Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

The Communication Complexity of Optimization

Santosh S. Vempala[*] Ruosong Wang[†] David P. Woodruff[‡]

Abstract

We consider the communication complexity of a number of distributed optimization problems. We start with the problem of solving a linear system. Suppose there is a coordinator together with s servers $P_1, \ldots, P_s$, the i-th of which holds a subset $A^{(i)}x = b^{(i)}$ of n_i constraints of a linear system in d variables, and the coordinator would like to output an $x \in \mathbb{R}^d$ for which $A^{(i)}x = b^{(i)}$ for $i = 1, \ldots, s$. We assume each coefficient of each constraint is specified using L bits. We first resolve the randomized and deterministic communication complexity in the point-to-point model of communication, showing it is $\widetilde{\Theta}(d^2L + sd)$ and $\widetilde{\Theta}(sd^2L)$, respectively. We obtain similar results for the blackboard communication model. As a result of independent interest, we show the probability a random matrix with integer entries in $\{-2^L, \ldots, 2^L\}$ is invertible is $1 - 2^{-\Theta(dL)}$, whereas previously only $1 - 2^{-\Theta(d)}$ was known.

When there is no solution to the linear system, a natural alternative is to find the solution minimizing the ℓ_p loss, which is the ℓ_p regression problem. While this problem has been studied, we give improved upper or lower bounds for every value of $p \geq 1$. One takeaway message is that sampling and sketching techniques, which are commonly used in earlier work on distributed optimization, are neither optimal in the dependence on d nor on the dependence on the approximation ε, thus motivating new techniques from optimization to solve these problems.

Towards this end, we consider the communication complexity of optimization tasks which generalize linear systems, such as linear, semidefinite, and convex programming. For linear programming, we first resolve the communication complexity when d is constant, showing it is $\widetilde{\Theta}(sL)$ in the point-to-point model. For general d and in the point-to-point model, we show an $\widetilde{O}(sd^3L)$ upper bound and an $\widetilde{\Omega}(d^2L + sd)$ lower bound. In fact, we show if one perturbs the coefficients randomly by numbers as small as $2^{-\Theta(L)}$, then the upper bound is $\widetilde{O}(sd^2L) + \mathrm{poly}(dL)$, and so this bound holds for almost all linear programs. Our study motivates understanding the bit complexity of linear programming, which is related to the running time in the unit cost RAM model with words of $O(\log(nd))$ bits, and we give the fastest known algorithms for linear programming in this model.

1 Introduction

Large-scale optimization problems often cannot fit into a single machine, and so they are distributed across a number s of machines. That is, each of servers $P_1, \ldots, P_s$ may hold a subset of constraints that it is given locally as input, and the goal of the servers is to communicate with each other to find a solution satisfying all constraints. Since communication is often a bottleneck in distributed computation, the goal of the servers is to communicate as little as possible.

There are several different standard communication models, including the point-to-point model and the blackboard model. In the point-to-point model, each pair of servers can talk directly with each other. This is often more conveniently modeled by looking at the *coordinator model*, for which there is an extra server called the coordinator, and all communication must pass through the coordinator. This is easily seen to be equivalent, from a total communication perspective, to the point-to-point model up to a factor of 2, for forwarding messages from server P_i to server P_j, and a term of $\log s$ per message to indicate which server the message should be forwarded to. Another model of computation is the *blackboard model*, in which there is a shared broadcast channel among all the s servers. When a server sends a message, it is visible to each of the other $s - 1$ servers and determines who speaks next, based upon an agreed upon protocol. We mostly consider randomized communication, in which for every input, we require the coordinator to output the solution to the optimization problem with high probability. For linear systems we also consider deterministic communication complexity.

A number of recent works in the theory community have looked at studying specific optimization problems in such communication models, such as principal component analysis [35, 39, 13] and kernel [8] and robust

[*]Georgia Tech. Email: `vempala@cc.gatech.edu`.
[†]Carnegie Mellon University. Email: `ruosongw@andrew.cmu.edu`.
[‡]Carnegie Mellon University. Email: `dwoodruf@cs.cmu.edu`.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

variants [59, 23], computing higher correlations [35], ℓ_p regression [55, 23] and sparse regression [15], estimating the mean of a Gaussian [61, 28, 15], database problems [31, 58], clustering [17], statistical [56], graph problems [43, 56] and many, many more.

There are also a large number of distributed learning and optimization papers, for example [7, 61, 63, 1, 14, 40, 60, 19, 44, 26, 24, 25, 32, 47, 46, 62, 4, 34]. With a few exceptions, these works do not study general communication complexity, but rather consider specific classes of algorithms. Namely, a number of these works only allow gradient and Hessian computations in each round, and do not allow arbitrary communication. Another aspect of these works is that they typically do not count total bit complexity, but rather only count number of rounds, whereas we are interested in total communication. In a number of optimization problems, the bit complexity of storing a single number in an intermediate computation may be as large as storing the entire original optimization problem. It is therefore infeasible to transmit such a number. While one could round this number, the effect of rounding is often unclear, and could destroy the desired approximation guarantee. One exception to the above is the work of [52], which studies the problem in which there are two servers, each holding a convex function, who would like to find a solution so as to *minimize the sum* of the two functions. The upper bounds are in a different communication model than ours, where the functions are added together, while the lower bounds only apply to a restricted class of protocols.

Noticeably absent from previous work is the communication complexity of *solving linear systems*, which is a fundamental primitive in many optimization tasks. Formally, suppose there is a coordinator together with s servers $P_1, \ldots, P_s$, the i-th of which holds a subset $A^{(i)}x = b^{(i)}$ of n_i constraints of a d-dimensional linear system, and the coordinator would like to output an $x \in \mathbb{R}^d$ for which $A^{(i)}x = b^{(i)}$ for $i = 1, \ldots, s$. We further assume each coefficient of each constraint is specified using L bits. The first question we ask is the following.

QUESTION 1.1. *What is the communication complexity of solving a linear system?*

When there is no solution to the linear system, a natural alternative is to find the solution minimizing the ℓ_p loss, which is the ℓ_p regression problem $\min_{x \in \mathbb{R}^d} \|Ax - b\|_p$, where for an n-dimensional vector y, $\|y\|_p = \left(\sum_{i=1}^n |y_i|^p\right)^{1/p}$ is its ℓ_p norm.

In the distributed ℓ_p regression problem, each server has a matrix $A^{(i)} \in \mathbb{R}^{n_i \times d}$ and a vector $b^{(i)} \in \mathbb{R}^{n_i}$, and the coordinator would like to output an $x \in \mathbb{R}^d$ so that $\|Ax - b\|_p$ is approximately minimized, namely, that $\|Ax - b\|_p \le (1 + \varepsilon) \min_{x'} \|Ax' - b\|_p$. Note that here $A \in \mathbb{R}^{n \times d}$ is the matrix obtained by stacking the matrices $A^{(1)}, \ldots, A^{(s)}$ on top of each other, where $n = \sum_{i=1}^s n_i$. Also, $b \in \mathbb{R}^n$ is the vector obtained by stacking the vectors $b^{(1)}, \ldots, b^{(s)}$ on top of each other. We assume that each entry of A and b is an L-bit integer, and we are interested in the randomized communication complexity of this problem.

While previous work [41, 55] has looked at the distributed ℓ_p regression problem, such work is based on two main ideas: sampling and sketching. Such techniques reduce a large optimization problem to a much smaller one, thereby allowing servers to send succinct synopses of their constraints in order to solve a global optimization problem.

Sampling and sketching are the key techniques of recent work on distributed low rank approximation [55, 35] and regression algorithms. A natural question, which will motivate our study of more complex optimization problems below, is whether other techniques in optimization can be used to obtain more communication-efficient algorithms for these problems.

QUESTION 1.2. *Are there tractable optimization problems for which sampling and sketching techniques are suboptimal in terms of total communication?*

To answer Question 1.2 it is useful to study optimization problems generalizing both linear systems and ℓ_p regression for certain values of p. Towards this end, we consider the communication complexity of linear, semidefinite, and convex programming. Formally, in the linear programming problem, suppose there is a coordinator together with s servers $P_1, \ldots, P_s$, the i-th of which holds a subset $A^{(i)}x \le b^{(i)}$ of n_i constraints of a d-dimensional linear system, and the coordinator, who holds a vector $c \in \mathbb{R}^d$, would like to output an $x \in \mathbb{R}^d$ for which $c^T x$ is maximized subject to $A^{(i)}x \le b^{(i)}$ for $i = 1, \ldots, s$. We further assume each coefficient of each constraint, as well as the objective function c, is specified using L bits.

QUESTION 1.3. *What is the communication complexity of solving a linear program?*

One could try to implement known linear programming algorithms as distributed protocols. The main challenge here is that known linear programming algorithms operate in the *real RAM model of computation*, meaning that basic arithmetic operations on real numbers can be performed in constant time. This is problematic in the distributed setting, since it might mean real numbers need to be communicated among the

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

servers, resulting in protocols that could have infinite communication. Thus, controlling the bit complexity of the underlying algorithm is essential, and this motivates the study of linear programming algorithms in the *unit cost RAM model of computation*, meaning that a word is $O(\log(nd))$ bits, and only basic arithmetic operations on words can be performed in constant time. Such a model is arguably more natural than the real RAM model. If one were to analyze the fastest linear programming algorithms in the unit cost RAM model, their time complexity would blow up by $\text{poly}(dL)$ factors, since the intermediate computations require manipulating numbers that grow exponentially large or small. Surprisingly, we are not aware of any work that has addressed this question:

QUESTION 1.4. *What is the best possible running time of an algorithm for linear programming in the unit cost RAM model?*

As far as time complexity is concerned, it is not even known if linear programming is inherently more difficult than just solving a linear system. Indeed, a long line of work on interior point methods, with the current most recent work of [20], suggests that solving a linear program may not be substantially harder than solving a linear system. One could ask the same question for communication.

QUESTION 1.5. *Is solving a linear program inherently harder than solving a linear system? What about just checking the feasibility of a linear program versus that of a linear system?*

Recent Independent Work. A recent independent work [5] also studies solving linear programs in the distributed setting, although their focus is to study the tradeoff between round complexity and communication complexity in low dimensions, while our focus is to study the communication complexity in arbitrary dimensions. Note, however, that we also provide nearly optimal bounds for constant dimensions for linear programming in both coordinator and blackboard models.

1.1 Our Contributions

We make progress on answering the above questions, with nearly tight bounds in many cases. For a function f, we let $\widetilde{O}(f) = f\,\text{polylog}(sndL/\varepsilon)$ and similarly define $\widetilde{\Theta}$ and $\widetilde{\Omega}$.

1.1.1 Linear Systems

We begin with linear systems, for which we obtain a complete answer for both randomized and deterministic communication, in both coordinator and blackboard models of communication.

THEOREM 1.1. *In the coordinator model, the randomized communication complexity of solving a linear system is $\widetilde{\Theta}(d^2L + sd)$, while the deterministic communication complexity is $\widetilde{\Theta}(sd^2L)$. In the blackboard model, both the randomized communication complexity and the deterministic communication complexity are $\widetilde{\Theta}(d^2L+s)$.*

Theorem 1.1 shows that randomization provably helps for solving linear systems. The theorem also shows that in the blackboard model the problem becomes substantially easier.

1.1.2 Approximate Linear Systems, i.e., ℓ_p Regression

We next study the ℓ_p regression problem in both the coordinator and blackboard models of communication. Finding a solution to a linear system is a special case of ℓ_p regression; indeed in the case that there is an x for which $Ax = b$ we must return such an x to achieve $(1+\varepsilon)$ relative error in objective function value for ℓ_p regression. Consequently, our lower bounds for linear systems apply also to ℓ_p regression for any $\varepsilon > 0$.

We first summarize our results in Table 1 and Table 2 for constant ε. We state our results primarily for randomized communication. However, in the case of ℓ_2 regression, we also discuss deterministic communication complexity.

One of the main takeaway messages from Table 1 is that sampling-based approaches, namely those based upon the so-called Lewis weights [22], would require $\Omega(d^{p/2})$ samples for ℓ_p regression when $p > 2$, and thus communication. Another way for solving ℓ_p regression for $p > 2$ is via *sketching*, as done in [55], but then the communication is $\Omega(n^{1-2/p})$. Our method, which is deeply tied to linear programming, discussed more below, solves this problem in $\widetilde{O}(sd^3L)$ communication. Thus, this gives a new method, departing from sampling and sketching techniques, which achieves much better communication. Our method involves embedding ℓ_p into ℓ_∞, and then using distributed algorithms for linear programming to solve ℓ_∞ regression.

As with linear systems, one takeaway message from the results in Table 2 is that the problems have significantly more communication-efficient upper bounds in the blackboard model than in the coordinator model. Indeed, here we obtain tight bounds for ℓ_1 and ℓ_2 regression, matching those that are known for the easier problem of linear systems.

We next describe our results for non-constant ε in both the coordinator model and the blackboard model. Here we focus on ℓ_1 and ℓ_2, which illustrate several surprises.

One of the most interesting aspects of the results

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Error Measure	Upper Bound	Lower Bound	Theorem
ℓ_1 (randomized)	$\widetilde{O}(sd^2L)$	$\widetilde{\Omega}(d^2L + sd)$	Theorem 7.1, 3.5
ℓ_1 (deterministic)	$\widetilde{O}(sd^2L)$	$\widetilde{\Omega}(sd^2L)$	Theorem 7.1, 3.3
ℓ_2 (randomized)	$\widetilde{O}(sd^2L)$	$\widetilde{\Omega}(d^2L + sd)$	Theorem 6.1, 3.5
ℓ_2 (deterministic)	$\widetilde{O}(sd^2L)$	$\widetilde{\Omega}(sd^2L)$	Theorem 6.1, 3.3
ℓ_p for constant $p > 2$	$\widetilde{O}(sd^3L)$	$\widetilde{\Omega}(d^2L + sd)$	Theorem 8.2, 3.5
ℓ_∞	$\widetilde{O}(sd^3L)$	$\widetilde{\Omega}(d^2L + sd)$	Theorem 8.1, 3.5

Table 1: Summary of our results for ℓ_p regression in the coordinator model for constant ε.

Error Measure	Upper Bound	Lower Bound	Theorem
ℓ_1	$\widetilde{O}(s + d^2L)$	$\widetilde{\Omega}(s + d^2L)$	Theorem 7.2, 3.5
ℓ_2	$\widetilde{O}(s + d^2L)$	$\widetilde{\Omega}(s + d^2L)$	Theorem 6.2, 3.5
ℓ_p for constant $p > 2$	$O(\min\{sd + d^4L, sd^3L\})$	$\widetilde{\Omega}(s + d^2L)$	Theorem 8.2, 3.5
ℓ_∞	$O(\min\{sd + d^4L, sd^3L\})$	$\widetilde{\Omega}(s + d^2L)$	Theorem 8.1, 3.5

Table 2: Summary of our results for ℓ_p regression in the blackboard model for constant ε.

Error Measure	Upper Bound	Lower Bound	Theorem
ℓ_1	$\widetilde{O}(\min(sd^2L + \frac{d^2L}{\varepsilon^2}, \frac{sd^3L}{\varepsilon})$	$\widetilde{\Omega}(d^2L + sd)$	Theorem 7.2, 7.3, 3.5
ℓ_2 (randomized)	$\widetilde{O}(sd^2L)$	$\widetilde{\Omega}(d^2L + sd)$	Theorem 6.1, 3.5
ℓ_2 (deterministic)	$\widetilde{O}(sd^2L)$	$\widetilde{\Omega}(sd^2L)$	Theorem 6.1, 3.3

Table 3: Summary of our results for ℓ_1 and ℓ_2 regression in the coordinator model for general ε.

in Table 3 is our dependence on ε for ℓ_1 regression, where for small enough ε relative to sd, we achieve a $1/\varepsilon$ instead of a $1/\varepsilon^2$ dependence. We note that all sampling [22] and sketching-based solutions [55] to ℓ_1 regression have a $1/\varepsilon^2$ dependence. Indeed, this dependence on ε comes from basic concentration inequalities. In contrast, our approach is based on preconditioned first-order methods described in more detail below.

A takeaway message from Table 4 is that our lower bound shows some dependence on ε is necessary both for ℓ_1 and ℓ_2 regression, provided ε is not too small. This shows that in the blackboard model, one cannot obtain the same $\widetilde{O}(d^2L + s)$ upper bound for these problems as for linear systems, thereby separating their complexity from that of solving a linear system.

1.1.3 Linear Programming

One of our main technical ingredients is to recast ℓ_p regression problems as linear programming problems and develop the first communication-efficient solutions for distributed linear programming. Despite this problem being one of the most important problems that we know how to solve in polynomial time, we are not aware of any previous work considering its communication complexity in generality besides a recent independent work [5]

First, when the dimension d is constant, we obtain nearly optimal upper and lower bounds.

Theorem 1.2. *In constant dimensions, the randomized communication complexity of linear programming is $\widetilde{\Theta}(sL)$ in the coordinator model and $\widetilde{\Omega}(s + L)$ in the blackboard model. Our upper bounds allow the coordinator to output the solution vector $x \in \mathbb{R}^d$, while the lower bounds hold already for testing if the linear program is feasible. Here the $\widetilde{\Theta}(\cdot)$ notation and the $\widetilde{\Omega}(\cdot)$ notation suppress only* polylog(sL) *factors.*

Despite the fact that we do not have tight upper bounds matching the $\widetilde{\Omega}(s + L)$ lower bounds in the blackboard model, under the additional assumption that each constraint in the linear program is placed on a random server, we develop an algorithm with a matching $\widetilde{O}(s + L)$ communication cost. Partitioning constraints randomly across servers instead is common in distributed computation, see, e.g., [6]. Neverthelss we leave it as an open problem in the blackboard model in constant dimensions, to remove this requirement.

For solving a linear system in constant dimensions, the randomized communication complexity is $\widetilde{\Theta}(s + L)$

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Error Measure	Upper Bound	Lower Bound	Theorem
ℓ_1	$\widetilde{O}(s+\frac{d^2L}{\varepsilon^2})$	$\widetilde{\Omega}(s+\frac{d}{\varepsilon}+d^2L)$ for $s>\Omega(1/\varepsilon)$	Theorem 7.2, 3.5, 5.1
ℓ_2	$\widetilde{O}(s+\frac{d^2L}{\varepsilon})$	$\widetilde{\Omega}(s+\frac{d}{\varepsilon^{1/2}}+d^2L)$ for $s>\Omega(1/\sqrt{\varepsilon})$	Theorem 6.2, 3.5, 5.2

Table 4: Summary of our results for ℓ_1 and ℓ_2 regression in the blackboard model for general ε.

in both models. Again, the $\widetilde{\Theta}(\cdot)$ notation suppresses only polylog(sL) factors. Thus, in the coordinator model, we separate the communication complexity of these problems. We can also separate the complexities in the blackboard model if we instead look at the feasibility problem. Here instead of requiring the coordinator to output the solution vector, we just want to see if the linear system or linear program is feasible. We have the following theorem for this.

Theorem 1.3. *In constant dimensions, the randomized communication complexity of checking whether a system of linear equations is feasible is $O(s \log L)$ in either the coordinator or blackboard model of communication.*

Combining Theorem 1.2 and Theorem 1.3, we see that for feasibility in the blackboard model, linear programming requires $\widetilde{\Omega}(s+L)$ bits, while linear system feasibility takes $\widetilde{O}(s)$ bits, and thus we separate these problems in the blackboard model as well.

Returning to linear programs, we next consider the complexity in arbitrary dimensions.

Theorem 1.4. *In the coordinator model, the randomized communication complexity of exactly solving a linear program $\max\{c^T x : Ax \leq b\}$ with n constraints in dimension d and all coefficients specified by L-bit numbers is $\widetilde{O}(sd^3L)$. Moreover it is lower bounded by $\widetilde{\Omega}(d^2L + sd)$. Here the upper and lower bounds require the coordinator to output the solution vector $x \in \mathbb{R}^d$.*

The lower bound in Theorem 1.4 just follows from our lower bound for linear systems. The upper bound is based on an optimized distributed cutting-plane algorithm. We describe the idea below.

While the upper bound is $\widetilde{O}(sd^3L)$, one can further improve it as follows. We show that if the coefficients of A in the input to the linear program are perturbed independently by i.i.d. discrete Gaussians with variance as small as $2^{-\Theta(L)}$, then we can improve the upper bound for solving this perturbed problem to $\widetilde{O}(sd^2L + d^4L)$, where now the success probability of the algorithm is taken over both the randomness of the algorithm and the random input instance, which is formed by a random perturbation of a worst-case instance. Note that this is an improvement for sufficiently large s. Our model coincides with the well-studied *smooth complexity* model of linear programming [48, 11, 49]. However, a major difference is that the variance of the perturbation needs to be at least inverse polynomial in their works, whereas we allow our variance to be as small as $2^{-\Theta(L)}$.

Theorem 1.5. *In the smoothed complexity model with discrete Gaussians of variance $2^{-\Theta(L)}$, the communication complexity of exactly solving a linear program $\max\{c^T x : Ax \leq b\}$ with n constraints in dimension d and all coefficients specified by L-bit numbers, with probability at least $9/10$ over the input distribution and randomness of the protocol, is $\widetilde{O}(sd^2L + d^4L)$ in the coordinator model.*

While our focus in this paper is on communication, our upper bounds also give a new technique for improving the time complexity in the unit cost RAM model of linear programming, where arithmetic operations on words of size $O(\log(nd))$ can be performed in constant time. For this fundamental problem we obtain the fastest known algorithm even in the *non-smoothed setting* of linear programming.

Theorem 1.6. *The time complexity of solving an $n \times d$ linear program with L-bit coefficients is $\widetilde{O}(nd^{\omega}L + \text{poly}(dL))$ in the unit cost RAM model.*

We note that this is for solving an LP *exactly* in the RAM model with words of size $O(\log(nd))$ bits. The current fastest linear programming algorithms [37, 38, 20] state the bounds in terms of additive error ε, which incurs a multiplicative factor of at least $\Omega(dL)$ to solve the problem exactly. Also such algorithms manipulate large numbers at intermediate points in the algorithm, which are at least L bits, which could take $\Omega(L)$ time to perform a single operation on. It seems that transferring such results to the unit cost RAM model with $O(\log(nd))$ bit words incurs time at least $\Omega(nd^{2.5}L^2 + d^{w+1.5}L^2)$. This holds true even of the recent work [20], which focuses on the setting $n = O(d)$ and does not improve the leading $nd^{2.5}L^2$ term. Even such a bit-complexity bound needs careful checking of the number of bits required as recent improvements use sophisticated inverse maintenance methods to save on the number of operations (an exercise that was carried out thoroughly for the Ellipsoid method in [29]).

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

1.1.4 Implications for Convex Optimization and Semidefinite Programming Our upper bounds also extend to more general convex optimization problems. For these, we must modify the problem statement to finding an ε-additive approximation rather than the exact solution. We obtain the following upper bound for a convex program in $\mathbb{R}^d$.

Theorem 1.7. *The communication complexity of solving the convex optimization problem* $\min\{c^T x : x \in \bigcap_i K_i\}$ *for convex sets* $K_i \subseteq RB^n$*, one per server, to within an additive error* ε*, i.e., finding a point* y *s.t.* $c^T y \leq OPT + \varepsilon$ *and* $y \in \bigcap_i K_i + \varepsilon B^n$ *is* $O(sd^2 \log(Rd/\varepsilon) \log d)$*.*

If the objective function is not known to all servers, we incur an additional $O(sdL)$ communication. For semidefinite programs with $d \times d$ symmetric matrices and n linear constraints this gives a bound of $\widetilde{O}(sd^4 \log(1/\varepsilon))$. Note that we can simply send all the constraints to one server in $O(nd^2L)$ communication, so this is always an upper bound.

1.2 Our Techniques

1.2.1 Linear Systems To solve linear systems in the distributed setting, the coordinator can go through the servers one by one. The coordinator and all servers maintain the same set C of linearly independent linear equations. For each server P_i, if there is a linear equation stored by P_i that is linearly independent with linear equations in C, then P_i sends that linear equation to all other servers and adds that linear equation into C. In the end, C will be a maximal set of linearly independent equations, and thus the coordinator can simply solve the linear equations in C. This protocol is deterministic and has communication complexity $O(sd^2L)$ in the coordinator model and $O(s + d^2L)$ in the blackboard model, since at most d linear equations will be added into the set C.

In fact, the preceding protocol is optimal for deterministic protocols, even just for testing the feasibility of linear systems. To prove lower bounds, we first prove the following new theorem about random matrices which may be of independent interest.

Theorem 1.8. (Informal version of Theorem 3.1) *Let* R *be a* $d \times d$ *matrix with i.i.d. random integer entries in* $\{-2^L, \ldots, 2^L\}$*. The probability that* R *is invertible is* $1 - 2^{-\Theta(dL)}$*.*

The previous best known probability bound was only $1 - 2^{-\Theta(d)}$ [50, 12]; we stress that the results of [12] are not sufficient[1] to prove our stronger bound with the extra factor of L in the exponent, which is crucial for our lower bound.

With Theorem 1.8, in Lemma 3.2, we use the probabilistic method to construct a set of $|\mathcal{H}| = 2^{\Omega(d^2L)}$ matrices $\mathcal{H} \subseteq \mathbb{R}^{d\times d}$ with integral entries in $[-2^L, 2^L]$, such that for any $S, T \in \mathcal{H}$, $S^{-1}e_d \neq T^{-1}e_d$, where e_d is the d-th standard basis vector.

Now consider any deterministic protocol for testing the feasibility of linear systems. Suppose the linear system on the i-th server is $H_i x = e_d$ for some $H_i \in \mathcal{H}$, then the entire linear system is feasible if and only if $H_1 = H_2 = \ldots = H_s$. This is equivalent to the problem in which each server receives a binary string of length $\log(|\mathcal{H}|)$, and the goal is to test whether all strings are the same or not. In the coordinator model, a deterministic lower bound of $\Omega(s \log(|\mathcal{H}|))$ for this problem can be proved using the symmetrization technique in [43, 57], which gives an optimal $\Omega(sd^2L)$ lower bound. An optimal $\Omega(s+d^2L)$ deterministic lower bound can also be proved in the blackboard model. More details can be found in Section 3.3.

For solving linear systems, an $\Omega(d^2L)$ lower bound holds even for randomized algorithms in the coordinator model. When there is only a single server which holds a linear system $Hx = e_d$ for some $H \in \mathcal{H}$, in order for the coordinator to know the solution $x = H^{-1}e_d$, standard information-theoretic argument shows that $\log(|\mathcal{H}|)$ bits of communication is necessary, which gives an $\Omega(d^2L)$ lower bound. This idea is formalized in Section 3.4. A natural question is whether the $O(sd^2L)$ upper bound is optimal for randomized protocols.

We first show that in order to test feasibility, it is *possible* to achieve a communication complexity of $O(sd^2 \log(dL))$, which can be exponentially better than the bound for deterministic protocols. The idea is to use hashing. With randomness, the servers can first agree on a random prime number p, and test the feasibility over the finite field $\mathbb{F}_p$. It suffices to have the prime number p randomly generated from the range $[2, \text{poly}(dL)]$, and thus the L factor in the communicataion complexity of deterministic protocols can be improved to $\log p = \log(dL)$. However, it is still unclear if *solving linear systems* in the coordinator model will require $\Omega(sd^2L)$ bits of communication for randomized protocols.

Quite surprisingly, we show that $O(sd^2L)$ is not the optimal bound for randomized protocols, and the optimal bound is $\widetilde{\Theta}(d^2L + sd)$. In the deterministic pro-

[1]We have verified this with Philip Matchett Wood, who is an author of [12]. The issue is that in their Corollary 1.2, they have an explicit constraint on the cardinality of the set S, i.e., $|S| = O(1)$. In their Theorem 2.2, it is assumed that $|S| = n^{o(n)}$. Thus, as far as we are aware, there are no known results sufficient to prove our singularity probability bound.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

tocol with communication complexity $O(sd^2L)$, most communication is wasted on synchronizing the set C, which requires the servers to send linear equations to all other servers. In our new protocol, *only the coordinator* maintains the set C. The issue now, however, is that the servers no longer know which linear equation they own is linearly independent with those equations in C. On the other hand, each server can simply generate a random linear combination of all linear equations it owns. We can show that if a server does have a linear equation that is linearly independent with those in C, with constant probability, the random linear combination is also linearly independent with those in C, and thus the coordinator can add the random linear combination into C. Notice that taking random linear combinations to preserve the rank of a matrix is a special case of dimensionality reduction or *sketching*, which comes up in a number of applications, see, for example compressed sensing [16, 10], data streams [3], and randomized numerical linear algebra [54]. Here though, a crucial difference is that we just need the fact that if a set of vectors S is not contained in the span of another set of vectors V, then a random linear combination of the vectors in S is also not in the span of V with high probability. This allows us to adaptively take as few linear combinations as possible to solve the linear system, enabling us to achieve much lower communication than would be possible by just sketching the linear systems at each server and non-adaptively combining them.

If we implement this protocol naïvely, then the communication complexity will be $\widetilde{O}(d^2L + sdL)$, since at most d linear equations will be added into C, and there is an $\widetilde{O}(dL)$ communication complexity associated with each of them. Furthermore, even if a server does not have any linear equation that is linearly independent with C, it still needs to send random linear combinations to the coordinator, which would require $\widetilde{O}(sdL)$ communication. To improve this further to $\widetilde{O}(sd)$, we can still use the hashing trick mentioned before. If a server generates a random linear combination, it can first test whether the linear combination is linearly independent with C over the finite field p, for a random prime p chosen in $[2, \mathrm{poly}(dL)]$. This will reduce the communication complexity to $\widetilde{O}(d)$ for each test. If the linear equation is indeed linearly independent with C, then the server sends the original linear equation (without taking the residual modulo p) to the coordinator. Again the total communication complexity for sending the original linear equations is upper bounded by $O(d^2L)$. Thus, the total communication complexity is upper bounded by $\widetilde{O}(d^2L + sd)$. See Section 4.2 for details.

By a reduction from the OR of $s-1$ copies of the two-server set-disjointness problem to solving linear systems, we can prove an extra $\widetilde{\Omega}(sd)$ lower bound, which holds even for testing feasibility of linear systems. Here the idea is to interpret vectors in $\{0,1\}^d$ as characteristic vectors of subsets of $[d]$. One of the servers will fix the solution of the linear system to be a predefined vector x. Each server P_i has a single linear equation $a_i^T x = 1$. By interpreting vectors as sets, $a_i^T x = 1$ implies the set represented by a_i and x are intersecting. Thus, the servers are actually solving the OR of $s-1$ copies of the two-server set-disjointness problem, which is known to have $\widetilde{\Omega}(sd)$ communication complexity [43, 56]. This lower bound is formally given in Section 3.4.

1.2.2 Linear Regression For an ℓ_2 regression instance $\min_x \|Ax - b\|_2$, the optimal solution can be calculated using the *normal equations*, i.e., the optimal solution x satisfies $A^TAx = A^Tb$. This already gives a simple yet nearly optimal deterministic protocol for ℓ_2 regression in the coordinator model: the coordinator calculates A^TA and A^Tb using only $\widetilde{O}(sd^2L)$ bits of communication by collecting the covariance matrices from each server and summing them up. The $\widetilde{O}(sd^2L)$ communication complexity matches our lower bound for solving linear systems for deterministic protocols in the coordinator model. However, when implemented in the blackboard model, the communication complexity of this protocol is still $\widetilde{O}(sd^2L)$. To improve this bound, we first show how to efficiently obtain approximations to leverage scores in both models. Our protocol is built upon the algorithm in [21], but implemented in a distributed manner. The resulting algorithm has $\widetilde{O}(sd^2L)$ communication complexity in the coordinator model but only $\widetilde{O}(s + d^2L)$ communication complexity in the blackboard model. With approximate leverage scores, the coordinator can then sample $\widetilde{O}(d/\varepsilon^2)$ rows of the matrix A to obtain a *subspace embeeding*, at which point it will be easy to calculate a $(1+\varepsilon)$-approximate solution to the ℓ_2 regression problem. The number of sampled rows can be further improved to $\widetilde{O}(d/\varepsilon)$ using Sárlos's argument [45] since solving ℓ_2 regression does not necessarily require a full $(1+\varepsilon)$ subspace embedding, which results in a protocol with communication complexity $\widetilde{O}(s + d^2L/\varepsilon)$ in the blackboard model. Full details can be found in Section 6.

One may wonder if the dependence on $1/\varepsilon$ is necessary for solving ℓ_2 regression in the blackboard model. In Section 5, we show that some dependence on $1/\varepsilon$ is actually necessary. We show an $\Omega(d/\sqrt{\varepsilon})$ lower bound whenever $s > \Omega(1/\sqrt{\varepsilon})$. The hardness follows from the fact that if the matrix A satisfies $A^{(i)} = I$ for all $i \in [s]$, then the optimal solution is just the *average* of $b^{(1)}, b^{(2)}, \ldots, b^{(s)}$. Thus, if we can get sufficiently

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

good approximation to the ℓ_2 regression problem, then we can actually recover the *sum* of $b^{(1)}, b^{(2)}, \ldots, b^{(s)}$, at which point we can resort to known communication complexity lower bound in the blackboard model [43]. This argument will also give an $\Omega(d/\varepsilon)$ lower bound for $(1+\varepsilon)$-approximate ℓ_1 regression in the blackboard model, whenever $s > \Omega(1/\varepsilon)$. Details can be found in Section 5.

For ℓ_1 regression, we can no longer use the normal equations. However, we can obtain approximations to ℓ_1 Lewis weights by using approximations to leverage scores, as shown in [22]. With approximate ℓ_1 Lewis weights of the A matrix, the coordinator can then obtain a $(1+\varepsilon)$ ℓ_1 subspace embedding by sampling $\widetilde{O}(d/\varepsilon^2)$ rows. This will give an $O(sd^2L + d^2L/\varepsilon^2)$ upper bound for $(1+\varepsilon)$-approximate ℓ_1 regression in the coordinator model, and an $O(s + d^2L/\varepsilon^2)$ upper bound in the blackboard model. It is unclear if the number of sampled rows can be further reduced since there is no known ℓ_1 version of Sárlos's argument. A natural question is whether the $1/\varepsilon^2$ dependence is optimal. We show that the dependence on ε can be further improved to $1/\varepsilon$, by using optimization techniques, or more specifically, first-order methods. Despite the fact that the objective function of ℓ_1 regression is neither smooth nor strongly-convex, it is known that by using Nesterov's Accelerated Gradient Descent and smoothing reductions [42], one can solve ℓ_1 regression using only $O(1/\varepsilon)$ full gradient calculations. On the other hand, the complexity of first-order methods usually has dependences on various parameters of the input matrix A, which can be unbounded in the worst case. Fortunately, recent developments in ℓ_1 regression [27] show how to *precondition* the matrix A by simply doing an ℓ_1 Lewis weights sampling, and then rotating the matrix appropriately. By carefully combining this preconditioning procedure with Accelerated Gradient Descent, we obtain an algorithm for $(1+\varepsilon)$-approximate ℓ_1 regression with communication complexity $\widetilde{O}(sd^3L/\varepsilon)$ in the coordinator model, which shows it is indeed possible to improve the ε dependence for ℓ_1 regression. More details can be found in Section 7.

For general ℓ_p regression, if we still use Lewis weights sampling, then the number of sampled rows and thus the communication complexity will be $\Omega(d^{p/2})$. Even worse, when $p = \infty$, Lewis weights sampling will require an unbounded number of samples. However, ℓ_∞ regression can be easily formulated as a linear program, which we show how to solve exactly in the distributed setting. Inspired by this approach, we further develop a general reduction from ℓ_p regression to linear programming. Our idea is to use the max-stability of exponential random variables [2] to embed ℓ_p into ℓ_∞, write the optimization problem in ℓ_∞ as a linear program and then solve the problem using linear program solvers. However, such embeddings based on exponential random variables usually produce heavy-tailed random variables and makes the dilation bound hard to analyze. Here, since our goal is just to solve a linear regression problem, we only need the dilation bound for the optimal solution of the regression problem. In Section 8, we show that $(1+\varepsilon)$-approximate ℓ_p regression can be reduced to solving a linear program with $\widetilde{O}(d/\varepsilon^2)$ variables, which implies a communication protocol for ℓ_p regression without the $\Omega(d^{p/2})$ dependence.

1.2.3 Linear and Convex Programs

We adapt two different algorithms from the literature for efficient communication and implement them in the distributed setting. The first is Clarkson's algorithm, which works by sampling $O(d^2)$ constraints in each iteration and finds an optimal solution to this subset; the sampling weights are maintained implicitly. In each iteration the total communication is $O(d^3L)$ for gathering the constraints and an additional $\widetilde{O}(sd^2L)$ per round to send the solution to this subset of constraints to all servers. This solution is used to update the sampling weights. Clarkson's algorithm has the nice guarantee that it needs only $O(d \log n)$ rounds with high probability. A careful examination of this algorithm shows that the bit complexity of the computation (not the communication) is dominated by checking whether a proposed solution satisfies all constraints, i.e., computing Ax for a given x. We show this can be done with time complexity $\widetilde{O}(nd^\omega L)$ in the unit cost RAM model and this is the leading term of the claimed time bound.

Notice that the $\widetilde{O}(sd^3L)$ term in the communication complexity of Clarkson's algorithm comes from the fact that the protocol needs to send an optimal solution x^* of a linear program with size $O(d^2) \times d$ for a total of $O(d \log n)$ times. However, when each server P_i receives x^*, all P_i will do is to check whether x^* satisfies the constraints stored on P_i or not. Notice that here entries in the constraints have bit complexity L, whereas the solution vector x^* has bit complexity $\widetilde{O}(dL)$ for each entry. Intuitively, for most linear programs, we don't need such a high precision for the solution vector x^*. This leads to the idea of smoothed analysis. We show that if the coefficients of A in the input to the linear program are perturbed independently by i.i.d. discrete Gaussians with variance as small as $2^{-\Theta(L)}$, then we can improve the upper bound for solving this perturbed problem to $\widetilde{O}(sd^2L + d^4L)$. The reason here is that with Gaussian noise, we can round each entry of the solution vector x^* to have bit complexity $\widetilde{O}(L)$,

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

which would suffice for verifying whether x^* satisfies the constraints or not, for most linear programs. Details regarding Clarkson's algorithm can be found in Section 10. The smoothed analysis model and running time of Clarkson's lgorithm in the unit cost RAM model can be found in the full version [53].

One minor drawback of Clarkson's algorithm is it has a dependence on $\log n$. In constant dimensions, our $\widetilde{\Omega}(s + L)$ lower bound in the blackboard model holds only when $n = 2^{\Omega(L)}$, in which case the communication complexity of Clarkson's algorithm will be $\widetilde{O}(sL + L^2)$.

Under the additional assumption that each constraint in the linear program is placed on a random server, we develop an algorithm with communication complexity $\widetilde{O}(s + L)$ in the blackboard model. To achieve this goal, we modify Seidel's classical algorithm and implement it in the distributed setting. Seidel's algorithm benefits from the additional assumption from two aspects. On the one hand, Seidel's classical algorithm needs to go through all the constraints in a random order, which can be easily achieved now since all constraints are placed on a random server. On the other hand, Seidel's classical algorithm needs to make a recursive call each time it finds one of d constraints that determines the optimal solution, and will make $\sum_{i=1}^{n} d/i = \Theta(d \log n)$ recursive calls in expectation. To implement Seidel's algorithm in the distributed setting, each time we find one of the d constraints that determines the optimal solution, the current server also needs to broadcast that constraint. Thus, naïvely we need to broadcast $O(d \log n)$ constraints during the execution, which would result in $O(s + L \log n)$ communication. Under the additional assumption, with good probability, the first server P_1 stores at least $\Omega(n/s)$ constraints. Since the first server P_1 does not need to make any recursive calls or broadcasts, the total number of recursive calls (and thus broadcasts) will be $\sum_{i=\Omega(n/s)}^{n} d/i = \Theta(d \log s)$. The formal analysis is given in the full version.

For convex programming, we have to use a more general algorithm. We use a refined version of the classical center-of-gravity method. The basic idea is to round violated constraints that are used as cutting planes to $O(d \log d)$ bits. We optimize over the ellipsoid method in the following two ways. First, we round the violated constraint sent in each iteration by locally maintaining an ellipsoid to ensure the rounding error does not affect the algorithm. Roughly speaking, each server maintains a well-rounded current feasible set, and the number of bits needed in each round is thus only $\widetilde{O}(d)$. Secondly, we use the center of gravity method to make sure the volume is cut by a constant factor rather than a $(1 - 1/d)$ factor in each iteration, even when constraints are rounded. See Section 11 for details.

2 Preliminaries

2.1 Notation For m matrices $A^{(1)} \in \mathbb{R}^{d \times n_1}, A^{(2)} \in \mathbb{R}^{d \times n_2}, \ldots, A^{(m)} \in \mathbb{R}^{d \times n_m}$, we use $[A^{(1)}\ A^{(2)}\ \cdots\ A^{(m)}]$ to denote the matrix in $\mathbb{R}^{d \times (n_1 + n_2 + \cdots + n_m)}$ whose first n_1 columns are the same as $A^{(1)}$, the next n_2 columns are the same as $A^{(2)}$, ..., and the last n_m columns are the same as $A^{(m)}$.

For a matrix $A \in \mathbb{R}^{n \times d}$, we use $\text{span}(A) = \{Ax \mid x \in \mathbb{R}^d\}$ to denote the subspace spanned by the columns of the matrix A. For a set of vectors $S \subseteq R^d$, we use $\text{span}(S)$ to denote the subspace spanned by the vectors in S. For a set of linear equations C, we also $\text{span}(C)$ to denote all linear combinations of linear equations in C. We use A_i to denote the i-th column of A and A^i to denote the i-th row of A. We use $A^\dagger$ to denote the Moore-Penrose inverse of A. We use $\text{rank}(A)$ to denote the rank of A over the real numbers and $\text{rank}_p(A)$ to denote the rank of A over the finite field $\mathbb{F}_p$.

For a vector $x \in \mathbb{R}^d$, we use $\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{1/p}$ to denote its ℓ_p norm. For two vectors x and y, we use $\langle x, y \rangle$ to denote their inner product.

For matrices A and B, we say $A \approx_\kappa B$ if and only if $\frac{1}{\kappa} B \preceq A \preceq \kappa B$, where $\preceq$ refers to the Löwner partial ordering of matrices, i.e., $A \preceq B$ if $B - A$ is positive semi-definite.

2.2 Models of Computation and Problem Settings We study the distributed linear regression problem in two distributed models: the coordinator model (a.k.a. the message passing model) and the blackboard model. The coordinator model represents distributed computation systems with point-to-point communication, while the blackboard model represents those where messages can be broadcasted to each party.

In the *coordinator model*, there are $s \geq 2$ servers $P_1, P_2, \ldots, P_s$, and one coordinator. These s servers can directly send messages to the coordinator through a two-way private channel. The computation is in terms of rounds: at the beginning of each round, the coordinator sends a message to some of the s servers, and then each of those servers that have been contacted by the coordinator sends a message back to the coordinator.

In the alternative *blackboard model*, the coordinator is simply a blackboard where the s servers $P_1, P_2, \ldots, P_s$ can share information; in other words, if one server sends a message to the coordinator/blackboard then the other $s - 1$ servers can see this information without further communication. The order for the servers to send messages is decided by the contents of the blackboard.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

For both models we measure the *communication cost* which is defined to be the total number of bits sent through the channels.

In the *distributed linear system problem*, there is a data matrix $A \in \mathbb{R}^{n\times d}$ and a vector b of observed values. All entries in A and b are integers between $[-2^L, 2^L]$, where L is the *bit complexity*. The matrix $[A\ b]$ is distributed row-wise among the s servers $P_1, P_2, \ldots, P_s$. More specifically, for each server P_i, there is a matrix $[A^{(i)}\ b^{(i)}]$ stored on P_i, which is a subset of rows of $[A\ b]$. Here we assume $\{[A^{(1)}\ b^{(1)}], [A^{(2)}\ b^{(2)}], \ldots, [A^{(s)}\ b^{(s)}]\}$ is a *partition* of all rows in $[A\ b]$. The goal of the *feasibility testing* problem is to design a protocol, such that upon termination of the protocol, the coordinator reports whether the linear system $Ax = b$ is feasible or not. The goal of the *linear system solving* problem is to design a protocol, such that upon termination of the protocol, either the coordinator outputs a vector $x^* \in \mathbb{R}^d$, such that $Ax^* = b$, or the coordinator reports the linear system $Ax = b$ is infeasible. It can be seen that the linear system solving problem is strictly harder than the feasibility testing problem.

In the *distributed linear regression problem*, there is a data matrix $A \in \mathbb{R}^{n\times d}$ and a vector b of observed values, which is distributed in the same way as in the distributed linear system problem. The goal of the distributed ℓ_p regression problem is to design a protocol, such that upon termination of the protocol, the coordinator outputs a vector $x^* \in \mathbb{R}^d$ to minimize $\|Ax - b\|_p$.

In the *distributed linear programming problem*, there is a matrix $A \in \mathbb{R}^{n\times d}$ and a vector b, which is distributed in the same way as in the distributed linear system problem. The goal of the *feasibility testing* problem is to design a protocol, such that upon termination of the protocol, the coordinator reports whether the linear program $Ax \le b$ is feasible or not. In the *linear programming solving* problem, the goal is to design a protocol, such that upon termination of the protocol, the coordinator outputs a vector $x^* \in \mathbb{R}^d$ such that $Ax^* \le b$ is satisfied. There can also be a vector $c \in \mathbb{R}^d$ which is known to all servers, and in this case the goal is to minimize (or maximize) $\langle c, x\rangle$ under the constraint that $Ax \le b$.

2.3 Row Sampling Algorithms

Definition 2.1. ([21]) *Given a matrix $A \in \mathbb{R}^{n\times d}$. The leverage score of a row A^i is defined to be $\tau_i(A) = A^i(A^TA)^\dagger(A^i)^T$. Given another matrix $B \in \mathbb{R}^{n'\times d}$, the generalized leverage score of a row A^i w.r.t. B is defined to be*

$$\tau_i^B(A) = \begin{cases} A^i(B^TB)^\dagger(A^i)^T & \text{if } A^i \perp \ker(B), \\ \infty & \text{otherwise.} \end{cases}$$

Definition 2.2. ([22]) *Given a matrix $A \in \mathbb{R}^{n\times d}$. The ℓ_1 Lewis weights $\{\overline{w}_i\}_{i=1}^n$ are the unique weights such that for each $i \in [n]$ we have $\overline{w}_i = \tau_i\left(\overline{W}^{-1/2}A\right)$, where $\overline{W}$ is the diagonal matrix formed by putting $\{\overline{w}_i\}_{i=1}^n$ on the diagonal.*

Theorem 2.1. ([21]) *There exists an absolute constant C such that for any matrix $A \in \mathbb{R}^{n\times d}$ and any set of sampling values p_i satisfying $p_i \ge C\tau_i(A)\log d\varepsilon^{-2}$, if we generate a matrix S with $N = \sum_{i=1}^n p_i$ rows, each chosen independently as the i-th basis vector, times $p_i^{-1/2}$ with probability p_i/N, then with probability at least 0.99, for all vector $x \in \mathbb{R}^d$, $(1-\varepsilon)\|Ax\|_2 \le \|SAx\|_2 \le (1+\varepsilon)\|Ax\|_2$.*

Theorem 2.2. ([22]) *There exists an absolute constant C such that for any matrix $A \in \mathbb{R}^{n\times d}$ and any set of sampling values p_i satisfying $p_i \ge C\overline{w}_i \log d\varepsilon^{-2}$, if we generate a matrix S with $N = \sum_{i=1}^n p_i$ rows, each chosen independently as the i-th basis vector, times p_i^{-1} with probability p_i/N, then with probability at least 0.99, for all vectors $x \in \mathbb{R}^d$, $(1-\varepsilon)\|Ax\|_1 \le \|SAx\|_1 \le (1+\varepsilon)\|Ax\|_1$. Here $\{\overline{w}_i\}_{i=1}^n$ are the ℓ_1 Lewis weights of the matrix A.*

3 Communication Complexity Lower Bound for Linear Systems

3.1 The Hard Instance In this section, we construct a family of matrices, which will be used to prove a communication complexity lower bound in the subsequent section.

We need the following theorem on the singularity probability of discrete random matrices.

Theorem 3.1. *Let $M_n \in \mathbb{R}^{n\times n}$ be a matrix whose entries are i.i.d. random variables with the same distribution as the summation of t random signs, for sufficiently large t, $\Pr[M_n$ is singular$] \le t^{-Cn}$, where $C > 0$ is an absolute constant.*

The proof of Theorem 3.1 closely follows previous approaches for bounding the singularity probability of random ±1 matrices (see, e.g., [33, 50, 51, 12].). For completeness, we include a proof of Theorem 3.1 in the full version [53].

Lemma 3.1. *For any $d > 0$ and sufficiently large t, there exists a set of matrices $\mathcal{T} \subseteq \mathbb{R}^{d\times(d-1)}$ with integral entries in $[-t, t]$ for which $|\mathcal{T}| = t^{\Omega(d^2)}$ and*

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

1. *For any $T \in \mathcal{T}$, $\operatorname{rank}(T) = d - 1$;*

2. *For any $S, T \in \mathcal{T}$ such that $S \neq T$, $\operatorname{span}([S\ T]) = \mathbb{R}^d$.*

Proof. We use the probabilisitic method to prove the existence. The proof can be found in the full version [53]. □

The following lemma can be easily proved using Lemma 3.1. See the full version [53] for the formal proof.

Lemma 3.2. *For any $d > 0$ and sufficiently large t, there exists a set of matrices $\mathcal{H} \subseteq \mathbb{R}^{d\times d}$ with integral entries in $[-t, t]$ for which $|\mathcal{H}| = t^{\Omega(d^2)}$ and*

1. *For any $T \in \mathcal{H}$, T is non-singular;*

2. *For any $S, T \in \mathcal{H}$, $S^{-1}e_d \neq T^{-1}e_d$, where e_d is the d-th standard basis vector.*

3.2 Deterministic Lower Bound for the Equality Problem In this section, we prove our deterministic communication complexity lower bound for the Equality problem in the coordinator model, which will be used as an intermediate problem in Section 3.3. In the Equality problem, each server P_i receives a binary string $t_i \in \{0, 1\}^n$. The goal is to test whether $t_1 = t_2 = \ldots = t_s$. We will prove an $\Omega(sn)$ lower bound for deterministic communication protocols. The case $s = 2$ has a well-known $\Omega(n)$ lower bound.

Lemma 3.3. (See, e.g., [36, p11]) *Any deterministic protocol for solving the Equality problem with $s = 2$ requires $\Omega(n)$ bits of communication.*

Our plan is to reduce the case $s = 2$ to the case $s > 2$, using the symmetrization technique [43, 57]. In the full version [53], we prove the following theorem.

Theorem 3.2. *Any deterministic protocol for solving the Equality problem with s servers in the coordinator model requires $\Omega(sn)$ bits of communication.*

3.3 Deterministic Lower Bound for Testing Feasibility of Linear Systems In this section, we present our deterministic communication complexity lower bound for testing the feasibility of linear systems, in the coordinator model and the blackboard model. The proof can be found in the full version [53].

Theorem 3.3. *For any deterministic protocol $\mathcal{P}$,*

- *If $\mathcal{P}$ can test whether $Ax = b$ is feasible or not in the coordinator model, then the communication complexity of $\mathcal{P}$ is $\Omega(sd^2L)$;*

- *If $\mathcal{P}$ can test whether $Ax = b$ is feasible or not in the blackboard model, then the communication complexity of $\mathcal{P}$ is $\Omega(s + d^2L)$;*

3.4 Randomized Lower Bound for Solving Linear Systems In this section, we prove randomized communication complexity lower bounds for solving linear systems. We first prove an $\Omega(d^2L)$ lower bound, which already holds for the case $s = 2$. When $s = 2$ the coordinator model and the blackboard model are equivalent in terms of communication complexity, and thus we shall not distinguish these two models in the remaining part of this proof.

Consider the set $\mathcal{H}$ constructed in Lemma 3.2 with $t = 2^L$. In the hard instance, only server P_1 receives a matrix $H \in \mathcal{H}$, and the goal is to let the coordinator output the solution to the linear system $Hx = e_d$. For any two $H_1, H_2 \in \mathcal{H}$ and $H_1 \neq H_2$, we must have $H_1^{-1}e_d \neq H_2^{-1}e_d$. Thus, by standard information-theoretic arguments, in order for the coordinator to output the solution to $Hx = e_d$, the communication complexity is at least $\Omega(\log(|\mathcal{H}|)) = \Omega(d^2L)$. Formally, we have proved the following theorem.

Theorem 3.4. *Any randomized protocol that succeeds with probability at least 0.99 for solving linear systems requires $\Omega(d^2L)$ bits of communication in the coordinator model and the blackboard model. The lower bound holds even when $s = 2$.*

In the full version [53], we prove another lower bound of $\widetilde{\Omega}(sd)$ for solving linear systems in the coordinator model, based on the lower bound for the OR of $s - 1$ copies of the two-player set-disjointness problem proved in [43, 56]. Combining this lower bound and the trivial $\Omega(s)$ lower bound in the blackboard model with Theorem 3.4, we have the following theorem.

Theorem 3.5. *Any randomized protocol that succeeds with probability at least 0.99 for solving linear systems requires $\widetilde{\Omega}(sd + d^2L)$ bits of communication in the coordinator model and $\Omega(s+d^2L)$ bits of communication in the blackboard model.*

4 Communication Protocols for Linear Systems

4.1 Testing Feasibility of Linear Systems In this section we present a randomized communication protocol for testing feasibility of linear systems, which has communication complexity $O(sd^2 \log(dL))$ in the coordinator model and $O(s + d^2 \log(dL))$ in the blackboard model. The protocol is described in Figure 1.

In the full version [53], we prove the following theorem.

Theorem 4.1. *The protocol in Figure 1 is a randomized protocol for testing feasibility of linear systems and has communication complexity $O(sd^2 \log(dL))$ in the coordinator model and $O(s + d^2 \log(dL))$ in the blackboard*

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

1. The coordinator generates a random prime number p in $[2, \text{poly}(dL)]$ and sends p to all servers.
2. Each server P_i tests the feasibility of its *own* linear system. If the linear system is infeasible then P_i terminates the protocol.
3. Each server maintains the same set of linear equations C. Initially C is the empty set.
4. For $i = 1, 2, \ldots, s$
   (a) P_i checks whether the linear system formed by all linear equations in C and all linear equations stored on P_i is feasible over the finite field $\mathbb{F}_p$. P_i terminates the protocol if it is infeasible.
   (b) For each linear equation stored on P_i that is linearly independent with linear equations in C over the finite field $\mathbb{F}_p$, P_i sends that linear equation to all servers, after taking the residual of each entry modulo p. All servers add that linear equation into C.

Figure 1: Randomized protocol for testing feasibility

model. The protocol succeeds with probability at least 0.99.

4.2 Solving Linear Systems In this section we present communication protocols for solving linear systems. We start with deterministic protocols, in which case we can get a protocol with communication complexity $O(sd^2L)$ in the coordinator model and $O(s + d^2L)$ in the blackboard model.

In order to solve linear systems, we can still use the protocol in Figure 1, but we don't use the prime number p any more. In Step 4a of the protocol, we no longer check the feasibility over the finite field. In Step 4b of the protocol, we no longer takes the residual modulo p before sending the linear equations. At the end of the protocol, each server can use the set of linear equations C, which is a maximal set of linear equations of the original linear system, to solve the linear system. The communication complexity is $O(sd^2L)$ in the coordinator model and $O(s + d^2L)$ in the blackboard model since at most d linear equations will be added into the set C, and each linear equation requires $O(dL)$ bits to describe.

Formally, we have proved the following theorem.

THEOREM 4.2. *There exists a deterministic protocol for solving linear systems which has communication complexity $O(sd^2L)$ in the coordinator model and $O(s+d^2L)$ in the blackboard model.*

Now we turn to randomized protocols. We describe a protocol for solving linear systems with communication complexity $\widetilde{O}(d^2L + sd)$ in the coordinator model. The description is given in Figure 2, and we prove the correctness of the protocol and analyze the communication complexity of the protocol in the full version [53].

1. Each server P_i tests the feasibility of their *own* linear system. If the linear system is infeasible then P_i terminates the protocol. Otherwise, each server P_i finds a maximal set of linearly independent linear equations, say S_i.
2. The coordinator maintains a set of linear equations C. At the beginning C is the empty set.
3. For $i = 1, 2, \ldots, s$
   (a) Repeat the followings for $O(\log d)$ times
      i. Server P_i calculates a linear equation $c = \sum_{t \in S_i} r_t \cdot t$, here $\{r_t\}_{t \in S_i}$ is a set of i.i.d. random signs. P_i sends the linear equation c to the coordinator.
      ii. The coordinator terminates the protocol if $C \cup \{c\}$ is infeasible. Otherwise if c is not a linear combination of those linear equations in C, then the coordinator adds c into C, and then goes to Step 3a, i.e., repeats another $O(\log d)$ times.
4. The coordinator obtains the solution by solving all equations in C.

Figure 2: Randomized protocol for solving linear systems in the coordinator model

THEOREM 4.3. *The protocol described in Figure 2 is a randomized protocol for solving linear systems which has communication complexity $\widetilde{O}(sd + d^2L)$ in the coordinator model. Here the $\widetilde{O}(\cdot)$ notation hides only* polylog(dL) *factors. The protocol succeeds with probability at least* 0.99.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

5 Communication Complexity Lower Bounds for Linear Regression in the Blackboard Model

In this section, we prove communication complexity lower bounds for linear regression in the blackboard model.

We first define the k-XOR problem and the k-MAJ problem. In the blackboard model, each server P_i receives a binary string $x_i \in \{0,1\}^d$. In the k-XOR problem, at the end of a communication protocol, the coordinator correctly outputs the coordinate-wise XOR of these vectors, for at least $0.99d$ coordinates. In the k-MAJ problem, at the end of a communication protocol, the coordinator correctly outputs the coordinate-wise majority of these vectors, for at least $0.99d$ coordinates.

We need the following lemma for our lower bound proof, whose proof can be found in the full version [53].

Lemma 5.1. *Any randomized communication protocol that solves the k-XOR problem or the k-MAJ problem and succeeds with probability at least* 0.99 *has communication complexity $\Omega(dk)$.*

In the full version [53], we give a reduction from k-MAJ to $(1+\varepsilon)$-approximate ℓ_1 regression in the blackboard model, and prove an $\Omega(d/\varepsilon)$ lower bound when $s > \Omega(1/\varepsilon)$. We also give a reduction from k-XOR to $(1+\varepsilon)$-approximate ℓ_2 regression in the blackboard model, and prove an $\Omega(d/\sqrt{\varepsilon})$ lower bound when $s > \Omega(1/\sqrt{\varepsilon})$. These reductions, together with Lemma 5.1, lead to the following theorems.

Theorem 5.1. *When $s > \Omega(1/\varepsilon)$, any randomized protocol that succeeds with probability at least* 0.99 *for solving $(1+\varepsilon)$-approximate ℓ_1 regression requires $\Omega(d/\varepsilon)$ bits of communication in the blackboard model.*

Theorem 5.2. *When $s > \Omega(1/\sqrt{\varepsilon})$, any randomized protocol that succeeds with probability at least* 0.99 *for solving $(1+\varepsilon)$-approximate ℓ_2 regression requires $\Omega(d/\sqrt{\varepsilon})$ bits of communication in the blackboard model.*

6 Communication Protocols for ℓ_2 Regression

In this section, we design distributed protocols for solving the ℓ_2 regression problem.

6.1 A Deterministic Protocol In this section, we design a simple deterministic protocol for ℓ_2 regression in the distributed setting with communication complexity $\widetilde{O}(sd^2L)$ in the coordinator model.

According to the normal equations, the optimal solution to the ℓ_2 regression problem $\min_x \|Ax-b\|_2$ can be attained by setting $x^* = (A^TA)^\dagger A^Tb$. In Figure 3, we show how to calculate A^TA and A^Tb in the distributed model.

1. Each server P_i calculates $(A^{(i)})^TA^{(i)}$ and $(A^{(i)})^Tb^{(i)}$, and then sends them to the coordinator.
2. The coordinator calculates $A^TA = \sum_{i=1}^s (A^{(i)})^TA^{(i)}$ and $A^Tb = \sum_{i=1}^s (A^{(i)})^Tb^{(i)}$, and then calculates $x = (A^TA)^\dagger \cdot A^Tb$.

Figure 3: Protocol for ℓ_2 regression in the coordinator model

Notice that the bit complexity of entries in A^TA and A^Tb is $O(L+\log n)$ since the bit complexity of entries in A and b is L, which implies the communication complexity of the protocol in Figure 3 is $O(sd^2(L+\log n))$, in both the coordinator model and the blackboard model.

Theorem 6.1. *The protocol in Figure 3 is a deterministic protocol which exactly solves ℓ_2 regression, and the communication complexity is $O(sd^2(L+\log n))$, in both the coordinator model and the blackboard model.*

6.2 A Protocol in the Blackboard Model In this section, we design a recursive protocol for obtaining constant approximations to leverage scores in the distributed setting, which is described in Figure 4. We then show how to solve ℓ_2 regression by using this protocol.

The protocol described in Figure 4 is basically Algorithm 2 in [21] for approximating leverage scores, implemented in the distributed setting. We analyze the communication complexity of the protocol in the full version [53].

Lemma 6.1. *The protocol described in Figure 4 is a randomized protocol with communication complexity $\widetilde{O}(sd^2L)$ in the coordinator model and $\widetilde{O}(s+d^2L)$ in the blackboard model, such that with constant probability, upon termination of the protocol, each server P_i has constant approximations to leverage scores of all rows in $A^{(i)}$.*

Our protocol for solving the ℓ_2 regression problem in the blackboard model is described in Figure 5. We analyze the communication complexity of the protocol in the full version [53].

Theorem 6.2. *The protocol described in Figure 5 is a randomized protocol which returns a $(1+\varepsilon)$-approximate solution to ℓ_2 regression with constant probability, and the communication complexity is $\widetilde{O}(s+d^2L/\varepsilon)$ in the blackboard model .*

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Input: An $n \times d$ matrix $A = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(s)} \end{bmatrix}$, where $A^{(i)}$ is stored on server P_i.

Output: An $O(d \log d) \times d$ matrix $\widetilde{A}$ such that

$$\Omega(1)\|Ax\|_2 \le \|\widetilde{A}x\|_2 \le O(1)\|Ax\|_2$$

for all $x \in \mathbb{R}^d$, which is stored on the coordinator and all servers.

1. If $n \le O(d \log d)$, then each server P_i sends $A^{(i)}$ to the coordinator, and then the coordinator sends A to each server, and returns.
2. Each server P_i locally uniformly samples half of the rows from $A^{(i)}$ to form $\left(A^{(i)}\right)'$. Then each server takes $\left(A^{(i)}\right)'$ as input and invokes the protocol recursively to compute $\widetilde{A'}$ for $A' = \begin{bmatrix} \left(A^{(1)}\right)' \\ \left(A^{(2)}\right)' \\ \vdots \\ \left(A^{(s)}\right)' \end{bmatrix}$ such that $\Omega(1)\|A'x\|_2 \le \|\widetilde{A'}x\|_2 \le O(1)\|A'x\|_2$ for all $x \in \mathbb{R}^d$.
3. Using $\widetilde{A'}$, each server P_i calculates generalized leverage scores (Definition 2.1) of all rows in $A^{(i)}$ with respect to $\widetilde{A'}$.
4. The coordinator and all servers obtain $\widetilde{A}$ by sampling and rescaling $O(d \log d)$ rows using Theorem 2.1, by setting $p_i \ge C\widetilde{\tau}_i \log d$, where

$$\widetilde{\tau}_i = \begin{cases} \tau_i^{\widetilde{A}}(A) & \text{if row } A^i \text{ is sampled in Step 2,} \\ \frac{1}{1+\frac{1}{\tau_i^{\widetilde{A}}(A)}} & \text{otherwise.} \end{cases}$$

Figure 4: Protocol for approximating leverage scores in the distributed setting

7 Communication Protocols for ℓ_1 Regression

In this section, we design distributed protocols for solving the ℓ_1 regression problem.

7.1 A Simple Protocol In this section, we design a simple protocol for obtaining a $(1+\varepsilon)$-approximate solution to the ℓ_1 regression problem in the distributed setting. The protocol is described in Figure 6. We prove its correctness and analyze the communication complexity in the full version [53].

1. Use the protocol in Figure 4 to approximate leverage scores of A.
2. The coordinator obtains SA and Sb by sampling and rescaling $O(d/\varepsilon + d \log d)$ rows of $[A\ b]$, using the sampling process in Theorem 2.1.
3. The coordinator calculates $x = \min_x \|SAx - Sb\|_2$.

Figure 5: Protocol for ℓ_2 regression in the blackboard model

1. Each server P_i calculates an $O(d \log d/\varepsilon^2) \times n$ matrix $S^{(i)}$ such that for all $x \in \mathbb{R}^d$, $(1-\varepsilon)\|A^{(i)}x - b^{(i)}\|_1 \le \|S^{(i)}A^{(i)}x - S^{(i)}b^{(i)}\|_1 \le (1+\varepsilon)\|A^{(i)}x - b^{(i)}\|_1$, and then sends $S^{(i)}A^{(i)}$ and $S^{(i)}b^{(i)}$ to the coordinator.
2. Let $\widetilde{A} = \begin{bmatrix} S^{(1)}A^{(1)} \\ S^{(2)}A^{(2)} \\ \vdots \\ S^{(s)}A^{(s)} \end{bmatrix}, \widetilde{b} = \begin{bmatrix} S^{(1)}b^{(1)} \\ S^{(2)}b^{(2)} \\ \vdots \\ S^{(s)}b^{(s)} \end{bmatrix}$. The coordinator solves $\min_x \|\widetilde{A}x - \widetilde{b}\|_1$.

Figure 6: Protocol for ℓ_1 regression in the coordinator model

THEOREM 7.1. *The protocol described in Figure 6 is a deterministic protocol which returns a $(1+\varepsilon)$-approximate solution to the ℓ_1 regression problem, and the communication complexity is $\widetilde{O}(sd^2L/\varepsilon^2)$ in both the coordinator model and the blackboard model.*

7.2 A Protocol Based on ℓ_1 Lewis Weights Sampling In this section, we first design a protocol for obtaining constant approximations to ℓ_1 Lewis weights in the distributed setting, which is described in Figure 7, and then solves the ℓ_1 regression problem based on this protocol.

The protocol described in Figure 7 is basically the

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

1. Each server P_i initializes $w_i = 1$ for all rows in $A^{(i)}$.
2. For $t = 1, 2, \dots, T$
   (a) Each server P_i obtains constant approximations to leverage scores of $W^{-1/2}A$, for all rows stored on P_i using the protocol in Lemma 6.1, where W is the diagonal matrix formed by putting the elements of w on the diagonal.
   (b) Set $w_i = (w_i \tau_i(W^{-1/2}A))^{1/2}$.

Figure 7: Protocol for approximating ℓ_1 Lewis weights

algorithm in Section 3 of [22] for approximating ℓ_1 Lewis weights, implemented in the distributed setting. Using the same analysis, by setting $T = O(\log \log n)$, we can show w_i are constant approximations to the ℓ_1 Lewis weights of A. We analyze its communication complexity in the full version [53].

Lemma 7.1. *The protocol described in Figure 7 is a randomized protocol with communication complexity $\widetilde{O}(s + d^2L)$ in the blackboard model and $\widetilde{O}(sd^2L)$ in the blackboard model, such that with constant probability, upon termination of the protocol, each server P_i has constant approximations to the ℓ_1 Lewis weights of all rows in $A^{(i)}$.*

Our protocol for solving the ℓ_1 regression problem in the blackboard model is described in Figure 8. We prove its correctness and analyze the communication complexity in the full version [53].

1. Use the protocol in Figure 7 to approximate ℓ_1 Lewis weights of $[A\ b]$.
2. The coordinator obtains SA and Sb by sampling and rescaling $O(d \log d/\varepsilon^2)$ rows of $[A\ b]$, using the sampling process in Theorem 2.2.
3. The coordinator calculates $x = \min_x \|SAx - Sb\|_1$.

Figure 8: Protocol for ℓ_1 regression in the blackboard model

Theorem 7.2. *The protocol described in Figure 8 is a randomized protocol which returns a $(1+\varepsilon)$-approximate solution to the ℓ_1 regression problem with constant probability, and the communication complexity is $\widetilde{O}(s + d^2L/\varepsilon)$ in the blackboard model and $\widetilde{O}(sd^2L + d^2L/\varepsilon^2)$ in the coordinator model.*

7.3 A Protocol Based on Accelerated Gradient Descent

In the full version, we present a protocol for the ℓ_1 regression problem in the coordinator model based on accelerated gradient descent, whose communication complexity is $\widetilde{O}(sd^3L/\varepsilon)$.

Theorem 7.3. *There is a randomized protocol which returns a $(1+\varepsilon)$-approximate solution to the ℓ_1 regression problem with constant probability, and the communication complexity is $\widetilde{O}(sd^3L/\varepsilon)$ in the coordinator model.*

8 Communication Protocols for ℓ_p Regression

In this section, we design distributed protocols for solving the ℓ_p regression problem, including $p = \infty$.

8.1 Communication Protocols for ℓ_∞ Regression

Any ℓ_∞ regression instance $\min_x \|Ax - b\|_\infty$ can be formulated as the following linear program,

$$\begin{aligned} &\text{minimize } v \\ &\text{subject to } \langle A^i, x\rangle - b^i \le v, \\ &\qquad\qquad\quad \langle A^i, x\rangle - b^i \ge -v, \end{aligned}$$

which has $2n$ constraints and $d+1$ variables. Thus, any linear programming protocol implies a protocol for solving the ℓ_∞ regression problem, with the same communication complexity. Using the linear program solvers in Section 10 and Section 11, we have the following theorem.

Theorem 8.1. *ℓ_∞ regression can be solved deterministically and exactly with communication complexity $\widetilde{O}(sd^3L)$ in the coordinator model, and randomly and exactly with communication complexity $\widetilde{O}(\min\{sd + d^4L, sd^3L\})$ in the blackboard model.*

8.2 Communication Protocols for ℓ_p Regression When $p > 2$

In the full version [53], we develop an approach that reduces $(1+\varepsilon)$-approximate ℓ_p regression to linear programs with $\widetilde{O}(d/\varepsilon^2)$ variables. Using the linear program solvers in Section 10 and Section 11, we have the following theorem.

Theorem 8.2. *$(1+\varepsilon)$-approximate ℓ_p regression can be solved by a randomized protocol with communication complexity $\widetilde{O}(sd^3L/\varepsilon^6)$ in the coordinator model, or*

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

by a randomized protocol with communication complexity $\widetilde{O}(\min\{sd^3L/\varepsilon^6, sd/\varepsilon^2 + d^4L/\varepsilon^8\})$ in the blackboard model.

9 Communication Complexity Lower Bound for Linear Programming

In this section, we prove a communication complexity lower bound for testing feasibility of linear programs.

We need the following lemma to construct our hard instance.

LEMMA 9.1. *Let L be a sufficiently large integer. We use $m_i \in \mathbb{R}^2$ to denote the vector $m_i = \left(\frac{i}{2^L}, 1 - \frac{i^2}{2\cdot 4^L}\right)$. For any $1 \le i, j \le 2^{L/100}$, we have*

1. $\|m_i\|_2^2 \ge 1 + \frac{1}{2^{4L+2}}$;
2. *For any $i \neq j$, $\langle m_i, m_j \rangle \le 1$.*

Now we reduce the *lopsided set disjiontness* problem to testing feasibility of linear programs. In this problem, for a choice of universe size U, the last server P_s receives an element $u \in [U]$, and for each $i < s$, server P_i receives a set $S_i \subseteq [U]$. The goal is to test whether there exists i such that $u \in S_i$. We reduce this problem with $U = 2^{L/100}$ to testing the feasibility of linear programs for $d = 2$, where L is the bit complexity of the linear program.

For the reduction, server P_s adds a constraint $x = m_u$, for the element $u \in [U]$ that P_s receives. I.e., server P_s forces the solution x to be m_u. For each $i < s$, for each $v \in S_i$, server P_i adds a constraint $\langle m_v, x\rangle \le 1$. Here m_u and m_v are as defined in Lemma 9.1. By Lemma 9.1, this linear program is feasible if and only if $u \notin \bigcup_{i<s} S_i$.

In the full version [53], we show the lopsided set disjointness problem has an $\Omega(s \log U / \log s)$ randomized communication complexity lower bound in the coordinator Formally, we have the following theorem.

THEOREM 9.1. *Any randomized protocol that succeeds with probability at least 0.99 for testing feasibility of linear programs requires $\Omega(s \log L / \log s)$ bits of communication in the coordinator model and $\Omega(s+L)$ bits of communication in the blackboard model. The lower bound holds even when $d = 2$.*

Notice that by Theorem 4.1, testing feasibility of linear systems for $d = 2$ requires only $O(s \log L)$ randomized communication complexity. This shows an exponential separation between testing feasibility of linear systems and linear programs, in the communication model.

10 Clarkson's Algorithm

In this section, we discuss how to implement Clarkson's algorithm to solve linear programs in the distributed setting. The protocol is described in Figure 9. During the protocol, each server P_i maintains a multi-set H_i of constraints (i.e., each constraint can appear more than once in H_i). Initially, H_i is the set of constraints stored on P_i. Furthermore, the coordinator maintains $|H_i|$, which is initially set to be the number of constraints stored on each server.

1. The coordinator obtains $9d^2$ constraints R, by sampling uniformly at random from $H_1 \cup H_2 \cup \ldots \cup H_s$.
2. The coordinator calculates the optimal solution x_R, which is the optimal solution to the linear program satisfying all constraints in R. The coordinator sends x_R to each server.
3. Each server P_i calculates the total number of constraints that are stored on P_i and violated by x_R, i.e., $|V_i|$ where $V_i = \{h \in H_i \mid x_R \text{ violates } h\}$.
4. The coordinator calculates $|V| = \sum_{i=1}^{s} |V_i|$ where $V = \{h \in H \setminus R \mid x_R \text{ violates } h\}$ and sends $|V|$ to each server.
   (a) If $|V| = 0$ then x_R is a feasible solution and the protocol terminates.
   (b) If $|V| \le \frac{2}{9d-1}|H|$, then each server updates $H_i \leftarrow H_i \cup V_i$ and the coordinator updates $|H_i| \leftarrow |H_i| + |V_i|$.
   (c) Goto Step 1.

Figure 9: Clarkson's Algorithm

The protocol in Figure 9 is basically Clarkson's algorithm [18], implemented in the distributed setting. Using the analysis in [18], the expected number of iterations is $O(d \log n)$. The correctness also directly follows from the analysis in [18]. We analyze its communication complexity in the full version [53].

THEOREM 10.1. *The expected communication complexity of the protocol in Figure 9 is $\widetilde{O}(sd^3L + d^4L)$ in the coordinator model and $\widetilde{O}(sd + d^4L)$ in the blackboard model*

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

11 The Center of Gravity Method

In this section, we discuss how to implement the center-of-gravity cutting-plane method [30] in the distributed setting. The description of the protocol can be found in Figure 10.

The servers each maintain a polytope P (the same one for all servers), adding a constraint in each iteration. Each server also maintains the center of the polytope z and its covariance C.

For any vector $a \in \mathbb{R}^d$, its ε-rounding $\widetilde{a}$ w.r.t. to C is defined as follows: Let $B = C^{1/2}$. We take the unit vector $B^T a / \|B^T a\|_2$, round it down to the nearest multiple of ε in each coordinate. So we have $\|\widetilde{a} - B^T a/\|B^T a\|_2\|_2 \leq \varepsilon\sqrt{d}$.

1. Set $P = [-R, R]^n, z = 0, C = I$.
2. Repeat, while no feasible solution found, and at most T times:
   (a) Set z to be the centroid of P and C to be the covariance matrix of the uniform distribution over P, i.e., $C = \mathrm{E}_{x \sim P}(x - z)(x - z)^T$.
   (b) Each server P_i checks z against their subset of constraints. If a violated constraint $a \cdot z > b_i$ is found, and no violation has been reported by other servers so far, then it rounds a to $\widetilde{a}$ and broadcasts. If a violation has been reported, it uses the $\widetilde{a}$ broadcast by the server that first found the violation.
   (c) Set $P = P \cap \{x : C^{-1/2}\widetilde{a} \cdot x \leq C^{-1/2}\widetilde{a} \cdot z + \varepsilon d^{3/2} \|C^{-1/2}\widetilde{a}\|_2\}$.

Figure 10: Center-of-Gravity Algorithm

If each server were to report the exact violated constraint, the volume of P would drop by a constant factor in each iteration. To reduce the communication, we round the constraint and shift it away a bit to make sure that the rounded constraint (1) is still valid for the target LP and (2) it is close enough that the volume still drops by a constant factor.

Lemma 11.1. ([9]) *Let z be the center of gravity of an isotropic convex body K in $\mathbb{R}^d$. Then, for any halfspace H within distance t of z, we have*

$$\mathsf{vol}(K \cap H) \geq \left(\frac{1}{e} - t\right) \mathsf{vol}(K).$$

The proof of the following lemma can be found in the full version [53].

Lemma 11.2. *For $\varepsilon < 0.1/d\sqrt{d}$, the volume of the polytope P maintained by each server drops by a constant factor in each iteration.*

We prove the following two theorems in the full version [53].

Theorem 11.1. *The protocol in Figure 10 is a deterministic protocol for solving linear programming with communication complexity $O(sd^3 L \log^2 d)$ in both the coordinator model and the blackboard model.*

Theorem 11.2. *The communication complexity of the protocol in Figure 10 for solving convex programming is $O(sd^2 \log d \log(Rd/\varepsilon))$.*

12 Discussion

The lens of communication complexity reveals surprising structure about well-known optimization problems. A very interesting open question is to fully resolve the randomized communication complexity of linear programming as a function of s, d, and L. Another interesting direction is to design more efficient linear programming algorithms in the RAM model with unit cost operations on words of size $O(\log(nd))$ bits; such algorithms while being inherently useful may also give rise to improved communication protocols. While our regression algorithms illustrated various shortcomings of previous techniques, there are still interesting gaps in our bounds to be resolved.

Acknowledgments

Santosh S. Vempala was supported in part by NSF awards CCF-1717349 and DMS-1839323. Ruosong Wang and David P. Woodruff were supported in part by Office of Naval Research (ONR) grant N00014-18-1-2562. Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing.

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

References

[1] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15(1):1111–1133, 2014.

[2] Alexandr Andoni. High frequency moments via max-stability. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 6364–6368. IEEE, 2017.

[3] Alexandr Andoni et al. Eigenvalues of a matrix in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1729–1737. Society for Industrial and Applied Mathematics, 2013.

[4] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1756–1764, 2015.

[5] Sepehr Assadi, Nikolai Karpov, and Qin Zhang. Distributed and streaming linear programming in low dimensions. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 236–253. ACM, 2019.

[6] Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 3–12, 2017.

[7] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, pages 26.1–26.22, 2012.

[8] Maria-Florina Balcan, Yingyu Liang, Le Song, David P. Woodruff, and Bo Xie. Communication efficient distributed kernel principal component analysis. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 725–734, 2016.

[9] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

[10] P Bickel, P Diggle, S Fienberg, U Gather, I Olkin, and S Zeger. Springer series in statistics. 2009.

[11] Avrim Blum and John Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 905–914. Society for Industrial and Applied Mathematics, 2002.

[12] Jean Bourgain, Van H Vu, and Philip Matchett Wood. On the singularity probability of discrete random matrices. *Journal of Functional Analysis*, 258(2):559–603, 2010.

[13] Christos Boutsidis, David P. Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 236–249, 2016.

[14] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[15] Mark Braverman, Ankit Garg, Tengyu Ma, Huy L. Nguyen, and David P. Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1011–1020, 2016.

[16] Emmanuel J Candès and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[17] Jiecao Chen, He Sun, David P. Woodruff, and Qin Zhang. Communication-optimal distributed clustering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3720–3728, 2016.

[18] Kenneth L Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995.

[19] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 205–214, 2009.

[20] M. B. Cohen, Y. Tat Lee, and Z. Song. Solving Linear Programs in the Current Matrix Multiplication Time. In *STOC*, 2019.

[21] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.

[22] Michael B Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 183–192. ACM, 2015.

[23] Graham Cormode, Charlie Dickens, and David P. Woodruff. Leveraging well-conditioned bases: Streaming and distributed summaries in minkowski p-norms. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1048–1056, 2018.

[24] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pages 1647–1655, 2011.

[25] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

[26] John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

[27] David Durfee, Kevin A Lai, and Saurabh Sawlani. ℓ_1 regression using lewis weights preconditioning and stochastic gradient descent. In *Conference On Learning Theory*, pages 1626–1656, 2018.

[28] Ankit Garg, Tengyu Ma, and Huy L. Nguyen. On communication cost of distributed statistical estimation and dimensionality. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2726–2734, 2014.

[29] M. Grotschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer, 1988.

[30] B. Grunbaum. Partitions of mass-distributions and convex bodies by hyperplanes. *Pacific J. Math.*, 10:1257–1261, 1960.

[31] Dirk Van Gucht, Ryan Williams, David P. Woodruff, and Qin Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 199–212, 2015.

[32] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in neural information processing systems*, pages 3068–3076, 2014.

[33] Jeff Kahn, János Komlós, and Endre Szemerédi. On the probability that a random$\pm$1-matrix is singular. *Journal of the American Mathematical Society*, 8(1):223–240, 1995.

[34] Daniel M Kane, Roi Livni, Shay Moran, and Amir Yehudayoff. On communication complexity of classification problems. *arXiv preprint arXiv:1711.05893*, 2017.

[35] Ravi Kannan, Santosh Vempala, and David P. Woodruff. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1040–1057, 2014.

[36] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, New York, NY, USA, 1997.

[37] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433, 2014.

[38] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 230–249, 2015.

[39] Yingyu Liang, Maria-Florina Balcan, Vandana Kanchanapally, and David P. Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3113–3121, 2014.

[40] Dhruv Mahajan, S Sathiya Keerthi, S Sundararajan, and Léon Bottou. A parallel sgd method with strong convergence. *arXiv preprint arXiv:1311.0636*, 2013.

[41] Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 91–100, 2013.

[42] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[43] Jeff M Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 486–501. SIAM, 2012.

[44] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

[45] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

[46] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 850–857. IEEE, 2014.

[47] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008, 2014.

[48] Daniel Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 296–305. ACM, 2001.

[49] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of termination of linear programming algorithms. *Mathematical Programming*, 97(1-2):375–404, 2003.

[50] Terence Tao and Van Vu. On random$\pm$1 matrices:

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited

singularity and determinant. *Random Structures & Algorithms*, 28(1):1–23, 2006.

[51] Terence Tao and Van Vu. On the singularity probability of random bernoulli matrices. *Journal of the American Mathematical Society*, 20(3):603–628, 2007.

[52] John N Tsitsiklis and Zhi-Quan Luo. Communication complexity of convex optimization. *Journal of Complexity*, 3(3):231–243, 1987.

[53] Santosh S Vempala, Ruosong Wang, and David P Woodruff. The communication complexity of optimization. *arXiv preprint arXiv:1906.05832*, 2019.

[54] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[55] David P. Woodruff and Qin Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 546–567, 2013.

[56] David P. Woodruff and Qin Zhang. When distributed computation is communication expensive. In *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, pages 16–30, 2013.

[57] David P Woodruff and Qin Zhang. An optimal lower bound for distinct elements in the message passing model. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 718–733. Society for Industrial and Applied Mathematics, 2014.

[58] David P. Woodruff and Qin Zhang. Distributed statistical estimation of matrix products with applications. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 383–394, 2018.

[59] David P. Woodruff and Peilin Zhong. Distributed low rank approximation of implicit functions of a matrix. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 847–858, 2016.

[60] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.

[61] Yuchen Zhang, John C. Duchi, and Martin J. Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 14(1):3321–3363, 2013.

[62] Yuchen Zhang and Xiao Lin. Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning*, pages 362–370, 2015.

[63] Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 2595–2603, 2010.

Downloaded 12/30/20 to 45.17.162.35. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2020 by SIAM
Unauthorized reproduction of this article is prohibited