequality (6.36)), we have

$$I(X^{(i_t)}; Y_t \mid Y_{1:t-1}) \le I(X; Y_t \mid Y_{1:t-1}).$$

By the chain rule for mutual information, we have $\sum_{t=1}^{T} I(X; Y_t \mid Y_{1:t-1}) = I(X; Y)$, so the proof is complete.

Proof of inequality (6.44) It remains to prove inequality (6.44): in order to do so, we establish a one-to-one correspondence between the variables in Lemma 34 and the variables in inequality (6.44). Let us begin by making the identifications

$$V \to A \qquad X^{(i_t)} \to B \qquad Y_{1:t-1} \to C \qquad Y_t \to D.$$

For Lemma 34 to hold, we must verify conditions (6.27), (6.28), and (6.29). For condition (6.27) to hold, Y_t must be independent of V given $\{Y_{1:t-1}, X^{(i_t)}\}$. Since the distribution of $P_{Y_t}(\cdot \mid Y_{1:t-1}, X^{(i_t)})$ is measurable-$\{Y_{1:t-1}, X^{(i_t)}\}$, condition (6.29) is satisfied by the assumption in Lemma 32.

Finally, for condition (6.28) to hold, we must be able to factor the conditional probability of $Y_{1:t-1}$ given $\{V, X^{(i_t)}\}$ as

$$P(Y_{1:t-1} = y_{1:t-1} \mid V, X^{(i_t)}) = \Psi_1(V, y_{1:t-1})\Psi_2(X^{(i_t)}, y_{1:t-1}). \tag{6.45}$$

To prove this decomposition, notice that

$$P(Y_{1:t-1} = y_{1:t-1} \mid V, X^{(i_t)}) = \prod_{k=1}^{t-1} P(Y_k = y_k \mid Y_{1:k-1}, V, X^{(i_t)}).$$

For any $k \in \{1, \ldots, t-1\}$, if $i_k = i_t$—that is, the message Y_k is generated based on sample $X^{(i_t)} = X^{(i_k)}$—then Y_k is independent of V given $\{X^{(i_t)}, Y_{1:k-1}\}$. Thus, $P_{Y_k}(\cdot \mid Y_{1:k-1}, V, X^{(i_t)})$ is measurable-$\{X^{(i_t)}, Y_{1:k-1}\}$. If the kth index $i_k \neq i_t$, then Y_k is independent of $X^{(i_t)}$ given $\{Y_{1:k-1}, V\}$ by construction, which means $P_{Y_k}(\cdot \mid Y_{1:k-1}, V, X^{(i_t)}) = P_{Y_k}(\cdot \mid Y_{1:k-1}, V)$, thereby verifying the decomposition (6.45). Thus, we have verified that each of the conditions of Lemma 34 holds, so that inequality (6.44) follows.

6.4.4.2 Proof of Lemma 33

To prove Lemma 33, fix an arbitrary realization $\nu_{\setminus j} \in \{-1, 1\}^{d-1}$ of $V_{\setminus j}$. Conditioning on $V_{\setminus j} = \nu_{\setminus j}$, note that $\nu_j \in \{-1, 1\}$, and consider the distributions of the jth coordinate of each (local) sample $X_j^{(i)} \in \mathbb{R}^n$,

$$P_{X_j^{(i)}}(\cdot \mid V_j = \nu_j, V_{\setminus j} = \nu_{\setminus j}) \quad \text{and} \quad P_{X_j^{(i)}}(\cdot \mid V_j = -\nu_j, V_{\setminus j} = \nu_{\setminus j}).$$

We claim that these distributions—with appropriate constants—satisfy the conditions of Lemma 32. Indeed, fix $a \geq 0$, take the set $G = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq \sqrt{n}a\}$, and set the log-likelihood ratio parameter $\alpha = \sqrt{n}\delta a/\sigma^2$. Then the random variable $E_j = 1$ if $X_j^{(i)} \in G$ for all $i = 1, \ldots, m$, and we may apply precisely proof of Lemma 32 (we still obtain the factorization (6.45) by conditioning everything on $V_{\setminus j} = \nu_{\setminus j}$). Thus we obtain

$$\begin{aligned} I(V_j; Y \mid V_{\setminus j} = \nu_{\setminus j}) \leq 2\left(e^{4\alpha} - 1\right)^2 I(X_j; Y \mid V_{\setminus j} = \nu_{\setminus j}) \\ + H(E_j \mid V_{\setminus j} = \nu_{\setminus j}) + P(E_j = 0 \mid V_{\setminus j} = \nu_{\setminus j}). \end{aligned} \tag{6.46}$$

Of course, the event E_j is independent of $V_{\setminus j}$ by construction, so that $P(E_j = 0 \mid V_{\setminus j}) = P(E_j = 0)$ and $H(E_j \mid V_{\setminus j} = \nu_{\setminus j}) = H(E_j)$, and standard Gaussian tail bounds (cf. the proof of Lemma 30 and inequality (6.40)) imply that

$$H(E_j) \le m h_2\left(2\exp\left(-\frac{(a-\sqrt{n}\delta)^2}{2\sigma^2}\right)\right) \quad \text{and} \quad P(E_j = 0) \le 2m\exp\left(-\frac{(a-\sqrt{n}\delta)^2}{2\sigma^2}\right).$$

Thus by integrating over $V_{\setminus j} = \nu_{\setminus j}$, inequality (6.46) implies the lemma.

Chapter 7

Communication complexity of matrix rank estimation

In this chapter, we study a specific linear algebraic problem: computing the generalized rank of a matrix. Given a parameter $c \geq 0$, the generalized rank of an $n \times n$ positive semidefinite matrix A corresponds to the number of eigenvalues that are larger than c. It is denoted by $\text{rank}(A, c)$, with the usual rank corresponding to the special case $c = 0$. Note that the generalized rank of a matrix cannot be represented as a closed-form expression of the matrix. In the distributed setting, the matrix A is stored across multiple machines, and we are interested in the communication complexity of this problem.

Estimating the generalized rank of a matrix is useful for many applications. In the context of large-scale principal component analysis (PCA) [67, 100], it is overly expensive to compute the full eigendecomposition before deciding when to truncate it. Thus, an important first step is to estimate the rank of the matrix of interest in order to determine how many dimensions will be sufficient to describe the data. The rank also provides useful information for determining the tuning parameter of robust PCA [38] and collaborative filtering algorithms [177, 173]. In the context of numerical linear algebra, a number of eigensolvers [182, 162, 176] for large-scale scientific applications are based on divide-and-conquer paradigms. It is a prerequisite of these algorithms to know the approximate number of eigenvalues located in a given interval. Estimating the generalized rank of a matrix is also needed in the context of sampling-based methods for randomized numerical linear algebra [83, 138]. For these methods, the rank of a matrix determines the number of samples required for a desired approximation accuracy.

Motivated by large-scale data analysis problems, we study the generalized rank estimation problem in a distributed setting, in which the matrix A can be decomposed as the the sum of m matrices

$$A := \sum_{i=1}^{m} A_i, \tag{7.1}$$

where each matrix A_i is stored on a separate machine i. Thus, a distributed algorithm needs

to communicate between m machines to perform the estimation. There are other equivalent formulations of this problem. For example, suppose that machine i has a design matrix $X_i \in \mathbb{R}^{n \times N_i}$ and we want to determine the rank of the aggregated design matrix

$$X := (X_1, X_2, \ldots, X_m) \in \mathbb{R}^{n \times N} \quad \text{where } N := \textstyle\sum_{i=1}^{m} N_i.$$

Recall that the singular values of matrix X are equal to the square root of the eigenvalues of the matrix XX^T. If we define $A_i := X_i X_i^T$, then equation (7.1) implies that

$$A = \sum_{i=1}^{m} A_i = \sum_{i=1}^{m} X_i X_i^T = XX^T.$$

Thus, determining the generalized rank of the matrix X reduces to the problem of determining the rank of the matrix A. In this chapter, we focus on the formulation given by equation (7.1).

The standard way of computing the generalized matrix rank, or more generally of computing the number of eigenvalues within a given interval, is to exploit Sylvester's law of inertia [77]. Concretely, if the matrix $A - cI$ admits the decomposition $A - cI = LDL^T$, where L is unit lower triangular and D is diagonal, then the number of eigenvalues of matrix A that are greater than c is the same as the number of positive entries in the diagonal of D. While this method yields an exact count, in the distributed setting it requires communicating the entire matrix A. Due to bandwidth limitations and network delays, the $\Theta(n^2)$ communication cost is a significant bottleneck on the algorithmic efficiency. For a matrix of rank r, the power method [77] can be used to compute the top r eigenvalues, which reduces the communication cost to $\Theta(rn)$. However, this cost is still prohibitive for moderate sizes of r. Recently, Napoli et al. [150] studied a more efficient randomized approach for approximating the eigenvalue counts based on Chebyshev polynomial approximation of high-pass filters. When applying this algorithm to the distributed setting, the communication cost is $\Theta(pn)$, where p is the degree of Chebyshev polynomials. However, the authors note that polynomials of high degree can be necessary.

In this chapter, we study the communication complexity of distributed algorithms for the problem of generalized rank estimation, in both the deterministic and randomized settings. We establish upper bounds by deriving practical, communication-efficient algorithms, and we also establish complexity-theoretic lower bounds. Our first main result shows that no deterministic algorithm is efficient in terms of communication. In particular, communicating $\Omega(n^2)$ bits is necessary for all deterministic algorithms to approximate the matrix rank with constant relative error. That such algorithms cannot be viewed as efficient is due to the fact that by communicating $\mathcal{O}(n^2)$ bits, we are able to compute all eigenvalues and the corresponding eigenvectors. In contrast to the inefficiency of deterministic algorithms, we propose a randomized algorithm that approximates matrix rank by communicating $\widetilde{\mathcal{O}}(n)$ bits. When the matrix is of rank r, the relative approximation error is $1/\sqrt{r}$. Under the same relative error, we show that $\Omega(n)$ bits of communication is necessary, establishing the optimality of our algorithm. This is in contrast with the $\Omega(rn)$ communication complexity

lower bound for randomized PCA [101]. The difference shows that estimating the eigenvalue count using a randomized algorithm is easier than estimating the top r eigenpairs.

7.1 Problem formulation

In this section, we begin with more details on the problem of estimating generalized matrix ranks, as well as the notations on communication complexity.

7.1.1 Generalized matrix rank

Given an $n \times n$ positive semidefinite matrix A, we use $\sigma_1(A) \geq \sigma_2(A) \geq \cdots \geq \sigma_n(A) \geq 0$ to denote its ordered eigenvalues. For a given constant $c \geq 0$, the generalized rank of order c is given by

$$\mathrm{rank}(A, c) = \sum_{k=1}^{n} \mathbb{I}[\sigma_k(A) > c], \tag{7.2}$$

where $\mathbb{I}[\sigma_k(A) > c]$ is a 0-1-valued indicator function for the event that $\sigma_k(A)$ is larger than c. Since $\mathrm{rank}(A, 0)$ is equal to the usual rank of a matrix, we see the motivation for using the generalized rank terminology. We assume that $\|A\|_2 = \sigma_1(A) \leq 1$ so that the problem remains on a standardized scale.

In an m-machine distributed setting, the matrix A can be decomposed as a sum $A = \sum_{i=1}^{m} A_i$, where the $n \times n$ matrix A_i is stored on machine i. We study distributed protocols, to be specified more precisely in the following section, in which each machine i performs local computation involving the matrix A_i, and the machines then exchange messages so to arrive at an estimate $\widehat{r}(A) \in [n] := \{0, \ldots, n\}$. Our goal is to obtain an estimate that is close to the rank of the matrix in the sense that

$$(1 - \delta)\mathrm{rank}(A, c_1) \leq \widehat{r}(A) \leq (1 + \delta)\mathrm{rank}(A, c_2), \tag{7.3}$$

where $c_1 > c_2 \geq 0$ and $\delta \in [0, 1)$ are user-specified constants. The parameter $\delta \in [0, 1)$ upper bounds the relative error of the approximation. The purpose of assuming different thresholds c_1 and c_2 in bound (7.3) is to handle the ambiguous case when the matrix A has many eigenvalues smaller but very close to c_1. If we were to set $c_1 = c_2$, then any estimator $\widehat{r}(A)$ would be strictly prohibited to take these eigenvalues into account. However, since these eigenvalues are so close to the threshold, distinguishing them from other eigenvalues just above the threshold is obviously difficult (but for an uninteresting reason). Setting $c_1 > c_2$ allows us to expose the more fundamental sources of difficulty in the problem of estimating generalized matrix ranks.

7.1.2 Communication complexity

We have introduced the basic concepts of communication complexity in Chapter 2. In this chapter, we adopt the bulk of the framework of communication complexity, but with one minor twist in how we define "correctness" in computing the function. For our problem, each machine is a player, and the i^{th} player holds the matrix A_i. Our function of interest is given by $F(A_1, \ldots, A_m) = \text{rank}(\sum_{i=1}^m A_i)$. The public blackboard setting corresponds to a broadcast-free model, in which each machine can send messages to a master node, then the master node broadcasts the messages to all other machines without additional communication cost.

Let us now clarify the notion of "correctness" used in this chapter. In the standard communication model, a protocol Π is said to correctly compute the function F if the output of the protocol is exactly equal to $F(A_1, \ldots, A_m)$. In this chapter, we allow approximation errors in the computation, as specified by the parameters (c_1, c_2), which loosen the matrix rank to the generalized matrix ranks, and the tolerance parameter $\delta \in (0, 1)$. More specifically, we say:

Definition 2. *A protocol Π correctly computes the rank of the matrix A up to tolerances (c_1, c_2, δ) if the output $\widehat{r}(A)$ satisfies inequality (7.3).*

Given this definition of correctness, we denote the deterministic communication complexity of the rank estimation problem by $\mathcal{D}(c_1, c_2, \delta)$, and the corresponding randomized communication complexity by $\mathcal{R}_\epsilon(c_1, c_2, \delta)$. The goal of this chapter is to study these two quantities, especially their dependence on the dimension n of matrices.

In addition to allowing for approximation error, our analysis—in contrast to most classical communication complexity—allows the input matrices $\{A_i\}_{i=1}^m$ to take real values. However, doing so does not make the problem substantially harder. Indeed, in order to approximate the matrices in elementwise ℓ_∞-norm up to τ rounding error, it suffices to discretize each matrix entry using $\mathcal{O}(\log(1/\tau))$ bits. As we discuss in more detail in the sequel, this type of discretization has little effect on the communication complexity.

7.2 Bounds for deterministic algorithms

We begin by studying the communication complexity of deterministic algorithms. Here our main result shows that the trivial algorithm—the one in which each machine transmits essentially its whole matrix—is optimal up to logarithmic factors. In the statement of the theorem, we assume that the n-dimensional matrix A is known to have rank in the interval[1] $[r, 2r]$ for some integer $r \leq n/4$.

Theorem 12. *For matrices A with rank in the interval $[r, 2r]$:*

(a) For all $0 \leq c_2 < c_1$ and $\delta \in (0, 1)$, we have $\mathcal{D}(c_1, c_2, \delta) = \mathcal{O}\Big(mrn \log\big(\frac{mrn}{c_1 - c_2}\big)\Big)$.

[1] We use an interval assumption, as the problem becomes trivial if the rank is fixed exactly.

(b) For two machines $m = 2$, constants $0 \leq c_2 < c_1 < 1/20$ and $\delta \in (0, 1/12)$, we have $\mathcal{D}(c_1, c_2, \delta) = \Omega(rn)$.

When the matrix A has rank r that grows proportionally with its dimension n, the lower bound in part (b) shows that deterministic communication complexity is surprisingly large: it scales as $\Theta(n^2)$, which is as large as transmitting the full matrices. Up to logarithmic factors, this scaling is matched by the upper bound in part (a). It is proved by analyzing an essentially trivial algorithm: for each index $i = 2, \ldots, m$, machine i encodes a reduced rank representation of the matrix A_i, representing each matrix entry by $\log_2 \left(\frac{12mrn}{c_1 - c_2} \right)$ bits. It sends this quantized matrix $\widetilde{A}_i$ to the first machine. Given these received messages, the first machine then computes the matrix sum $\widetilde{A} := A_1 + \sum_{i=2}^{m} \widetilde{A}_i$, and it outputs $\widehat{r}(A)$ to be the largest integer k such that $\sigma_k(\widetilde{A}) > (c_1 + c_2)/2$.

On the other hand, in order to prove the lower bound, we consider a two-party rank testing problem. Consider two agents holding matrices A_1 and A_2, respectively, such that the matrix sum $A := A_1 + A_2$ has operator norm at most one. Suppose that exactly one of the two following conditions are known to hold:

- the matrix A has rank r, or
- the matrix A has rank between $\frac{6r}{5}$ and $2r$, and in addition its $(6r/5)^{th}$ eigenvalue is lower bounded as $\sigma_{\frac{6r}{5}}(A) > \frac{1}{20}$.

The goal is to decide which case is true by exchanging the minimal number of bits between the two agents. Denoting this problem by `RankTest`, the proof of part (a) proceeds by showing first that $D(\texttt{RankTest}) = \Omega(rn)$, and then reducing from the `RankTest` problem to the matrix rank estimation problem. See Section 7.4.1 for the proof.

7.3 Bounds for randomized algorithms

We now turn to the study of randomized algorithms, for which we see that the communication complexity is substantially lower. In Section 7.3.1, we propose a randomized algorithm with $\widetilde{\mathcal{O}}(n)$ communication cost, and in Section 7.3.3, we establish a lower bound that matches this upper bound in various regimes.

7.3.1 Upper bounds via a practical algorithm

In this section, we present an algorithm based on uniform polynomial approximations for estimating the generalized matrix rank. Let us first provide some intuition for the algorithm before defining it more precisely. For a fixed pair of scalars $c_1 > c_2 \geq 0$, consider the function

$H_{c_1,c_2} : \mathbb{R} \to [0,1]$ given by

$$H_{c_1,c_2}(x) := \begin{cases} 1 & \text{if } x > c_1 \\ 0 & \text{if } x < c_2 \\ \frac{x-c_2}{c_1-c_2} & \text{otherwise.} \end{cases} \tag{7.4}$$

As illustrated in Figure 7.1, it is a piecewise linear approximation to a step function. The squared function $H^2_{c_1,c_2}$ is useful in that it can be used to sandwich the generalized ranks of a matrix A. In particular, given a positive semidefinite matrix A with ordered eigenvalues $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_n(A) \geq 0$, observe that we have

$$\operatorname{rank}(A, c_1) \leq \sum_{i=1}^{n} H^2_{c_1,c_2}(\sigma_i(A)) \leq \operatorname{rank}(A, c_2). \tag{7.5}$$

Our algorithm exploits this sandwich relation in estimating the generalized rank.

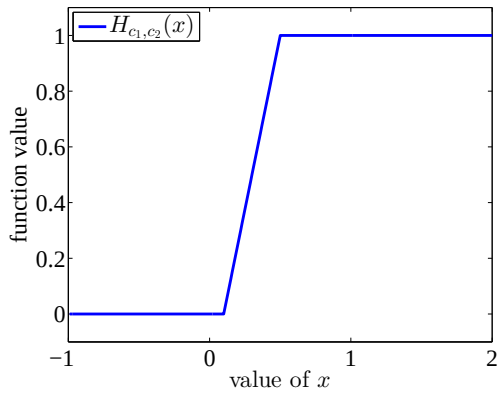


Figure 7.1: An illustration of the function $x \mapsto H_{c_1,c_2}(x)$ with $c_1 = 0.5$ and $c_2 = 0.1$.

In particular, suppose that we can find a polynomial function $f : \mathbb{R} \to \mathbb{R}$ such that $f \approx H_{c_1,c_2}$, and which is extended to a function on the cone of PSD matrices in the standard way. Observe that if σ is an eigenvalue of A, then the spectral mapping theorem [23] ensures that $f(\sigma)$ is an eigenvalue of $f(A)$. Consequently, letting $g \sim N(0, I_{n\times n})$ be a standard Gaussian vector, we have the useful relation

$$\mathbb{E}\Big[\|f(A)g\|_2^2\Big] = \sum_{i=1}^{n} f^2(\sigma_i(A)) \approx \sum_{i=1}^{n} H^2_{c_1,c_2}(\sigma_i(A)). \tag{7.6}$$

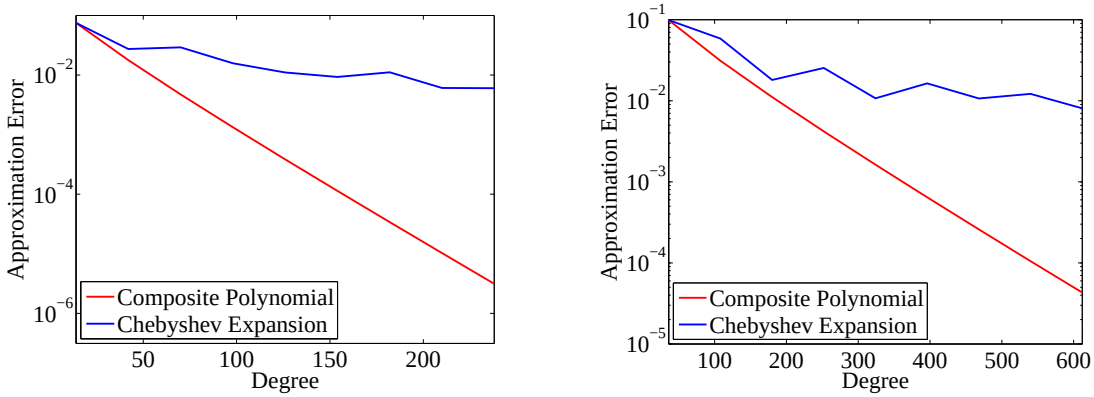
Combined with the sandwich relation (7.5), we see that a polynomial approximation f to the function H_{c_1,c_2} can be used to estimate the generalized rank.

If f is a polynomial function of degree p, then the vector $f(A)g$ can be computed through p rounds of communication. In more detail, in one round of communciation, we can first

compute the matrix-vector product $Ag = \sum_{i=1}^{m} A_i g$. Given the vector Ag, a second round of communication suffices to compute the quantity $A^2 g$. Iterating a total of p times, the first machine is equipped with the collection of vectors $\{g, Ag, A^2 g, \ldots, A^p g\}$, from which it can compute $f(A)g$.

Let us now consider how to obtain a suitable polynomial approximation of the function H_{c_1,c_2}. The most natural choice is a Chebyshev polynomial approximation of the first kind: more precisely, since H_{c_1,c_2} is a continuous function with bounded variation, classical theory [141, Theorem 5.7] guarantees that the Chebyshev expansion converges uniformly to H_{c_1,c_2} over the interval $[0, 1]$. Consequently, we may assume that there is a finite-degree Chebyshev polynomial q_1 of the first kind such that

$$\sup_{x \in [0,1]} |q_1(x) - H_{c_1,c_2}(x)| \leq 0.1. \tag{7.7a}$$



(a) Thresholds $(c_1, c_2) = (0.2, 0.1)$ (b) Thresholds $(c_1, c_2) = (0.02, 0.01)$

Figure 7.2. Comparison of the composite polynomial approximation in Algorithm 6 with the Chebyshev polynomial expansion. The error is measured with the ℓ_∞-norm on the interval $[0, c_2] \cup [c_1, 1]$. The composite polynomial approximation achieves a linear convergence rate as the degree is increased, while the Chebyshev expansion converges at a much slower rate.

By increasing the degree of the Chebyshev polynomial, we could reduce the approximation error (set to 0.1 in the expansion (7.7a)) to an arbitrarily small level. However, a very high degree could be necessary to obtain an arbitrary accuracy. Instead, our strategy is to start with the Chebyshev polynomial q_1 that guarantees the 0.1-approximation error (7.7a), and then construct a second polynomial q_2 such that the composite polynomial function $f = q_2 \circ q_1$ has an approximation error, when measured over the intervals $[0, c_2]$ and $[c_1, 1]$ of interest, that converges linearly in the degree of function f. More precisely, consider the polynomial of degree $2p + 1$ given by

$$q_2(x) = \frac{1}{B(p+1, p+1)} \int_0^x t^p (1-t)^p dt \quad \text{where } B(\cdot, \cdot) \text{ is the Beta function.} \tag{7.7b}$$

Lemma 35. *Consider the composite polynomial* $f(x) := q_2(q_1(x))$*, where the base polynomials* q_1 *and* q_2 *were previously defined in equations* (7.7a) *and* (7.7b) *respectively. Then* $f(x) \in [0,1]$ *for all* $x \in [0,1]$*, and moreover*

$$|f(x) - H_{c_1,c_2}(x)| \leq 2^{-p} \qquad \text{for all } x \in [0,c_2] \cup [c_1,1]. \tag{7.8}$$

See Section 7.6.1 for the proof.

Figure 7.2 provides a comparison of the error in approximating H_{c_1,c_2} for the standard Chebyshev polynomial and the composite polynomial. In order to conduct a fair comparison, we show the approximations obtained by Chebyshev and composite polynomials of the same final degree, and we evaluate the ℓ_∞-norm approximation error on interval $[0,c_2] \cup [c_1,1]$—namely, for a given polynomial approximation h, the quantity

$$\text{Error}(h) := \sup_{x \in [0,c_2] \cup [c_1,1]} |h(x) - H_{c_1,c_2}(x)|.$$

As shown in Figure 7.2 shows, the composite polynomial function achieves a linear convergence rate with respect to its degree. In contrast, the convergence rate of the Chebyshev expansion is sub-linear, and substantially slower than that of the composite function. The comparison highlights the advantage of our approach over the method only based on Chebyshev expansions.

Given the composite polynomial $f = q_2 \circ q_1$, we first evaluate the vector $f(A)g$ in a two-stage procedure. In the first stage, we evaluate $q_1(A)g$, $q_1^2(A)g$, …, $q_1^{2p+1}(A)g$ using the Clenshaw recurrence [50], a procedure proven to be numerically stable [141]. The details are given in Algorithm 5. In the second stage, we substitute the coefficients of q_2 so as to evaluate $q_2(q_1(A))b$. The overall procedure is summarized in Algorithm 6.

Algorithm 5: Evaluation of Chebyshev Polynomial

Input: m machines hold $A_1, A_2, \ldots, A_m \in \mathbb{R}^{n \times n}$; vector $v \in \mathbb{R}^d$; Chebyshev polynomial expansion $q(x) = \frac{1}{2}a_0 T_0(x) + \sum_{i=1}^{d} a_i T_i(x)$.

Output: matrix-vector product $q(A)v$.

1. Initialize vector $b_{d+1} = b_{d+2} = \mathbf{0} \in \mathbb{R}^n$.
2. For $j = d, \ldots, 1, 0$: the first machine broadcasts b_{j+1} to all other machines. Machine i computes $A_i b_{j+1}$ and sends it back to the first machine. The first machine computes

$$b_j := \left(4 \sum_{i=1}^{m} A_i b_{j+1}\right) - 2b_{j+1} - b_{j+2} + a_j v.$$

3. Output $\frac{1}{2}(a_0 v + b_1 - b_3)$;

The following result provides a guarantee for the overall procedure (combination of Algorithm 5 and Algorithm 6) when run with degree $p = \lceil \log_2(2n) \rceil$:

Algorithm 6: Randomized Algorithm for Rank Estimation

Input: Each of m machines hold matrices $A_1, A_2, \ldots, A_m \in \mathbb{R}^{n\times n}$. Tolerance parameters (c_1, c_2), polynomial degree p, and number of repetitions T.

1. (a) Find a Chebyshev expansion q_1 of the function H_{c_1,c_2} satisfying the uniform bound (7.7a).
   (b) Define the degree $2p+1$ polynomial function q_2 by equation (7.7b).
2. (a) Generate a random Gaussian vector $g \sim N(0, I_{n\times n})$.
   (b) Apply Algorithm 5 to compute $q_1(A)g$, and sequentially apply the same algorithm to compute $q_1^2(A)g, \ldots, q_1^{2p+1}(A)g$.
   (c) Evaluate the vector $y := f(A)g = q_2(q_1(A))g$ on the first machine.
3. Repeat Step 2 for T times, obtaining a collection of n-vectors $\{y_1, \ldots, y_T\}$, and output the estimate $\widehat{r}(A) = \frac{1}{T}\sum_{i=1}^{T} \|y_i\|_2^2$.

Theorem 13. *For any $0 \le \delta < 1$, with probability at least $1 - 2\exp\left(-\frac{T\delta^2 \mathrm{rank}(A,c_1)}{32}\right)$, the output of Algorithm 6 satisfies the bounds*

$$(1-\delta)\mathrm{rank}(A, c_1) - 1 \le \widehat{r}(A) \le (1+\delta)(\mathrm{rank}(A, c_2) + 1). \tag{7.9}$$

Moreover, we have the following upper bound on the randomized communication complexity of estimating the generalized matrix rank:

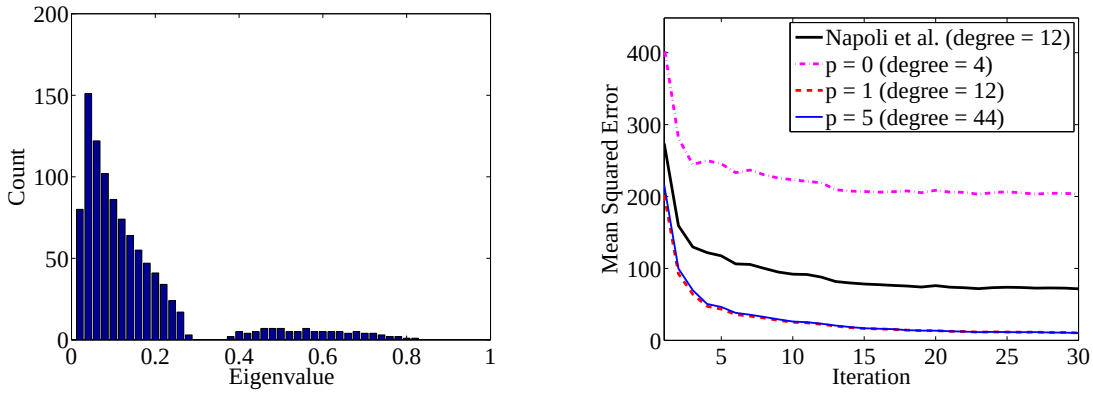
$$\mathcal{R}_\epsilon\left(c_1, c_2, 1/\sqrt{\mathrm{rank}(A, c_1)}\right) = \widetilde{\mathcal{O}}(mn). \tag{7.10}$$

We show in Section 7.3.3 that the upper bound (7.10) is unimprovable up to the logarithmic pre-factors. For now, let us turn to the results of some numerical experiments using Algorithm 6, which show that in addition to being an order-optimal algorithm, it is also practically useful.

7.3.2 Numerical experiments

Given $m = 2$ machines, suppose that machine i (for $i = 1, 2$) receives $N_i = 1000$ data points of dimension $n = 1000$. Each data point x is independently generated as $x = a + \varepsilon$, where $a \sim N(0, \lambda\Sigma)$ and $\varepsilon \sim N(0, \sigma^2 I_{n\times n})$ are random Gaussian vectors. Here $\Sigma \in \mathbb{R}^{n\times n}$ is a low-rank covariance matrix of the form $\Sigma := \sum_{i=1}^{r} u_i u_i^T$, where $\{u_i\}_{i=1}^r$ are an orthonormal set of vectors in $\mathbb{R}^n$ drawn uniformly at random. The goal is to estimate the rank r from the observed $N_1 + N_2 = 2000$ data points.

Let us now describe how to estimate the rank using the covariance matrix of the samples. Notice that $\mathbb{E}[xx^T] = \lambda^2\Sigma + \sigma^2 I_{n\times n}$, of which there are r eigenvalues equal to $\lambda + \sigma^2$ and the



(a) Eigenvalue Distribution (b) Rank Estimation Error

Figure 7.3. Panel (a): distribution of eigenvalues of matrix A. Panel (b): mean squared error of rank estimation versus the number of iterations for the baseline method by Napoli et al. [150], and three versions of Algorithm 6 (with parameters $p \in \{0, 1, 5\}$).

remaining eigenvalues are equal to σ^2. Letting $x_{i,j} \in \mathbb{R}^n$ denote the j-th data point received by machine i, that machine can compute the local sample covariance matrix

$$A_i = \frac{1}{N_1 + N_2} \sum_{j=1}^{N_i} x_{i,j} x_{i,j}^T, \quad \text{for } i = 1, 2.$$

The full sample covariance matrix is given by the sum $A := A_1 + A_2$, and its rank can be estimated using Algorithm 6.

In order to generate the data, we choose the parameters $r = 100$, $\lambda = 0.4$ and $\sigma^2 = 0.1$. These choices motivate the thresholds $c_1 = \lambda + \sigma^2 = 0.5$ and $c_2 = \sigma^2 = 0.1$ in Algorithm 6. We illustrate the behavior of the algorithm for three different choices of the degree parameter p—specifically, $p \in \{0, 1, 5\}$—and for a range of repetitions $T \in \{1, 2, \ldots, 30\}$. Letting $\widehat{r}(A)$ denote the output of the algorithm, we evaluate the mean squared error, $\mathbb{E}[(\widehat{r}(A) - r)^2]$, based on 100 independent runs of the algorithm.

We plot the results of this experiment in Figure 7.3. Panel (a) shows the distribution of eigenvalues of the matrix A. In this plot, there is a gap between the large eigenvalues generated by the low-rank covariance matrix Σ, and small eigenvalues generated by the random Gaussian noise, showing that the problem is relatively easy to solve in the centralized setting. Panel (b) shows the estimation error achieved by the communication-efficient distributed algorithm; notice how the estimation error stabilizes after $T = 30$ repetitions or iterations. We compare our algorithm for $p \in \{0, 1, 5\}$, corresponding to polynomial approximations with degree in $\{4, 12, 44\}$. For the case $p = 0$, the polynomial approximation is implemented by the Chebyshev expansion. For the case $p = 1$ and $p = 5$, the approximation is achieved by the composite function f. As a baseline method, we also implement Napoli et al.'s algorithm [150] in the distributed setting. In particular, their method replaces the function f in Algorithm 6 by a Chebyshev expansion of the high-pass filter $\mathbb{I}(x \geq \frac{c_1+c_2}{2})$. It is observed

that both the Chebyshev expansion with $p = 0$ and the baseline method incur a large bias in the rank estimate, while the composite function's estimation errors are substantially smaller. After $T = 30$ iterations, Algorithm 6 with $p = 1$ achieves a mean squared error close to 10, which means that the relative error of the estimation is around 3%.

7.3.3 Lower Bound

It is natural to wonder if the communication efficiency of Algorithm 6 is optimal. The following theorem shows that, in order to achieve the same $1/\sqrt{r}$ relative error, it is necessary to send $\Omega(n)$ bits. As in our upper bound, we assume that the matrix A satisfies the spectral norm bound $\|A\|_2 \leq 1$. Given an arbirary integer r in the interval $[16, n/4]$, suppose that the generalized matrix ranks satisfy the sandwich relation $r \leq \operatorname{rank}(A, c_1) \leq \operatorname{rank}(A, c_2) \leq 2r$. Under these conditions, we have the following guarantee:

Theorem 14. *For any c_1, c_2 satisfying $c_1 < 2c_2 \leq 1$ and any $\epsilon \leq \epsilon_0$ for some numerical constant ϵ_0, we have*

$$\mathcal{R}_\epsilon\left(c_1, c_2, 1/\sqrt{r}\right) = \Omega(n). \tag{7.11}$$

See Section 7.4.3 for the proof of this lower bound.

According to Theorem 14, for matrices with true rank in the interval $[16, n/2]$, the communication complexity for estimating the rank with relative error $1/\sqrt{r}$ is lower bounded by $\Omega(n)$. This lower bound matches the upper bound provided by Theorem 13. In particular, choosing $r = 16$ yields the worst-case lower bound

$$\mathcal{R}_\epsilon(c_1, c_2, 1/4) = \Omega(n),$$

showing that $\Omega(n)$ bits of communication are necessary for achieving a constant relative error. This lower bound is not trivial relative to the coding length of the correct answer: given that the matrix rank is known to be between r and $2r$, this coding length scales only as $\Omega(\log r)$.

There are several open problems suggested by the result of Theorem 14. First, it would be interesting to strengthen the lower bound (7.11) from $\Omega(n)$ to $\Omega(mn)$, incorporating the natural scaling with the number of machines m. Doing so requires a deeper investigation into the multi-party structure of the problem. Another open problem is to lower bound the communication complexity for arbitrary values of the tolerance parameter δ, say as small as $1/r$. When δ is very small, communicating $\mathcal{O}(mn^2)$ bits is an obvious upper bound, and we are not currently aware of better upper bounds. On the other hand, whether it is possible to prove an $\Omega(n^2)$ lower bound for small δ remains an open question.

7.4 Proofs of main results

In this section, we provide the proofs of our main results, with the proofs of some more technical lemmas deferred to Section 7.6.

7.4.1 Proof of Theorem 12

Let us begin with our first main result on the deterministic communication complexity of the generalized rank problem.

7.4.1.1 Proof of lower bound

We first prove the lower bound stated in part (a) of Theorem 12. Let us recall the `RankTest` problem previously described after the statement of Theorem 12. Alice holds a matrix $A_1 \in \mathbb{R}^{n\times n}$ and Bob holds a matrix $A_2 \in \mathbb{R}^{n\times n}$ such that the matrix sum $A := A_1 + A_2$ has operator norm at most one. Either the matrix A has rank r, or the matrix A has rank between $\frac{6r}{5}$ and $2r$, and in addition its $6r/5$ eigenvalue is lower bounded as $\sigma_{\frac{6r}{5}}(A) > \frac{1}{20}$. The `RankTest` problem is to decide which of these two mutually exclusive alternatives holds. The following lemma provides a lower bound on the deterministic communication complexity of this problem:

Lemma 36. *For any $r \leq n/4$, we have $D(\texttt{RankTest}) = \Omega(rn)$.*

We use Lemma 36 to lower bound $D(c_1, c_2, \delta)$, in particular by reducing to it from the `RankTest` problem. Given a `RankTest` instance, since there are $m \geq 2$ machines, the first two machines can simulate Alice and Bob, holding A_1 and A_2 respectively. All other machines hold a zero matrix. Suppose that $c_1 \leq 1/20$ and $\delta \leq 1/12$. If there is an algorithm achieving the bound (7.3), then if $A = A_1 + A_2$ is of rank r, then

$$\widehat{r}(A) \leq (1+\delta)\text{rank}(A, c_2) \leq \Big(1 + \frac{1}{12}\Big) r = \frac{13r}{12}. \tag{7.12a}$$

Otherwise, the $\frac{6r}{5}$-th eigenvalue of A is greater than $1/20$, so that

$$\widehat{r}(A) \geq (1-\delta)\text{rank}(A, c_1) \geq \Big(1 - \frac{1}{12}\Big)\frac{6r}{5} = \frac{11r}{10} > \frac{13r}{12}. \tag{7.12b}$$

In conjunction, inequality (7.12a) and (7.12b) show that we can solve the `RankTest` problem by testing whether or not $\widehat{r}(A) \leq \frac{13r}{12}$. Consequently, the deterministic communication complexity $D(c_1, c_2, \delta)$ is lower bounded by the communication complexity of `RankTest`.

In order to complete the proof of Theorem 12(a), it remains to prove Lemma 36, and we do so using a randomized construction. Let us say that a matrix $Q \in \mathbb{R}^{r\times n}$ is sampled from the orthogonal ensemble if it is sampled in the following way: let $U \in \mathbb{R}^{n\times n}$ be a matrix uniformly sampled from the group of orthogonal matrices, then Q is the sub-matrix consisting of the first r rows of U. We have the following claim.

Lemma 37. *Given matrices $Q_1 \in \mathbb{R}^{r\times n}$ and $Q_2 \in \mathbb{R}^{r\times n}$ independently sampled from the orthogonal ensemble, we have $\sigma_{\frac{6r}{5}}(Q_1^T Q_1 + Q_2^T Q_2) > \frac{1}{10}$ with probability at least $1 - e^{-\frac{3rn}{100}}$.*

See Section 7.6.2 for the proof.

Taking Lemma 37 as given, introduce the shorthand $N = \lfloor \frac{rn}{50} \rfloor$. Suppose that we independently sample 2^N matrices of dimensions $r \times n$ from the orthogonal ensemble. Since there are $2^N(2^N-1)/2$ distinct pairs of matrices in our sample, the union bound in conjunction with Lemma 37 implies that

$$\mathbb{P}\Big[\forall i \neq j :\ \sigma_{\frac{6r}{5}}(Q_i^T Q_i + Q_j^T Q_j) > \frac{1}{10}\Big] \geq 1 - \frac{2^N(2^N-1)}{2} \exp\Big(-\frac{3rn}{100}\Big). \tag{7.13}$$

With our choice of N, it can be verified that the right-hand side of inequality (7.13) is positive. Thus, there exists a realization of orthogonal matrices $Q_1, \ldots, Q_{2^N} \in \mathbb{R}^{r\times n}$ such that for all $i \neq j$ we have $\sigma_{\frac{6r}{5}}(Q_i^T Q_i + Q_j^T Q_j) > \frac{1}{10}$.

We use this collection of orthogonal matrices in order to reduce the classical `Equality` problem to the rank estimation problem. In the `Equality` problem, Alice has a binary string $x_1 \in \{0,1\}^N$ and Bob has another binary string $x_2 \in \{0,1\}^N$, and their goal is to compute the function

$$\texttt{Equality}(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 = x_2; \\ 0 & \text{otherwise;} \end{cases}$$

It is well-known [110] that the deterministic communication complexity of the `Equality` problem is $D(\texttt{Equality}) = N + 1$.

In order to perform the reduction, given binary strings x_1 and x_2 of length N, we construct two matrices A_1 and A_2 such that their sum $A = A_1 + A_2$ has rank r if and only if $x_1 = x_2$. Since both x_1 and x_2 are of length N, each of them encodes an integer between 1 and 2^N. Defining $A_1 = \frac{Q_{x_1}^T Q_{x_1}}{2}$ and $A_2 = \frac{Q_{x_2}^T Q_{x_2}}{2}$, the triangle inequality guarantees that

$$\|A\|_2 \leq \|A_1\|_2 + \|A_2\|_2 = \frac{\|Q_{x_1}^T Q_{x_1}\|_2 + \|Q_{x_2}^T Q_{x_2}\|_2}{2} \leq 1,$$

showing that A satisfies the required operator norm bound. If $x_1 = x_2$, then $A = Q_{x_1}^T Q_{x_1}$, which is a matrix of rank r. If $x_1 \neq x_2$, then by our construction of Q_{x_1} and Q_{x_2}, we know that the matrix A has rank between $\frac{6r}{5}$ and $2r$ and moreover that $\sigma_{\frac{6r}{5}}(A) > \frac{1}{20}$. Thus, we can output $\texttt{Equality}(x_1, x_2) = 1$ if we detect the rank of matrix A to be r and output $\texttt{Equality}(x_1, x_2) = 0$ otherwise. Using this protocol, the `Equality` evaluation is always correct. As a consequence, the deterministic communication complexity of `RankTest` is lower bounded by that of `Equality`. Finally, noting that $D(\texttt{RankTest}) \geq D(\texttt{Equality}) = N + 1 > \frac{rn}{50}$ completes the proof.

7.4.1.2 Proof of upper bound

In order to prove the upper bound stated in part (b), we analyze the algorithm described following the theorem statement. If the matrix $A = \sum_{i=1}^{m} A_i$ has rank at most $2r$, then given the PSD nature of the component matrices, each matrix A_i also has rank at most $2r$. Consequently, we can find a factorization of the form $A_i = B_i B_i^T$ where $B_i \in \mathbb{R}^{n \times r}$. Let $\widetilde{B}_i$ be a quantization of the matrix B_i, allocating $\log_2\left(\frac{12mrn}{c_1 - c_2}\right)$ bits to each entry. Note that each machine must transmit at most $rn \log_2\left(\frac{12mrn}{c_1 - c_2}\right)$ bits in order to convey the quantized matrix $\widetilde{B}_i$.

Let us now analyze the approximation error. By our choice of quantization, we have

$$|||\widetilde{B}_i - B_i|||_{\mathrm{op}} \leq |||\widetilde{B}_i - B_i|||_{\mathrm{F}} \;\leq\; \sqrt{2rn}\frac{c_1 - c_2}{12mrn} \;=\; \frac{c_1 - c_2}{6m\sqrt{2rn}}.$$

Defining $\widetilde{A}_i = \widetilde{B}_i \widetilde{B}_i^T$ we have

$$\begin{aligned} |||\widetilde{A}_i - A_i|||_{\mathrm{F}} \leq |||\widetilde{B}_i - B_i|||_{\mathrm{F}} \sqrt{2rn}\Big(|||B_i|||_{\mathrm{op}} + |||\widetilde{B}_i|||_{\mathrm{op}}\Big) &\leq \frac{c_1 - c_2}{6m}\Big(2 + \frac{c_1 - c_2}{6m\sqrt{2rn}}\Big) \\ &\leq \frac{c_1 - c_2}{2m}, \end{aligned}$$

where the final inequality follows as long as $\frac{c_1 - c_2}{6m\sqrt{2rn}} \leq 1$.

Consequently, the sum $\widetilde{A} = \sum_{i=1}^{m} \widetilde{A}_i$ satisfies the bound

$$|||\widetilde{A} - A|||_F \leq \sum_{i=2}^{m} |||\widetilde{A}_i - A_i|||_F \leq \frac{(c_1 - c_2)}{2}.$$

Applying the Wielandt-Hoffman inequality [93] yields the upper bound

$$|\sigma_k(\widetilde{A}) - \sigma_k(A)| \leq |||\widetilde{A} - A|||_F \leq (c_1 - c_2)/2 \quad \text{for all } k \in [n]. \tag{7.14}$$

Recalling that $\widehat{r}(A)$ is the largest integer k such that $\sigma_k(\widetilde{A}) > (c_1 + c_2)/2$, inequality (7.14) implies that

$$(c_1 + c_2)/2 \geq \sigma_{\widehat{r}(A)+1}(\widetilde{A}) \geq \sigma_{\widehat{r}(A)+1}(A) - (c_1 - c_2)/2,$$

which implies $\sigma_{\widehat{r}(A)+1}(A) \leq c_1$. This upper bound verifies that $\widehat{r}(A) \geq \widehat{r}(A, c_1)$. On the other hand, inequality (7.14) also yields

$$(c_1 + c_2)/2 < \sigma_{\widehat{r}(A)}(\widetilde{A}) \leq \sigma_{\widehat{r}(A)}(A) + (c_1 - c_2)/2,$$

which implies $\sigma_{\widehat{r}(A)}(A) > c_2$ and $\widehat{r}(A) \leq \widehat{r}(A, c_2)$. Combining the above two inequalities yields the claim (7.3).

7.4.2 Proof of Theorem 13

We split the proof into two parts, corresponding to the upper bounds (7.9) and (7.10) respectively.

Proof of upper bound (7.9): Let λ_j be the j-th largest eigenvalue of A and let v_j be the associated eigenvector. Let function f be defined as $f(x) := q_2(q_1(x))$. Using basic linear algebra, we have

$$\|y\|_2^2 = \sum_{j=1}^{n} f^2(\lambda_j)(v_j^T g)^2. \tag{7.15}$$

Since g is an isotropic Gaussian random vector, the random variables $Z_j = (v_j^T g)^2$ are i.i.d., each with χ^2 distribution with one degree of freedom. To analyze the concentration behavior of Z variables, we recall the notion of a sub-exponential random variable.

A random variable Y is called *sub-exponential* with parameter (σ^2, β) if $\mathbb{E}[Y] = 0$ and the moment generating function is upper bounded as $\mathbb{E}[e^{tY}] \leq e^{t^2\sigma^2/2}$ for all $|t| \leq 1/\beta$. The following lemma, proved in Section 7.6.3, characterizes some basic properties of sub-exponential random variables.

Lemma 38. *(a) If $Z \sim \chi^2$, then both $Z - 1$ and $1 - Z$ are sub-exponential with parameter $(4, 4)$.*

(b) Given an independent sequence $\{Y_i\}_{i=1}^n$ in which Y_i is sub-exponential with parameter (σ_i^2, β_i), then for any choice of non-negative weights $\{\alpha\}_{i=1}^n$, the weighted sum $\sum_{i=1}^n \alpha_i Y_i$ is sub-exponential with parameters $(\sum_{i=1}^n \alpha_i^2\sigma_i^2, \max_{i\in[n]}\{\alpha_i\beta_i\})$.

(c) If Y is sub-exponential with parameter (σ^2, β), then

$$\mathbb{P}\big[Y \geq t\big] \leq e^{-\frac{t^2}{2\sigma^2}} \quad \text{for all } t \in [0, \tfrac{\sigma^2}{\beta}).$$

We consider $\|y\|_2^2$ as well as the associated lower bound $L = \sum_{j=1}^{\text{rank}(A,c_1)} f^2(\lambda_j)(v_j^T b)^2$. By parts (a) and (b) of Lemma 38, the variable $\|y\|_2^2 - \mathbb{E}[\|y\|_2^2]$ is sub-exponential with parameter $(4\sum_{i=1}^n f^2(\lambda_i), 4)$, and the variable $\mathbb{E}[L] - L$ is sub-exponential with parameter $(4\sum_{i=1}^{\text{rank}(A,c_1)} f^2(\lambda_i), 4)$. In order to apply part (c) of Lemma 38, we need upper bounds on the sum $\sum_{i=1}^n f^2(\lambda_i)$, as well as upper/lower bounds on the sum $\sum_{i=1}^{\text{rank}(A,c_1)} f^2(\lambda_i)$. For the first sum, we have

$$\begin{aligned}\sum_{j=1}^{n} f^2(\lambda_j) &= \sum_{j=1}^{\text{rank}(A,c_2)} f^2(\lambda_j) + \sum_{j=\text{rank}(A,c_2)+1}^{n} f^2(\lambda_j) \\ &\leq \text{rank}(A, c_2) + n2^{-p} \\ &\leq \text{rank}(A, c_2) + 1. \end{aligned} \tag{7.16}$$

where the last two inequalities use Lemma 35 and the fact that $p = \lceil \log_2(2n) \rceil$. For the second sum, using Lemma 35 implies that

$$\begin{aligned} \operatorname{rank}(A, c_1) &\geq \sum_{i=1}^{\operatorname{rank}(A,c_1)} f^2(\lambda_i) \geq \operatorname{rank}(A, c_1)(1 - 2^{-p})^2 \\ &\overset{(i)}{\geq} \operatorname{rank}(A, c_1)(1 - 1/(2n))^2 \overset{(ii)}{\geq} \operatorname{rank}(A, c_1) - 1. \end{aligned}$$

where inequality (i) follows since $2^{-p} \leq 1/(2n)$; inequality (ii) follows since $(1 - 1/(2n))^2 \geq 1 - 1/n$. Thus, we have

$$\mathbb{E}[\|y\|_2^2] \leq \operatorname{rank}(A, c_2) + 1 \quad \text{and} \quad \mathbb{E}[L] \geq \operatorname{rank}(A, c_1) - 1. \tag{7.17}$$

Putting together the pieces, we see that $\|y\|_2^2 - \mathbb{E}[\|y\|_2^2]$ is sub-exponential with parameter $(4(\operatorname{rank}(A, c_2) + 1), 4)$ and $\mathbb{E}[L] - L$ is sub-exponential with parameter $(4 \operatorname{rank}(A, c_1), 4)$.

Let $\widehat{r}$ be the average of T independent copies of $\|y\|_2$, and let $\widehat{r}_L$ be the average of T independent copies of L. By Lemma 38 (b), we know that $\widehat{r} - \mathbb{E}[\widehat{r}]$ is sub-exponential with parameter $(4(\operatorname{rank}(A, c_2) + 1)/T, 4/T)$, and $\mathbb{E}[\widehat{r}_L] - \widehat{r}_L$ is sub-exponential with parameter $(4 \operatorname{rank}(A, c_1)/T, 4/T)$. Plugging these parameters into Lemma 38 (c), for any $0 \leq \delta < 1$, we find that

$$\mathbb{P}\Big[\widehat{r} \leq \mathbb{E}[\widehat{r}] + \delta(\operatorname{rank}(A, c_2) + 1)\Big] \geq 1 - \exp\Big(-\frac{T\delta^2(\operatorname{rank}(A, c_2) + 1)}{32}\Big) \tag{7.18a}$$

$$\mathbb{P}\Big[\widehat{r}_L \geq \mathbb{E}[\widehat{r}_L] - \delta \operatorname{rank}(A, c_1)\Big] \geq 1 - \exp\Big(-\frac{T\delta^2 \operatorname{rank}(A, c_1)}{32}\Big). \tag{7.18b}$$

Combining inequalities (7.17), (7.18a), and (7.18b) yields

$$\mathbb{P}\Big[(1 - \delta)\operatorname{rank}(A, c_1) - 1 \leq \widehat{r}_L \leq \widehat{r} \leq (1 + \delta)(\operatorname{rank}(A, c_2) + 1)\Big] \leq 1 - 2e^{-\frac{T\delta^2 \operatorname{rank}(A,c_1)}{32}}, \tag{7.19}$$

which completes the proof of inequality (7.9).

Proof of upper bound (7.10): It remains to show to establish the upper bound (7.10) on the randomized communication complexity. The subtle issue is that in a discrete message model, we cannot calculate $f(A)g$ without rounding errors. Indeed, in order to make the rounding error of each individual message bounded by τ, each machine needs $\mathcal{O}(n \log(1/\tau))$ bits to encode a message. Consequently, the overall communication complexity scales as $\mathcal{O}(Tmdpn \log(1/\tau))$, where T is the number of iterations of Algorithm 6; m is the number of machines, the quantities d and p are the degrees of q_1 and q_2, and n is the matrix dimenson. With the choices given, we have $d = \mathcal{O}(1)$ and $p = \mathcal{O}(\log n)$. In order to make inequality (7.9) hold with probability at least $1 - \epsilon$, the upper bound (7.19) suggests choosing $T = \Theta(\log(1/\epsilon))$.

Finally, we need to upper bound the quantity $\mathcal{O}(\log(1/\tau))$. In order to do so, let us revisit Algorithm 6 to see how rounding errors affect the final output. For each integer $k = 1, \ldots, 2p+1$, let us denote by δ_k the error of evaluating $q_1^k(A)g$ using Algorithm 5. It is known [141, Chapter 2.4.2] that the rounding error of evaluating a Chebyshev expansion is bounded by $md\tau$. Thus, we have $\delta_{k+1} \leq \|q_1(A)\|_2 \delta_k + md\tau$. Since $\|q_1(A)\|_2 \leq 1.1$ by construction, we have the upper bound

$$\delta_k \leq 10(1.1^{k+1} - 1)md\tau. \tag{7.20}$$

For a polynomial of the form $q_2(x) = \sum_{i=0}^{2p+1} a_i x^i$, we have $y = \sum_{i=0}^{2p+1} a_i q_1^i(A)b$. As a consequence, there is a universal constant C such that error in evaluating y is bounded by

$$C \sum_{i=0}^{2p+1} \delta_i |a_i| \leq C' \, (1.1)^{2p+1} md\tau \sum_{i=0}^{2p+1} |a_i|.$$

By the definition of the polynomial q_2 and the binomial theorem, we have

$$\sum_{i=0}^{2p+1} |a_i| \leq \frac{2^p}{B(p+1, p+1)} = \frac{2^p (2p+1)!}{(p!)^2} \leq 2^{3p}.$$

Putting the pieces together, in order to make the overall error small, it suffices to choose τ of the order $(mdn)^{-1} 2^{-4p}$. Doing so ensures that $\log(1/\tau) = \mathcal{O}(p \log(mdn))$, which when combined with our earlier upper bounds on d, p and T, establishes the claim (7.10).

7.4.3 Proof of Theorem 14

In order to prove Theorem 14, it suffices to consider the two-player setting, since the first two machines can always simulate the two players Alice and Bob. Our proof proceeds via reduction from the 2-SUM problem [214], in which Alice and Bob have inputs $(U_1, \ldots, U_r)$ and $(V_1, \ldots, V_r)$, where each U_i and V_i are subsets of $\{1, \ldots, L\}$. It is promised that for every index $i \in \{1, \ldots, r\}$, the intersection of U_i and V_i contains at most one element. The goal is to compute the sum $\sum_{i=1}^r |U_i \cap V_i|$ up to an additive error of $\sqrt{r}/2$. Woodruff and Zhang [214] showed that randomized communication complexity of the 2-SUM problem is lower bounded as $\Omega(rL)$.

We note here that when $r \geq 16$, the same communication complexity lower bound holds if we allow the additive error to be $2\sqrt{r}$. To see this, suppose that Alice and Bob have inputs of length $r/16$ instead of r. By replicating their inputs 16 times, each of Alice and Bob can begin with an input of length r. Assume that by using some algorithm, they can compute the 2-SUM for the replicated input with additive error at most $2\sqrt{r}$. In this way, they have computed the 2-SUM for the original input with additive error at most $\sqrt{r}/8$. Note that $\sqrt{r/8} = \sqrt{r/16}/2$. The lower bound on the 2-SUM problem implies that the communication cost of the algorithm is $\Omega(rL/16)$, which is on the same order of $\Omega(rL)$.

To perform the reduction, let $L = \lfloor n/r - 1 \rfloor$. Since $r \leq n/2$, we have $L \geq 1$. Suppose that Alice and Bob are given subsets $(U_1, \ldots, U_r)$ and $(V_1, \ldots, V_r)$, which define an underlying instance of the 2-SUM problem. Based on these subsets, we construct two n-dimensional matrices A_1 and A_2 and the matrix sum $A := A_1 + A_2$; we then argue that any algorithm that can estimate the generalized matrix rank of A can solve the underlying 2-SUM problem.

The reduction consists of the following steps. First, Alice constructs a matrix X of dimensions $rL \times n$ as follows. For each $i \in \{1, \ldots, r\}$ and $j \in \{1, \ldots, L\}$, define $t(i,j) = (i-1)L + j$, and let $X_{t(i,j)}$ denote the associated row of X. Letting $e_{t(i,j)} \in \mathbb{R}^n$ denote the canonical basis vector (with a single one in entry $t(i,j)$), we define

$$X_{t(i,j)} = \begin{cases} e_{t(i,j)} & \text{if } j \in U_i \\ 0 & \text{otherwise.} \end{cases}$$

Second, Bob constructs a matrix Y of dimensions $rL \times n$ following the same rule as Alice, but using the subset $(V_1, \ldots, V_L)$ in place of $(U_1, \ldots, U_L)$. Now define the $n \times n$ matrices

$$A_1 := c_2\Big(X^T X + \sum_{i=1}^{r} e_{rL+i} e_{rL+i}^T\Big) \quad \text{and} \quad A_2 := c_2\Big(Y^T Y + \sum_{i=1}^{r} e_{rL+i} e_{rL+i}^T\Big).$$

With these definitions, it can be verified that $\|A\|_2 \leq 2c_2 \leq 1$, and moreover that all eigenvalues of A are either equal to $2c_2$ or at most c_2. Since $c_1 < 2c_2$, the quantities $\mathrm{rank}(A, c_1)$ and $\mathrm{rank}(A, c_2)$ are equal, and equal to the number of eigenvalues at $2c_2$. The second term in the definition of A_1 and A_2 ensures that there are at least r eigenvalues equal to $2c_2$. For all (i,j) pairs such that $j \in U_i \cap V_i$, the construction of X and Y implies that there are two corresponding rows in X and Y equal to each other, and both of them are canonical basis vectors. Consequently, they create a $2c_2$ eigenvalue in matrix A. Overall, we have $\mathrm{rank}(A, c_1) = \mathrm{rank}(A, c_2) = r + \sum_{i=1}^{r} |U_i \cap V_i|$, Since the problem set-up ensures that $|U_i \cap V_i| \leq 1$, we conclude $r \leq \mathrm{rank}(A, c_1) \leq 2r$.

Now suppose that there is a randomized algorithm estimating the rank of A such that

$$(1-\delta)\mathrm{rank}(A, c_1) \leq \widehat{r}(A) \leq (1+\delta)\mathrm{rank}(A, c_2).$$

Introducing the shorthand $s := \sum_{i=1}^{k} |U_i \cap V_i|$, when $\delta = 1/\sqrt{r}$, we have

$$r + s - (r+s)/\sqrt{r} \leq \widehat{r}(A) \leq r + s + (r+s)/\sqrt{r}.$$

Thus, the estimator $\widehat{r}(A) - r$ computes s up to additive error $(r+s)/\sqrt{r}$, which is upper bounded by $2\sqrt{r}$. It means that the rank estimation algorithm solves the 2-SUM problem. As a consequence, the randomized communication complexity of the rank estimation problem is lower bounded by $\Omega(rL) = \Omega(n)$.

7.5 Connections to other problems

In this chapter, we have studied the problem of estimating the generalized rank of matrices. Our main results are to show that in the deterministic setting, sending $\Theta(n^2)$ bits is both necessary and sufficient in order to obtain any constant relative error. In contrast, when randomized algorithms are allowed, this scaling is reduced to $\widetilde{\Theta}(n)$.

Our work suggests an important problem, one whose resolution has a number of interesting consequences. In the current chapter, we establish the $\widetilde{\Theta}(n)$ scaling of communication complexity for achieving a relative error $\delta = 1/\sqrt{r}$ where r is the matrix rank. Moreover, Algorithm 6 does not guarantee higher accuracies (e.g., $\delta = 1/r$), and as discussed in Section 7.3.3, it is unknown whether the $\Omega(n)$ lower bound is tight. The same question remains open even for the special case when all the matrix eigenvalues are either greater than constant c or equal to zero. In this special case, if we were to set $c_1 = c$ and $c_2 = 0$ in Algorithm 6, then it would compute *ordinary* matrix rank with relative error $\delta = 1/\sqrt{r}$. Although the problem is easier in the sense that all eigenvalue are promised to lie in the subset $\{0\} \cup (c, 1]$, we are currently not aware of any algorithm with $\widetilde{\mathcal{O}}(n)$ communication cost achieving better error rate. On the other hand, proving a tight lower bound for arbitrary δ remains an open problem.

The special case described above is of fundamental interest because it can be reduced to many classical problems in linear algebra and convex optimization, as we describe here. More precisely, if there is an algorithm solving any of these problems, then it can be used for computing the matrix rank with relative error $\delta = 0$. On the other hand, if we obtain a tight lower bound for computing the matrix rank, then it implies a lower bound for a larger family of problems. We list a subset of these problems giving a rough intuition for the reduction.

To understand the connection, we begin by observing that the problem of rank computation can be reduced to that of matrix rank testing, in which the goal is to determine whether a given matrix sum $A := A_1 + \cdots + A_m$ has rank at most $r - 1$, or rank at least r, assuming that all eigenvalues belong to $\{0\} \cup (c, +\infty)$. If there is an algorithm solving this problem for arbitrary integer $r \leq n$, then we can use it for computing the rank. The reduction is by performing a series of binary searches, each step deciding whether the rank is above or below a threshold. In turn, the rank test problem can be further reduced to the following problems:

Singularity testing: The goal of singularity testing is to determine if the sum of matrices $B := B_1 + \cdots + B_m$ is singular, where machine i stores the PSD matrix B_i. Algorithms for singularity testing can be used for rank testing. The reduction is by using a public random coin to generate a shared random projection matrix $Q \in \mathbb{R}^{r \times n}$ on each machine and then setting $B_i := QA_iQ^T$. The inclusion of the public coin only increases the communication complexity by a moderate amount [118], in particular by an additive term $\mathcal{O}(\log(n))$. On the other hand, with high probability the matrix A has rank at most $r - 1$ if and only if the matrix B is singular.

Solving linear equations: Now suppose that machine i stores a strictly positive definite matrix C_i and a vector y. The goal is to compute the vector x satisfying $Cx = y$ for $C := C_1 + \cdots + C_m$. Algorithms for solving linear equations can be used for the singularity test. In particular, let $C_i := B_i + \lambda I$ and take y to be a random Gaussian vector. If the matrix B is singular, then the norm $\|x\|_2 \to \infty$ as $\lambda \to 0$. Otherwise, it remains finite as $\lambda \to 0$. Thus, we can test for $\lambda = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ to decide if the matrix is singular. Note that the solution need not be exact, since we only test if the ℓ_2-norm remains finite.

Convex optimization: Suppose that each machine has a strictly convex function f_i, and the overall goal is to compute a vector x that minimizes the function $x \mapsto f(x) := f_1(x) + \cdots + f_m(x)$. The algorithms solving this problem can be used for solving linear equations. In particular, for a strictly positive definite matrix C_i, the function $f_i(x) := \frac{1}{2}x^T C_i x - \frac{1}{m}y^T x$ is strictly convex, and with these chocies, the function f is uniquely minimized at $C^{-1}y$. (Since the linear equation solver doesn't need to be exact, the solution here is also allowed to be approximate.)

This reduction chain suggests the importance of studying matrix rank estimation, especially for characterizing lower bounds on communication complexity. We hope the results in this paper are a meaningful first step in exploring this problem area.

7.6 Proof of technical results

7.6.1 Proof of Lemma 35

The function q_2 is monotonically increasing on $[0, 1]$. In addition, we have $q_2(0) = 0$ and $q_2(1) = 1$, and hence $q_2(z) \in [0, 1]$ for all $z \in [0, 1]$. Let us refine this analysis on two end intervals: namely, $z \in [-0.1, 0.1]$ and $z \in [0.9, 1.1]$. For $z \in [-0.1, 0.1]$, it is easy to observe from the definition of q_2 that $q_2(z) \geq 0$. Moreover, for $z \in [-0.1, 0.1]$ we have $|z(1 - z)| \leq 0.11$. Thus,

$$q_2(z) = \frac{\int_0^z t^p(1-t)^p dt}{\int_0^1 t^p(1-t)^p dt} \leq \frac{\int_0^z t^p(1-t)^p dt}{\int_{0.4}^{0.6} t^p(1-t)^p dt} \leq \frac{0.1 \times (0.11)^p}{0.2 \times (0.24)^p} < 2^{-p}.$$

The function q_2 is symmetric in the sense that $q_2(z) + q_2(1 - z) = 1$. Thus, for $z \in [0.9, 1.1]$, we have $q_2(z) = 1 - q_2(1 - z) \in [1 - 2^{-p}, 1]$. In summary, we have proved that

$$0 \leq q_2(z) \leq 1 \text{ for } z \in [-0.1, 1.1], \tag{7.21a}$$
$$q_2(z) \leq 2^{-p} \text{ for } z \in [-0.1, 0.1], \tag{7.21b}$$
$$q_2(z) \geq 1 - 2^{-p} \text{ for } z \in [0.9, 1.1]. \tag{7.21c}$$

By the standard uniform Chebyshev approximation, we are guaranteed that $q_1(x) \in [-0.1, 1.1]$ for all $x \in [0, 1]$. Thus, inequality (7.21a) implies that $q_2(q_1(x)) \in [0, 1]$ for all $x \in [0, 1]$. If $x \in [0, c_2]$, then $q_1(x) \in [-0.1, 0.1]$, and thus inequality (7.21b) implies

$q_2(q_1(x)) \le 2^{-p}$. If $x \in [c_1, 1]$, then $q_1(x) \in [0.9, 1.1]$, and thus inequality (7.21c) implies $q_2(q_1(x)) \ge 1 - 2^{-p}$. Combining the last two inequalities yields that

$$|q_2(q_1(x)) - H_{c_1,c_2}(x)| \le 2^{-p} \qquad \text{for all } x \in [0, c_2] \cup [c_1, 1].$$

7.6.2 Proof of Lemma 37

Let q_t be the t-th row of Q_2, and let $Q^{(t)} \in \mathbb{R}^{r+t}$ be the matrix whose first r rows are the rows of Q_1, and its remaining t rows are $q_1, \ldots, q_t$. Let $q_{t+1}^{\|}$ be the projection of q_{t+1} to the subspace generated by the rows of $Q^{(t)}$ and let $q_{t+1}^{\perp} := q_{t+1} - q_{t+1}^{\|}$. We have

$$\begin{aligned}(Q^{(t+1)})^T Q^{(t+1)} &= (Q^{(t)})^T Q^{(t)} + q_t^T q_t = (Q^{(t)})^T Q^{(t)} + (q_{t+1}^{\|})^T q_{t+1}^{\|} + (q_{t+1}^{\perp})^T q_{t+1}^{\perp} \\ &\succeq (Q^{(t)})^T Q^{(t)} + (q_{t+1}^{\perp})^T q_{t+1}^{\perp}.\end{aligned}$$

This inequality yields the lower bound

$$Q_1^T Q_1 + Q_2^T Q_2 \succeq Q_1^T Q_1 + \sum_{t=1}^{r} (q_t^{\perp})^T q_t^{\perp}, \tag{7.22}$$

where $\succeq$ denotes ordering in the positive semidefinite cone. Note that the rows of Q_1 and $\{q_t^{\perp}\}_{t=1}^{r}$ are mutually orthogonal. To prove that the $\frac{6k}{5}$-th largest eigenvalue of $Q_1^T Q_1 + Q_2^T Q_2$ is greater than $1/10$, it suffices to prove that there are at least $r/5$ vectors in $\{q_t^{\perp}\}_{t=1}^{r}$ which satisfy $\|q_t^{\perp}\|_2^2 > 1/10$.

Let $\mathcal{S}_1$ be the linear subspace generated by $q_1, \ldots, q_{t-1}$ and let $\mathcal{S}_1^{\perp}$ be its orthogonal subspace. The vector q_t is uniformly sampled from a unit sphere in $\mathcal{S}_1^{\perp}$. Let $\mathcal{S}_2$ be the linear subspace generated by the rows of $Q^{(t-1)}$. Since $Q^{(t-1)}$ has $r + t - 1$ rows, the subspace has at most $r + t - 1$ dimensions. Without loss of generality, we assume that $\mathcal{S}_2$ has $r + t - 1$ dimensions (otherwise, we expand it to reach the desired dimensionality). We let $\mathcal{S}_2^{\perp}$ be the orthogonal subspace of $\mathcal{S}_2$. By definition, $q_t^{\perp}$ is the projection of q_t to $\mathcal{S}_2^{\perp}$ (or a linear space that contains $\mathcal{S}_2^{\perp}$ if the subspace $\mathcal{S}_2$ has been expanded to reach the $r+t-1$ dimensionality). Let q_t' be the projection of q_t to $\mathcal{S}_1^{\perp} \cap \mathcal{S}_2^{\perp}$, then we have

$$\|q_t^{\perp}\|_2^2 \ge \|q_t'\|_2^2. \tag{7.23}$$

Note that $\mathcal{S}_1^{\perp}$ is of dimension $n - t + 1$ and $\mathcal{S}_2^{\perp}$ is of dimension $n - r - t + 1$. Thus, the dimension of $\mathcal{S}_1^{\perp} \cap \mathcal{S}_2^{\perp}$ is at least $n - r - 2t + 2$. Constructing q_t' is equivalent to projecting a random vector in the $(n - t + 1)$-dimension sphere to a $(n - r - 2t + 2)$-dimension subspace. It is a standard result (e.g. [54, Lemma 2.2]) that

$$\mathbb{P}\Big[\|q_t'\|_2^2 \le \beta \cdot \frac{n - r - 2t + 2}{n - t + 1}\Big] \le \exp\Big(\frac{n - r - 2t + 2}{2}(1 - \beta + \log(\beta))\Big) \quad \text{for any } \beta < 1.$$

Setting $\beta = 0.3$ and using the fact that $t \le r \le n/4$, we find that

$$\mathbb{P}\Big[\|q_t'\|_2^2 \le 1/10\Big] \le \exp\Big(\frac{n - n/4 - n/2 + 2}{2}(1 - 0.3 + \log(0.3))\Big) \le \exp(-n/16). \tag{7.24}$$

Defining the event $\mathcal{E}_t := \{\|q'_t\|_2^2 \leq 1/10\}$, note that inequality (7.24) yields $\mathbb{P}[\mathcal{E}_t] \leq \exp(-n/16)$. Since q'_t is the projection of a random unit vector to a subspace of constant dimension, the events $\{\mathcal{E}_j\}_{j=1}^t$ are mutually independent, and hence

$$\mathbb{P}\Big[\text{at least } \tfrac{4k}{5} \text{ events in } \{\mathcal{E}_j\}_{j=1}^t \text{ occur }\Big] \leq \binom{r}{4r/5}(\exp(-n/16))^{\frac{4r}{5}} \leq \exp\Big(\frac{r\log(r)}{5} - \frac{rn}{20}\Big)$$
$$\leq \exp\Big(-\frac{3rn}{100}\Big),$$

where the last inequality follows since any integer r satisfies $\log(r) \leq \frac{2r}{5} \leq \frac{n}{10}$. Thus, with probability at least $1 - \exp(-\frac{3rn}{100})$, there are at least $r/5$ rows satisfying $\|q'_t\|_2^2 > 1/10$. Combining this result with inequality (7.22) and (7.23) completes the proof.

7.6.3 Proof of Lemma 38

The claimed facts about sub-exponential random variables are standard [35], but we provide proofs here for completeness.

Part (a): Let Z be χ^2 variable with one degree of freedom. Its moment generating function takes the form

$$\mathbb{E}[\exp(t(Z-1))] = (1-2t)^{-1/2}e^{-t} \quad \text{for } t < 1/2.$$

Some elementary algebra shows that $(1-2t)^{-1/2}e^{-t} \leq e^{2t^2}$ for any $t \in [-1/4, 1/4]$. Thus, we have $\mathbb{E}[\exp(t(Z-1))] \leq e^{2t^2}$ for $|t| \leq 1/4$, verifying the recentered variable $X = Z - 1$ is sub-exponential with parameter $(4, 4)$. Also by the moment generating function of Z, we have

$$\mathbb{E}[\exp(t(1-Z))] = (1+2t)^{-1/2}e^{t} \quad \text{for } t > -1/2.$$

Replacing t by $-t$ and comparing with the previous conclusion reveals that $1 - Z$ is sub-exponential with parameter $(4, 4)$.

Part (b): Suppose that $Z_1, \ldots, Z_n$ are independent and Z_i is sub-exponential with parameter (σ_i^2, β_i). By the definition of sub-exponential random variable, we have

$$\mathbb{E}\Big[\exp\Big(t\sum_{i=1}^n \alpha_i Z_i\Big)\Big] = \prod_{i=1}^n \mathbb{E}[\exp(t\alpha_i Z_i)] \leq \prod_{i=1}^n \exp((t\alpha_i)^2\sigma^2/2) = \exp\Big(\frac{t^2\sum_{i=1}^n \alpha_i^2\sigma_i^2}{2}\Big)$$

for all $t \leq \max_{i\in[n]}\{1/(\alpha_i\beta_i)\}$. This bound establishes that $\sum_{i=1}^n \alpha_i Z_i$ is sub-exponential with parameter $(\sum_{i=1}^n \alpha_i^2\sigma_i^2, \max_{i\in[n]}\{\alpha_i\beta_i\})$, as claimed.

Part (c): Notice that $\mathbb{P}[Z \geq t] = \mathbb{P}[e^{\lambda Z} \geq e^{\lambda t}]$ with any $\lambda > 0$. Applying Markov's inequality yields

$$\mathbb{P}[Z \geq t] \leq \frac{\mathbb{E}[\exp(\lambda Z)]}{e^{\lambda t}} \leq \exp\Big(- \lambda t + \frac{\lambda^2 \sigma^2}{2} \Big) \quad \text{for } \lambda \leq 1/\beta,$$

where the last step follows since Z is sub-exponential with parameter (σ^2, β). Notice that the minimum of $-\lambda t + \frac{\lambda^2 \sigma^2}{2}$ occurs when $\lambda^* = t/\sigma^2$. Since $t < \sigma^2/\beta$, we have $\lambda^* < 1/\beta$, verifying the validness of λ^*. Plugging λ^* in the previous inequality completes the proof.

Part IV

Distributed systems

Chapter 8

Programming interface for parallelizing stochastic algorithms

In this chapter, we present a general framework for parallelizing stochastic algorithms for machine learning applications. Stochastic algorithms process a large-scale dataset by sequentially processing random subsamples. This processing scheme makes the per-iteration cost of the algorithm much cheaper than that of batch processing algorithms while still yielding effective descent. Indeed, for convex optimization, the efficiency of stochastic gradient descent (SGD) and its variants has been established both in theory and in practice [226, 31, 216, 62, 181, 99]. For non-convex optimization, stochastic methods achieve state-of-the-art performance on a broad class of problems, including matrix factorization [107], neural networks [109] and representation learning [209]. Stochastic algorithms are also widely used in the Bayesian setting for finding approximations to posterior distributions; examples include Markov chain Monte Carlo, expectation propagation [147] and stochastic variational inference [92].

Although classical stochastic approximation procedures are sequential, it is clear that they also present opportunities for parallel and distributed implementations that may yield significant additional speedups. One active line of research studies asynchronous parallel updating schemes in the setting of a lock-free shared memory [172, 64, 129, 236, 89]. When the time delay of concurrent updates are bounded, it is known that such updates preserve statistical correctness [4, 129]. Such asynchronous algorithms yield significant speedups on multi-core machines. On distributed systems connected by commodity networks, however, the communication requirements of such algorithms can be overly expensive. If messages are frequently exchanged across the network, the communication cost will easily dominate the computation cost.

There has also been a flurry of research studying the implementation of stochastic algorithms in the fully distributed setting [238, 231, 158, 75, 128]. Although promising results have been reported, the implementations proposed to date have their limitations—they have been designed for specific algorithms, or they require careful partitioning of the data to avoid inconsistency.

In this paper, we propose a general framework for parallelizing stochastic algorithms on multi-node distributed systems. Our framework is called Splash (**S**ystem for **P**arallelizing **L**earning **A**lgorithms with **S**tochastic Met**h**ods). Splash consists of a programming interface and an execution engine. Using the programming interface, the user develops sequential stochastic algorithms without thinking about issues of distributed computing. The algorithm is then automatically parallelized by the execution engine. The parallelization is communication efficient, meaning that its separate threads don't communicate with each other until all of them have processed a large bulk of data. Thus, the inter-node communication need not be a performance bottleneck.

Programming Interface The programming interface is designed around a key paradigm: implementing incremental updates that processes weighted data. Unlike existing distributed machine learning systems [57, 217, 121, 149] which requires the user to explicitly specify a distributed algorithm, Splash asks the user to implement a processing function that takes an individual data element as input to incrementally update the corresponding variables. When this function is iteratively called on a sequence of samples, it defines a sequential stochastic algorithm. It can also be called in a distributed manner for constructing parallel algorithms, which is the job of the execution engine. This programming paradigm allows one algorithmic module working on different computing environments, no matter if it is a single-core processor or a large-scale cluster. As a consequence, the challenge of parallelizing these algorithms has been transferred from the developer side to the system side.

To ensure parallelizability, the user is asked to implement a slightly stronger version of the base sequential algorithm: it needs to be capable of processing *weighted samples*. An m-weighted sample tells the processing function that the sample appears m times consecutively in the sequence. Many stochastic algorithms can be generalized to processing weighted samples without sacrificing computational efficiency. We will demonstrate SGD and collapsed Gibbs sampling as two concrete examples. Since the processing of weighted samples can be carried out within a sequential paradigm, this requirement does not force the user to think about a distributed implementation.

Execution Engine In order to parallelize the algorithm, Splash converts a distributed processing task into a sequential processing task using distributed versions of *averaging* and *reweighting*. During the execution of the algorithm, we let every thread sequentially process its local data. The local updates are iteratively averaged to construct the global update. Critically, however, although averaging reduces the variance of the local updates, it doesn't reduce their bias. In contrast to the sequential case in which a thread processes a full sequence of random samples, in the distributed setting every individual thread touches only a small subset of samples, resulting in a significant bias relative to the full update. Our reweighting scheme addresses this problem by feeding the algorithm with weighted samples, ensuring that the total weight processed by each thread is equal to the number of samples in the full sequence. This helps individual threads to generate nearly-unbiased estimates of the

full update. Using this approach, Splash automatically detects the best degree of parallelism for the algorithm.

Theoretically, we prove that Splash achieves the optimal rate of convergence for parallelizing SGD, assuming that the objective function is smooth and strongly convex. We conduct extensive experiments on a variety of stochastic algorithms, including algorithms for logistic regression, collaborative filtering and topic modeling. The experiments verify that Splash can yield orders-of-magnitude speedups over single-thread stochastic algorithms and over state-of-the-art batch algorithms.

Besides its performance, Splash is a contribution on the distributed computing systems front, providing a flexible interface for the implementation of stochastic algorithms. We build Splash on top of Apache Spark [224], a popular distributed data-processing framework for batch algorithms. Splash takes the standard Resilient Distributed Dataset (RDD) of Spark as input and generates an RDD as output. The data structure also supports default RDD operators such as Map and Reduce, ensuring convenient interaction with Spark. Because of this integration, Splash works seamlessly with other data analytics tools in the Spark ecosystem, enabling a single system to address the entire analytics pipeline.

8.1 Shared and local variables

In this paper, we focus on the stochastic algorithms which take the following general form. At step t, the algorithm receives a data element z_t and a vector of *shared variables* v_t. Based on these values the algorithm performs an incremental update $\Delta(z_t, v_t)$ on the shared variable:

$$v_{t+1} \leftarrow v_t + \Delta(z_t, v_t) \tag{8.1}$$

For example, stochastic gradient descent (SGD) fits this general framework. Letting x denote a random data element x and letting w denote a parameter vector, SGD performs the update:

$$t \leftarrow t + 1 \quad \text{and} \quad w \leftarrow w - \eta_t \nabla \ell(w; x) \tag{8.2}$$

where $\ell(\cdot; x)$ is the loss function associated with the element and η_t is the stepsize at time t. In this case both w and t are shared variables.

There are several stochastic algorithms using *local variables* in their computation. Every local variable is associated with a specific data element. For example, the collapsed Gibbs sampling algorithm for LDA [78] maintains a topic assignment for each word in the corpus. Suppose that a topic $k \in \{1, \ldots, K\}$ has been sampled for a word w, which is in document d. The collapsed Gibbs sampling algorithm updates the word-topic counter n_{wk} and the document-topic counter n_{dk} by

$$n_{wk} \leftarrow n_{wk} + 1 \quad \text{and} \quad n_{dk} \leftarrow n_{dk} + 1. \tag{8.3}$$

The algorithm iteratively resample topics for every word until the model parameters converge. When a new topic is sampled for the word w, the following operation removes the old topic before drawing the new one:

$$n_{wk} \leftarrow n_{wk} - 1 \quad \text{and} \quad n_{dk} \leftarrow n_{dk} - 1. \tag{8.4}$$

Update (8.3) and update (8.4) are executed at different stages of the algorithm but they share the same topic k. Thus, there should be a local variable associated with the word w storing the topic. Splash supports creating and updating local variables during the algorithm execution.

The usage of local variables can sometimes be tricky. Since the system carries out automatic reweighting and rescaling (refer to Section 8.3.2), any improper usage of the local variable may cause inconsistent scaling issues. The system thus provides a more robust interface called "delayed operator" which substitutes the functionality of local variables in many situations. In particular, the user can declare an operation such as (8.4) as a delayed operation and suspend its execution to the next time when the same element is processed. The scaling consistency of the delay operation is guaranteed by the system.

Shared variables and local variables are stored separately. In particular, shared variables are replicated on every data partition. Their values are synchronized. The local variables, in contrast, are stored with the associated data elements and will never be synchronized. This storage scheme optimizes the communication efficiency and allows for convenient element-wise operations.

8.2 Programming with Splash

Splash allows the user to write self-contained Scala applications using its programming interface. The goal of the programming interface is to make distributed computing transparent to the user. Splash extends Apache Spark to provide an abstraction called a *Parametrized RDD* for storing and maintaining the distributed dataset. The Parametrized RDD is based on the Resilient Distributed Dataset (RDD) [224] used by Apache Spark. It can be created from a standard RDD object:

```
val paramRdd = new ParametrizedRDD(rdd).
```

We provide a rich collection of interfaces to convert the components of Parametrized RDD to standard RDDs, facilitating the interaction between Splash and Spark. To run algorithms on the Parametrized RDD, the user creates a function called `process` which implements the stochastic algorithm, then calls the method

```
paramRdd.run(process)
```

to start running the algorithm. In the default setting, the execution engine takes a full pass over the dataset by calling `run()` once. This is called one *iteration* of the algorithm

execution. The inter-node communication occurs only at the end of the iteration. The user may call `run()` multiple times to take multiple passes over the dataset.

The `process` function is implemented using the following format:

```
def process(elem : Any, weight : Int, sharedVar : VarSet, localVar : VarSet){...}
```

It takes four arguments as input: a single element `elem`, the weight of the element, the shared variable `sharedVar` and the local variable `localVar` associated with the element. The goal is to update `sharedVar` and `localVar` according to the input.

Splash provides multiple ways to manipulate these variables. Both local and shared variables are manipulated as key-value pairs. The key must be a string; the value can be either a real number or an array of real numbers. Inside the `process` implementation, the value of local or shared variables can be accessed by `localVar.get(key)` or `sharedVar.get(key)`. The local variable can be updated by setting a new value for it:

```
localVar.set(key, value)
```

The shared variable is updated by *operators*. For example, using the *add* operator, the expression

```
sharedVar.add(key, delta)
```

adds a scalar `delta` to the variable. The SGD updates (8.2) can be implemented via several add operators. Other operators supported by the programming interface, including *delayed add* and *multiply*, are introduced in Section 8.3.2. Similar to the standard RDD, the user can perform `map` and `reduce` operations directly on the Parametrized RDD. For example, after the algorithm terminates, the expression

```
val loss = paramRdd.map(evalLoss).sum()
```

evaluates the element-wise losses and aggregates them across the dataset.

8.3 Strategy for parallelization

In this section, we first discuss two naive strategies for parallelizing a stochastic algorithm and their respective limitations. These limitations motivate the strategy that Splash employs.

8.3.1 Two naive strategies

We denote by $\Delta(\mathcal{S})$ the incremental update on variable v after processing the set of samples $\mathcal{S}$. Suppose that there are m threads and each thread processes a subset S_i of $\mathcal{S}$.

If the i-th thread increments the shared variable by $\Delta(S_i)$, then the *accumulation scheme* constructs a global update by accumulating local updates:

$$v_{\text{new}} = v_{\text{old}} + \sum_{i=1}^{m} \Delta(S_i). \tag{8.5}$$

The scheme (8.5) provides a good approximation to the full update if the batch size $|D_i|$ is sufficiently small [4]. However, frequent communication is necessary to ensure a small batch size. For distributed systems connected by commodity networks, frequent communication is prohibitively expensive, even if the communication is asynchronous.

Applying scheme (8.5) on a large batch may easily lead to divergence. Taking SGD as an example: if all threads starts from the same vector w_{old}, then after processing a large batch, the new vector on each thread will be close to the optimal solution w^*. If the variable is updated by formula (8.5), then we have

$$w_{\text{new}} - w^* = w_{\text{old}} - w^* + \sum_{i=1}^{m} \Delta(S_i) \approx w_{\text{old}} - w^* + \sum_{i=1}^{m} (w^* - w_{\text{old}}) = (m-1)(w^* - w_{\text{old}}).$$

Clearly SGD will diverge if $m \geq 3$.

One way to avoid divergence is to multiply the incremental change by a small coefficient. When the coefficient is $1/m$, the variable is updated by

$$v_{\text{new}} = v_{\text{old}} + \frac{1}{m} \sum_{i=1}^{m} \Delta(S_i). \tag{8.6}$$

This *averaging scheme* usually avoids divergence. However, since the local updates are computed on $1/m^{th}$ of $\mathcal{S}$, they make little progress comparing to the full sequential update. Thus the algorithm converges substantially slower than its sequential counterpart after processing the same amount of data. See Section 8.3.4 for an empirical evidence of this claim.

8.3.2 Our strategy

We now turn to describe the Splash strategy for combining parallel updates. First we introduce the operators that Splash supports for manipulating shared variables. Then we illustrate how conflicting updates are combined by the reweighting scheme.

Operators The programming interface allows the user to manipulate shared variables inside their algorithm implementation via *operators*. An operator is a function that maps a real number to another real number. Splash supports three types of operators: *add*, *delayed add* and *multiply*. The system employs different strategies for parallelizing different types of operators.

The *add* operator is the the most commonly used operator. When the operation is performed on variable v, the variable is updated by $v \leftarrow v + \delta$ where δ is a user-specified scalar. The SGD update (8.2) can be implemented using this operator.

The *delayed add* operator performs the same mapping $v \leftarrow v + \delta$; however, the operation will not be executed until the next time that the same element is processed by the system. Delayed operations are useful in implementing sampling-based stochastic algorithms. In particular, before the new value is sampled, the old value should be removed. This "reverse" operation can be declared as a delayed operator when the old value was sampled, and executed before the new value is sampled. See Section 8.3.3 for a concrete example.

The *multiply* operator scales the variable by $v \leftarrow \gamma \cdot v$ where γ is a user-specified scalar. The multiply operator is especially efficient for scaling high-dimensional arrays. The array multiplication costs $\mathcal{O}(1)$ computation time, independent of the dimension of the array. The fast performance is achieved by a "lazy update" scheme. For every array u, there is a variable V maintaining the product of all multipliers applied to the array. The multiply operator updates $V \leftarrow \gamma \cdot V$ with $\mathcal{O}(1)$ time. For the i-th element u_i, a variable V_i maintains the product of all multipliers applied to the element. When the element is accessed, the system updates u_i and V_i by

$$u_i \leftarrow \frac{V}{V_i} \cdot u_i \quad \text{and} \quad V_i \leftarrow V.$$

In other words, we delay the multiplication on individual element until it is used by the program. As a consequence, those infrequently used elements won't be a bottleneck on the algorithm's performance. See Section 8.3.3 for a concrete example.

Reweighting Assume that there are m thread running in parallel. Note that all Splash operators are linear transformations. When these operators are applied sequentially, they merge into a single linear transformation. Let S_i be the sequence of samples processed by thread i, which is a fraction $1/m$ of the full sequence $\mathcal{S}$. For an arbitrary shared variable v, we can write thread i's transformation of this variable in the following form:

$$v \leftarrow \Gamma(S_i) \cdot v + \Delta(S_i) + T(S_i), \tag{8.7}$$

Here, both $\Gamma(S_i)$, $\Delta(S_i)$ and $T(S_i)$ are thread-level operators constructed by the execution engine: $\Gamma(S_i)$ is the aggregated multiply operator, $\Delta(S_i)$ is the term resulting from the add operators, and $T(S_i)$ is the term resulting from the delayed add operators executed in the current iteration. A detailed construction of $\Gamma(S_i)$, $\Delta(S_i)$ and $T(S_i)$ is given in Section 8.6.1.

As discussed in Section 8.3.1, directly combining these transformations leads to divergence or slow convergence (or both). The reweighting scheme addresses this dilemma by assigning weights to the samples. Since the update (8.7) is constructed on a fraction $1/m$ of the full sequence $\mathcal{S}$, we reweight every element by m in the local sequence. After reweighting, the data distribution of S_i will approximate the data distribution of $\mathcal{S}$. If the update (8.7) is a (randomized) function of the data distribution of S_i, then it will approximate the full sequential update after the reweighting, thus generating a nearly unbiased update.

More concretely, the algorithm manipulates the variable by taking sample weights into account. An m-weighted sample tells the algorithm that it appears m times consecutively in the sequence. We rename the transformations in (8.7) by $\Gamma(mS_i)$, $\Delta(mS_i)$ and $T(mS_i)$, emphasizing that they are constructed by processing m-weighted samples. Then we redefine the transformation of thread i by

$$v \leftarrow \Gamma(mS_i) \cdot v + \Delta(mS_i) + T(mS_i) \tag{8.8}$$

and define the global update by

$$v_{\text{new}} = \frac{1}{m} \sum_{i=1}^{m} \Big(\Gamma(mG_i) \cdot v_{\text{old}} + \Delta(mS_i) \Big) + \sum_{i=1}^{m} T(mS_i). \tag{8.9}$$

Equation (8.9) combines the transformations of all threads. The terms $\Gamma(mS_i)$ and $\Delta(mS_i)$ are scaled by a factor $1/m$ because they were constructed on m times the amount of data. The term $T(mS_i)$ is not scaled, because the delayed operators were declared in earlier iterations, independent of the reweighting. Finally, the scaling factor $1/m$ should be multiplied to all delayed operators declared in the current iteration, because these delayed operators were also constructed on m times the amount of data.

Determining the degree of parallelism To determine the thread number m, the execution engine partitions the available cores into different-sized groups. Suppose that group i contains m_i cores. These cores will execute the algorithm tentatively on m_i parallel threads. The best thread number is then determined by cross-validation and is dynamically updated. The cross-validation requires the user to implement a loss function, which takes the variable set and an individual data element as input to return the loss value. See Section 8.6.2 for a detailed description. To find the best degree of parallelism, the base algorithm needs to be robust in terms of processing a wide range of sample weights.

8.3.3 Generalizing stochastic algorithms

Many stochastic algorithms can be generalized to processing weighted samples without sacrificing computational efficiency. The most straightforward generalization is to repeat the single-element update m times. For example, one can generalize the SGD updates (8.2) by

$$t \leftarrow t + m \quad \text{and} \quad w \leftarrow w - \eta_{t,m} \nabla \ell(w; x) \tag{8.10}$$

where $\eta_{t,m} := \sum_{i=t-m+1}^{t} \eta_i$ is the sum of all stepsizes in the time interval $[t - m + 1, t]$, and η_i is the stepsize for the unit-weight sequential SGD. If m is large, computing $\eta_{t,m}$ might be expensive. We may approximate it by

$$\eta_{t,m} \approx \int_{t-m+1}^{t+1} \eta_z dz$$

if the right-hand side has a closed-form solution, or simply approximate it by $\eta_{t,m} \approx m\eta_t$.

In many applications, the loss function $\ell(w; x)$ can be decomposed as $\ell(w; x) := f(w; x) + \frac{\lambda}{2}\|w\|_2^2$ where the second term is the ℓ_2-norm regularization. Thus, we have $\nabla\ell(w; x) = \nabla f(w; x) + \lambda w$. If the feature vector x is sparse, then $\nabla f(w; x)$ is usually sparse as well. In this case, we have a more efficient implementation of (8.10):

$$\begin{aligned} &t \leftarrow t + m && \text{(via add operator)},\\ &w \leftarrow (1 - \eta_{t,m}\lambda) \cdot w && \text{(via multiply operator)},\\ &w \leftarrow w - \eta_{t,m}\nabla f(w; x) && \text{(via add operator)}. \end{aligned}$$

Note that the multiply operator has complexity $\mathcal{O}(1)$. Thus, the overall complexity is proportional to the number of non-zero components of $\nabla f(w; x)$. If $\nabla f(w; x)$ is a sparse vector, then this update will be more efficient than (8.10). It demonstrates the benefit of combining different types of operators.

Note that equation (8.10) scales the stepsize with respect to m, which might be unsafe m is very large. Karampatziakis and Langford [102] propose a robust approach to dealing with large importance weights in SGD. The programming interface allows the user to implement the approach by Karampatziakis and Langford [102].

We take the collpased Gibbs sampling algorithm for LDA as a second example. The algorithm iteratively draw a word w from document d, and sample the topic of w by

$$P(\text{topic} = k|d, w) \propto \frac{(n_{dk} + \alpha)(n_{wk} + \beta)}{n_k + \beta W}. \tag{8.11}$$

Here, W is the size of the vocabulary; n_{dk} is the number of words in document d that has been assigned topic k; n_{wk} is the total number of times that word w is assigned to topic k and $n_k := \sum_w n_{wk}$. The constants α and β are hyper-parameters of the LDA model. When a topic k is sampled for the word, the algorithm updates n_{wk} and n_{dk} by (8.3). When a new topic will be sampled for the same word, the algorithm removes the old topic k by (8.4). If the current word has weight m, then we can implement the algorithm by

$$n_{wk} \leftarrow n_{wk} + m \quad \text{and} \quad n_{dk} \leftarrow n_{dk} + m \qquad \text{(via add operator)}, \tag{8.12}$$

$$n_{wk} \leftarrow n_{wk} - m \quad \text{and} \quad n_{dk} \leftarrow n_{dk} - m \qquad \text{(via delayed add operator)}. \tag{8.13}$$

As a consequence, the update (8.12) will be executed instantly. The update (8.13) will be executed at the next time when the same word is processed.

8.3.4 A toy example

We present a toy example illustrating the strategy described in Section 8.3.2. Consider the following convex optimization problem. There are $N = 3,000$ two-dimensional vectors represented by $x_1, \ldots, x_N$, such that x_i is randomly and independently drawn from the normal distribution $x \sim N(0, I_{2\times 2})$. The goal is to find a two-dimensional vector w which

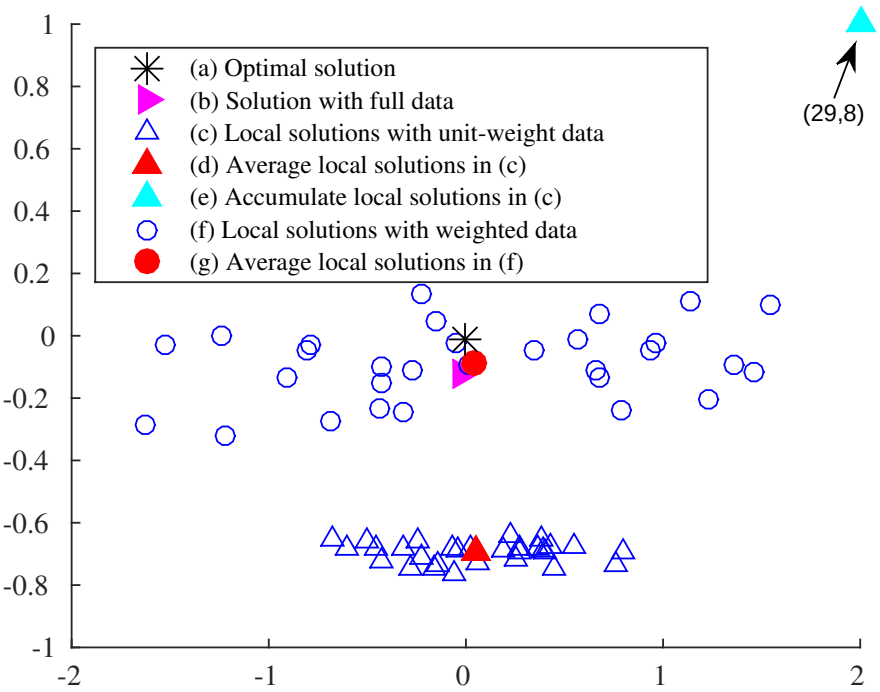


Figure 8.1. Comparing parallelization schemes on a simple convex optimization problem. Totally $N = 3,000$ samples are partitioned into $m = 30$ batches. Each batch is processed by an independent thread running stochastic gradient descent. Each thread uses either unit-weight data or weighted data (weight $= 30$). The local solutions are combined by either averaging or accumulation. From the plot, we find that combining weighted solutions achieves the best performance.

minimizes the weighted distance to all samples. More precisely, the loss function on sample x_i is defined by

$$\ell(w; x_i) := (x_i - w)^T \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{100} \end{pmatrix} (x_i - w)$$

and the overall objective function is $L(w) := \frac{1}{N} \sum_{i=1}^{N} \ell(w; x_i)$. We want to find the vector that minimizes the objective function $L(w)$.

We use the SGD update (8.10) to solve the problem. The algorithm is initialized by $w_0 = (-1, -1)^T$ and the stepsize is chosen by $\eta_t = 1/\sqrt{t}$. For parallel execution, the dataset is evenly partitioned into $m = 30$ disjoint subsets, such that each thread accesses to a single subset, containing 1/30 faction of data. The sequential implementation and the parallel implementations are compared in Figure 8.1. Specifically, we compare seven types of implementations defined by different strategies:

(a) The exact minimizer of $L(w)$.

(b) The solution of SGD achieved by taking a full pass over the dataset. The dataset contains $N = 3,000$ samples.

(c) The local solutions by 30 parallel threads. Each thread runs SGD by taking one pass over its local data. The local dataset contains 100 samples.

(d) Averaging local solutions in (c). This is the averaging scheme described by formula (8.6).

(e) Aggregating local solutions in (c). This is the accumulation scheme described by formula (8.5).

(f) The local solution by 30 parallel threads processing weighted data. Each element is weighted by 30. Each thread runs SGD by taking one pass over its local data.

(g) Combining parallel updates by formula (8.9), setting sample weight $m = 30$. Under this setting, formula (8.9) is equivalent to averaging local solutions in (f).

In Figure 8.1, we observe that solution (b) and solution (g) achieve the best performance. Solution (b) is obtained by a sequential implementation of SGD: it is the baseline solution that parallel algorithms target at approaching. Solution (g) is obtained by Splash with the reweighting scheme. The solutions obtained by other parallelization schemes, namely solution (d) and (e), have poor performances. In particular, the averaging scheme (d) has a large bias relative to the optimal solution. The accumulation scheme (e) diverges far apart from the optimal solution.

To see why Splash is better, we compare local solutions (c) and (f). They correspond to the unweighted SGD and the weighted SGD respectively. We find that solutions (c) have a significant bias but relatively small variance. In contrast, solutions (f) have greater variance but much smaller bias. It verifies our intuition that reweighting helps to reduce the bias by enlarging the local dataset. Note that averaging reduces the variance but doesn't change the bias. It explains why averaging works better with reweighting.

8.4 Convergence analysis

In this section, we study the SGD convergence when it is parallelized by Splash. The goal of SGD is to minimize an empirical risk function

$$L(w) = \frac{1}{|S|} \sum_{x \in S} \ell(w; x),$$

where S is a fixed dataset and $w \in \mathbb{R}^d$ is the vector to be minimized over. Suppose that there are m threads running in parallel. At every iteration, thread i randomly draws (with replacement) a subset of samples S_i of length n from the dataset S. The thread sequentially processes S_i by SGD. The per-iteration update is

$$t \leftarrow t + m \quad \text{and} \quad w \leftarrow w + \big(\Pi_W(w - m\eta_t \nabla \ell(w; x)) - w\big), \tag{8.14}$$

where the sample weight is equal to m. We have generalized the update (8.10) by introducing $\Pi_W(\cdot)$ as a projector to a feasible set W of the vector space. Projecting to the feasible set is a standard post-processing step for an SGD iterate. At the end of the iteration, updates are synchronized by equation (8.9). This is equivalent to computing:

$$t_{\text{new}} = t_{\text{old}} + mn \quad \text{and} \quad w_{\text{new}} = \frac{1}{m}\sum_{i=1}^{m}\Big(w_{\text{old}} + \Delta(mD_i)\Big). \tag{8.15}$$

We denote by $w_\star := \arg\min_{w\in W} L(w)$ the minimizer of the objective function, and denote by w^T the combined vector after the T-th iteration.

General convex function For general convex functions, we start by introducing three additional terms. Let $w_{i,j}^k$ be the value of vector w at iteration k, when thread i is processing the j-th element of S_i. Let $\eta_{i,j}^k$ be the stepsize associated with that update. We define a weighted average vector:

$$\overline{w}^T = \frac{\sum_{k=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n}\eta_{i,j}^k w_{i,j}^k}{\sum_{k=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n}\eta_{i,j}^k}.$$

Note that $\overline{w}^T$ can be computed together with w^T. For general convex L, the function value $L(\overline{w}^T)$ converges to $L(w_\star)$. See Section 8.6.3 for the proof.

Theorem 15. *Assume that $\|\nabla\ell(w;x)\|_2$ is bounded for all $(w,x)\in W\times S$. Also assume that η_t is a monotonically decreasing function of t such that $\sum_{t=1}^{\infty}\eta_t=\infty$ and $\sum_{t=1}^{\infty}\eta_t^2<\infty$. Then we have*

$$\lim_{T\to\infty}\mathbb{E}[L(\overline{w}^T)-L(w^*)]=0.$$

Smooth and strongly convex function We now turn to study smooth and strongly convex objective functions. We make three assumptions on the objective function. Assumption J restricts the optimization problem in a bounded convex set. Assumption K and Assumption L require the objective function to be sufficiently smooth and strongly convex in that set.

Assumption J. *The feasible set $W\subset\mathbb{R}^d$ is a compact convex set of finite diameter R. Moreover, $w_\star$ is an interior point of W; i.e., there is a set $U_\rho:=\{w\in\mathbb{R}^d:\|w-w_\star\|_2<\rho\}$ such that $U_\rho\subset W$.*

Assumption K. *There are finite constants L, G and H such that $\|\nabla^2 L(w;x)-\nabla^2\ell(w_\star;x)\|_2\le L\|w-w_\star\|_2$, $\|\nabla\ell(w;x)\|_2\le G$ and $\|\nabla^2\ell(w;x)\|_2\le H$ for all $(w,x)\in W\times S$.*

Assumption L. *The objective function L is λ-strongly convex over the space W, meaning that $\nabla^2 L(w)\succeq\lambda I_{d\times d}$ for all $w\in W$.*

As a preprocessing step, we construct an Euclidean ball B of diameter $D := \frac{\lambda}{4(L+G/\rho^2)}$ which contains the optimal solution $w_\star$. The ball center can be found by running the sequential SGD for a constant number of steps. During the Splash execution, if the combined vector $w^T \notin B$, then we project it to B, ensuring that the distance between w^T and $w_\star$ is bounded by D. Introducing this projection step simplifies the theoretical analysis, but it may not be necessary in practice.

Under these assumptions, we provide an upper bound on the mean-squared error of w^T. The following theorem shows that the mean-square error decays as $1/(Tmn)$, inversely proportionally to the total number of processed samples. It is the optimal rate of convergence among all optimization algorithms which relies on noisy gradients [167]. See Section 8.6.4 for the proof.

Theorem 16. *Under Assumptions J-L, if we choose the stepsize $\eta_t = \frac{2}{\lambda t}$, then the output w^T has mean-squared error:*

$$\mathbb{E}\left[\|w^T - w_\star\|_2^2\right] \leq \frac{4G^2}{\lambda^2 Tmn} + \frac{C_1}{Tm^{1/2}n^{3/2}} + \frac{C_2}{Tn^2}, \tag{8.16}$$

where C_1 and C_2 are constants independent of T, m and n.

When the local sample size n is sufficiently larger than the thread number m (which is typically true), the last two terms on the right-hand side of bound (8.16) are negligibly small. Thus, the mean-squared error is dominated by the first term, which scales as $1/(Tmn)$.

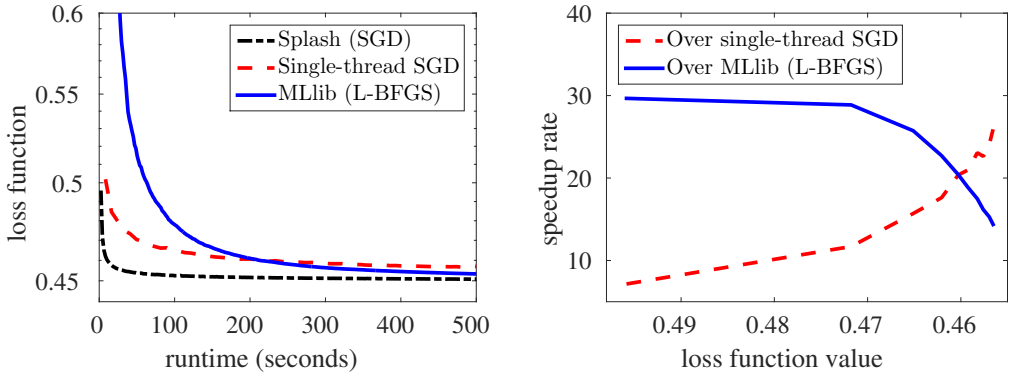
8.5 Experiments

In this section, we report the empirical performance of Splash on three machine learning tasks: logistic regression, collaborative filtering and topic modeling. Our implementation of Splash runs on an Amazon EC2 cluster with eight nodes. Each node is powered by an eight-core Intel Xeon E5-2665 with 30GB of memory and was connected to a commodity 1GB network, so that the cluster contains 64 cores. For all experiments, we compare Splash with MLlib v1.3 [146] — the official distributed machine learning library for Spark. We also compare Splash against single-thread stochastic algorithms.

8.5.1 Logistic regression

We solve a digit recognition problem on the MNIST 8M dataset [130] using multi-class logistic regression. The dataset contains 8 million hand-written digits. Each digit is represented by a feature vector of dimension $d = 784$. There are ten classes representing the digits 0-9. The goal is to minimize the following objective function:

$$L(w) := \frac{1}{n}\sum_{i=1}^{n} -\langle w_{y_i}, x_i\rangle + \log\Big(\sum_{k=0}^{9}\exp^{\langle w_k, x_i\rangle}\Big)$$



(a) Loss function convergence (b) Splash speedup rates

Figure 8.2. Multi-class logistic regression on the MNIST 8M digit recognition dataset. (a) The convergence of different methods; (b) The speedup over other methods for achieving the same loss function value.

where $x_i \in \mathbb{R}^d$ is the feature vector of the i-th element and $y_i \in \{0, \ldots, 9\}$ is its label. The vectors $w_0, \ldots, w_9 \in \mathbb{R}^d$ are parameters of the logistic regression model.

Splash solves the optimization problem by SGD. We use equation (8.10) to generalize SGD to processing weighted samples (the stepsize $\eta_{t,m}$ is approximated by $m\eta_t$). The stepsize η_t is determined by the adaptive subgradient method (AdaGrad) [62]. We compare Splash against the single-thread SGD (with AdaGrad) and the MLlib implementation of L-BFGS [159]. Note that MLlib also provides a mini-batch SGD method, but in practice we found it converging substantially slower than L-BFGS.

Figure 8.2(a) shows the convergence plots of the three methods. Splash converges in a few seconds to a good solution. The single-thread AdaGrad and the L-BFGS algorithm converges to the same accuracy in much longer time. Figure 8.2(b) demonstrates Splash's speedup over other methods. When the target loss decreases, the speedup rate over the single-thread SGD grows larger, while the speedup rate over MLlib drops lower. Thus, Splash is 15x - 30x faster than MLlib. Note that Splash runs a stochastic algorithm and L-BFGS is a batch method. It highlights the advantage of the stochastic method in processing large dataset.

8.5.2 Collaborative filtering

We now turn to a personalized movie recommendation task. For this task, we use the Netflix prize dataset [18], which contains 100 million movie ratings made by 480k users on 17k movies. We split the dataset randomly into a training set and a test set, which contains 90% and 10% of the ratings respectively. The goal is to predict the ratings in the test set given ratings in the training set.

The problem can be solved using collaborative filtering. Assume that each user i is associated with a latent vector $u_i \in \mathbb{R}^d$, and each movie j is associated with a latent vector $v_j \in \mathbb{R}^d$. The affinity score between the user and the movie is measure by the inner product

$\langle u_i, v_j \rangle$. Given ratings in the training set, we define the objective function by:

$$L(\{u_i\}, \{v_j\}) := \sum_{(i,j,r_{ij}) \in S} \left((\langle u_i, v_j \rangle - r_{ij})^2 + \lambda \|u_i\|_2^2 + \lambda \|v_j\|_2^2 \right), \tag{8.17}$$

where $\mathcal{S}$ represents the training set; The triplet (i, j, r_{ij}) represents that the user i gives rating r_{ij} to the movie j. In the training phase, we fit the user vectors $\{u_j\}$ and the movie vectors $\{v_j\}$ by minimizing (8.17). In the testing phase, we predict the ratings of a user i which might not be in the training set. Let $\{r_{ij}\}_{j \in J}$ be the observed ratings from the user, we compute the user vector u_i by

$$u_i := \arg\min_{u \in \mathbb{R}^d} \sum_{j \in J} (\langle u, v_j \rangle - r_{ij})^2 + \lambda \|u\|_2^2,$$

then predict the ratings on other movies by $\langle u_i, v_j \rangle$. The prediction loss is measured by the mean-squared error.

To minimize the objective function (8.17), we employ a SGD method. Let I be the set of users. Let J_i represents the set of movies that user i has rated in the training set. We define an objective function with respect to $\{v_j\}$ as:

$$\begin{aligned} L(\{v_j\}) &:= \min_{\{u_i\}} L(\{u_i\}, \{v_j\}) \\ &= \sum_{i \in I} \left(\min_{u \in \mathbb{R}^d} \left\{ \sum_{j \in J_i} (\langle u, v_j \rangle - r_{ij})^2 + \lambda \|u\|_2^2 \right\} + \lambda \sum_{j \in J_i} \|v_j\|_2^2 \right), \end{aligned} \tag{8.18}$$

so that the movie vectors are obtained by minimizing (8.18). SGD suffices to solve this problem because the objective function is a sum of individual losses.

In practice, we choose the dimension $d = 100$ and regularization parameter $\lambda = 0.02$. Thus the number of parameters to be learned is 65 million. The movie vectors are initialized by a random unit vector on the "first quadrant" (all coordinates are positive). Splash runs the generalized SGD algorithm (8.10) with stepsizes determined by AdaGrad. We compare Splash against the single-thread SGD method and the MLlib implementation of alternating least square (ALS) method. The ALS method minimizes the objective function (8.17) by alternating minimization with respect to $\{u_i\}$ and with respect to $\{v_j\}$.

According to Figure 8.3, Splash converges much faster than the single-thread SGD and the ALS. This is because that SGD can learn accurate movie vectors by processing a fraction of the the data. For example, to achieve a prediction loss lower than 0.70, it takes Splash only 13 seconds, processing 60% of the training set. To achieve the same prediction loss, it takes the ALS 480 seconds, taking 40 passes over the full training set. In other words, Splash features a 36x speedup over the MLlib.

8.5.3 Topic modeling

We use the NYTimes article dataset from the UCI machine learning repository [124]. The dataset contains 300k documents and 100 million word tokens. The vocabulary size is 100k.

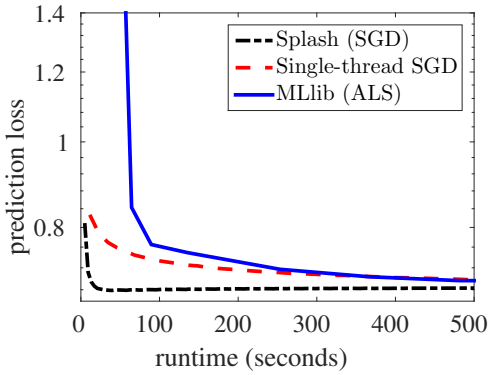


Figure 8.3: Collaborative filtering on the Netflix prize dataset.

The goal is to learn $K = 500$ topics from these documents. Each topic is represented by a multinomial distribution of words. The number of parameters to be learned is 200 million.

We employ the LDA model [30] and choose hyper-parameters $\alpha = \beta = 0.1$. Splash runs the generalized collapsed Gibbs sampling algorithm (8.12)-(8.13). We also use the over-sampling technique [235], that is, for each word the algorithm independently samples 10 topics, each topic carrying 1/10 of the word's weight. We compare Splash with the single-thread collapsed Gibbs sampling algorithm and the MLlib implementation of the variational inference (VI) method [30].

To evaluate the algorithm's performance, we resort to the predictive log-likelihood metric by Hoffman et al. [92]. In particular, we partition the dataset into a training set $\mathcal{S}$ and a test set $\mathcal{T}$. The test set contains 10k documents. For each test document in $\mathcal{T}$, we partition its words into a set of observed words $\mathbf{w}_{\text{obs}}$ and held-out words $\mathbf{w}_{\text{ho}}$, keeping the sets of unique words in $\mathbf{w}_{\text{obs}}$ and $\mathbf{w}_{\text{ho}}$ disjoint. We learn the topics from the training data $\mathcal{S}$, and then use that knowledge and the word set $\mathbf{w}_{\text{obs}}$ to estimate the topic distribution for the test documents. Finally, the predictive log-likelihood of the held-out words, namely $\log p(\mathbf{w}_{\text{new}}|\mathbf{w}_{\text{obs}}, \mathcal{S})$, are computed. The performance of the algorithm is measured by the average predictive log-likelihood per held-out word.

Figure 8.4 plots the predictive log-likelihoods. Among the three methods, the single-thread collapsed Gibbs sampling algorithm exhibits little progress in the first 3,000 seconds. But when the algorithm is parallelized by Splash, it converges faster and better than the MLlib implementation of variational inference (VI). In particular, Splash converges to a predictive log-likelihoods of -8.12, while MLlib converges to -8.36. When measured at fixed target scores, Splash is 3x - 6x faster than MLlib.

8.5.4 Runtime analysis

The runtime of a distributed algorithm can be decomposed into three parts: the computation time, the waiting time and the communication time. The waiting time is the latency that the fast threads wait for the slowest thread. The communication time is the amount of time

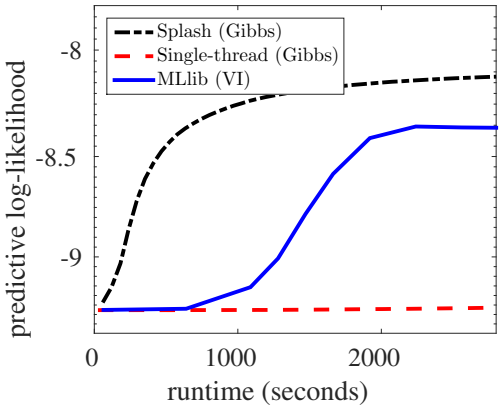


Figure 8.4. Topic modeling on the NYTimes dataset. The LDA model learns $K = 500$ topics.

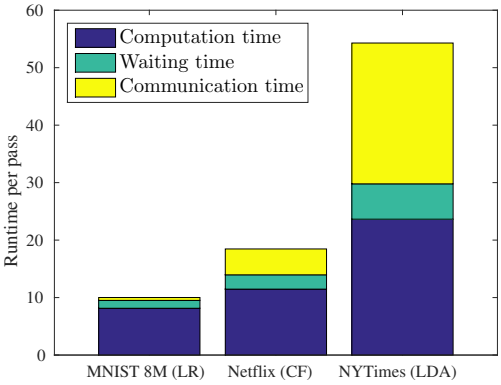


Figure 8.5. Runtime of Splash for taking one pass over the training set. Three machine learning tasks: logistic regression (LR), collaborative filtering (CF) and topic modeling (LDA)

spent on synchronization.

We present runtime analysis on the three machine lear.ning tasks. For logistic regression and collaborative filtering, we let Splash workers synchronize five times per taking one pass over the training set. For topic modeling, we let the workers synchronize once per taking one pass. Figure 8.5 breaks down the runtime of Splash into the three parts. For the three tasks, the waiting time is 16%, 21% and 26% of the computation time. This ratio will increase if the algorithm is parallelized on more machines. In contrast, the communication time is 6%, 39% and 103% of the computation time — it is proportional to the number of parameters to be learned and logarithmically proportional to the number of workers (via TreeReduce). The communication time can be reduced by decreasing the synchronization frequency.

8.6 Technical details

In this section, we present technical details that have been omitted in the main text of the chapter. Then we provide proofs of the convergence results.

8.6.1 Constructing linear transformation on a thread

When element-wise operators are sequentially applied, they merge into a single linear transformation. Assume that after processing a local subset S, the resulting transformation can be represented by

$$v \leftarrow \Gamma(S) \cdot v + \Delta(S) + T(S)$$

where $\Gamma(S)$ is the scale factor, $\Delta(S)$ is the term resulting from the element-wise add operators, and $T(S)$ is the term resulting from the element-wise delayed add operators declared before the last synchronization.

We construct $\Gamma(S)$, $\Delta(S)$ and $T(S)$ incrementally. Let P be the set of processed elements. At the beginning, the set of processed elements is empty, so that we initialize them by

$$\Gamma(P) = 1, \quad \Delta(P) = 0 \quad \text{and} \quad \xi(P) = 0 \quad \text{for } P = \emptyset.$$

After processing element z, we assume that the user has performed all types of operations, resulting in a transformation taking the form

$$v \leftarrow \gamma(v + t) + \delta \tag{8.19}$$

where the scalars γ and δ result from instant operators and t results from the delayed operator. Concatenating transformation (8.19) with the transformation constructed on set P, we have

$$\begin{aligned} v &\leftarrow \gamma \cdot \Big(\Gamma(P) \cdot v + \Delta(P) + T(P) + t\Big) + \delta \\ &= \gamma \cdot \Gamma(P) \cdot v + \Big(\gamma \cdot \Delta(P) + \delta\Big) + \Big(\gamma \cdot T(P) + \gamma t\Big). \end{aligned}$$

Accordingly, we update the terms Γ, Δ and T by

$$\Gamma(P \cup \{z\}) = \gamma \cdot \Gamma(P), \quad \Delta(P \cup \{z\}) = \gamma \cdot \Delta(P) + \delta \quad \text{and} \quad T(P \cup \{z\}) = \gamma \cdot T(P) + \gamma t \tag{8.20}$$

and update the set of processed elements by $P \leftarrow P \cup \{z\}$. After processing the entire local subset, the set P will be equal to S, so that we obtain $\Gamma(S)$, $\Delta(S)$ and $T(S)$.

8.6.2 Determining thread number

Suppose that there are M available cores in the cluster. The execution engine partitions these cores into several groups. Suppose that the i-th group contains m_i cores. The group sizes are determined by the following allocation scheme:

- Let $4m_0$ be the thread number adopted by the last iteration. Let $4m_0 := 1$ at the first iteration.
- For $i = 1, 2, \ldots,$ if $8m_{i-1} \leq M - \sum_{j=1}^{i-1} m_j$, the let $m_i := 4m_{i-1}$. Otherwise, let $m_i := M - \sum_{j=1}^{i-1} m_j$. Terminate when $\sum_{j=1}^{i} m_j = M$.

It can be easily verified that the candidate thread numbers (which are the group sizes) in the current iteration are at least as large as that of the last iteration. The candidate thread numbers are $4m_0, 16m_0, \ldots$ until they consume all of the available cores.

The i-th group is randomly allocated with m_i Parametrized RDD partitions for training, and allocated with another m_i Parametrized RDD partitions for testing. In the training phase, they execute the algorithm on m_i parallel threads, following the parallelization strategy described in Section 8.3.2. In the testing phase, the training results are broadcast to all the partitions. The thread number associated with the smallest testing loss will be chosen. The user is asked to provide an evaluation function $\ell : W \times S \to \mathbb{R}$ which maps a variable-sample pair to a loss value. This function, for example, can be chosen as the element-wise loss for optimization problems, or the negative log-likelihood of probabilistic models. If the user doesn't specify an evaluation function, then the largest m_i will be chosen by the system.

Once a thread number is chosen, its training result will be applied to all Parametrized RDD partitions. The allocation scheme ensures that the largest thread number is at least $3/4$ of M. Thus, in case that M is the best degree of parallelism, the computation power will not be badly wasted. The allocation scheme also ensures that M will be the only candidate of parallelism if the last iteration's thread number is greater than $M/2$. Thus, the degree of parallelism will quickly converge to M if it outperforms other degrees. Finally, the thread number is not updated in every iteration. If the same thread number has been chosen by multiple consecutive tests, then the system will continue using it for a long time, until some retesting criterion is satisfied.

8.6.3 Proof of Theorem 15

We assume that $\|\nabla \ell(w; x)\|_2 \leq G$ for any $(w, x) \in W \times S$. The theorem will be established if the following inequality holds:

$$\sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} 2\eta_{i,j}^k \mathbb{E}[L(w_{i,j}^k) - L(w_\star)] \leq m\mathbb{E}[\|w_0 - w_\star\|_2 - \|w^T - w_\star\|_2] + G^2 \sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} (\eta_{i,j}^k)^2 \tag{8.21}$$

To see how inequality (8.21) proves the theorem, notice that the convexity of function L yields

$$\mathbb{E}[L(\overline{w}_j) - L(w^*)] \leq \frac{\sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} \eta_{i,j}^k \mathbb{E}[L(w_{i,j}^k) - L(w_\star)]}{\sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} \eta_{i,j}^k}.$$

Thus, inequality (8.21) implies

$$\mathbb{E}[L(\overline{w}_j) - L(w^*)] \leq \frac{m\mathbb{E}[\|w_0 - w_\star\|_2 - \|w^T - w_\star\|_2] + G^2 \sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} (\eta_{i,j}^k)^2}{2 \sum_{k=1}^{T} \sum_{i=1}^{m} \sum_{j=1}^{n} \eta_{i,j}^k}.$$

By the assumptions on η_t, it is easy to see that the numerator of right-hand side is bounded, but the denominator is unbounded. Thus, the fraction converges to zero as $T \to \infty$.

It remains to prove inequality (8.21). We prove it by induction. The inequality trivially holds for $T = 0$. For any integer $k > 0$, we assume that the inequality holds for $T = k - 1$. At iteration k, every thread starts from the shared vector w^{k-1}, so that $w_{i,1}^k \equiv w^{k-1}$. For any $j \in \{1, \ldots, n\}$, let $g_{i,j}^k$ be a shorthand for $\nabla \ell(w_{i,j}^k; x)$. A bit of algebraic transformation yields:

$$\begin{aligned} \|w_{i,j+1}^k - w_\star\|_2^2 &= \|\Pi_W(w_{i,j}^k - \eta_{i,j}^k g_{i,j}^k) - w_\star\|_2^2 \leq \|w_{i,j}^k - \eta_{i,j}^k g_{i,j}^k - w_\star\|_2^2 \\ &= \|w_{i,j}^k - w_\star\|_2^2 + (\eta_{i,j}^k)^2 \|g_{i,j}^k\|_2^2 - 2\eta_{i,j}^k \langle w_{i,j}^k - w_\star, g_{i,j}^k \rangle, \end{aligned}$$

where the inequality holds since $w_\star \in W$ and Π_W is the projection onto W. Taking expectation on both sides of the inequality and using the assumption that $\|g_{i,j}^k\|_2 \leq G$, we have

$$\mathbb{E}[\|w_{i,j+1}^k - w_\star\|_2^2] \leq \mathbb{E}[\|w_{i,j}^k - w_\star\|_2^2] + G^2(\eta_{i,j}^k)^2 - 2\eta_{i,j}^k \mathbb{E}[\langle w_{i,j}^k - w_\star, \nabla L(w_{i,j}^k) \rangle].$$

By the convexity of function L, we have $\langle w_{i,j}^k - w_\star, \nabla L(w_{i,j}^k) \rangle \geq L(w_{i,j}^k) - L(w_\star)$. Plugging in this inequality, we have

$$2\eta_{i,j}^k \mathbb{E}[L(w_{i,j}^k) - L(w_\star)] \leq \mathbb{E}[\|w_{i,j}^k - w_\star\|_2^2] - \mathbb{E}[\|w_{i,j+1}^k - w_\star\|_2^2] + G^2(\eta_{i,j}^k)^2. \tag{8.22}$$

Summing up inequality (8.22) for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, we obtain

$$\begin{aligned} \sum_{i=1}^{m} \sum_{j=1}^{n} 2\eta_{i,j}^k \mathbb{E}[L(w_{i,j}^k) - L(w_\star)] \leq m\mathbb{E}[\|w^{k-1} - w_\star\|_2^2] - \sum_{i=1}^{m} \mathbb{E}[\|w_{i,n+1}^k - w_\star\|_2^2] \\ + \sum_{i=1}^{m} \sum_{j=1}^{n} G^2(\eta_{i,j}^k)^2. \end{aligned} \tag{8.23}$$

Notice that $w^k = \frac{1}{m} w_{i,n+1}^k$. Thus, Jensen's inequality implies

$$\sum_{i=1}^{m} \|w_{i,n+1}^k - w_\star\|_2^2 \geq m\|w^k - w_\star\|_2.$$

Plugging this inequality to upper bound (8.23) yields

$$\sum_{i=1}^{m} \sum_{j=1}^{n} 2\eta_{i,j}^k \mathbb{E}[L(w_{i,j}^k) - L(w_\star)] \leq m\mathbb{E}[\|w^{k-1} - w_\star\|_2^2 - \|w^k - w_\star\|_2^2] + \sum_{i=1}^{m} \sum_{j=1}^{n} G^2(\eta_{i,j}^k)^2. \tag{8.24}$$

The induction is complete by combining upper bound (8.24) with the inductive hypothesis.

8.6.4 Proof of Theorem 16

Recall that w^k is the value of vector w after iteration k. Let w_i^k be the output of thread i at the end of iteration k. According to the update formula, we have $w^k = \Pi_B(\frac{1}{m}\sum_{i=1}^m w_i^k)$, where $\Pi_B(\cdot)$ is the projector to the set B. The set B contains the optimal solution $w_\star$. Since projecting to a convex set doesn't increase the point's distance to the elements in the set, and because that w_i^k $(i = 1, \ldots, m)$ are mutually independent conditioning on w^{k-1}, we have

$$\begin{aligned}
\mathbb{E}[\|w^k - w_\star\|_2^2] &\leq \mathbb{E}\Big[\mathbb{E}\Big[\Big\|\frac{1}{m}\sum_{i=1}^m w_i^k - w_\star\Big\|_2^2\Big|w^{k-1}\Big]\Big] \\
&= \frac{1}{m^2}\sum_{i=1}^m \mathbb{E}[\mathbb{E}[\|w_i^k - w_\star\|_2^2|w^{k-1}]] + \frac{1}{m^2}\sum_{i\neq j}\mathbb{E}[\mathbb{E}[\langle w_i^k - w_\star, w_j^k - w_\star\rangle|w^{k-1}]] \\
&= \frac{1}{m}\mathbb{E}[\|w_1^k - w_\star\|_2^2] + \frac{m-1}{m}\mathbb{E}[\|\mathbb{E}[w_1^k|w^{k-1}] - w_\star\|_2^2] \qquad (8.25)
\end{aligned}$$

Equation (8.25) implies that we could upper bound the two terms on the right-hand side respectively. To this end, we introduce three shorthand notations:

$$\begin{aligned}
a_k &:= \mathbb{E}[\|w^k - w_\star\|_2^2], \\
b_k &:= \mathbb{E}[\|w_1^k - w_\star\|_2^2], \\
c_k &:= \mathbb{E}[\|\mathbb{E}[w_1^k|w^{k-1}] - w_\star\|_2^2].
\end{aligned}$$

Essentially, equation (8.25) implies $a_k \leq \frac{1}{m}b_k + \frac{m-1}{m}c_k$. Let $a_0 := \|w^0 - w_\star\|_2$ where w^0 is the initial vector. The following two lemmas upper bounds b_{k+1} and c_{k+1} as functions of a_k. We defer their proofs to the end of this section.

Lemma 39. *For any integer $k \geq 0$, we have*

$$b_{k+1} \leq \frac{k^2}{(k+1)^2}a_k + \frac{\beta_1}{(k+1)^2 n} \quad \textit{where} \quad \beta_1 := 4G^2/\lambda^2.$$

Lemma 40. *We have $c_1 \leq \beta_2^2/n^2$ and for any integer $k \geq 1$,*

$$c_{k+1} \leq \frac{k^2}{(k+1)^2}a_k + \frac{2\beta_2\sqrt{a_k} + \beta_2^2/n}{(k+1)^2 n} \quad \textit{where} \quad \beta_2 := \max\left\{\lceil 2H/\lambda\rceil R, \frac{8G^2(L + G/\rho^2)}{\lambda^3}\right\}.$$

Combining equation (8.25) with the results of Lemma (39) and Lemma (40), we obtain an upper bound on a_1:

$$a_1 \leq \frac{\beta_1}{mn} + \frac{\beta_2^2}{n^2} := \beta_3. \qquad (8.26)$$

Furthermore, Lemma (39) and Lemma (40) upper bound a_{k+1} as a function of a_k:

$$a_{k+1} \leq \frac{k^2}{(k+1)^2}a_k + \frac{\beta_3 + 2\beta_2\sqrt{a_k}/n}{(k+1)^2}. \qquad (8.27)$$

Using upper bounds (8.26) and (8.27), we claim that

$$a_k \leq \frac{\beta_3 + 2\beta_2\sqrt{\beta_3}/n}{k} \quad \text{for} \quad k = 1, 2, \ldots \tag{8.28}$$

By inequality (8.26), the claim is true for $k = 1$. We assume that the claim holds for k and prove it for $k + 1$. Using the inductive hypothesis, we have $a_k \leq \beta_3$. Thus, inequality (8.27) implies

$$a_{k+1} \leq \frac{k^2}{(k+1)^2} \cdot \frac{\beta_3 + 2\beta_2\sqrt{\beta_3}/n}{k} + \frac{\beta_3 + 2\beta_2\sqrt{\beta_3}/n}{(k+1)^2} = \frac{\beta_3 + 2\beta_2\sqrt{\beta_3}/n}{(k+1)n}$$

which completes the induction. Note that both β_1 and β_2 are constants that are independent of k, m and n. Plugging the definition of β_3, we can rewrite inequality (8.28) as

$$a_k \leq \frac{4G^2}{\lambda^2 kmn} + \frac{C_1}{km^{1/2}n^{3/2}} + \frac{C_2}{kn^2}.$$

where C_1 and C_2 are constants that are independent of k, m and n. This completes the proof of the theorem.

8.6.4.1 Proof of Lemma 39

In this proof, we use w_j as a shorthand to denote the value of vector w at iteration $k + 1$ when the first thread is processing the j-th element. We drop the notation's dependence on the iteration number and on the thread index since they are explicit from the context. Let $g_j = \nabla\ell(w_j; x_j)$ be the gradient of loss function ℓ with respect to w_j on the j-th element. Let η_j be the stepsize parameter when w_j is updated. It is easy to verify that $\eta_j = \frac{2}{\lambda(kn+j)}$.

We start by upper bounding the expectation of $\|w_1^{k+1} - w_\star\|_2^2$ conditioning on w^k. By the strong convexity of L and the fact that $w_\star$ minimizes L, we have

$$\langle \mathbb{E}[g_j], w_j - w_\star \rangle \geq L(w_j) - L(w_\star) + \frac{\lambda}{2} \|w_j - w_\star\|_2^2 .$$

as well as

$$L(w_j) - L(w_\star) \geq \frac{\lambda}{2} \|w_j - w_\star\|_2^2 .$$

Hence, we have

$$\langle \mathbb{E}[g_j], w_j - w_\star \rangle \geq \lambda \|w_j - w_\star\|_2^2 \tag{8.29}$$

Recall that $\Pi_W(\cdot)$ denotes the projection onto set W. By the convexity of W, we have $\|\Pi_W(u) - v\|_2 \leq \|u - v\|_2$ for any $u, v \in W$. Using these inequalities, we have the following:

$$\begin{aligned}
\mathbb{E}[\|w_{j+1} - w_\star\|_2^2 | w^k] &= \mathbb{E}[\|\Pi_W(w_j - \eta_j g_j) - w_\star\|_2^2 | w^k] \\
&\leq \mathbb{E}[\|w_j - \eta_j g_j - w_\star\|_2^2 | w^k] \\
&= \mathbb{E}[\|w_j - w_\star\|_2^2 | w^k] - 2\eta_j \mathbb{E}\left[\langle g_j, w_j - w_\star \rangle | w^k\right] + \eta_j^2 \mathbb{E}[\|g_j\|_2^2 | w^k].
\end{aligned}$$

Note that the gradient g_j is independent of w_j conditioning on w^{k-1}. Thus, we have

$$\mathbb{E}\left[\langle g_j, w_j - w_\star\rangle | w^k\right] = \mathbb{E}[\langle \mathbb{E}[g_j], w_j - w_\star\rangle | w^k] \geq \lambda \mathbb{E}[\|w_j - w_\star\|_2^2 \, | w^k].$$

where the last inequality follows from inequality (8.29). As a consequence, we have

$$\mathbb{E}[\|w_{j+1} - w_\star\|_2^2 | w^k] \leq (1 - 2\eta_j\lambda)\mathbb{E}[\|w_j - w_\star\|_2^2 | w^k] + \eta_j^2 G^2.$$

Plugging in $\eta_j = \frac{2}{\lambda(kn+j)}$, we obtain

$$\mathbb{E}[\|w_{j+1} - w_\star\|_2^2 | w^k] \leq \left(1 - \frac{4}{kn+j}\right) \mathbb{E}[\|w_j - w_\star\|_2^2 | w^k] + \frac{4G^2}{\lambda^2(kn+j)^2}. \tag{8.30}$$

Case $k = 0$: We claim that any $j \geq 1$,

$$\mathbb{E}[\|w_j - w_\star\|_2^2] \leq \frac{4G^2}{\lambda^2 j} \tag{8.31}$$

Since $w_1^1 = w_{n+1}$, the claim establishes the lemma. We prove the claim by induction. The claim holds for $j = 1$ because inequality (8.29) yields

$$\|w_1 - w_\star\|_2^2 \leq \frac{\langle \mathbb{E}[g_1], w_1 - w_\star\rangle}{\lambda} \leq \frac{G\|w_1 - w_\star\|_2}{\lambda} \quad \Rightarrow \quad \|w_1 - w_\star\|_2 \leq G/\lambda.$$

Otherwise, we assume that the claim holds for j. Then inequality (8.30) yields

$$\begin{aligned}\mathbb{E}[\|w_{j+1} - w_\star\|_2^2] &\leq \left(1 - \frac{4}{j}\right)\frac{4G^2}{\lambda^2 j} + \frac{4G^2}{\lambda^2 j^2} \\ &= \frac{4G^2}{\lambda^2}\frac{j-4+1}{j^2} \leq \frac{4G^2}{\lambda^2(j+1)},\end{aligned}$$

which completes the induction.

Case $k > 0$: We claim that for any $j \geq 1$,

$$\mathbb{E}[\|w_j - w_\star\|_2^2 | w^k] \leq \frac{1}{(kn+j-1)^2}\left((kn)^2\|w^k - w_\star\|_2^2 + \frac{4G^2(j-1)}{\lambda^2}\right) \tag{8.32}$$

We prove (8.32) by induction. The claim is obviously true for $j = 1$. Otherwise, we assume that the claim holds for j and prove it for $j+1$. Since $1 - \frac{4}{kn+j} \leq (\frac{kn+j-1}{kn+j})^2$, combining the inductive hypothesis and inequality (8.30), we have

$$\begin{aligned}&\mathbb{E}[\|w_{j+1} - w_\star\|_2^2 | w^k] \\ &\quad\leq \frac{1}{(kn+j)^2}\left((kn)^2\|w^k - w_\star\|_2^2 + \frac{4G^2(j-1)}{\lambda^2}\right) + \frac{4G^2}{\lambda^2(kn+j)^2} \\ &\quad= \frac{1}{(kn+j)^2}\left((kn)^2\|w^k - w_\star\|_2^2 + \frac{4G^2 j}{\lambda^2}\right).\end{aligned}$$

which completes the induction. Note that claim (8.32) establishes the lemma since $w_1^k = w_{n+1}$.

8.6.4.2 Proof of Lemma 40

In this proof, we use w_j as a shorthand to denote the value of vector w at iteration $k+1$ when the first thread is processing the j-th element. We drop the notation's dependence on the iteration number and on the thread index since they are explicit from the context. Let $g_j = \nabla \ell(w_j; x_j)$ be the gradient of loss function ℓ with respect to w_j on the j-th element. Let η_j be the stepsize parameter when w_j is updated. It is easy to verify that $\eta_j = \frac{2}{\lambda(kn+j)}$.

Recall the neighborhood $U_\rho \subset W$ in Assumption J, and note that

$$\begin{aligned} w_{j+1} - w_\star &= \Pi_W(w_j - \eta_j g_j - w_\star) \\ &= w_j - \eta_j g_j - w_\star + \mathbb{I}(w_{j+1} \notin U_\rho)\left(\Pi_W(w_j - \eta_j g_j) - (w_j - \eta_j g_j)\right) \end{aligned}$$

since when $w \in U_\rho$, we have $\Pi_W(w) = w$. Consequently, an application of the triangle inequality and Jensen's inequality gives

$$\begin{aligned} \|\mathbb{E}[w_{j+1} - w_\star | w^k]\|_2 \leq \|\mathbb{E}[w_j - \eta_j g_j - w_\star | w^k]\|_2 \\ + \mathbb{E}\left[\|(\Pi_W(w_j - \eta_j g_j) - (w_j - \eta_j g_j))1(w_{j+1} \notin U_\rho)\|_2 \, | w^k\right]. \end{aligned}$$

By the definition of the projection and the fact that $w_j \in W$, we additionally have

$$\|\Pi_W(w_j - \eta_j g_j) - (w_j - \eta_j g_j)\|_2 \leq \|w_j - (w_j - \eta_j g_j))\|_2 \leq \eta_j \|g_j\|_2 .$$

Thus, by combining the above two inequalities, and applying Assumption K, we have

$$\begin{aligned} \|\mathbb{E}[w_{j+1} - w_\star | w^k]\|_2 &\leq \|\mathbb{E}[w_j - \eta_j g_j - w_\star | w^k]\|_2 + \eta_j \mathbb{E}[\|g_j\|_2 \, 1_{(w_{j+1} \notin U_\rho)} | w^k] \\ &\leq \|\mathbb{E}[w_j - \eta_j g_j - w_\star | w^k]\|_2 + \eta_j G \cdot P(w_j \notin U_\rho | w^k) \\ &\leq \|\mathbb{E}[w_j - \eta_j g_j - w_\star | w^k]\|_2 + \eta_j G \cdot \frac{\mathbb{E}[\|w_{j+1} - w_\star\|_2^2 | w^k]}{\rho^2}, \end{aligned} \tag{8.33}$$

where the last inequality follows from the Markov's inequality.

Now we turn to controlling the rate at which $w_j - \eta_j - g_j$ goes to zero. Let $\ell_j(\cdot) = \ell(\cdot; x_j)$ be a shorthand for the loss evaluated on the j-th data element. By defining

$$r_j := g_j - \nabla \ell_j(w_\star) - \nabla^2 \ell_j(w_\star)(w_j - w_\star),$$

a bit of algebra yields

$$g_j = \nabla \ell_j(w_\star) + \nabla^2 \ell_j(w_\star)(w_j - w_\star) + r_j.$$

First, we note that $\mathbb{E}[\nabla \ell_j(w_\star) | w^k] = \nabla L(w_\star) = 0$. Second, the Hessian $\nabla^2 \ell_j(w_\star)$ is independent of w_j. Hence we have

$$\begin{aligned} \mathbb{E}[g_j | w^k] &= \mathbb{E}[\nabla \ell_j(w_\star)] + \mathbb{E}[\nabla^2 \ell_j(w_\star) | w^k] \cdot \mathbb{E}[w_j - w_\star | w^k] + \mathbb{E}[r_j | w^k] \\ &= \nabla^2 L(w_\star) \mathbb{E}[w_j - w_\star | w^k] + \mathbb{E}[r_j | w^k]. \end{aligned} \tag{8.34}$$

Taylor's theorem implies that r_j is the Lagrange remainder

$$r_j = (\nabla^2 \ell_j(w') - \nabla^2 \ell_j(w_\star))(w' - w_\star),$$

where $w' = \kappa w_j + (1-\kappa) w_\star$ for some $\kappa \in [0,1]$. Applying Assumption K, we find that

$$\begin{aligned}\mathbb{E}[\|r_j\|_2 | w^k] &\le \mathbb{E}[\|\nabla^2 \ell_j(w') - \nabla^2 \ell_j(w_\star)\|_2 \, \|w_j - w_\star\|_2 \, | w^k] \\ &\le L \mathbb{E}[\|w_j - w_\star\|_2^2 | w^k]. \end{aligned} \tag{8.35}$$

By combining the expansion (8.34) with the bound (8.35), we find that

$$\begin{aligned}\|\mathbb{E}[w_j - \eta_j g_j - w_\star | w^k]\|_2 &= \left\|\mathbb{E}[(I - \eta_j \nabla^2 F_0(w_\star))(w_j - w_\star) + \eta_j r_j | w^k]\right\|_2 \\ &\le \|(I - \eta_j \nabla^2 L(w_\star)) \mathbb{E}[w_j - w_\star | w^k]\|_2 + \eta_j L \mathbb{E}[\|w_j - w_\star\|_2^2 | w^k].\end{aligned}$$

Using the earlier bound (8.33) and plugging in the assignment $\eta_j = \frac{2}{\lambda(kn+j)}$, this inequality then yields

$$\begin{aligned}\|\mathbb{E}[w_{j+1} - w_\star | w^k]\|_2 \le \left\|I - \eta_j \nabla^2 L(w_\star)\right\|_2 \|\mathbb{E}[w_j - w_\star | w^k]\|_2 \\ + \frac{2}{\lambda(kn+j)} \left(L \mathbb{E}[\|w_j - w_\star\|_2^2 | w^k] + \frac{G \mathbb{E}[\|w_{j+1} - w_\star\|_2^2 | w^k]}{\rho^2} \right). \end{aligned} \tag{8.36}$$

Next, we split the proof into two cases when $k = 1$ and $k > 1$.

Case $k = 0$: Note that by strong convexity and our condition that $\|\nabla^2 L(w_\star)\|_2 \le H$, whenever $\eta_j H \le 1$ we have

$$\|I - \eta_j \nabla^2 L(w_\star)\|_2 = 1 - \eta_j \lambda_{\min}(\nabla^2 L(w_\star)) \le 1 - \eta_j \lambda$$

Define $\tau_0 = \lceil 2H/\lambda \rceil$; then for $j \ge \tau_0$, we have $\eta_j H \le 1$. As a consequence, inequality (8.31) (in the proof of Lemma 39) and inequality (8.36) yield that for any $j \ge \tau_0$,

$$\|\mathbb{E}[w_{j+1} - w_\star]\|_2 \le (1 - 2/j) \, \|\mathbb{E}[w_j - w_\star]\|_2 + \frac{8G^2}{\lambda^3 j^2} \left(L + G/\rho^2\right). \tag{8.37}$$

As shorthand notations, we define two intermediate variables

$$u_t = \|\mathbb{E}(w_j - w_\star)\|_2 \quad \text{and} \quad b_1 = \frac{8G^2}{\lambda^3} \left(L + G/\rho^2\right).$$

Inequality (8.37) then implies the inductive relation

$$u_{j+1} \le (1 - 2/j) u_j + b_1/j^2 \quad \text{for any } j \ge \tau_0.$$

Now we claim that by defining $b_2 := \max\{\tau_0 R, b_1\}$, we have $u_j \le \beta/j$. Indeed, it is clear that $u_j \le \tau_0 R/j$ for $j = 1, 2, \ldots, \tau_0$. For $t > \tau_0$, using the inductive hypothesis, we have

$$u_{j+1} \le \frac{(1-2/j) b_2}{j} + \frac{b_1}{j^2} \le \frac{b_2 j - 2b_2 + b_2}{j^2} = \frac{b_2(j-1)}{j^2} \le \frac{b_2}{j+1}.$$

This completes the induction and establishes the lemma for $k = 0$.

Case $k > 0$: Let $u_j = \|\mathbb{E}[w_j - w_\star | w^k]\|_2$ and $\delta = \|w^k - w_\star\|_2$ as shorthands. Combining inequality (8.32) (in the proof of Lemma 39) and inequality (8.36) yield

$$\begin{aligned} u_{j+1} &\leq \left(1 - \frac{2}{kn+j}\right) u_j + \frac{2(L + G/\rho^2)}{\lambda(kn+j)(kn+j-1)^2} \left((kn)^2\delta^2 + \frac{4G^2 j}{\lambda^2}\right) \\ &\leq \left(1 - \frac{2}{kn+j}\right) u_j + \frac{2(L + G/\rho^2)}{\lambda(kn+j)(kn+j-1)kn} \left((kn)^2\delta^2 + \frac{4G^2 n}{\lambda^2}\right) \\ &= \frac{(kn+j-2)(kn+j-1)}{(kn+j-1)(kn+j)} u_j + \frac{b_1 kn\delta^2 + b_2/k}{(kn+j-1)(kn+j)} \end{aligned} \tag{8.38}$$

where we have introduced shorthand notations $b_1 := \frac{2(L+G/\rho^2)}{\lambda}$ and $b_2 := \frac{8G^2(L+G/\rho^2)}{\lambda^3}$. With these notations, we claim that

$$u_j \leq \frac{(kn-1)kn\delta + (j-1)(b_1 kn\delta^2 + b_2/k)}{(kn+j-2)(kn+j-1)}. \tag{8.39}$$

We prove the claim by induction. Indeed, since $u_1 = \delta$, the claim obviously holds for $j = 1$. Otherwise, we assume that the claim holds for j, then inequality (8.38) yields

$$\begin{aligned} u_{j+1} &\leq \frac{(kn-1)kn\delta + (j-1)(b_1 kn\delta^2 + b_2/k)}{(kn+j-1)(kn+j)} + \frac{b_1 kn\delta^2 + b_2/k}{(kn+j-1)(kn+j)} \\ &= \frac{(kn-1)kn\delta + j(b_1 kn\delta^2 + b_2/k)}{(kn+j-1)(kn+j)}, \end{aligned}$$

which completes the induction. As a consequence, a bit of algebraic transformation yields

$$\begin{aligned} \|\mathbb{E}[w_1^{k+1} - w_\star | w^k]\|_2 = u_{n+1} &\leq \frac{(kn-1)kn\delta + n(b_1 kn\delta^2 + b_2/k)}{((k+1)n-1)(k+1)n} \\ &\leq \frac{k^2 n^2 \delta}{(k+1)^2 n^2} + \frac{n b_1 kn\delta^2}{kn(k+1)n} + \frac{n b_2/k}{kn(k+1)n} \\ &\leq \left(\frac{k}{k+1}\right)^2 \delta + \frac{b_1\delta^2}{k+1} + \frac{b_2}{k(k+1)n} \\ &= \frac{k}{k+1}\left(\frac{k\delta + \frac{k+1}{k} b_1\delta^2}{k+1} + \frac{b_2}{k^2 n}\right) \end{aligned} \tag{8.40}$$

By the fact that $w^k \in B$, we have $\frac{k+1}{k} b_1 \delta \leq \frac{k+1}{k} b_1 D \leq 1$. Thus, inequality (8.40) implies

$$\|\mathbb{E}[w_1^{k+1} - w_\star | w^k]\|_2^2 \leq \left(\frac{k}{k+1}\right)^2 \left(\delta + \frac{b_2}{k^2 n}\right)^2$$

Taking expectation on both sides of the inequality, then applying Jensen's inequality, we obtain

$$\mathbb{E}[\|\mathbb{E}[w_1^{k+1} - w_\star | w^k]\|_2^2] \leq \frac{k^2 \mathbb{E}[\delta^2]}{(k+1)^2} + \frac{2b_2\sqrt{\mathbb{E}[\delta^2]} + b_2^2/n}{(k+1)^2 n}.$$

Hence, the lemma is established.

Chapter 9

Conclusion and future directions

In this chapter, we summarize the key contributions made by the thesis, and suggest several future directions. The chapter is organized in three sections, corresponding to the three parts of the main content of the thesis. In each section, we first summarize the main ideas presented in this part, then layout a roadmap for things that could likely follow.

9.1 Conclusion on distributed algorithms

In the first part of the thesis, we have presented three types of distributed algorithms for statistical optimization. In Chapter 3, we analyzed the one-shot averaging algorithm and proposed an improved algorithm using the technique of booststrap. The idea is to partition the dataset randomly into multiple pieces, compute a local solution based on each piece of data, then combine the local solutions by averaging. Despite the simplicity of this approach, we have shown that it achieves the optimal statistical accuracy under particular conditions. The most critical assumption of this chapter is the strong convexity assumption of the population risk. This assumption is satisfied by the parametric models on many real datasets. More precisely, we have shown that the simple one-shot averaging algorithm can tolerate the number of parallel machines proportional to the square root of the total number of samples. It means that we can parallelize the learning algorithm on thousands or more machines. The degree of parallelisim may be further increased if we use the bootstrapping bias correction technique. The advantage of the bootstrapping approach has been confirmed both theoretically and empirically.

There are several interesting questions that remain open. First, the chapter makes an assumption that the loss function must be third-order differentiable. This assumption might be too strong for particular models, such as the support vector machine modelo, where the loss function is continuous but not smooth. It is important to study the efficient parallelization of optimization algorithms for these non-smooth loss functions. Second, we have assumed the network topology to be star shaped or fully connected. it may also be interesting to study the effects of subsampled or bootstrap-based estimators in other distributed environments.

In Chapter 4, we have extended the idea of divide-and-conquer method to non-parametric learning. We focused on the problem of kernel ridge regression and demonstrated that the optimal statistical accuracy can be achieved by a divide-and-conquer algorithm. The idea is to train an estimator on each subset of data, with carefully chosen regularization parameters. Then the local estimators are averaged to form the global estimator. We show that the divide-and-conquer strategy substantially improves the computation efficiency, as the algorithm is dependent on only a small portion of entries of the kernel matrix.

It is interesting to consider the number of kernel evaluations required to implement our method. Recall that our method partitions the dataset into m pieces and quantifies $\gamma(\lambda)$ to be the *effective dimension* of the RKHS. Our estimator requires m sub-matrices of the full kernel (Gram) matrix, each of size $N/m \times N/m$. Since we require $m \leq N/\gamma^2(\lambda)$, in the best case, the algorithm requires at most $N\gamma^2(\lambda)$ kernel evaluations. By contrast, Bach [10] shows that Nyström-based subsampling can be used to form an estimator within a constant factor of optimal as long as the number of N-dimensional subsampled columns of the kernel matrix scales roughly as the *marginal dimension* $\widetilde{\gamma}(\lambda) = N \left\|\mathrm{diag}(K(K + \lambda NI)^{-1})\right\|_\infty$. Consequently, using roughly $N\widetilde{\gamma}(\lambda)$ kernel evaluations, Nyström subsampling can achieve optimal convergence rates. These two scalings–namely, $N\gamma^2(\lambda)$ versus $N\widetilde{\gamma}(\lambda)$—are currently not comparable: in some situations, such as when the data is not compactly supported, $\widetilde{\gamma}(\lambda)$ can scale linearly with N, while in others it appears to scale roughly as the true effective dimensionality $\gamma(\lambda)$. A natural question arising from these lines of work is to understand the true optimal scaling for these different estimators: is one fundamentally better than the other? Are there natural computational tradeoffs that can be leveraged at large scale? As datasets grow substantially larger and more complex, these questions should become even more important.

In Chapter 5, we have proposed a distributed optimization algorithm for a broader class of objective functions. We assume that the objective function may not be strongly convex. Instead, it is regularized by a squared ℓ_2-norm whose coefficient diminishes to zero as the sample size grows to infinity. Under this setting, the simple averaging strategy is not able to achieve the optimal convergence rate. The proposed DiSCO algorithm is an iterative approach and it uses both the first-order information and the second-order information of the objective function to speedup the convergence. The algorithm's computation complexity is as efficient as the first-order method, but its iteration complexity for achieving a particular optimality gap is as low as second-order methods. As a consequence, we showed that the iteration complexity of the algorithm doesn't depend on the total number of data points in the dataset.

The DiSCO algorithm has several limitations. It requires the objective function to be self-concordant, which is not satisfiable by non-smooth functions. The efficiency of DiSCO relies on the i.i.d. property of the data points, thus its convergence rate guarantee is not as strong as those optimization methods for worst-case inputs. It is interesting to study new algorithms that overcome these limitations. To simplify the problem, we may assume that there are only two machines, but the data on these machines are not i.i.d., and the objective function is the least-square loss for linear regression. Finding a communication-efficient

algorithm for solving this problem is of both theoretical and practical interest.

9.2 Conclusion on theories of distributed computing

In the second part of the thesis, we have studied fundamental limits of distributed algorithms. In Chapter 6, we have established lower bounds on the amount of communication required for several statistical estimation problems. Our lower bounds are information-theoretic in nature, based on variants of Fano's and Le Cam's methods. In particular, they rely on novel types of quantitative data processing inequalities that characterize the effect of bit constraints on the mutual information between parameters and messages.

Several open questions remain in Chapter 6. Our arguments are somewhat complex, and our upper and lower bounds differ by logarithmic factors. It would be interesting to understand which of our bounds can be sharpened; tightening the upper bounds would lead to interesting new distributed inference protocols, while improving the lower bounds could require new technical insights. We believe it will also be interesting to explore the application and extension of our results and techniques to other — perhaps more complex — problems in statistical estimation.

In Chapter 7, we have studied the problem of estimating the generalized rank of n-by-n matrices stored on multiple machines. Our main results are to show that in the deterministic setting, sending $\Theta(n^2)$ bits is both necessary and sufficient in order to obtain any constant relative error. In contrast, when randomized algorithms are allowed, this scaling is reduced to $\widetilde{\Theta}(n)$.

We raised an open question of how to estimate the matrix rank with relative error $\delta \ll 1/\sqrt{r}$ where r is the matrix rank. In Section 7.5, we demonstrated the connection between this open problem and the study of communication complexity for a broad class of linear algebraic computation problems. We have shown that if one can prove a tight communication complexity lower bound for estimating matrix rank with very small error, then it implies tight lower bounds for the problems listed in Section 7.5. This connection suggests the importance of studying matrix rank estimation, especially for characterizing lower bounds on communication complexity. We hope that the results in Chapter 7 are a meaningful first step in exploring this problem area.

9.3 Conclusion on machine learning systems

The last part of the thesis is devoted to practical machine learning systems. Chapter 8, we have presented Splash — a general framework for parallelizing stochastic algorithms. The programming paradigm of Splash is designed around a key concept: implementing incremental updates that processes weighted data. This paradigm allows the system to automatically parallelize the algorithm on commodity clusters. On machine learning tasks, Splash is orders-of-magnitude faster than state-of-the-art implementations adopting the it-

erative MapReduce. The fast performance is partially due to the superiority of stochastic algorithms over the batch algorithm, and partially due to the communication-efficient feature of the system. In addition, Splash is built on top of Spark which allows it seamlessly integrating with the existing data analytics stack.

For the parallelization strategy of Splash, we have provided theoretical analysis on stochastic gradient descent, and proved the optimal rate of convergence for strongly convex objective functions. It remains unknown if the same optimality guarantee holds for general convex functions, and more generally, if it holds for other problems besides convex optimization. These questions suggest theoretical studies of Splash for the future work. On the computer systems front, we are interested in adapting the framework to a more diversified system. For example, it is still challenging to perform distributed machine learning on a hybrid system of shared memories across cores and a network connecting machines. It is also important to design a system which can efficiently run large-scale optimization algorithms on GPU clusters, while simultaneously preserving the ease of implementation.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org.*

[2] H. Abelson. Lower bounds on information transfer in distributed computations. *Journal of the ACM (JACM)*, 27(2):384–392, 1980.

[3] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kegl, and D. Rousseau. Learning to discover: the higgs boson machine learning challenge. *URL http://higgsml. lal. in2p3. fr/documentation*, 2014.

[4] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *NIPS*, pages 873–881, 2011.

[5] A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, May 2012.

[6] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and E. Smola. Scalable inference in latent variable models. In *In WSDM*, pages 123–132, 2012.

[7] P. Assouad. Deux remarques sur l'estimation. *Comptes rendus des séances de l'Académie des sciences. Série 1, Mathématique*, 296(23):1021–1024, 1983.

[8] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, 1976.

[9] F. Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4:384–414, 2010.

[10] F. Bach. Sharp analysis of low-rank kernel matrix approximations. In *Proceedings of the Twenty Sixth Annual Conference on Computational Learning Theory*, 2013.

[11] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[12] K. Ball. An elementary introduction to modern convex geometry. In S. Levy, editor, *Flavors of Geometry*, pages 1–58. MSRI Publications, 1997.

[13] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 209–218. IEEE, 2002.

[14] B. Barak, M. Braverman, X. Chen, and A. Rao. How to compress interactive communication. *SIAM Journal on Computing*, 42(3):1327–1363, 2013.

[15] P. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *Annals of Statistics*, 33(4):1497–1537, 2005.

[16] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3:463–482, 2003.

[17] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[18] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[19] E. R. Berlekamp. *Algebraic Coding Theory: Revised Edition*. World Scientific, 2015.

[20] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic, 2004.

[21] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.

[22] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[23] R. Bhatia. *Matrix Analysis*. Graduate Texts in Mathematics. Springer-Verlag, New York, NY, 1997.

[24] L. Birgé. Approximation dans les espaces métriques et théorie de l'estimation. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 65(2):181–237, 1983.

[25] L. Birgé and D. V. SARWATE. A new lower bound for multiple hypothesis testing. *IEEE transactions on information theory*, 51(4):1611–1615, 2005.

[26] M. Birman and M. Solomjak. Piecewise-polynomial approximations of functions of the classes W_p^α. *Sbornik: Mathematics*, 2(3):295–317, 1967.

[27] J. A. Blackard, D. J. Dean, and C. W. Anderson. Covertype data set. In K. Bache and M. Lichman, editors, *UCI Machine Learning Repository*, URL: http://archive.ics.uci.edu/ml, 2013. University of California, Irvine, School of Information and Computer Sciences.

[28] C. Blake and C. J. Merz. {UCI} repository of machine learning databases. 1998.

[29] G. Blanchard and N. Krämer. Optimal learning rates for kernel conjugate gradient regression. In *Advances in Neural Information Processing Systems 24*, 2010.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[31] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[32] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[33] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[34] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[35] V. V. Buldygin and Y. V. Kozachenko. *Metric Characterization of Random Variables and Random Processes*. American Mathematical Society, Providence, RI, 2000.

[36] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[37] E. J. Candes and M. A. Davenport. How well can we estimate a sparse vector? *Applied and Computational Harmonic Analysis*, 34(2):317–323, 2013.

[38] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

[39] A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[40] F. Cappello and D. Etiemble. Mpi versus mpi+ openmp on the ibm sp for the nas benchmarks. In *Supercomputing, ACM/IEEE 2000 Conference*, pages 12–12. IEEE, 2000.

[41] A. Chakrabart, Y. Shi, A. Wirth, and A. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 270–278. IEEE, 2001.

[42] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1): 120–145, 2011.

[43] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[44] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pages 1–10. ACM, 2009.

[45] R. Chen, A. Gittens, and J. A. Tropp. The masked sample covariance estimator: an analysis using matrix concentration inequalities. *Information and Inference*, to appear, 2012.

[46] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 571–582, 2014.

[47] J. I. Chu and G. Schnitger. The communication complexity of several problems in matrix computation. *Journal of Complexity*, 7(4):395–407, 1991.

[48] J. I. Chu and G. Schnitger. Communication complexity of matrix computation over finite fields. *Mathematical Systems Theory*, 28(3):215–228, 1995.

[49] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pages 205–214. ACM, 2009.

[50] C. Clenshaw. A note on the summation of Chebyshev series. *Mathematics of Computation*, 9(51):118–120, 1955.

[51] E. P. Consortium et al. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57–74, 2012.

[52] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[53] T. M. Cover and J. A. Thomas. *Elements of Information Theory, Second Edition*. Wiley, 2006.

[54] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.

[55] A. de Acosta. Inequalities for b-valued random vectors with applications to the strong law of large numbers. *The Annals of Probability*, 9:157–161, 1981.

[56] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[57] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

[58] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc., 2014.

[59] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13(1):165–202, 2012.

[60] W. Deng and W. Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. CAAM Technical Report 12-14, Rice University, 2012.

[61] S. Deorowicz and S. Grabowski. Data compression for sequencing data. *Algorithms for Molecular Biology*, 8(1):1, 2013.

[62] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[63] J. C. Duchi and M. J. Wainwright. Distance-based and continuum fano inequalities with applications to statistical estimation. *arXiv [cs.IT]*, to appear, 2013.

[64] J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *Automatic Control, IEEE Transactions on*, 57(3):592–606, 2012.

[65] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[66] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. *arXiv:1302.3203 [math.ST]*, 2013. URL http://arXiv.org/abs/1302.3203.

[67] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[68] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[69] A. El Gamal and Y.-H. Kim. *Network information theory*. Cambridge university press, 2011.

[70] C. Evangelinos and C. Hill. Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2. *ratio*, 2(2.40):2–34, 2008.

[71] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264, 2002.

[72] T. A. S. Foundation. Apache hadoop nextgen mapreduce (yarn).

[73] T. A. S. Foundation. Mahout project. 2012.

[74] S. A. Geer. *Empirical Processes in M-estimation*, volume 6. Cambridge university press, 2000.

[75] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *SIGKDD*, pages 69–77. ACM, 2011.

[76] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, MD, third edition, 1996.

[77] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. JHU Press, 2012.

[78] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[79] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6): 789–828, 1996.

[80] C. Gu. *Smoothing Spline ANOVA Models*. Springer, 2002.

[81] A. Guntuboyina. Lower bounds for the minimax risk using-divergences, and applications. *Information Theory, IEEE Transactions on*, 57(4):2386–2399, 2011.

[82] L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[83] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[84] P. Hall. *The bootstrap and Edgeworth expansion*. Springer Science & Business Media, 2013.

[85] T. S. Han and S.-I. Amari. Statistical inference under multiterminal data compression. *Information Theory, IEEE Transactions on*, 44(6):2300–2324, 1998.

[86] T. Hastie and R. Tibshirani. *Generalized additive models*. Chapman & Hall, 1995.

[87] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27 (2):83–85, 2005.

[88] E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

[89] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing. More effective distributed ML via a stale synchronous parallel parameter server. In *NIPS*, pages 1223–1231, 2013.

[90] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[91] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[92] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[93] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[94] J. Hromkovič. *Communication complexity and parallel computing*. Springer Science & Business Media, 2013.

[95] D. Hsu, S. Kakade, and T. Zhang. Random design analysis of ridge regression. In *Proceedings of the 25nd Annual Conference on Learning Theory*, 2012.

[96] I. A. Ibragimov and R. Z. Has' minskii. *Statistical estimation: asymptotic theory*, volume 16. Springer Science & Business Media, 2013.

[97] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *NIPS*, pages 3068–3076, 2014.

[98] B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2009.

[99] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pages 315–323, 2013.

[100] I. Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005.

[101] R. Kannan, S. S. Vempala, and D. P. Woodruff. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory*, pages 1040–1057, 2014.

[102] N. Karampatziakis and J. Langford. Online importance weight aware updates. *arXiv preprint arXiv:1011.1576*, 2010.

[103] S. S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6:341–361, 2005.

[104] R. Khas' minskii. A lower bound on the risks of non-parametric estimates of densities in the uniform metric. *Theory of Probability & Its Applications*, 23(4):794–798, 1979.

[105] A. N. Kolmogorov and V. M. Tikhomirov. ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Matematicheskikh Nauk*, 14(2):3–86, 1959.

[106] V. Koltchinskii. Local Rademacher complexities and oracle inequalities in risk minimization. *Annals of Statistics*, 34(6):2593–2656, 2006.

[107] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[108] V. M. Krasnopolsky and M. S. Fox-Rabinovitz. Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction. *Neural Networks*, 19(2):122–134, 2006.

[109] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[110] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[111] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[112] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[113] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

[114] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. In *Advances in neural information processing systems*, pages 905–912, 2009.

[115] L. Le Cam. *Asymptotic methods in statistical decision theory*. Springer Science & Business Media, 2012.

[116] M. Ledoux and M. Talagrand. *Probability in Banach Spaces*. Springer, 1991.

[117] M. Ledoux and M. Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

[118] T. Lee and A. Shraibman. *Lower bounds in communication complexity*. Now Publishers Inc, 2009.

[119] E. L. Lehmann and G. Casella. *Theory of Point Estimation, Second Edition*. Springer, 1998.

[120] D. D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[121] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *Proc. OSDI*, pages 583–598, 2014.

[122] Y. Li, X. Sun, C. Wang, and D. P. Woodruff. On the communication complexity of linear algebraic problems in the message passing model. In *Distributed Computing*, pages 499–513. Springer, 2014.

[123] M. W. Libbrecht and W. S. Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332, 2015.

[124] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[125] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin. Large-scale logistic regression and linear support vector machines using Spark. In *Proceedings of the IEEE Conference on Big Data*, Washington DC, USA, 2014.

[126] Q. Lin and L. Xiao. An adaptive accelerated proximal gradient method and its homotopy contiuation for sparse optimization. *Computational Optimization and Applications*, published online, September 2014.

[127] Q. Lin, Z. Lu, and L. Xiao. An accelerated proximal coordinate gradient method and its application to regularized empirical risk minimization. Technical Report MSR-TR-2014-94, Microsoft Research, 2014. arXiv:1407.1296.

[128] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In *WWW*, pages 681–690. ACM, 2010.

[129] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *arXiv preprint arXiv:1311.1873*, 2013.

[130] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

[131] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[132] D. Luenberger. *Optimization by Vector Space Methods*. Wiley, 1969.

[133] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, New York, 1973.

[134] Z.-Q. Luo. Universal decentralized estimation in a bandwidth constrained sensor network. *IEEE Transactions on Information Theory*, 51(6):2210–2219, 2005.

[135] Z.-Q. Luo and J. N. Tsitsiklis. On the communication complexity of distributed algebraic computation. *Journal of the ACM*, 40(5):1019–1047, 1993.

[136] Z.-Q. Luo and J. N. Tsitsiklis. Data fusion with minimal communication. *Information Theory, IEEE Transactions on*, 40(5):1551–1563, 1994.

[137] L. Mackey, M. I. Jordan, R. Y. Chen, B. Farrell, J. A. Tropp, et al. Matrix concentration inequalities via the method of exchangeable pairs. *The Annals of Probability*, 42 (3):906–945, 2014.

[138] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

[139] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[140] R. P. Martin, A. M. Vahdat, D. E. Culler, and T. E. Anderson. *Effects of communication latency, overhead, and bandwidth in a cluster architecture*, volume 25. ACM, 1997.

[141] J. C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. CRC Press, 2010.

[142] R. Mcdonald, M. Mohri, N. Silberman, D. Walker, and G. S. Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.

[143] R. McDonald, K. Hall, and G. Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics, 2010.

[144] S. Mendelson. Geometric parameters of kernel machines. In *Proceedings of the Fifteenth Annual Conference on Computational Learning Theory*, pages 29–43, 2002.

[145] S. Mendelson. Improving the sample complexity using global data. *Information Theory, IEEE Transactions on*, 48(7):1977–1991, 2002.

[146] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *arXiv preprint arXiv:1505.06807*, 2015.

[147] T. P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[148] P. Moritz, R. Nishihara, I. Stoica, and M. I. Jordan. Sparknet: Training deep networks in spark. *arXiv preprint arXiv:1511.06051*, 2015.

[149] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi. Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 439–455. ACM, 2013.

[150] E. Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *arXiv:1308.4275*, 2013.

[151] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1):48–61, 2009.

[152] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[153] A. Nemirovsky and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. J. Wiley & Sons, New York, 1983.

[154] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

[155] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[156] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[157] Y. Nesterov and A. Nemirovski. *Interior Point Polynomial Time Methods in Convex Programming*. SIAM, Philadelphia, 1994.

[158] D. Newman, P. Smyth, M. Welling, and A. U. Asuncion. Distributed inference for latent Dirichlet allocation. In *NIPS*, pages 1081–1088, 2007.

[159] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[160] A. Olshevsky and J. N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48(1):33–55, 2009.

[161] D. N. Politis, J. P. Romano, and M. Wolf. *Subsampling*. Springer, 1999.

[162] E. Polizzi. Density-matrix-based algorithm for solving eigenvalue problems. *Physical Review B*, 79(11):115112, 2009.

[163] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[164] R. Power and J. Li. Piccolo: Building fast, distributed programs with partitioned tables. In *OSDI*, volume 10, pages 1–14, 2010.

[165] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM, 2005.

[166] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, 2007.

[167] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

[168] S. S. Ram, A. Nedić, and V. V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.

[169] G. Raskutti, M. Wainwright, and B. Yu. Early stopping for non-parametric regression: An optimal data-dependent stopping rule. In *49th Annual Allerton Conference on Communication, Control, and Computing*, pages 1318–1325, 2011.

[170] G. Raskutti, M. J. Wainwright, and B. Yu. Minimax rates of estimation for high-dimensional linear regression over-balls. *Information Theory, IEEE Transactions on*, 57(10):6976–6994, 2011.

[171] G. Raskutti, M. J. Wainwright, and B. Yu. Minimax-optimal rates for sparse additive models over kernel classes via convex programming. *Journal of Machine Learning Research*, 12:389–427, March 2012.

[172] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[173] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

[174] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[175] N. L. Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pages 2672–2680. 2012.

[176] T. Sakurai and H. Sugiura. A projection method for generalized eigenvalue problems using numerical integration. *Journal of Computational and Applied Mathematics*, 159 (1):119–128, 2003.

[177] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 285–295. ACM, 2001.

[178] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 1–7. ACM, 1996.

[179] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998.

[180] M. Schmidt, N. L. Roux, and F. R. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pages 1458–1466, 2011.

[181] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[182] G. Schofield, J. R. Chelikowsky, and Y. Saad. A spectrum slicing method for the Kohn–Sham problem. *Computer Physics Communications*, 183(3):497–505, 2012.

[183] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[184] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. arXiv:1309.2375.

[185] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *arXiv preprint arXiv:1209.1873*, 2012.

[186] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[187] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Stochastic convex optimization. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, 2009.

[188] O. Shamir and N. Srebro. On distributed stochastic optimization and learning. In *Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing*, 2014.

[189] O. Shamir, N. Srebro, and T. Zhang. Communication efficient distributed optimization using an approximate newton-type method. *arXiv preprint arXiv:1312.7853*, 2013.

[190] O. Shamir, N. Srebro, and T. Zhang. Communication efficient distributed optimization using an approximate Newton-type method. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*. JMLR: W&CP volume 32, 2014.

[191] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. MPS-SIAM Series on Optimization. SIAM-MPS, Philadelphia, PA, 2009.

[192] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[193] J. Shiers. The worldwide lhc computing grid (worldwide lcg). *Computer physics communications*, 177(1):219–223, 2007.

[194] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[195] Y. Singer and J. C. Duchi. Efficient learning using forward-backward splitting. In *Advances in Neural Information Processing Systems*, pages 495–503, 2009.

[196] E. R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M. J. Franklin, M. Jordan, T. Kraska, et al. Mli: An api for distributed machine learning. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1187–1192. IEEE, 2013.

[197] I. Steinwart, D. Hush, and C. Scovel. Optimal rates for regularized least squares regression. In *Proceedings of the 22nd Annual Conference on Learning Theory*, pages 79–93, 2009.

[198] C. J. Stone. Optimal global rates of convergence for non-parametric regression. *Annals of Statistics*, 10(4):1040–1053, 1982.

[199] G. Sun. KDD cup track 2 `soso.com` ads prediction challenge, 2012. URL http://www.kddcup2012.org/c/kddcup2012-track2. Accessed August 1, 2012.

[200] X. Sun and C. Wang. Randomized communication complexity for linear algebra problems over finite fields. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[201] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[202] M. Talagrand. A new look at independence. *The Annals of probability*, pages 1–34, 1996.

[203] J. N. Tsitsiklis and Z.-Q. Luo. Communication complexity of convex optimization. In *Decision and Control, 1986 25th IEEE Conference on*, pages 608–611. IEEE, 1986.

[204] J. N. Tsitsiklis et al. Decentralized detection. *Advances in Statistical Signal Processing*, 2(2):297–344, 1993.

[205] A. B. Tsybakov. Introduction to nonparametric estimation. revised and extended from the 2004 french original. translated by vladimir zaiats, 2009.

[206] A. W. van der Vaart. *Asymptotic Statistics*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998. ISBN 0-521-49603-9.

[207] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of Complexity*, pages 11–30. Springer, 2015.

[208] V. N. Vapnik and V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[209] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103. ACM, 2008.

[210] G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, PN, 1990.

[211] L. Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

[212] S. Whiteson and D. Whiteson. Machine learning for event selection in high energy physics. *Engineering Applications of Artificial Intelligence*, 22(8):1203–1217, 2009.

[213] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161322, pages 682–688, 2001.

[214] D. P. Woodruff and Q. Zhang. An optimal lower bound for distinct elements in the message passing model. In *Symposium on Discrete Algorithms*, pages 718–733. SIAM, 2014.

[215] D. Wyatt. *Akka concurrency*. Artima Incorporation, 2013.

[216] L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *NIPS*, pages 2116–2124, 2009.

[217] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. Petuum: A new platform for distributed machine learning on big data. *arXiv preprint arXiv:1312.7651*, 2013.

[218] Z. Xu and K. Hwang. Modeling communication overhead: Mpi and mpl performance on the ibm sp2. *Parallel & Distributed Technology: Systems & Applications, IEEE*, 4 (1):9–24, 1996.

[219] Y. Yang and A. Barron. Information-theoretic determination of minimax rates of convergence. *Annals of Statistics*, pages 1564–1599, 1999.

[220] Y. Yang, M. Pilanci, and M. J. Wainwright. Randomized sketches for kernels: Fast and optimal non-parametric regression. *arXiv:1501.06195 [stat.ml]*, 2015.

[221] A. C.-C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 209–213. ACM, 1979.

[222] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[223] B. Yu. Assouad, fano, and le cam. In *Festschrift for Lucien Le Cam*, pages 423–435. Springer, 1997.

[224] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*. USENIX Association, 2012.

[225] T. Zhang. Leave-one-out bounds for kernel methods. *Neural Computation*, 15(6): 1397–1437, 2003.

[226] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML*, page 116. ACM, 2004.

[227] T. Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.

[228] Y. Zhang and M. I. Jordan. Splash: User-friendly programming interface for parallelizing stochastic algorithms. *arXiv preprint arXiv:1506.07552*, 2015.

[229] Y. Zhang and L. Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. Technical Report MSR-TR-2014-123, Microsoft Research, 2014. arXiv:1409.3257.

[230] Y. Zhang and L. Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv preprint arXiv:1501.00263*, 2015.

[231] Y. Zhang, M. J. Wainwright, and J. C. Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

[232] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013.

[233] Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, pages 592–617, 2013.

[234] Y. Zhang, M. J. Wainwright, and M. I. Jordan. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. *arXiv preprint arXiv:1502.01403*, 2015.

[235] H. Zhao, B. Jiang, and J. Canny. Same but different: Fast and high-quality gibbs parameter estimation. *arXiv preprint arXiv:1409.5402*, 2014.

[236] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin. A fast parallel SGD for matrix factorization in shared memory systems. In *RecSys*, pages 249–256. ACM, 2013.

[237] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin. Distributed newton method for regularized logistic regression. Technical report, Department of Computer Science, National Taiwan University, 2014.

[238] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.