Improving the Bit Complexity of Communication for Distributed Convex Optimization

Mehrdad Ghadiri
Massachusetts Institute of Technology
Cambridge, USA
mehrdadg@mit.edu

Yin Tat Lee
University of Washington
Seattle, USA
Microsoft Research
Redmond, USA
yintat@uw.edu

Swati Padmanabhan
Massachusetts Institute of Technology
Cambridge, USA
pswt@mit.edu

William Swartworth
Carnegie Mellon University
Pittsburgh, USA
wswartwo@andrew.cmu.edu

David P. Woodruff
Carnegie Mellon University
Pittsburgh, USA
dwoodruf@cs.cmu.edu

Guanghao Ye
Massachusetts Institute of Technology
Cambridge, USA
ghye@mit.edu

ABSTRACT

We consider the communication complexity of some fundamental convex optimization problems in the point-to-point (coordinator) and blackboard communication models. We strengthen known bounds for approximately solving linear regression, p-norm regression (for $1 \leq p \leq 2$), linear programming, minimizing the sum of finitely many convex nonsmooth functions with varying supports, and low rank approximation; for a number of these fundamental problems our bounds are optimal, as proven by our lower bounds.

For example, for solving least squares regression in the coordinator model with s servers, n examples, d dimensions, and coefficients specified using at most L bits, we improve the prior communication bound of Vempala, Wang, and Woodruff (SODA, 2020) from $\tilde{O}(sd^2L)$ to $\tilde{O}(sdL + d^2\varepsilon^{-1}L)$, which is optimal up to logarithmic factors. We also study the problem of solving least squares regression in the coordinator model to *high accuracy*, for which we provide an algorithm with a communication complexity of $\widetilde{O}(sd(L + \log\kappa)\log(\varepsilon^{-1}) + d^2L)$, matching our improved lower bound for well-conditioned matrices up to a $\log(\varepsilon^{-1})$ factor. Among our techniques, we use the notion of *block leverage scores*, which have been relatively unexplored in this context, as well as dropping all but the "middle" bits in Richardson-style algorithms. We also introduce a new communication problem for accurately approximating inner products and establish a lower bound using the spherical Radon transform. Our lower bound can be used to show the first separation of linear programming and linear systems in the distributed model when the number of constraints is polynomial, addressing an open question in prior work.

We also give an improved algorithm for high-accuracy linear programming in the coordinator model that computes an approximate solution on well-conditioned inputs using $\tilde{O}(sd^{1.5}L + d^2L)$ communication. This improves over the previous bound of sd^2L. Finally, we give an improved algorithm, in the blackboard model of communication, for the problem $\min_{\theta\in\mathbb{R}^d}\sum_{i=1}^{s} f_i(\theta)$ where each f_i is convex, Lipschitz, and supported on $d_i \leq d$ (potentially overlapping) coordinates of θ using $\tilde{O}(\sum_{i=1}^{s} d_i^2 L)$ communication. Our techniques yield improved rates for decomposable submodular function minimization in the non-distributed setting as well.

CCS CONCEPTS

• **Mathematics of computing** → **Continuous optimization**; • **Theory of computation** → **Design and analysis of algorithms**; • **Computing methodologies** → **Symbolic and algebraic algorithms**.

KEYWORDS

bit complexity, communication complexity, distributed optimization, convex optimization

ACM Reference Format:
Mehrdad Ghadiri, Yin Tat Lee, Swati Padmanabhan, William Swartworth, David P. Woodruff, and Guanghao Ye. 2024. Improving the Bit Complexity of Communication for Distributed Convex Optimization. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3618260.3649787

1 INTRODUCTION

The scale of modern optimization problems often necessitates working with datasets that are distributed across multiple machines, which then communicate with each other to solve the optimization problem at hand. A crucial performance metric for algorithms in such distributed settings is the communication complexity. Traditionally, this has referred to the *number of rounds* of communication needed between the machines to solve the problem, and there has been a long line of work (which we shortly describe) optimizing this

*The ordering of authors is alphabetical. The full version of the paper is available at https://arxiv.org/abs/2403.19146

STOC '24, June 24–28, 2024, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649787

metric. However, as was highlighted in [31, 59, 76], in many core algorithmic primitives underlying recent advances in continuous optimization, the claimed (theoretical) runtimes are predicated on the assumption of exact computations with infinite precision. When analyzed under the finite-precision model, the true runtimes can be substantially higher. Consequently, inferring the true cost of distributed optimization algorithms built with these components requires a careful analysis. To address this need, our focus in this paper is on designing, for some fundamental optimization problems, distributed algorithms efficient in the *total number of bits communicated*.

Before describing our setup and results, we first provide a brief overview of prior work in the related area of distributed optimization, a mature field encompassing problems spanning engineering, control theory, signal processing, and machine learning. For instance, multi-agent coordination, distributed tracking and localization, estimation problems in sensor networks, opinion dynamics, and packet routing are all naturally cast as distributed convex minimization [11, 45, 68]. Classically, the primary goal in these problems was to design a communication strategy between the computational agents so that they eventually arrive at the optimal objective value [73]. A considerable body of work [33, 54, 72, 78] has therefore been devoted to obtaining asymptotic convergence guarantees for these problem classes. Going beyond asymptotic analysis, recent years have witnessed extensive progress in obtaining *non-asymptotic rates* (typically in terms of the number of rounds of communication) for problems in distributed machine learning such as distributed PAC learning [9], distributed online prediction [22], distributed estimation [26, 35, 54], and distributed delayed stochastic optimization [2, 53].

A related paradigm that has recently emerged in distributed computing is that of *federated learning* [37]. In this paradigm, the processes of data acquisition, processing, and model training are largely carried out on a network's edge nodes such as smartphones [13], wearables [32], location-based services [67], and IoT sensors [40, 52], under the orchestration of a central coordinator. Similar to the recent works on distributed machine learning mentioned in the preceding paragraph, for the works in this setting as well, it is the number of rounds of communication that is typically used as a proxy for total communication cost. Additional important concerns for works in federated learning include user privacy and robustness to distribution shifts in users' samples [63] and to heterogeneity in the computational capabilities of the nodes [64]. Finally, while our focus in this paper is the theory, we note that advances in the practice of distributed computing have been tremendously spurred by the development of programming models like MapReduce [21], which enable parallelizing the computation, distributing the data, and handling failures across thousands of machines.

Our setup. As mentioned earlier, only recently has there been a surge of interest in studying the bit complexity of optimization algorithms [31, 59, 76]. In this paper, we hope to continue pushing efforts in this direction and study the number of bits communicated to solve various *distributed* convex optimization problems under two models of communication, defined next. Our goal is to compute approximate solutions with efficient communication complexity.

Definition 1.1 (Coordinator Model)**.** *There are s machines (servers) and a central coordinator. Each machine can send information to and receive information from the coordinator. Any bit communicated with the coordinator counts toward the communication complexity of the algorithm.*

Definition 1.2 (Blackboard Model)**.** *There are s machines and a coordinator (blackboard). Each machine can send information to and receive information from the coordinator. Only bits sent to the coordinator count toward the communication complexity of the algorithm.*

The coordinator model is equivalent, up to a factor of two and an additive $\log s$ bits per message, to the *point-to-point model of computation*, in which machines directly interact with each other. The blackboard model may be viewed as having a shared memory between the machines, since it costs the machines only to write on to the blackboard, while reading from the blackboard is free.

We consider several fundamental optimization problems that have been studied extensively outside the distributed setting: least squares regression, low rank approximation, linear programming, and optimizing a sum of convex nonsmooth functions. We provide improved communication upper and lower bounds for these problems in the aforementioned distributed settings. While we obtain nearly tight upper and lower bounds for several of these problems in the "worst-case" settings, e.g., when matrices are arbitrarily poorly conditioned, another important component of our work is in improving bounds for well-behaved inputs, e.g., well-conditioned matrices or decomposable functions.

1.1 Our Contributions

In this paper, we address the communication complexity of least squares regression, low-rank approximation, and linear programming in the coordinator model, and finite-sum minimization of Lipschitz functions in the blackboard model. Our central technical novelty lies in developing efficient — in terms of bit complexity — methods for leverage score sampling, inverse maintenance, cutting-plane methods, and the use of block leverage scores in the distributed setting and in finite arithmetic. We summarize all formal statements in this section.

General Setup. In all problems, we consider a matrix that is divided among s servers as per the *row-partition* model. This is in contrast to the arbitrary partition model, in which each server holds a matrix $\mathbf{A}^{(i)}$, with $\mathbf{A} = \sum_{i\in[s]} \mathbf{A}^{(i)}$. In our model, the i^{th} machine stores a matrix $\mathbf{A}^{(i)} \in \mathbb{R}^{n_i\times d}$, and our problem matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, with $n = \sum_{i=1}^{s} n_i$, is formed by vertically stacking all the $\mathbf{A}^{(i)}$ matrices, i.e., $\mathbf{A} = [\mathbf{A}^{(i)}]$.

For least squares regression and linear programming, each server additionally holds a vector $\mathbf{b}^{(i)} \in \mathbb{R}^{n_i}$ whose vertical concatenation we denote by $\mathbb{R}^n \ni \mathbf{b} = [\mathbf{b}^{(i)}]$, with $n = \sum_{i=1}^{s} n_i$. When considering linear programming and finite-sum minimization, the vector $\mathbf{c}$ (where $\mathbf{c}$ is the vector that appears in the objective obtained by reducing the original finite-sum minimization using an epigraph trick) is also shared between the machines (or can be shared with $O(sd)$ communication). We explicitly describe the setup for each problem in its corresponding section.

We assume that the entries of $\mathbf{A}^{(i)}$ and $\mathbf{b}^{(i)}$ can be represented with L bits. We often model this by assuming that all entries are integers in $\{-2^L+1, \ldots, 2^L\}$. Sometimes it will be more convenient to work with normalized vectors and matrices, in which case we

allow entries to be of the form $c\,2^{-L}$ with $c \in \{-2^L+1, \ldots, 2^L\}$. We say that such numbers are expressed to L bits of precision.

1.1.1 Least Squares Regression, ℓ_p Regression, and Low-Rank Approximation. In many large-scale machine learning applications, one is faced with a large, potentially noisy regression problem for which a constant factor approximation is acceptable. Specifically, we are interested in computing an approximate solution $\widehat{\mathbf{x}}$ satisfying, for a given constant ε, the bound

$$\|\mathbf{A}\widehat{\mathbf{x}} - \mathbf{b}\|_2 \leq (1+\varepsilon) \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2. \tag{1.1}$$

We formalize our setup below.

Problem 1.3 (Setup in the Coordinator Model)**.** *Suppose there is a coordinator and s machines that communicate with each other as per the coordinator model of communication (Definition 1.1) with shared randomness. Suppose each machine $i \in [s]$ holds a matrix $\mathbf{A}^{(i)} \in \mathbb{R}^{n_i \times d}$ and a vector $\mathbf{b}^{(i)} \in \mathbb{R}^{n_i}$. Denote $\mathbf{A} = [\mathbf{A}^{(i)}] \in \mathbb{R}^{n \times d}$ and $\mathbf{b} = [\mathbf{b}^{(i)}] \in \mathbb{R}^n$, both represented with L bits in fixed-point arithmetic. Moreover, suppose the condition number of $\mathbf{A}$ is bounded by κ*[1]*.*

For least squares regression in this model, [76] gave upper and lower bounds of $\widetilde{O}(sd^2L)$ and $\Omega(sd+d^2L)$, respectively. Their upper bound comes from sending $(\mathbf{A}^{(i)})^\top \mathbf{A}^{(i)}$'s and $(\mathbf{A}^{(i)})^\top \mathbf{b}^{(i)}$'s to the coordinator which then computes the exact solution by the normal equations. On the other hand, they show that consistent[2] linear systems can be solved exactly using only $\widetilde{O}(sd + d^2L)$ communication. Furthermore, for consistent systems, the optimal regression error is zero, and so a regression algorithm must output the precise solution. Least squares regression is, therefore, certainly as hard as solving consistent linear systems. This motivates the following question: *Is solving least squares regression to constant accuracy harder than solving a consistent linear system?*

Our key (and surprising) takeaway message for this setting is that for constant L, *regression is no harder than solving linear systems*. Specifically, we give a protocol which, for any constant $\varepsilon > 0$, achieves $\widetilde{O}(sdL + d^2L)$ bits of communication for least squares regression, thus improving upon [76]'s $\widetilde{O}(sd^2L)$ upper bound and matching its lower bound of $\Omega(sd + d^2L)$ for constant L. Our upper bound also gives the first separation for least squares regression between the row-partition model and the arbitrary partition model, for which [46] showed an $\Omega(sd^2)$ lower bound.

Theorem 1.4 (ℓ_2 Regression in the Coordinator Model). *Given $\varepsilon > 0$ and a least squares regression problem in the setup of Problem 1.3 with input matrix $\mathbf{A} = [\mathbf{A}^{(i)}] \in \mathbb{R}^{n \times d}$ and vector $\mathbf{b} = [\mathbf{b}^{(i)}] \in \mathbb{R}^n$, there is a randomized protocol that allows the coordinator to solve the least squares regression problem with constant probability and relative error $(1 \pm \varepsilon)$ using*

$$\widetilde{O}\left(sdL + d^2\varepsilon^{-1}L\right) \text{ bits of communication.}$$

Additionally, if κ is a known upper bound on the condition number of $\mathbf{A}$ then there is a protocol using $\widetilde{O}(sd\log\kappa + d^2\varepsilon^{-1}L)$ communication.

[1]Note that not all of our bounds depend on κ.
[2]The system $(\mathbf{A}, \mathbf{b})$ is consistent if for some $\mathbf{x}$, we have $\mathbf{A}\mathbf{x} = \mathbf{b}$.

If L is not constant, there still remains a gap between the above bound and [76]'s lower bound of $\widetilde{\Omega}(sd + d^2L)$. By proving an improved $\widetilde{\Omega}(sdL)$ lower bound for ℓ_2 regression under a mild restriction on the number of rounds of the protocol, we close this gap (cf. Section 1.2.5).

Our upper bound from Theorem 1.4 extends to ℓ_p regression for $1 \leq p < 2$, as captured by Theorem 1.5. Notably, our protocols for regression have a small $\widetilde{O}(1)$ number of rounds of communication, with no dependence on the condition number of $\mathbf{A}$.

Theorem 1.5 (ℓ_p Regression for $1 \leq p < 2$ in the Coordinator Model). *For the setup described in Problem 1.3, there exists a randomized protocol that, with a probability of at least $1-\delta$, allows the coordinator to produce an ε-distortion ℓ_p subspace embedding for the column span of $\mathbf{A}$ using only*

$$\widetilde{O}\left((sdL + d^2\varepsilon^{-4}L)\log(\delta^{-1})\right) \text{ bits of communication.}$$

As a result, the coordinator can solve ℓ_p regression (for $1 \leq p < 2$) with the same communication.

While the focus of our work for regression has been on the coordinator model (Theorem 1.4 and Theorem 1.5), we note that [76] already provide optimal communication cost algorithms for constant-accuracy regression in the *blackboard model*, as remarked below.

Remark 1.6. *For constant-accuracy ℓ_1 and ℓ_2 regression in the blackboard model, [76] provides optimal algorithms with communication cost $\widetilde{O}(s + d^2L)$.*

Low Rank Approximation. As an application of our aforementioned least squares regression techniques, we obtain improved bounds for low-rank approximation in the distributed setting, a problem several prior works [12, 14, 29, 38] have considered. Notably, [14] studied the variant of the problem wherein the rows[3] of $\mathbf{A}$ are partitioned among s servers, and all servers must learn a projection Π that yields an approximately optimal Frobenius-norm error:

$$\|\mathbf{A}\Pi - \mathbf{A}\|_{\mathrm{F}} \leq (1+\varepsilon)\|\mathbf{A}_k - \mathbf{A}\|_{\mathrm{F}}, \tag{1.2}$$

where $\mathbf{A}_k$ is the best rank-k approximation of $\mathbf{A}$. In this setting, [14] provide an upper bound of $O(skdL)$ for constant ε, along with a nearly matching lower bound of $\Omega(skd)$. However, their lower bound crucially requires *all* servers to learn the projection. A natural question we answer is if relaxing this constraint could yield a better communication complexity. In other words: *Is it possible to do better when only the coordinator needs to learn the projection?*

Theorem 1.7 (Low-Rank Approximation in the Coordinator Model). *For the setup described in Problem 1.3, suppose that the s servers have shared randomness. Then there is a randomized protocol using*

$$\widetilde{O}\left(kL \cdot (d\varepsilon^{-2} + s\varepsilon^{-1})\right) \text{ bits of communication,}$$

that with constant probability lets the coordinator produce a rank-k orthogonal projection $\Pi \in \mathbb{R}^{d \times k}$ (where $k \leq d$) satisfying Inequality (1.2).

[3]Their matrices are transposed relative to ours, so their columns are partitioned among servers.

1.1.2 High-Accuracy Least Squares Regression. Complementing our constant-factor regression results in the previous paragraphs, we study *regression solved to machine precision*. We show that when the matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ has a small condition number (i.e., $\text{poly}(d)$), we obtain an $\widetilde{O}(sdL + d^2L\log(\varepsilon^{-1}))$ communication complexity of solving least squares regression to high accuracy. Specifically, we obtain the following result.

Theorem 1.8 (High-Accuracy ℓ_2 Regression in the Coordinator Model). *Given $\varepsilon > 0$ and the least squares regression setting of Problem 1.3 with input matrix $\mathbf{A} = [\mathbf{A}^{(i)}] \in \mathbb{R}^{n\times d}$ and vector $\mathbf{b} = [\mathbf{b}^{(i)}] \in \mathbb{R}^n$, there is a randomized algorithm that, with high probability, outputs a vector $\widehat{\mathbf{x}}$ such that*

$$\|\mathbf{A}\widehat{\mathbf{x}} - \mathbf{b}\|_2 \le \varepsilon \cdot \|\mathbf{A}(\mathbf{A}^\top\mathbf{A})^\dagger\mathbf{A}^\top\mathbf{b}\|_2 + \min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2. \tag{1.3}$$

Let κ be the condition number of $\mathbf{A}$. Then the algorithm uses

$$\widetilde{O}(sd(L + \log\kappa)\log(\varepsilon^{-1}) + d^2L) \text{ bits of communication.}$$

Moreover, the vector $\widehat{\mathbf{x}}$ is available on all the machines at the end of the algorithm.

This result improves upon the $\widetilde{O}(sd^2L)$ bound of [76] (which also gives an associated lower bound of $\widetilde{\Omega}(sd + d^2L)$). The error guarantee of Theorem 1.8 is different than that of Theorem 1.4 in two ways. First, the error is additive in Theorem 1.8 instead of multiplicative. The main reason is that the solution produced by the algorithm of Theorem 1.8 is available to all the machines instead of only being available only to the coordinator. This is needed when we use this result for each iteration of linear programming (Theorem 1.10). We note that the solution produced by the algorithm of Theorem 1.4 can be shared among all the machines, but we would need to use the rational number representation to share it, and the communication cost would increase significantly in this case. The second difference is that the dependence of the running time on the error parameter ε is logarithmic in Theorem 1.8. This allows us to achieve high-accuracy solutions, which are again needed for linear programming results to deal with adaptive adversary issues.

Our improvement is achieved by a novel rounding procedure for Richardson's iteration with preconditioning and has consequences outside the distributed setting as well. In particular, it implies an improvement for the bit complexity of solving a least squares regression problem (with an input that has constant bit complexity) from $\widetilde{O}((d^\omega + (\text{nnz}(\mathbf{A}) + d^2)\cdot\log^2(\varepsilon^{-1}))\cdot\log\kappa)$ [31] to $\widetilde{O}((d^\omega + (\text{nnz}(\mathbf{A}) + d^2)\cdot\log(\varepsilon^{-1}))\cdot\log\kappa)$, where $\text{nnz}(\mathbf{A})$ is the number of nonzero entries of $\mathbf{A}$.

Remark 1.9. *While our result in Theorem 1.8 operates only in the coordinator model, [76] studies this problem in the blackboard model as well. In particular, for ℓ_2 regression in the blackboard model with general accuracy parameter ε, [76] provides an algorithm with communication cost $\widetilde{O}(s + d^2L\varepsilon^{-1})$, with an associated lower bound of $\widetilde{\Omega}(s + d\varepsilon^{-1/2} + d^2L)$ for $s \ge \Omega(\varepsilon^{-1/2})$.*

1.1.3 High-Accuracy Linear Programming. A core technical component in achieving the results of Section 1.1.2 is the communication-efficient computation of a spectral approximation of a matrix via its intimate connection to its approximate leverage scores. We utilize this idea to develop communication-efficient high-accuracy linear programming too, as we describe next.

The work of [76] studied this problem and gave an upper bound of $\widetilde{O}(sd^3L + d^4L)$ by implementing Clarkson's algorithm [18] in the coordinator model. To obtain this bound, [76] first note that following the analysis of the original algorithm in [18], the total number of rounds of communication is $O(d\log d)$. In each round, the coordinator sends to all the s servers a vector $\mathbf{x}_R$, which is an optimal solution to the linear program $\mathbf{A}\mathbf{x} \le \mathbf{b}$. By polyhedral theory, there exists a non-singular subsystem $\mathbf{B}\mathbf{x} \le \mathbf{c}$, such that $\mathbf{x}_R$ is the unique solution of $\mathbf{B}\mathbf{x} = \mathbf{c}$. By Cramer's rule, each of d entries of $\mathbf{x}$ is a ratio of integers between $-d!2^{dL}$ and $d!2^{dL}$ and can therefore be represented in $\widetilde{O}(dL)$ bits. Multiplying all these quantities yields the claimed communication complexity.

We take a different approach and improve upon [76]'s above bound of $\widetilde{O}(sd^3L + d^4L)$ to $\widetilde{O}(sd^{1.5}L + d^2L)$. Our improvement is achieved by essentially adapting to the distributed setting recent advances in interior point methods for solving linear programs [42, 43, 75], with the associated toolkit of a weighted central path approach, efficient inverse maintenance, and data structures for efficient matrix-vector operations. Our rate holds for linear programs that have a small outer radius and a well-conditioned constraint matrix $\mathbf{A}$, as we formalize next.

Theorem 1.10 (Linear Programming in the Coordinator Model). *Given $\varepsilon > 0$, input matrix $\mathbf{A} = [\mathbf{A}^{(i)}] \in \mathbb{R}^{n\times d}$, and vectors $\mathbf{c} = [\mathbf{c}^{(i)}] \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^d$ in the setup of Problem 1.3, there is a randomized algorithm that, with high probability, outputs a vector $\widehat{\mathbf{x}} \in \mathbb{R}^n$ such that*

$$\|\mathbf{A}^\top\widehat{\mathbf{x}} - \mathbf{b}\|_2 \le \varepsilon\cdot(\|\mathbf{A}\|_{\mathrm{F}}\cdot R + \|\mathbf{b}\|_2)$$

and

$$\mathbf{c}^\top\widehat{\mathbf{x}} \le \min_{\mathbf{x}:\mathbf{A}^\top\mathbf{x}=\mathbf{b},\mathbf{x}\ge 0} \mathbf{c}^\top\mathbf{x} + \varepsilon\cdot\|\mathbf{c}\|_2\cdot R, \tag{1.4}$$

where R is the linear program's outer radius, i.e., $\|\mathbf{x}\|_2 \le R$ for all feasible $\mathbf{x}$. The algorithm uses

$$\widetilde{O}((sd^{1.5}(L + \log(\kappa R r^{-1}\varepsilon^{-1})) + d^2L\log(\varepsilon^{-1}))\cdot\log(\varepsilon^{-1}))$$

bits of communication, where κ is the condition number of $\mathbf{A}$, and r is the linear program's inner radius, i.e., there exists a feasible $\mathbf{x}$ with $\mathbf{x}_i \ge r$ for all $i \in [n]$. Moreover, the vector $\widehat{\mathbf{x}}$ is available on all the machines at the end of the algorithm.

As a special case of our Theorem 1.10, when our linear program has parameters κ, R, and ε^{-1} of the scale $\text{poly}(d)$, we obtain a communication complexity of $\widetilde{O}(sd^{1.5}L + d^2L)$, which is also an improvement over [76]'s previous bound.

Remark 1.11. *While we study linear programming only in the coordinator model, [76] studies this in the blackboard model as well. Specifically, in constant dimensions, [76] provides a randomized communication complexity of $\widetilde{\Omega}(s + L)$ for linear programming in the blackboard model.*

1.1.4 Finite-Sum Minimization with Varying Supports. Another problem class naturally amenable to study in the distributed setting is that of finite-sum minimization. We consider, in the blackboard model, the problem $\min_{\mathbf{x}}\sum_{i=1}^s f_i(\mathbf{x})$ where each $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ is μ-Lipschitz, convex, nonsmooth, and supported on (potentially overlapping) d_i coordinates. We call this problem "decomposable

nonsmooth convex optimization". The assumption of varying supports appears prominently in decomposable submodular function minimization [8, 61] and was recently studied by [25]. This problem, without this assumption, has seen extensive progress in variants of stochastic gradient descent (cf. Section 1.2.4). We formalize below the problem setup in the blackboard model.

Problem 1.12 (Decomposable Nonsmooth Convex Optimization Setup)**.** *Suppose there is a blackboard/coordinator and s machines that communicate with each other as per the blackboard model of communication (Definition 1.2). Suppose each machine $i \in [s]$ holds an oracle O_i that returns a subgradient (represented with L bits in fixed-point arithmetic) of the function $f_i : \mathbb{R}^d \mapsto \mathbb{R}$.*

Directly adapting the algorithm of [25] to the above model yields a communication cost of $\widetilde{O}(\max_{j\in[s]} d_j L \sum_{i=1}^s d_i)$. In this work, we improve this cost to $\widetilde{O}(\sum_{i=1}^s d_i^2 L)$, as formalized next.

Theorem 1.13 (Distributed Decomposable Nonsmooth Convex Optimization). *Given $\varepsilon > 0$ and the setup of Problem 1.12, consider the problem $\min_\theta \sum_{i=1}^s f_i(\theta)$, where each $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ is convex, μ-Lipschitz, and dependent on d_i coordinates of θ. Define $\theta^\star := \arg\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^s f_i(\theta)$. Suppose further that we know an initial $\theta^{(0)} \in \mathbb{R}^d$ such that $\|\theta^\star - \theta^{(0)}\|_2 \le D$. Then, there is an algorithm that outputs a vector $\theta \in \mathbb{R}^d$ such that*

$$\sum_{i=1}^s f_i(\theta) \le \sum_{i=1}^s f_i(\theta^\star) + \varepsilon \cdot \mu D.$$

Our algorithm uses

$$O\left(\sum_{i=1}^s d_i^2 \log(sd\varepsilon^{-1}) \cdot L\right) \text{ bits of communication,}$$

where $L = O(\log d)$ is the word length. At the end of our algorithm, all servers hold this solution.

Our technical novelty — modifying the analysis and slightly modifying the algorithm of [25] — yields an improvement in not just the distributed setting but also in the (non-distributed) setting [25] studied this problem in. Specifically, as a corollary (Theorem 1.14), we improve the total oracle cost of decomposable nonsmooth convex optimization from $\widetilde{O}(O_{\max} \cdot \sum_{i=1}^s d_i)$ to $\widetilde{O}(\sum_{i=1}^s O_i \cdot d_i)$, where O_i is the cost of invoking the i^{th} separation oracle, and $O_{\max}$ is the maximum of all O_i.

Theorem 1.14 (Solving Problem 1.12.). *Consider the problem $\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^s f_i(\theta)$ with each $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ convex, μ-Lipschitz, possibly non-smooth functions, depending on d_i coordinates of θ, and accessible via a (sub-)gradient oracle. Define the minimizer $\theta^\star := \arg\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^s f_i(\theta)$. Suppose we are given a vector $\theta^{(0)} \in \mathbb{R}^d$ such that $\|\theta^\star - \theta^{(0)}\|_2 \le D$. Then, given a weight vector $\mathbf{w} \in \mathbb{R}^s_{\ge 1}$ with which we define $m \stackrel{\text{def}}{=} \sum_{i\in[s]} w_i d_i$, there is an algorithm that, in time $\text{poly}(m \log(\varepsilon^{-1}))$, outputs a vector $\theta \in \mathbb{R}^d$ such that*

$$\sum_{i=1}^s f_i(\theta) \le \sum_{i=1}^s f_i(\theta^\star) + \varepsilon \cdot \mu D.$$

Moreover, let n_i be the number of subgradient oracle calls to f_i. Then, the algorithm's total oracle cost is

$$\sum_{i=1}^s w_i \cdot n_i = O(m \log(m/\varepsilon)).$$

As alluded to earlier, an important special case of decomposable nonsmooth convex optimization is decomposable submodular function minimization, which in turn has witnessed a long history of research [8, 27, 34, 39, 57]. Therefore, outside of distributed optimization, an immediate application of Theorem 1.14 is an improved cost of decomposable submodular function minimization, as we describe in Corollary 1.15.

Corollary 1.15 (Faster Submodular Function Minimization)**.** *Let $V = \{1, 2, \ldots, m\}$, and $F : 2^V \mapsto [-1, 1]$ be given by $F(S) = \sum_{i=1}^n F_i(S \cap V_i)$, where each $F_i : 2^{V_i} \mapsto \mathbb{R}$ is a submodular function on $V_i \subseteq V$. We can find an ε-additive approximate minimizer of F in*

$$O\left(\sum_{i=1}^n |V_i|^2 \log(n\varepsilon^{-1})\right) \text{ evaluation oracle calls.}$$

To contextualize our above result for decomposable SFM, [25] improved upon the cost of $O\left(V_{\max}^2 \sum_{i=1}^n |V_i|^4 \log(n\varepsilon^{-1})\right)$ by [8] to get a cost of $O\left(V_{\max} \sum_{i=1}^n |V_i| \log(n\varepsilon^{-1})\right)$. In cases where the $|V_i|$ are highly non-uniform, our result of $O\left(\sum_{i=1}^n |V_i|^2 \log(n\varepsilon^{-1})\right)$ evaluation oracle calls is therefore an improvement upon what is, to the best of our knowledge, the previous fastest result.

1.1.5 Lower Bounds. Finally, we complement our upper bounds results from the previous sections with lower bounds. [76] asked the following question: from the perspective of communication complexity, is solving a linear program harder than (exactly) solving a linear system? Towards answering this question, they showed that, in constant dimensions, checking feasibility of a linear program requires $\widetilde{\Omega}(sL)$ communication in the coordinator model, while feasibility for linear systems requires only $\widetilde{O}(s + L)$ communication, thereby demonstrating an exponential separation between the two problems. However their lower bound for linear programs was based on a hard instance with $2^{\Omega(L)}$ constraints. So for linear program feasibility problems with n constraints they leave open the possibility of a protocol with communication cost $O(s \log n) + o(sL)$. This is an important limitation of their lower bound, since they show for example, that a modified Clarkson's Algorithm [18, 76], indeed gives a protocol with $\log n$ dependence. This, therefore, motivates the following question: *is there an exponential separation between checking feasibility of linear programs and solving linear systems when there are only $\text{poly}(s + d)$ constraints?*

We answer this question in the affirmative, showing that such a separation does in fact hold, even for linear feasibility problems with $O(s + d)$ constraints.

Theorem 1.16. *Any protocol solving Linear Feasibility in the coordinator model requires at least $\Omega(sdL)$ communication for protocols that exchange at most $cL/\log L$ rounds of messages with each server and with $\log d \le 5L$. This bound holds even when the number of constraints is promised to be at most $O(s + d)$. For constant d the $\Omega(sL)$ lower bound holds with no assumption on the number of rounds.*

In addition to linear programming, one could also ask to get tight lower bounds for relative error least squares regression as discussed above. [76] gave a lower bound of $\Omega(sd+d^2L)$ for constant ε, however our algorithm requires $\widetilde{O}(sdL + d^2L)$ bits. We close this gap by showing that the sdL term is unavoidable. Interestingly, our regression lower bound follows from the same techniques that we we use to derive our linear programming lower bound.

1.2 Technical Overview

Before providing the details of our algorithms and analyses for each of the aforementioned results, we give high-level overviews of the techniques we use for each of them.

1.2.1 Least Squares Regression and Subspace Embeddings. We give two protocols for the regression problem instance of $\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{b}\|_2$. The first is based on sketching what we refer to as the *block leverage scores*, which for us is simply the sum of the leverage scores of the rows in that block (leverage scores computed with respect to $\mathbf{A}$). Our second protocol is based on *non-adaptive adaptive sketches* [50] from the in the data streaming literature. Both of our protocols for regression operate by constructing a subspace embedding[4] matrix $\mathbf{S}$ for the span of the columns of $\mathbf{A}$ and $\mathbf{b}$. This is a stronger guarantee than solving the regression problem, as the coordinator may compute $\mathbf{SAx} - \mathbf{Sb} = \mathbf{S}(\mathbf{Ax} - \mathbf{b})$ and output the solution to the sketched regression problem [77]. The subspace embedding construction proves useful in contexts other than regression too. Indeed, we require the subspace embedding construction to get improved communication for low-rank approximation. We now describe our two approaches below.

Block Leverage Scores. While block leverage scores have previously been considered in various forms [41, 51, 58, 60, 80] as far as we are aware they have (naturally) been used only in the context of sampling entire blocks at a time. In our setting, we are ultimately interested only in sampling rows, but find that approximating the block leverage score of each server is a useful subroutine. Specifically we show that for small k, sampling a $k \times d$ row-sketch of each block is almost sufficient to estimate all the block leverage scores. The catch is that we fail to accurately estimate block leverage scores that are larger than k. Intuitively, this is because such blocks could have more than k "important" rows. So our approach is to attempt to estimate the leverage score of all blocks via sketching using a small value of k. We might find that a small number of blocks have leverage scores that are too big for the estimates of their leverage scores to be accurate. To fix this, we focus on those blocks and sample a larger row-sketch from them in order to get a better estimate of their block leverage scores. Taking a larger sketch requires more communication per block. Crucially, however, the number of servers with leverage score greater than k is at most d/k. Thus we may proceed in a series of rounds, where in round r we focus on servers with leverage score at least 2^r. There are at most $d/2^r$ such servers, and for each server we take a sketch of total size roughly $2^r d$, so each round after (of which there are only $O(\log d)$) uses roughly d^2 communication. We note that the first round requires a roughly $1 \times d$ sized sketch from all servers, which yields an sd dependence. Once we have estimates of the block leverage scores, we observe that sampling sketched rows from the blocks proportional to the block leverage scores suffices to obtain a subspace embedding for $\mathbf{A}$.

Non-adaptive Adaptive Sketching. When $p = 2$, our protocol runs the recursive leverage score sampling procedure of [19] adapted to the distributed setting. One potential approach is to run this algorithm by sketching the inverse spectral approximations and broadcasting them to the servers. Unfortunately, when $\mathbf{A}$ is nearly singular, these sketches can have a high bit complexity. To avoid this, we instead use a version of an ℓ_2 *sampling sketch* which can be applied on the servers' sides and sent to the coordinator, which allows the coordinator to sample from the appropriate (relative) leverage score distribution. An issue arises if some relative scores are much larger than one, as we need to truncate them to roughly one before using them as sampling probabilities (up to scaling). To fix this, we first give a subroutine to identify this subset of outlying rows.

ℓ_p Regression beyond $p = 2$. Our "non-adaptive adaptive" protocol above extends to give optimal guarantees for ℓ_1 regression and ℓ_p regression for $1 \le p \le 2$ essentially by using the more general recursive Lewis weight sampling protocol of [20].

For $2 < p < 4$, the recursive Lewis weight sampling algorithm of [20] can also be run exactly to construct an ℓ_p subspace embedding, simply by broadcasting the approximate Lewis quadratic form to all servers on each round. Since the quadratic form is a $d \times d$ matrix, this broadcasting incurs an $\widetilde{O}(sd^2L)$ cost per round and hence an $\widetilde{O}(sd^2L)$ cost for computing approximate Lewis weights for all rows. The coordinator then must sample $d^{p/2}$ rows, resulting in a cost of $\widetilde{O}(sd^2L + d^{\max(p/2,1)+1})$.

If one is interested in sampling a coreset of rows to obtain an ℓ_p subspace embedding, then the $d^{p/2+1}$ term is unavoidable as we need to sample at least $d^{p/2}$ rows [48]. However for $1 \le p \le 2$ our approach for ℓ_2 regression shows that the sd^2L term can be improved to sdL. Whether the sd^2L term can be improved for all p is an interesting question that we leave to future work.

1.2.2 High-Accuracy Least Squares Regression. While constant-factor approximations often suffice, in certain settings it is important to ask for a solution that is optimal to within machine precision, e.g., if such solutions are used in iterative methods for solving a larger optimization problem. In this setting we consider the problem instance $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2$, where, given a row-partitioned system $\mathbf{A}$, $\mathbf{b}$, the goal of the coordinator is to output an $\mathbf{x}$ for which $\|\mathbf{Ax} - \mathbf{b}\|_2 \le \min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2 + \varepsilon \cdot \|\mathbf{A}(\mathbf{A}^\top\mathbf{A})^\dagger\mathbf{A}^\top\mathbf{b}\|_2$. A direct application of gradient descent requires about κ iterations, where κ is the condition number of matrix $\mathbf{A}$.

Our protocol for solving this problem in the distributed setting to a *high accuracy* is Richardson's iteration with preconditioning, coupled with careful rounding. We precondition using a constant factor spectral approximation of the matrix to reduce the number of iterations to only $\log(\varepsilon^{-1})$. This version of Richardson's iteration is equivalent to performing Newton's method with an approximate Hessian since the preconditioner spectrally approximates the Hessian inverse $(\mathbf{A}^\top\mathbf{A})^{-1}$. Computing a spectral approximation to $\mathbf{A}$ is equivalent to computing a subspace embedding, so to compute the preconditioner we employ our regression protocol from above. We note that the refinement sampling procedures we use in our two

[4] Recall that $\mathbf{S}$ is an ε-distortion ℓ_p subspace embedding for $\mathbf{A}$ if $\|\mathbf{SAx}\|_p = (1 \pm \varepsilon)\|\mathbf{Ax}\|_p$ for all $\mathbf{x}$.

regression algorithms are very similar, since both are based on the same algorithm from [19]. However, we believe there are sufficient differences in the specifics to merit writing out the latter in full. Alternatively, it is possible to employ a somewhat simpler protocol (which also has the advantage of computing approximations to *all* leverage scores) since in the high-precision setting we allow for a condition-number dependence.

The key novelty in the implementation of our Richardson-style iteration is to communicate, in each step, only a partial number of bits of the residual vector. The idea here is that as the solution converges, the bits with high place values do not change much between consecutive iterations and therefore need not be sent every time. Using a similar idea, we show that the Richardson iteration is robust to a small amount of noise, which helps us avoid updating the lowest order bits. Overall, via a careful perturbation analysis, we show that communicating the updates on only the $O(L \log \kappa)$ middle bits of each entry suffices to guarantee the convergence of Richardson's iteration.

1.2.3 High-Accuracy Linear Programming. Similar to Section 1.2.2, we ask the question of solving *linear programs* to a high accuracy. These require different techniques than ones in fast *first-order* algorithms for linear programs with runtimes depending polynomially on $1/\varepsilon$ [6, 7, 79]. Specifically, interior-point methods and cutting-plane methods are the standard approaches in the high-accuracy regime. Recent advances in fast high-accuracy algorithms for linear programs [42, 43, 75] were spurred by developments in the novel use of the Lewis weight barrier, techniques for efficient maintenance of the approximate inverse of a slowly-changing matrix, and efficient data structures for various linear algebraic primitives. Our approach for a communication-efficient high-accuracy linear program solver builds upon these developments, effectively adapting them into the coordinator model.

We first describe the standard framework of interior-point methods. In this paradigm, one reduces solving the problem of $\min_{\mathbf{u} \in \mathcal{S}} \mathbf{c}^\top \mathbf{u}$ to that of solving a sequence of slowly-changing unconstrained problems $\min_{\mathbf{u}} \Psi_t(\mathbf{u}) \stackrel{\text{def}}{=} \left\{ t \cdot \mathbf{c}^\top \mathbf{u} + \psi_{\mathcal{S}}(\mathbf{u}) \right\}$ parametrized by t, with a *self-concordant barrier* $\psi_{\mathcal{S}}$ that enforces feasibility by becoming unbounded as $\mathbf{x} \to \partial \mathcal{S}$. The algorithm starts at $t = 0$, for which an approximate minimizer $\mathbf{x}_0^\star$ of $\psi_{\mathcal{S}}$ is known, and it alternates between increasing t and updating, via Newton's method, $\mathbf{x}$ to an approximate minimizer $\mathbf{x}_t^\star$ of the new Ψ_t. For a sufficiently large t, the minimizer $\mathbf{x}_t^\star$ also approximately optimizes the original problem $\min_{\mathbf{u} \in \mathcal{S}} \mathbf{c}^\top \mathbf{u}$ with sub-optimality gap $O(\nu/t)$, where ν is the self-concordance parameter of the barrier function used. This self-concordance parameter typically also appears in the iteration complexity.

While this is the classical interior-point method as pioneered by [55], there has been a flurry of recent effort focusing on improving different components of this paradigm. The papers we use for our purposes are those by [42, 43, 75], which developed variants of the aforementioned central path method essentially by reducing the original LP to certain data structure problems such as inverse maintenance and heavy-hitters. Adapting these approaches to the coordinator model, we provide an algorithm for approximately solving LPs with $\widetilde{O}((sd^{1.5}(L + \log(\kappa R)) + d^2 L) \cdot \log(\varepsilon^{-1}))$ bits of communication (Theorem 1.10). Among the tools we employ for our analysis are those for matrix spectral approximation we develop and our result for the communication complexity of leverage scores, which we use to bound the communication complexity of iteratively computing Lewis weights for computing an initial feasible solution.

1.2.4 Decomposable Nonsmooth Convex Optimization. For decomposable nonsmooth convex optimization in the blackboard model, we improve an algorithm from the literature and then adapt this improved algorithm to the distributed setting. Specifically, we study $\min_{\mathbf{x} \in \mathbb{R}^d} \sum_{i=1}^s f_i(\mathbf{x})$, where each f_i is μ-Lipschitz, convex, and dependent on some d_i coordinates of $\mathbf{x}$ — note that the different f_i could have overlapping supports — and the i^{th} machine has subgradient-oracle access to the i^{th} function.

Most prior works [36, 66, 69] and their accelerated variants [1, 3, 30, 49, 82] designed in the non-distributed variant of finite-sum minimization assumed f_i to be smooth and strongly convex. Those designed for the distributed setting [15, 70, 81] also typically imposed this assumption (some exceptions include [26]), but additionally also used as their performance metric only the *number of rounds of communication*, as opposed to the *total number of bits communicated*, which is what we focus on. Variants of gradient descent [56] that are typically applicable to this problem also require a bounded condition number. There has also been work on non-smooth empirical risk minimization, but usually it requires that the objective be a sum of a smooth loss and a non-smooth regularizer. The formulation we study is a more general form of empirical risk minimization: In particular, our setting allows all f_i to be non-smooth.

The work of [25] combines ideas from classical cutting-plane and interior-point methods to obtain a nearly-linear (in total effective dimension) number of subgradient oracle queries for solving the problem in the non-distributed setting. This is the algorithm we modify and adapt to the distributed setting; our modification also yields improvements in the non-distributed setting.

We first describe the result obtained by simply adapting the algorithm of [25] to the blackboard model. Following [25], we first use a simple epigraph trick to reduce this problem to one with a linear objective and constrained to be on an intersection of parametrized epigraphs of f_i: $\min_{\mathbf{x}:\mathbf{x}_i \in \mathcal{K}_i \subseteq \mathbb{R}^{d_i+1} \forall i \in [s], \mathbf{A}\mathbf{x}=\mathbf{b}} \mathbf{c}^\top \mathbf{x}$, where $\mathcal{K}_i$ are all convex sets. All servers hold identical copies of the problem data at all times. However, each server i has only separation-oracle access to the set $\mathcal{K}_i$, which comes from the equivalence to the subgradient-oracle access to f_i by the result of [44].

We maintain crude outer and inner set approximations, $\mathcal{K}_{\text{in},i}$ and $\mathcal{K}_{\text{out},i}$, to each set $\mathcal{K}_i$ (such that $\mathcal{K}_{\text{in},i} \subseteq \mathcal{K}_i \subseteq \mathcal{K}_{\text{out},i}$) and update $\mathbf{x}$, our candidate minimizer of the (new) objective $\mathbf{c}^\top \mathbf{x}$ using an interior-point method. Ideally, for some choice of barrier function defined over $\mathcal{K} \cap \{\mathbf{A}\mathbf{x} = \mathbf{b}\}$, we would update our candidate minimizer to move along the central path through this set. However, since we do not explicitly know $\mathcal{K}$, we instead use a barrier function defined over its proxy, $\mathcal{K}_{\text{out}} \cap \{\mathbf{A}\mathbf{x} = \mathbf{b}\}$. We improve our approximations of $\mathcal{K}_{\text{in}}$ and $\mathcal{K}_{\text{out}}$ using ideas inspired from classical cutting plane methods [74]. Thus, our algorithm essentially alternates between performing a cutting-plane step (to improve our set approximation of $\mathcal{K}$) and performing an interior-point method step (to enable the candidate minimizer $\mathbf{x}$ to make progress along the central path).

Each server runs a copy of the above algorithm. After updating the parameter t and computing $\mathbf{x}^\star_{\text{out}}$ — the current target for the interior-point method step — each server tests feasibility of $\mathbf{x}^\star_{\text{out}}$. If there is a potential infeasibility of the i^{th} block, $\mathbf{x}^\star_{\text{out},i}$, then the server queries $\mathbf{x}^\star_{\text{out},i}$ (the i^{th} block of the current target point) and sends to the blackboard a separating hyperplane to update $\mathcal{K}_{\text{out},i}$ or a bit to indicate otherwise. The other servers then read this information and update either the set $\mathcal{K}_{\text{out},i}$ or $\mathcal{K}_{\text{in},i}$ on their ends. It was shown in [25] that this algorithm (without the distributed setting) has an oracle query complexity of $\widetilde{O}(\sum_{i=1}^s d_i)$. In the distributed setting, this would translate to a communication complexity of $\widetilde{O}(\max_{j\in[s]} d_j L \cdot \sum_{i=1}^s d_i)$.

Our main novelty is to modify the prior analysis (and slightly modify a specific parameter of the algorithm) so as to obtain the more fine-grained oracle cost of $\widetilde{O}(\sum_{i=1}^s w_i d_i L)$, for any arbitrary weight vector $\mathbf{w} \geq 0$. In our distributed algorithm, we set $w_i = d_i$, since the only communication that happens in a round is when a server sends hyperplane information to the blackboard. Thus, this translates to the communication cost of $\widetilde{O}(\sum_{i=1}^s d_i^2 L)$, an improvement over the bound of $\widetilde{O}(\max_{j\in[s]} d_j L \cdot \sum_{i=1}^s d_i)$ obtained from adapting [25] to the blackboard setting.

1.2.5 Lower Bounds. We are interested in obtaining tight lower bounds for least squares regression and low-rank approximation that capture the dependence on the bit complexity L. When proving such lower bounds, it is common to reduce from communication games such as multi-player set-disjointness [65]. However, it is not at all clear how one could encode such a combinatorial problem into an instance of regression that would yield a good bit-complexity lower bound. Indeed, most natural reductions from the standard communication problems would result in a single bit entry of $\mathbf{A}$. This motivates us to introduce a new communication game (Problem 1.17) that forces the players to communicate a large number of bits of their inputs.

Problem 1.17. *The coordinator holds an (infinite-precision) unit vector $\mathbf{v} \in \mathbb{R}^d$ with $d \geq 3$, and the s servers hold unit vectors $\mathbf{w}_1, \ldots, \mathbf{w}_s \in \mathbb{R}^d$ respectively. The coordinator must decide between (a) $\mathbf{v}^\top \mathbf{w}_k = 0$ for all $k \in [s]$ and (b) For some k, $|\mathbf{v}^\top \mathbf{w}_k| \geq \frac{\varepsilon}{d}$ and $\mathbf{v}^\top \mathbf{w}_i = 0$ for all $i \neq k$.*

The two player-version of Problem 1.17 is reminiscent of the promise inner product problem (PromiseIP$_d$) over $\mathbf{F}_p$, where the goal is to distinguish between $\mathbf{v}^\top \mathbf{w} = 0$ and $\mathbf{v}^\top \mathbf{w} = 1$ for $\mathbf{v}, \mathbf{w} \in \mathbf{F}_p^d$. This problem was introduced by [71] who gave an $\Omega(d \log p)$ lower bound and further considered by [47] who developed an s-player version. We note that their s-player version is for the "generalized inner product" and is therefore quite different from the game that we introduce. Furthermore we are not aware of a version of PromiseIP$_d$ over $\mathbb{R}$ that is suitable for our purposes, even though real versions of the inner product problem have been studied [5].
We give the following lower bound for our problem:

Theorem 1.18. *A protocol that solves Problem 1.17 with probability at least 0.9 requires at least $\Omega(sd \log(\varepsilon^{-1}))$ communication for protocols that exchange at most $c \log(\varepsilon^{-1})/\log\log(\varepsilon^{-1})$ rounds of messages with each server.*

To prove this, we begin by considering $d = 3$, and $s = 1$ so that the game involves two players, say Alice and Bob, holding vectors $\mathbf{v}$ and $\mathbf{w}$ in $\mathbb{R}^3$. We borrow techniques from Fourier analysis on the sphere to prove an $\Omega(\varepsilon^{-1})$ communication lower bound. Our techniques are reminiscent of those in [62], although we require somewhat less sophisticated machinery. One might wonder why we choose to start with $d = 3$ rather than $d = 2$. It turns out that when $d = 2$ the $\Omega(\varepsilon^{-1})$ lower bound does not hold! Indeed Alice can form the vector $\mathbf{v}^\perp$ so that the problem reduces to checking if $\mathbf{v}^\perp$ and $\mathbf{w}$ are approximately equal up to sign. This reduces to checking exact equality after truncating to approximately $\log(\varepsilon^{-1})$ bits. But this is easy to accomplish with $O(1)$ bits of communication by communicating an appropriate hash. It is not immediately clear whether a similar trick could apply in higher dimensions. In particular any proof of the lower bound must explain the difference between $d = 2$ and $d = 3$. The difference turns out to be that the spherical Radon transform is smoothing in dimensions 3 and higher, but not in dimension 2.

Given the $d = 3$ case, we boost our result to higher dimensions by a viewing a d-dimensional vector as the concatenation of $d/3$ vectors each of 3-dimensions and then applying the direct-sum technique of [10]. This requires us to first prove an information lower bound on a particular input distribution. This turns out to be easier to accomplish for public-coin protocols, and we then upgrade to general (private-coin) protocols using a "reverse-Newman" result of [17]. This last step is where our bounded round assumption arises from. We note that this is a purely technical artifact of our proof and can likely be avoided. Finally, we show how to extend our lower bound from two players to s players. With this result, we are able to deduce new lower bounds for least-squares regression and testing feasibility of linear programs.

Least Squares Regression. [76] studied the communication complexity of the least squares regression problem and showed a communication lower bound of $\widetilde{\Omega}(sd + d^2 L)$. We show that obtaining a constant factor approximation to a least-squares regression problem requires $\Omega(sdL)$ communication, at least for protocols that use at most roughly L rounds of communication. This bounded round assumption is mild since our algorithms need only $\widetilde{O}(1)$ rounds, which is desirable.

The reduction is from Problem 1.17 above. Our approach is to construct a matrix from the inputs whose smallest singular value is roughly 2^{-L} in case (a) and roughly $2^{-L/2}$ in case (b). To create such a matrix $\mathbf{A}$ we stack the vectors $\alpha\mathbf{v}, \mathbf{w}_1, \ldots, \mathbf{w}_s$ and additionally append an orthonormal basis for $\mathbf{v}^\perp$. We choose α to be an extremely small constant so that in either case, $\mathbf{v}$ is approximately the singular vector of $\mathbf{A}$ corresponding to $\sigma_{\min}(\mathbf{A})$. In case (a) we will arrange for $\sigma_{\min}(\mathbf{A})$ to be roughly 2^{-L} whereas in case (b) $\mathbf{w}_k$ will cause $\sigma_{\min}(\mathbf{A})$ to increase since $\mathbf{w}_k$ has positive inner product with $\mathbf{v}$. While the additive change in $\sigma_{\min}(\mathbf{A})$ is small, the multiplicative effect will be large. We then set up a regression problem involving $\mathbf{A}$ so that an approximate least squares solution has norm roughly $1/\sigma_{\min}(\mathbf{A})$ in either case, allowing us to distinguish cases (a) and (b).

Linear programming. Given our new communication lower bound, our reduction to linear feasibility is simple. We pick a collection of linear constraints that forces a feasible point $\mathbf{x}$ to satisfy $\mathbf{x} = \mathbf{v}$

and $\mathbf{x}^\top \mathbf{w}_k = 0$. In fact, this is just a linear system so how can our lower bound apply to it, given the better upper bounds for linear systems in [76]? The issue is that we need our linear constraints to have fixed bit precision whereas Problem 1.17 involves vectors with infinite precision. So we create inequalities enforcing the machine precision instead of requiring inner products exactly zero. Our lower bound gives a new way to obtain lower bounds depending on the condition number in this context, which may be useful for other problems.

High-Accuracy Regression. Finally, in the high-accuracy regime we show an $\widetilde{\Omega}(sd\log(\varepsilon^{-1}))$ lower bound for solving least squares regression to ε additive error. This shows that the $sd\log(\varepsilon^{-1})$ dependence in our high precision algorithm is unavoidable, and in fact shows that our upper bound is tight in the common setting where L and $\log\kappa$ are $O(1)$.

Theorem 1.19. *Consider a distributed least squares regression problem with the rows* $\mathbf{A}$ *and* $\mathbf{b}$ *distributed across* s *servers, and with* $\|\mathbf{b}\| = 1$. *Let* $\mathbf{x}_\star$ *minimize* $\|\mathbf{A}\mathbf{x}_\star - b\|_2$. *A protocol in the coordinator model that produces* $\widehat{\mathbf{x}}$ *satisfying*

$$\|\mathbf{A}\widehat{\mathbf{x}} - \mathbf{b}\|_2 \leq \varepsilon + \|\mathbf{A}\mathbf{x}_\star - \mathbf{b}\|_2$$

with probability at least 0.98 *requires* $\widetilde{\Omega}\left(sd\min(\log(\varepsilon^{-1}), L)\right)$ *communication. This lower bound holds even if* $\mathbf{A}$ *is promised to have condition number* $O(1)$.

2 CONCLUSION AND FUTURE DIRECTIONS

In this section, we discuss a few open problems and possible directions for future research regarding the communication complexity of the convex optimization problems we discussed in the paper.

Linear regression on matrices with special structure. Many optimization algorithms use linear regression as a subprocedure. One such example is IPMs, which are used to solve linear programs, which we discussed in this paper. In some scenarios, the linear regression problem has a special structure, e.g., the alternating least squares algorithm in tensor decomposition [24, 28] and least squares with non-negative data [23], which appears in many real-world problems. It would be interesting to investigate the communication complexity of solving linear regression problems that have matrices with special structures.

Inverse maintenance. Many recent improvements for the running time of convex optimization problems, such as linear programming and semi-definite programming, have relied on the use of inverse maintenance. We also used this to improve the communication complexity of solving linear programs when we use IPMs in a distributed setting. There has recently been some progress on analyzing inverse maintenance for general matrix formulas in an attempt to unify the analysis of many algorithms [4, 16]. An interesting direction is to analyze the communication complexity of inverse maintenance for such general matrix formulas.

ACKNOWLEDGEMENTS

We are very grateful to the anonymous reviewers of STOC'24 for their detailed and constructive suggestions. We thank Krishna Pillutla for helpful references to related works in distributed optimization. Research of Yin Tat Lee was supported by NSF awards CCF-1749609, DMS-1839116, DMS-2023166, CCF-2105772, a Microsoft Research Faculty Fellowship, a Sloan Research Fellowship, and a Packard Fellowship. Research of Swati Padmanabhan was supported by NSF awards CCF-1749609, DMS-1839116, DMS-2023166, and CCF-2105772. Research of Guanghao Ye was supported by NSF awards CCF-1955217 and DMS-2022448. Research of William Swartworth and David P. Woodruff were supported by a Simons Investigator Award. Part of this work was done while visiting the Simons Insitute for the Theory of Computing and Google Research.

REFERENCES

[1] Alekh Agarwal and Leon Bottou. 2015. A lower bound for the optimization of finite sums. In *International conference on machine learning*.
[2] Alekh Agarwal and John C Duchi. 2011. Distributed delayed stochastic optimization. *Advances in neural information processing systems* (2011).
[3] Zeyuan Allen-Zhu. 2017. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research* (2017).
[4] Emile Anand, Jan van den Brand, Mehrdad Ghadiri, and Daniel Zhang. 2024. The Bit Complexity of Dynamic Algebraic Formulas and their Determinants. *arXiv preprint arXiv:2401.11127* (2024).
[5] Alexandr Andoni, Jarosław Błasiok, and Arnold Filtser. 2022. Communication Complexity of Inner Product in Symmetric Normed Spaces. *arXiv preprint arXiv:2211.13473* (2022).
[6] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. 2021. Practical large-scale linear programming using primal-dual hybrid gradient. *Advances in Neural Information Processing Systems* (2021).
[7] David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. 2023. Faster first-order primal-dual methods for linear programming using restarts and sharpness. *Mathematical Programming* (2023).
[8] Kyriakos Axiotis, Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu. 2021. Decomposable submodular function minimization via maximum flow. In *International Conference on Machine Learning*.
[9] Maria Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. 2012. Distributed learning, communication complexity and privacy. In *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings.
[10] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.* 68, 4 (2004).
[11] Dimitri Bertsekas and John Tsitsiklis. 2015. *Parallel and distributed computation: numerical methods*. Athena Scientific.
[12] Srinadh Bhojanapalli, Prateek Jain, and Sujay Sanghavi. 2014. Tighter low-rank approximation via sampling the leveraged element. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*.
[13] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems* (2019).
[14] Christos Boutsidis, David P Woodruff, and Peilin Zhong. 2016. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*.
[15] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning* (2011).
[16] Jan van den Brand. 2021. Unifying Matrix Data Structures: Simplifying and Speeding up Iterative Algorithms. In *Symposium on Simplicity in Algorithms (SOSA)*.
[17] Mark Braverman and Ankit Garg. 2014. Public vs private coin in bounded-round information. In *International Colloquium on Automata, Languages, and Programming*. Springer.
[18] Kenneth L Clarkson. 1995. Las Vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)* (1995).
[19] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. 2015. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*.
[20] Michael B Cohen and Richard Peng. 2015. Lp row sampling by lewis weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*.
[21] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* (2008).
[22] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. 2012. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research* (2012).

[23] Jelena Diakonikolas, Chenghui Li, Swati Padmanabhan, and Chaobing Song. 2022. A Fast Scale-Invariant Algorithm for Non-negative Least Squares with Non-negative Data. *Advances in Neural Information Processing Systems* (2022).

[24] Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David Woodruff. 2019. Optimal sketching for kronecker product regression and low rank approximation. *Advances in neural information processing systems* (2019).

[25] Sally Dong, Haotian Jiang, Yin Tat Lee, Swati Padmanabhan, and Guanghao Ye. 2022. Decomposable Non-Smooth Convex Optimization with Nearly-Linear Gradient Oracle Complexity. *Advances in Neural Information Processing Systems* 35 (2022).

[26] John C Duchi, Alekh Agarwal, and Martin J Wainwright. 2012. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control* (2012).

[27] Alina Ene, Huy Nguyen, and László A Végh. 2017. Decomposable submodular function minimization: discrete and continuous. *Advances in neural information processing systems* 30 (2017).

[28] Matthew Fahrbach, Gang Fu, and Mehrdad Ghadiri. 2022. Subquadratic kronecker regression with applications to tensor decomposition. *Advances in Neural Information Processing Systems* (2022).

[29] Dan Feldman, Melanie Schmidt, and Christian Sohler. 2020. Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering. *SIAM J. Comput.* (2020).

[30] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. 2015. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*.

[31] Mehrdad Ghadiri, Richard Peng, and Santosh Vempala. 2023. The Bit Complexity of Efficient Continuous Optimization. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. https://www.computer.org/csdl/proceedings-article/focs/2023/189400c059/1T9796LmQ80

[32] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. 2020. LoAdaBoost: Loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data. *Plos one* (2020).

[33] Ali Jadbabaie, Jie Lin, and A Stephen Morse. 2003. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control* (2003).

[34] Stefanie Jegelka, Francis Bach, and Suvrit Sra. 2013. Reflection methods for user-friendly submodular optimization. *Advances in Neural Information Processing Systems* (2013).

[35] Björn Johansson, Maben Rabi, and Mikael Johansson. 2010. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization* (2010).

[36] Rie Johnson and Tong Zhang. 2013. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems* 26 (2013).

[37] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14 (2021).

[38] Ravi Kannan, Santosh Vempala, and David Woodruff. 2014. Principal component analysis and higher correlations for distributed data. In *Conference on Learning Theory*. PMLR.

[39] Senanayak Sesh Kumar Karri, Francis Bach, and Thomas Pock. 2019. Fast decomposable submodular function minimization using constrained total variation. *Advances in Neural Information Processing Systems* 32 (2019).

[40] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[41] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. 2016. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*.

[42] Yin Tat Lee and Aaron Sidford. 2014. Path Finding Methods for Linear Programming: Solving Linear Programs in $\tilde{O}(\sqrt{\text{rank}})$ Iterations and Faster Algorithms for Maximum Flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*.

[43] Yin Tat Lee and Aaron Sidford. 2015. Efficient Inverse Maintenance and Faster Algorithms for Linear Programming. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*.

[44] Yin Tat Lee, Aaron Sidford, and Santosh S Vempala. 2018. Efficient convex optimization with membership oracles. In *Conference On Learning Theory*. PMLR.

[45] Victor Lesser, Charles L Ortiz Jr, and Milind Tambe. 2003. *Distributed sensor networks: A multiagent perspective*. Springer Science & Business Media.

[46] Yi Li, Honghao Lin, and David Woodruff. 2023. ℓ_p-Regression in the Arbitrary Partition Model of Communication. In *The Thirty Sixth Annual Conference on Learning Theory*.

[47] Yi Li, Xiaoming Sun, Chengu Wang, and David P Woodruff. 2014. On the communication complexity of linear algebraic problems in the message passing model. In *Distributed Computing: 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings 28*. Springer.

[48] Yi Li, Ruosong Wang, and David P Woodruff. 2021. Tight bounds for the subspace sketch problem with applications. *SIAM J. Comput.* (2021).

[49] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. 2015. A universal catalyst for first-order optimization. *Advances in neural information processing systems* (2015).

[50] Sepideh Mahabadi, Ilya Razenshteyn, David P Woodruff, and Samson Zhou. 2020. Non-adaptive adaptive sampling on turnstile streams. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*.

[51] Naren Sarayu Manoj and Max Ovsiankin. 2023. The Change-of-Measure Method, Block Lewis Weights, and Approximating Matrix Block Norms. arXiv:2311.10013 [math.FA]

[52] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*.

[53] Angelia Nedić, Dimitri P Bertsekas, and Vivek S Borkar. 2001. Distributed asynchronous incremental subgradient methods. *Studies in Computational Mathematics* (2001).

[54] Angelia Nedic and Asuman Ozdaglar. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Automat. Control* (2009).

[55] Yurii Nesterov and Arkadii Nemirovskii. 1994. *Interior-point polynomial algorithms in convex programming*. SIAM.

[56] Yurii E Nesterov. 1983. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. akad. nauk Sssr*, Vol. 269.

[57] Robert Nishihara, Stefanie Jegelka, and Michael I Jordan. 2014. On the convergence rate of decomposable submodular function minimization. *Advances in Neural Information Processing Systems* 27 (2014).

[58] Urvashi Oswal, Swayambhoo Jain, Kevin S Xu, and Brian Eriksson. 2019. Block cur: Decomposing matrices using groups of columns. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II 18*.

[59] Richard Peng and Santosh Vempala. 2021. Solving sparse linear systems faster than matrix multiplication. In *Proceedings of the 2021 ACM-SIAM symposium on discrete algorithms (SODA)*. SIAM.

[60] Alessandro Perelli and Martin S Andersen. 2021. Regularization by denoising sub-sampled Newton method for spectral CT multi-material decomposition. *Philosophical Transactions of the Royal Society A* (2021).

[61] Akbar Rafiey and Yuichi Yoshida. 2022. Sparsification of decomposable submodular functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[62] Oded Regev and Bo'az Klartag. 2011. Quantum one-way communication can be exponentially stronger than classical communication. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*.

[63] Amirhossein Reisizadeh, Farzan Farnia, Ramtin Pedarsani, and Ali Jadbabaie. 2020. Robust federated learning: The case of affine distribution shifts. *Advances in Neural Information Processing Systems* (2020).

[64] Amirhossein Reisizadeh, Isidoros Tziotis, Hamed Hassani, Aryan Mokhtari, and Ramtin Pedarsani. 2022. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *IEEE Journal on Selected Areas in Information Theory* (2022).

[65] Tim Roughgarden et al. 2016. Communication complexity (for algorithm designers). *Foundations and Trends® in Theoretical Computer Science* (2016).

[66] Nicolas Roux, Mark Schmidt, and Francis Bach. 2012. A stochastic gradient method with an exponential convergence rate for finite training sets. *Advances in neural information processing systems* 25 (2012).

[67] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. 2019. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications* (2019).

[68] Ali H Sayed et al. 2014. Adaptation, learning, and optimization over networks. *Foundations and Trends® in Machine Learning* (2014).

[69] Shai Shalev-Shwartz and Tong Zhang. 2013. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research* 14, 2 (2013).

[70] Ohad Shamir, Nati Srebro, and Tong Zhang. 2014. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*.

[71] Xiaoming Sun, Chengu Wang, and Wei Yu. 2012. The relationship between inner product and counting cycles. In *Latin American Symposium on Theoretical Informatics*.

[72] S Sundhar Ram, Angelia Nedić, and Venugopal V Veeravalli. 2010. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications* (2010).

[73] John N Tsitsiklis. 1984. *Problems in decentralized decision making and computation*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[74] Pravin M Vaidya. 1989. A new algorithm for minimizing convex functions over convex sets. In *30th Annual Symposium on Foundations of Computer Science*.

[75] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. 2020. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*.
[76] Santosh S Vempala, Ruosong Wang, and David P Woodruff. 2020. The Communication Complexity of Optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*.
[77] David P Woodruff et al. 2014. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science* (2014).
[78] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. 2007. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing* (2007).
[79] Zikai Xiong and Robert Michael Freund. 2023. Computational Guarantees for Restarted PDHG for LP based on" Limiting Error Ratios" and LP Sharpness. *arXiv preprint arXiv:2312.14774* (2023).
[80] Peng Xu, Jiyan Yang, Fred Roosta, Christopher Ré, and Michael W Mahoney. 2016. Sub-sampled Newton methods with non-uniform sampling. *Advances in Neural Information Processing Systems* (2016).
[81] Yuchen Zhang and Xiao Lin. 2015. DiSCO: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning*.
[82] Yuchen Zhang and Xiao Lin. 2015. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *International Conference on Machine Learning*.

Received 13-NOV-2023; accepted 2024-02-11