Journal of Machine Learning Research 18 (2017) 1-43 Submitted 12/16; Revised 8/17; Published 11/17

Distributed Stochastic Variance Reduced Gradient Methods by Sampling Extra Data with Replacement

Jason D. Lee JASONLEE@MARSHALL.USC.EDU
Marshall School of Business
University of Southern California
Los Angeles, CA 90089, USA

Qihang Lin QIHANG-LIN@UIOWA.EDU
Tippie College of Business
University of Iowa
Iowa City, IA 52242, USA

Tengyu Ma TENGYU@CS.PRINCETON.EDU
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA

Tianbao Yang TIANBAO-YANG@UIOWA.EDU
Department of Computer Science
University of Iowa
Iowa City, IA 52242, USA

Editor: Francis Bach

Abstract

We study the round complexity of minimizing the average of convex functions under a new setting of distributed optimization where each machine can receive two subsets of functions. The first subset is from a random partition and the second subset is randomly sampled with replacement. Under this setting, we define a broad class of distributed algorithms whose local computation can utilize both subsets and design a distributed stochastic variance reduced gradient method belonging to in this class. When the condition number of the problem is small, our method achieves the optimal parallel runtime, amount of communication and rounds of communication among all distributed first-order methods up to constant factors. When the condition number is relatively large, a lower bound is provided for the number of rounds of communication needed by any algorithm in this class. Then, we present an accelerated version of our method whose the rounds of communication matches the lower bound up to logarithmic terms, which establishes that this accelerated algorithm has the lowest round complexity among all algorithms in our class under this new setting.

Keywords: distributed optimization, communication complexity, first-order method, lower bound, stochastic variance reduced gradient

1. Introduction

In this paper, we consider the *distributed optimization* problem of minimizing the average of N convex functions $f_i : \mathbb{R}^d \to \mathbb{R}$ for $i = 1, \ldots, N$, i.e.,

$$\min_{x\in\mathbb{R}^d} \left\{ f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x) \right\} \tag{1}$$

using m machines. For simplicity of notation, we assume $N = mn$ for an integer n but all of our results can be easily generalized for an arbitrary N. In this paper, the norm $\|\cdot\|$ represents the Euclidean norm and $\langle\cdot,\cdot\rangle$ represents the inner product in $\mathbb{R}^d$. Throughout the whole paper, we make the following standard assumptions on problem (1).

Assumption 1 *The functions f_i for $i = 1, \ldots, N$ in (1) satisfy the following properties:*

- *Each function f_i is convex and L-smooth, which means f_i is differentiable and its gradient ∇f_i is L-Lipschitz continuous, i.e., $\|\nabla f_i(x)-\nabla f_i(y)\| \le L\|x-y\|$, $\forall x, y \in \mathbb{R}^d$.*
- *The average function f is μ-convex with $\mu \ge 0$, i.e., $f(x) \ge f(y) + \langle \nabla f(y), x - y\rangle + \frac{\mu}{2}\|x - y\|^2$, $\forall x, y \in \mathbb{R}^d$.*

When $\mu > 0$, f is *μ-strongly convex* and we call $\kappa = \frac{L}{\mu}$ the *condition number* of f. Note that the function f itself can be L_f-smooth, namely, $\|\nabla f(x)-\nabla f(y)\| \le L_f\|x-y\|$, $\forall x, y \in \mathbb{R}^d$, for a constant $L_f \le L$. Let x^* be an optimal solution of (1) and a deterministic solution $\hat{x}$ is called an *ϵ-optimal solution* to (1) if $f(\hat{x}) - f(x^*) \le \epsilon$. If $\hat{x}$ is a random variable generated by a stochastic algorithm, we call it an ϵ-optimal solution if $\mathbb{E}[f(\hat{x}) - f(x^*)] \le \epsilon$.

One of the most important applications of problem (1) is *empirical risk minimization* (ERM) in statistics and machine learning. Let $\{\xi_1, \xi_2, \ldots, \xi_N\}$ be a set of i.i.d. samples from an unknown distribution D. A regularized ERM problem with can be formulated as

$$\min_{x\in\mathbb{R}^d} \left\{ f(x) = \frac{1}{N}\sum_{i=1}^{N} \phi(x, \xi_i) + r(x) \right\}, \tag{2}$$

where x represents a vector of parameters of a predictive model, $\phi(x, \xi)$ is a loss function, and $r(x)$ is a regularization term. Note that (2) has the form of (1) with $f_i(x) = \phi(x, \xi_i) + r(x)$. A commonly used regularization term is $r(x) = \frac{\lambda}{2}\|x\|^2$ and the regularization parameter λ significantly influences μ and κ. The value of λ is typically in the order of $\Theta(1/\sqrt{N}) = \Theta(1/\sqrt{mn})$ as justified by Shamir and Srebro (2014), Shamir et al. (2014), Shalev-Shwartz et al. (2009) and Zhang and Lin (2015). For simplicity, in the rest of the paper, we will call f_i the ith data point or the ith function interchangeably.

We consider a situation where all N functions are initially stored in a large storage space that has limited computation power. Then, these N functions are partitioned, sampled and distributed to m machines where the main computation is performed for solving (1). One common practice under a classical setting is to randomly and evenly partition the N functions into subsets $S_1, S_2, \ldots, S_m$ with $|S_j| = \frac{N}{m} = n$ and store S_j in machine j. In this paper, we consider a *new setting* where an extra set of functions R_j with $|R_j| \le O(|S_j|)$ is sampled with replacement from these N functions so that the set $S_j \cup R_j$ is stored in

machine j for the local computation. Since no machine can access all N functions, we consider solving (1) by synchronized distributed algorithms that alternate between a local computation procedure at each machine, and a round of communication to share information among the machines.

One contribution of this paper is to develop a distributed algorithm for (1) under this new setting. Compared to the previous methods where only S_j is stored in machine j, our method requires more storage space or machines but significantly fewer rounds of communication to achieve an ϵ-optimal solution for (1) when the problem's parameters are in a practical regime, showing the advantage of the new setting. Moreover, we define a broad class of distributed algorithms for (1) under the new setting, which includes the proposed method, and establish a lower bound for the number of rounds of communication needed by any algorithm in this class to find an ϵ-optimal solution. This lower bound matches the number of rounds required by the proposed method up to logarithmic factors, indicating that our lower bound is tight and the proposed method is nearly optimal in this class.

1.1 Performance Metrics

In the design of distributed algorithms, one often needs to consider more performance metrics than in the single-machine scenario. A priori, the most important performance metric is the total runtime of algorithm—the time difference between the start of the first machine and the end of the last machine. However, there are other factors than the machines' CPU time that affect the total runtime such as the communication between machines. Moreover, the latency in initiating communication is also non-negligible, and often the bottleneck of the entire system.

We use the following synchronous message passing model from the distributed computation literature, e.g., Gropp et al. (1996) and Dean and Ghemawat (2008): We assume the communication occurs in rounds—in each round, machines exchanges messages with each other and, between two rounds, machines only perform computation based on their local information (local data points and messages received before). Under this model, three metrics are of main interests to us:

- **Parallel runtime**: The longest running time of the m machines spent in local parallel computation. In distributed first-order optimization methods, we measure it by the number of gradient computations, i.e., $\nabla f_i(x)$ computed for any i and x, in parallel.
- **The amount of communication**: The total amount of communication among m machines, measured by the number of bits transmitted through the network. It does not include the bits of the communication for distributing functions to machines before an algorithm starts. For the algorithms that only communicate iterates x and gradients $\nabla f_i(x)$, the amount of communication can be measured by the number of vectors of size $O(d)$ transmitted through the network.
- **The number of rounds of communication**: How many times m machines have to pause their local computation and initiate a round of communication to exchange messages. We also refer it as "rounds" or "round complexity" for simplicity.

Typically, trade-offs exist among these three metrics and algorithms need to be designed with a balance of these metrics. In this paper, we mostly focus on the third metric, the

number of rounds of communication an algorithm needs for solving (1), which has a significant impact on the total runtime of an algorithm when the speed of network transmission is slow or initiating the communication between machines in each round is time consuming.

1.2 Main Contributions

Distributed algorithms with low round complexity: First, we propose a *distributed stochastic variance reduced gradient* (DSVRG) method in the new distributed setting where machine j has access to two subsets of $\{f_i\}_{i\in[N]}$: a subset S_j of size n from a random partition of $\{f_i\}_{i\in[N]}$ and a subset R_j of size αn with $\alpha > 0$ sampled with replacement from $\{f_i\}_{i\in[N]}$. This method is a simple distributed implementation of the single-machine stochastic variance reduced gradient (SVRG) method (Johnson and Zhang, 2013; Mahdavi et al., 2013; Xiao and Zhang, 2014; Konečnỳ and Richtárik, 2017). We show in Theorem 14 that the DSVRG method with $\alpha = \Theta(1)$ requires $O\left(\left(\frac{\kappa}{n}\right)\log\left(\frac{1}{\epsilon}\right)\right)$ rounds of communication to find an ϵ-optimal solution for (1) in a realistic regime of ϵ and κ that contains most of ERM problems in practices.

Second, using the acceleration techniques developed by Frostig et al. (2015) and Lin et al. (2015), we propose a *distributed accelerated stochastic variance reduced gradient* (DASVRG) method that further improves the theoretical performance of DSVRG. In a practical regime of ϵ and κ similar to DSVRG, we show in Theorem 16 that DASVRG with $\alpha = \Theta(1)$ requires only $O\left(\left(\sqrt{\frac{\kappa}{n}}\right)\log\left(\frac{\kappa}{n}\right)\log\left(\frac{1}{\epsilon}\right)\right)$ rounds of communication to find an ϵ-optimal solution, which is lower than the round complexity of many existing techniques in the traditional setting with $R_j = \emptyset$, showing the advantage of the new distributed setting. The proposed algorithms may require more machines than previous approaches in order to have extra space for storing R_j in additional to S_j. However, when $\alpha = \Theta(1)$ (e.g., $\alpha = 1$), the required number of machines will increase only by a constant factor under the new setting.

Round complexity lower bounds: We also contribute to the answer of a fundamental question about the round complexity for solving (1) under the new setting:

When each machine receives functions from both random partition and sampling with replacement, what is the minimum number of rounds of communication a distributed optimization algorithm needs in order to obtain an ϵ-optimal solution for (1)?

Since different algorithms utilize different operations in local computation which may influence how many rounds are needed, a class of algorithms must be specified before we can answer the question above affirmatively. Hence, we define a broad class of algorithms called *distributed (extended) first-order algorithm*, denoted by $\mathcal{F}_\alpha$ (see Definition 1).

In $\mathcal{F}_\alpha$ with $\alpha \geq 0$, machine j has access to the subsets S_j and R_j of $\{f_i\}_{i\in[N]}$ as described in the DSVRG and DASVRG methods above with $|R_j| = \alpha n$. One realistic setting is when $\alpha \leq O(1)$ so that the memory space required in each machine remains $|R_j| + |S_j| = \Theta(|S_j|) = \Theta(n)$. It is only interesting to consider $\alpha < m - 1$, since, otherwise, $|R_j| + |S_j| \geq N$ which means the memory space of machine j is large enough to store all the N functions so that distributed computing is no longer needed. In $\mathcal{F}_\alpha$, each machine locally maintains a set of vectors W which can be shared with other machines and merged the W sets sent from other machines during a round of communication. Between the rounds of communication, machine j can add to W arbitrarily many linear combinations of the vectors in W and the (proximal) gradients of the functions in $S_j \cup R_j$ evaluated on the

vectors in W. With flexible schemes of communication and computation, the class $\mathcal{F}_\alpha$ with $\alpha = 0$ (i.e., $R_j = \emptyset$) covers many exist methods in the classical setting. When $\alpha > 0$, the class $\mathcal{F}_\alpha$ covers the proposed DSVRG and DASVRG methods.

We show that, when f is μ-strongly convex ($\mu > 0$), any algorithm in $\mathcal{F}_\alpha$ needs at least $\tilde{\Omega}\big(\sqrt{\frac{\kappa/n+\alpha}{(\alpha+1)^3}}\log\big(\frac{1}{\epsilon}\big)\big)$ rounds of communication in order to find an ϵ-optimal solution. Here and elsewhere in the paper, $\tilde{\Omega}(\cdot)$ and $\tilde{O}(\cdot)$ hide a logarithmic factor of m. In the scenario where $\alpha \leq O(1)$, this lower bound becomes $\tilde{\Omega}\left(\sqrt{\frac{\kappa}{n}}\log\left(\frac{1}{\epsilon}\right)\right)$, which suggests that the round complexity of DASVRG with $\alpha = \Theta(1)$ is nearly optimal and our lower bound is tight up to logarithmic factors for a practical regime of parameters. Briefly speaking, this lower bound is proved by carefully designing the "worst" $\{f_i\}_{i\in[N]}$ such that, with a high probability, the random subset $S_j \cup R_j$ in any single machine j does not contain enough (first-order) information of f in (1) so that many rounds of communication must be performed. Since this proof is much more challenging than the round complexity analysis of the DSVRG and DASVRG methods and is the main technical result of this paper, we provide it first in the main body of the paper before introducing the new algorithms.

Optimal parallel runtime, amount of communication, and rounds: In some practical regime of parameters, we show that our DSVRG algorithm can be tuned so that it achieves the optimal parallel runtime, the optimal amount of communication, and the optimal number of rounds of communication *simultaneously* for solving (1). Concretely, when $\kappa = O(n^{1-2\delta})$ with a constant $0 < \delta < \frac{1}{2}$, with appropriate choices of the control parameters (shown in Proposition 15), DSVRG finds an ϵ-optimal solution with a parallel runtime of $O(n)$, an $O(m)$ amount of communication and $O(1)$ rounds of communication for any $\epsilon = \Omega(\frac{1}{n^s})$ where s is any universal positive constant.

Note that $\kappa = O(n^{1-2\delta})$ is a common setting for machine learning applications. For example, as argued in Shamir and Srebro (2014), Shamir et al. (2014), Shalev-Shwartz et al. (2009) and Zhang and Lin (2015), ERM often has the condition number κ in the order of $\Theta(\sqrt{N}) = \Theta(\sqrt{mn})$. Therefore, when the number of machines m is not too large, e.g., when $m \leq n^{0.8}$, we have that $\kappa = \Theta(\sqrt{mn}) \leq n^{0.9}$ (so that $\delta = 0.05$).[1] Moreover, $\epsilon = n^{-10}$ (so that $s = 10$) is certainly a high enough accuracy for most machine learning applications since it exceeds the statistically optimal accuracy.

These performance guarantees of DSVRG, under the specific setting where $\kappa = O(n^{1-2\delta})$ and $\epsilon = \Omega(\frac{1}{n^s})$, are optimal up to constant factors among all the distributed first-order methods. First, to solve (1), all m machines together need to compute at least $\Omega(N)$ gradients (Agarwal and Bottou, 2015) in total so that each function in $\{f_i\}_{i\in[N]}$ can be accessed at least once. Therefore, at least one machine needs to compute at least $\Omega(n)$ gradients in serial given any possible allocation of functions. Second, the amount of communication is at least $\Omega(m)$ for even simple Gaussian mean estimation problems (Braverman et al., 2016), which can be solved as optimization (1). Third, at least $O(1)$ rounds of communication is needed to integrate the computation results from machines into a final output.

Extension to non-strongly convex problem: Although we mainly focus on strongly convex problems, some of our results can be extended to the non-strongly convex case. Using a similar approach to the strongly convex case, we show that when $\mu = 0$, any algorithm in $\mathcal{F}_\alpha$ needs at least $\tilde{\Omega}\Big(\sqrt{\frac{L}{(\alpha+1)^3 n\epsilon}}\Big)$ rounds of communication to find an

1. If $n = 10^5$, then $n^{0.8} = 10^4$ which is already much more than the number of machines in most clusters.

ϵ-optimal solution for (1). In the scenario where $\alpha \leq O(1)$, this lower bound becomes $\tilde{\Omega}\Big(\sqrt{\frac{L}{n\epsilon}}\Big)$. To achieve of this lower bound, we apply DASVRG to the strongly convex problem $\min_{x\in\mathbb{R}^d}\{f(x) + \frac{\sigma_\epsilon}{2}\|x\|^2\}$ with $\sigma_\epsilon = \Theta(\epsilon)$. According to the round complexity of DASVRG in the strongly convex case, we show in Theorem 17 that, in a certain regime of ϵ and κ, an ϵ-optimal solution can be found by DASVRG with $\alpha = \Theta(1)$ in $O\Big(\Big(\sqrt{\frac{L}{n\epsilon}}\Big)\log\Big(\frac{L}{n\epsilon}\Big)\log\big(\frac{1}{\epsilon}\big)\Big)$ rounds of communication, showing that our lower bound is tight up to logarithmic factors in that regime.

1.3 Notations and Outline

For a positive integer K, let $[K]$ denote the set $\{1, 2, \ldots, K\}$. We define a *multi-set* as a collection of items where some items can be repeated. We say two multi-sets are equal if they contain the same sets of unique elements and each unique element appears for the same number of times in both multi-sets. For a multi-set R, the notation $R\backslash\{i\}$ represents a multi-set identical to R except that the occurrence of i, if $i \in R$, is reduced by one. The union of a regular set S and a multi-set R, denoted by $S \cup R$, is defined as a regular set consisting of the items in either S or R without repetition. Let $A \otimes B$ denote the Kronecker product of matrices A and B and let $\mathbf{1}_n$ denotes the all-one vector in $\mathbb{R}^n$.

The rest of this paper is organized as follows. In Section 2, we review the related literature and compare the theoretical performance of our methods with existing work in distributed optimization. In Section 3, we define a broad family of distributed optimization algorithms and state our main results on the lower bounds for the rounds of communication for this family. In Section 4, we provide the proof of the main results. In Section 5 and Section 6, we present the DSVRG and DASVRG algorithms, respectively, and show their theoretical guarantees. We briefly describe how to apply DASVRG to non-strongly convex problems in Section 7. Finally, we present some numerical results in Section 8 and conclude the paper in Section 9.

2. Related Work

The work most closely related to our paper is Arjevani and Shamir (2015), where a lower bound for the rounds of communication was established for solving

$$\min_{x\in\mathbb{R}^d}\left\{\frac{1}{m}\sum_{j=1}^{m}\bar{f}_j(x)\right\} \tag{3}$$

using a broad class of distributed algorithms when machine j has only access to the local function $\bar{f}_j(x)$ for $j = 1, 2, \ldots, m$. To connect (3) to (1), we can define the local function as

$$\bar{f}_j(x) := \frac{1}{|S_j|}\sum_{i\in S_j} f_i(x) \tag{4}$$

for a given partition $\{S_j\}_{j\in[m]}$ of $\{f_i\}_{i\in[N]}$. Arjevani and Shamir (2015) proved that, if the local functions $\{\bar{f}_j\}_{j\in[m]}$ are δ-related (see Arjevani and Shamir (2015) for the definition) and f is strongly convex, the class of algorithms they considered needs at least $\Omega\big(\big(\sqrt{\delta\kappa}\big)\log\big(\frac{1}{\epsilon}\big)\big)$

rounds of communication to find an ϵ-optimal solution for (3). When $\delta = \Omega(1)$, their lower bound can be achieved by a straightforward centralized distributed implementation of accelerated gradient methods. In a specific context of linear regression with functions in S_j being a i.i.d. sample, it was shown by Shamir et al. (2014) that $\delta = O(\frac{1}{\sqrt{n}})$. In this case, the lower bound by Arjevani and Shamir (2015) becomes $O(\frac{\sqrt{\kappa}}{n^{.25}}\log(\frac{1}{\epsilon}))$, which has already been achieved by DISCO (Zhang and Lin, 2015) and AIDE (Reddi et al., 2016).

With our notation, the aforementioned algorithms all belong to the family $\mathcal{F}_0$ where each machine only receives the subset S_j from the random partition of $\{f_i\}_{i\in[N]}$. Unlike all previous work, the DSVRG and DASVRG algorithms we propose belong to $\mathcal{F}_\alpha$ with $\alpha > 0$ so that each machine has access to a subset R_j sampled with replacement from $\{f_i\}_{i\in[N]}$ in addition to S_j. Moreover, in the setting of Arjevani and Shamir (2015), machine j processes the entire local average function $\bar{f}_j$ in (4) in the local computation while machine j in our algorithms can process each individual function in $S_j \cup R_j$. Because of these key differences, DSVRG and DASVRG do not fall into the class of algorithms subject to the lower bound in Arjevani and Shamir (2015). That's why the communication lower bound we establish for $\mathcal{F}_\alpha$ and the communication upper bound ensured by DASVRG can be both lower than the lower bound by Arjevani and Shamir (2015). The "worst" set of functions $\{f_i\}_{i\in[N]}$ constructed in our main proof is a generalization of the set of functions in Arjevani and Shamir (2015). Chen et al. (2017) established lower bounds for the round complexity when the data is partitioned on features which is different from the setting in our paper and Arjevani and Shamir (2015) where the data is partitioned on samples.

Recently, there have been many distributed optimization algorithms proposed for problem (1) when f is strongly convex. We list several of them, including a distributed implementation of the accelerated gradient method (Accel Grad) by Nesterov (2013), in Table 1 and present their rounds and key assumptions for a clear comparison. Except DSVRG and DASVRG, all algorithms in Table 1 belong to $\mathcal{F}_0$ (i.e., $R_j = \emptyset$).

The $O(\sqrt{\kappa_f}\log(\frac{1}{\epsilon}))$ rounds of communication ($\kappa_f := \frac{L_f}{\mu}$) of Accel Grad in Table 1 is directly from the iteration complexity of the single-machine accelerated gradient method, and the $O(\sqrt{\kappa}\log(\frac{1}{\epsilon}))$ round complexity of centralized ADMM was given by Deng and Yin (2016). The distributed dual coordinate ascent method, including DisDCA (Yang, 2013), CoCoA (Jaggi et al., 2014) and CoCoA+ (Ma et al., 2015), is a class of distributed coordinate optimization algorithms which can be applied to the conjugate dual formulation of (2) when $f_i(x) = \phi(x, \xi_i) = \phi_i(\xi_i^T x)$ for ϕ_i on $\mathbb{R}^1$. By Ma et al. (2015) and Jaggi et al. (2014), CoCoA+ requires $O(\kappa\log(1/\epsilon))$ rounds of communication to find an ϵ-optimal solution.[2] Therefore, when applicable, both DSVRG and DASVRG with $\alpha = \Theta(1)$ require fewer rounds of communication than the methods mentioned above.

Assuming the problem (1) has the form of (2) with ξ_i's i.i.d. sampled from a distribution D (denoted by $\xi_i \overset{iid}{\sim} D$), the DANE (Shamir et al., 2014) and DISCO (Zhang and Lin, 2015) algorithms require $O((1+\frac{\sqrt{\kappa}}{n^{.25}})\log(1/\epsilon))$ and $O((1+\frac{\kappa^2}{n})\log(1/\epsilon))$ rounds of communication, respectively. Hence, DSVRG uses fewer rounds of communication than DANE and fewer than DISCO when $\kappa \leq n^{1.5}$. If applicable, DASVRG always uses fewer rounds of communication than DISCO and DANE. Furthermore, DANE only has the theoretical guarantee

2. CoCoA+ has a better theoretical performance than CoCoA. According to Ma et al. (2015), CoCoA+ is equivalent to DisDCA with "practical updates" (Yang, 2013) under certain choices of parameters.

Algorithm	Rounds	Settings (when Assumption 1 holds)
DSVRG	$(1+\frac{\kappa}{n})\log\frac{1}{\epsilon}$	$\kappa\log\left(\frac{1}{\epsilon}\right)\leq O(N)$, $\alpha=\Theta(1)$
DASVRG	$(1+\sqrt{\frac{\kappa}{n}})\log(1+\frac{\kappa}{n})\log\frac{1}{\epsilon}$	$(1+\sqrt{\frac{\kappa}{n}})\log(1+\frac{\kappa}{n})\log\frac{1}{\epsilon}\leq O(m)$, $\alpha=\Theta(1)$
DISCO	$(1+\frac{\sqrt{\kappa}}{n^{.25}})\log\frac{1}{\epsilon}$	(2) with $\xi_i\overset{iid}{\sim}D$ and quadratic $\phi(x,\xi_i)$
	$d^{.25}\left((1+\frac{\sqrt{\kappa}}{n^{.25}})\log\frac{1}{\epsilon}+\frac{\kappa^{1.5}}{n^{.75}}\right)$	(2) with $\xi_i\overset{iid}{\sim}D$
DANE	$\delta^2\kappa^2\log\frac{1}{\epsilon}$	δ-related $\{\bar{f}_j\}_{j\in[m]}$
	$(1+\frac{\kappa^2}{n})\log\frac{1}{\epsilon}$	(2) with $\xi_i\overset{iid}{\sim}D$ and quadratic $\phi(x,\xi_i)$
	$\kappa_f\log\frac{1}{\epsilon}$	
AIDE	$\sqrt{\delta\kappa}\log\frac{1}{\epsilon}$	δ-related $\{\bar{f}_j\}_{j\in[m]}$
	$(1+\frac{\sqrt{\kappa}}{n^{.25}})\log\frac{1}{\epsilon}$	(2) with $\xi_i\overset{iid}{\sim}D$ and quadratic $\phi(x,\xi_i)$
	$\sqrt{\kappa_f}\log\frac{1}{\epsilon}$	
CoCoA+	$\kappa\log\frac{1}{\epsilon}$	(2) with $\phi(x,\xi_i)=\phi_i(\xi_i^Tx)$
Accel Grad	$\sqrt{\kappa_f}\log\frac{1}{\epsilon}$	
ADMM	$\sqrt{\kappa}\log\frac{1}{\epsilon}$	μ-strongly convex f_i

Table 1: Rounds and settings of different distributed optimization algorithms. Except DSVRG and DASVRG, all algorithms in this table only require $\alpha=0$ (i.e., they do not require a subset R_j sampled with replacement).

mentioned above when it is applied to quadratic problems. Also, DISCO only applies to self-concordant functions with easily computed Hessian[3] and makes strong statistical assumptions on the data points. On the contrary, DSVRG and DASVRG works for a more general problem (1) and do not assume $\xi_i\overset{iid}{\sim}D$ for (2).

In each iteration, DANE requires solving a non-trivial sub-problem to optimality. Reddi et al. (2016) proposed an InexactDane algorithm that is similar to DANE but only needs to solve the sub-problem approximately and still achieves the same round complexity as DANE up to a logarithmic factor. Applying the acceleration techniques developed by Frostig et al. (2015) and Lin et al. (2015) to InexactDane, Reddi et al. (2016) further proposed an Accelerated Inexact DanE (AIDE) algorithm that matches the round complexity of DISCO up to a logarithmic factor.

Shamir (2016) consider a without-replacement SVRG which, if implemented in a distributed way, is similar to the DSVRG proposed here. The main difference between his method and DSVRG is that his method sequentially accesses the data points in S_j while while DSVRG accesses the data points in R_j for the local update. Since Shamir (2016) assumes that the concatenation of $\{S_j\}_{j\in[m]}$ forms a random permutation of $\{f_i\}_{[N]}$, his method essentially performs the iterative update of SVRG by sampling from $\{f_i\}_{[N]}$ without replacement. When κ is smaller than N and ϵ is not extremely small (the same as what DSVRG requires), his method obtains the same round complexity as DSVRG. However, his complexity analysis only applies for quadratic $\{f_i\}_{[N]}$. Konečný et al. (2016) proposed a

3. The examples in Zhang and Lin (2015) all take the form of $f_i(x)=\phi_i(\xi_i^Tx)$ for some function ϕ_i on $\mathbb{R}^1$, which is more specific than $\phi(x,\xi_i)$, so that it is relatively easy to compute the Hessian of f_i.

Federated SVRG algorithm which is also a distributed implementation of SVRG and can be viewed as a special case of InexactDane with specific choices of parameters (Reddi et al., 2016). However, the performance analysis of InexactDane under those specific choices of parameters is only available when $\{f_i\}_{[N]}$ are quadratic and δ-related.

The methods in comparison in Table 1 are centralized methods which are typically implemented in a star network with a center server. There also exist many distributed (sub)gradient methods (Nedić and Ozdaglar, 2007, 2009; Duchi et al., 2012; Jakovetić et al., 2012; Chen and Ozdaglar, 2012; Jakovetić et al., 2014; Beck et al., 2014; Shi et al., 2015a,b; Aybat et al., 2015) and distributed ADMM methods (Boyd et al., 2011; Wei and Ozdaglar, 2012; Mota et al., 2013; Shi et al., 2014; Chang et al., 2015; Makhdoumi and Ozdaglar, 2017; Deng and Yin, 2016; Mokhtari et al., 2016) for decentralized optimization over general networks. In general, the decentralized distributed methods require more rounds of communication than the centralized ones. For example, the number of communication rounds needed by the decentralized ADMM in Makhdoumi and Ozdaglar (2017) is $O(\sqrt{\kappa}\log(\frac{1}{\epsilon}))$ multiplied by a network-dependent factor that is typically greater than one.

3. Lower Bounds on Rounds of Communication

In this section, we first define a broad family of distributed (extended) first-order methods which contains many existing distributed optimization algorithms in the literature. Then, we prove the lower bounds for the round complexity for this family of algorithms to find an ϵ-optimal solution for (1) under Assumption 1 with $\mu > 0$ and $\mu = 0$, respectively.

The algorithms in the family we consider consist of a data distribution stage where the functions $\{f_i\}_{i\in[N]}$ are distributed onto m machines, and a distributed computation stage where, in each round, machines can not only use first-order information of the functions stored locally but also apply preconditioning using local second-order information.

Definition 1 (Distributed (extended) first-order algorithms $\mathcal{F}_\alpha$) *Suppose there is a stage of data distribution where $\{f_i\}_{i\in[N]}$ are distributed to m machines such that:*

1. *The index set $[N]$ is randomly and evenly partitioned into $S_1, S_2, \ldots, S_m$ with $|S_j| = n$ for $j = 1, \ldots, m$.*

2. *A multi-set R_j of size αn is created by sampling with replacement from $[N]$ for $j = 1, \ldots, m$, where $\alpha \geq 0$ is a constant. When $\alpha = 0$, we have $R_j = \emptyset$ for all j.*

3. *Let $S_j' = S_j \cup R_j$. Machine j acquires functions in $\{f_i\}_{i\in S_j'}$ for $j = 1, \ldots, m$.*

We say an algorithm $\mathcal{A}$ for solving (1) belongs to the family $\mathcal{F}_\alpha$ ($\mathcal{A} \in \mathcal{F}_\alpha$) of distributed (extended) first-order algorithms if the machines do the following operations in rounds:

1. *Machine j maintains a local working set of vectors $W_j \subset \mathbb{R}^d$ initialized to be $W_j = \{\mathbf{0}\}$, where $\mathbf{0}$ is the all-zero vector in $\mathbb{R}^d$.*

2. *In each round, for arbitrarily many times, machine j can add any $w \in \mathbb{R}^d$ to W_j if w satisfies (c.f. Arjevani and Shamir (2015))*

$$\gamma w + \nu \nabla F_j(w) \in span\Big\{ w', \nabla F_j(w'), (\nabla^2 F_j(w') + D)w'', (\nabla^2 F_j(w') + D)^{-1} w'' \Big| \\ w', w'' \in W_j, D \text{ is diagonal}, \nabla^2 F_j(w') \text{ and } (\nabla^2 F_j(w') + D)^{-1} \text{ exist} \Big\} \quad (5)$$

for some constants $\gamma \geq 0$ and $\nu \geq 0$ with $\gamma\nu \neq 0$, where $F_j = \sum_{i \in U_j \subset S'_j} f_i$ with an arbitrary subset U_j of S'_j.

3. *At the end of the round, all machines can simultaneously send any vectors in W_j to any other machines, and each machine can add the vectors received from other machines to its local working set.*

4. *The final output is a vector in the linear span of W_j for any j.*

We use $\mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$ to represent the output vector of $\mathcal{A}$ when it is applied to (1) for H rounds with the inputs $\{f_i\}_{i\in[N]}$ and the sets $\{S'_j\}_{j\in[m]}$ generated in the data distribution stage described above.

Besides the randomness due to the data distribution stage, the algorithm $\mathcal{A}$ itself can be a randomized algorithm. Hence, the output $\mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$ can be a random vector. Next, we provide some discussions on this class of algorithms.

In $\mathcal{F}_\alpha$, machines can only share iterates but not data points explicitly. Therefore, the sets $\{S'_j\}_{j\in[m]}$ will not be changed during the algorithm. The class $\mathcal{F}_\alpha$ allows gradient update $w = w' - \eta \nabla F_j(w')$ with $\eta > 0$ in local computation, which corresponds to (5) with $\gamma = 1, \nu = 0$. It also allows proximal mapping update $w = \arg\min_v \frac{1}{2\eta}\|v - w'\|^2 + F_j(v)$ with $\eta > 0$ in local computation, which has an optimality condition $w + \eta \nabla F_j(w) = w'$ and thus corresponds to (5) with $\gamma = 1, \nu = \eta$. As a result, the class $\mathcal{F}_0$ covers distributed Accel Grad, ADMM, DANE and AIDE. Although the algorithms in $\mathcal{F}_\alpha$ can use the local second-order information like $\nabla^2 f_i(x)$ in each machine, Newton's method is still not contained in $\mathcal{F}_\alpha$ since Newton's method requires the access to the global second-order information such as $\nabla^2 f(x)$ (machines are not allowed to share Hessian matrices with each other). That being said, one can still use a distributed iteration method which can multiply $\nabla^2 f_i(x)$ to a local vector in order to solve the inversion of $\nabla^2 f(x)$ approximately. This scheme will lead to a distributed inexact Newton method such as DISCO, which also belongs to $\mathcal{F}_0$. The DSVRG and DASVRG algorithms proposed in this paper belong to $\mathcal{F}_\alpha$ with $\alpha > 0$.

We are ready to present the lower bounds for the rounds of communications.

Theorem 2 *Suppose $m \geq (e + 4\max\{1, \alpha\})^2$ and there exists an algorithm $\mathcal{A} \in \mathcal{F}_\alpha$ with the following property:*

"For any $\epsilon > 0$ and any N convex functions $\{f_i\}_{i\in[N]}$ satisfying Assumption 1 with $\mu > 0$ and $\kappa \geq 1.5n$, there exists H_ϵ such that the output $\hat{x} = \mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H_\epsilon)$ satisfies $\mathbb{E}[f(\hat{x}) - f(x^*)] \leq \epsilon$, where the sets $\{S'_j\}_{j\in[m]}$ are generated as in Definition 1."

Then, with $k \equiv \lceil (e + 4\max\{1, \alpha\}) \log m \rceil$, we must have

$$H_\epsilon \geq \left(\frac{\sqrt{2\kappa/(3n) + k - 1} - \sqrt{k}}{4k\sqrt{k}} \right) \log\left(\left(1 - \frac{1}{e}\right) \frac{\mu\|x^*\|^2}{4\epsilon} \right) \geq \tilde{\Omega}\left(\sqrt{\frac{\kappa/n + \alpha}{(\alpha+1)^3}} \log\left(\frac{\mu\|x^*\|^2}{\epsilon} \right) \right).$$

In the definition of $\mathcal{F}_\alpha$, we allow the algorithm to access both randomly partitioned data and independently sampled data, and allow the algorithm to use local Hessian for preconditioning. This makes our lower bounds in Theorem 2 **stronger**: Even with more options in distributing data and with algorithms more powerful than first-order methods (in terms of the class of operations it can take), the number of rounds needed to find an ϵ-optimal solution still cannot be reduced.

The $\kappa \geq 1.5n$ condition is not critical because, when $\kappa \leq \frac{n^{1-2\delta}}{32} < 1.5n$ for a constant $0 < \delta < \frac{1}{2}$, we will show in Proposition 15 in Section 5.2 that DSVRG only needs $O(\frac{\log(1/\epsilon)}{\delta \log n})$ rounds, which almost meets the trivial lower bound $\Omega(1)$.

A similar result can be obtained for the non-strongly convex case.

Theorem 3 *Suppose $m \geq (e + 4\max\{1, \alpha\})^2$ and there exists an algorithm $\mathcal{A} \in \mathcal{F}_\alpha$ with the following property:*

"For any $\epsilon > 0$ and any N convex functions $\{f_i\}_{i\in[N]}$ satisfying Assumption 1 with $\mu = 0$, there exists H_ϵ such that the output $\hat{x} = \mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H_\epsilon)$ satisfies $\mathbb{E}[f(\hat{x}) - f(x^*)] \leq \epsilon$, where the sets $\{S'_j\}_{j\in[m]}$ are generated as in Definition 1."

Then, with $k \equiv \lceil (e + 4\max\{1, \alpha\}) \log m \rceil$, we must have $H_\epsilon \geq \sqrt{\left(1 - \frac{1}{e}\right) \frac{L\|x^*\|^2}{64k^3 n\epsilon}}$.

4. Proofs of Main Results

In this section, we will provide the proofs for Theorem 2 and 3. Before presenting all technical details, we will first give an informal sketch of the proofs.

4.1 Sketch of Proof for Main Results

We will only provide an informal proof sketch for Theorem 2. The proof of Theorem 3 follows a similar strategy.

Let k be an integer with $k = \Theta(\log(m))$, $\mu' = n\mu$ and $\kappa' = \frac{L}{\mu'}$. For simplicity, we assume $\frac{m}{k}$ is an integer in this proof sketch although Theorem 2 and 3 hold without this assumption. Let $E_0 = \{\mathbf{0}\}$ where $\mathbf{0}$ is the all-zero vector in $\mathbb{R}^b$ and let E_t be the subspace of $\mathbb{R}^b$ consisting of the vectors whose non-zero values only appear in the first t coordinates.

Motivated by the proof for the lower bound of the iteration complexity of first-order methods (Nesterov, 2013), we construct a convex quadratic function $\bar{p} : \mathbb{R}^b \to \mathbb{R}$ and represent it as the average of k convex quadratic functions $p_s : \mathbb{R}^b \to \mathbb{R}$ for $s = 1, 2, \ldots, k$. In particular, we construct $\bar{p}(w) = \frac{1}{2}w^T\Sigma w + q^T w + \frac{\mu'}{2}\|w\|^2 = \frac{1}{k}\sum_{s=1}^k p_s$, where $\bar{p}_s(w) = \frac{1}{2}w^T\Sigma_s w + q_s^T w + \frac{\mu'}{2}\|w\|^2$ for $s = 1, 2, \ldots, k$. We are able to choose Σ, Σ_s, q and q_s here such that the following properties are satisfied

1. The matrix $\Sigma = \frac{1}{k}\sum_{s=1}^k \Sigma_s$. The matrix Σ is tridiagonal and the matrix Σ_s is block diagonal with a 2×2 block and $k - 2$ consecutive zero entries alternatively appearing along its diagonal. The sparsity patterns of Σ and Σ_s can be seen in Figure 1 in an example with $b = 15$ and $k = 3$.

2. The vector $q = \frac{1}{k}\sum_{s=1}^k q_s$ with $q, q_s \in E_1$ for $s = 1, \ldots, k$.

3. Each p_s is L-smooth and $\bar{p}$ is μ'-strongly convex.

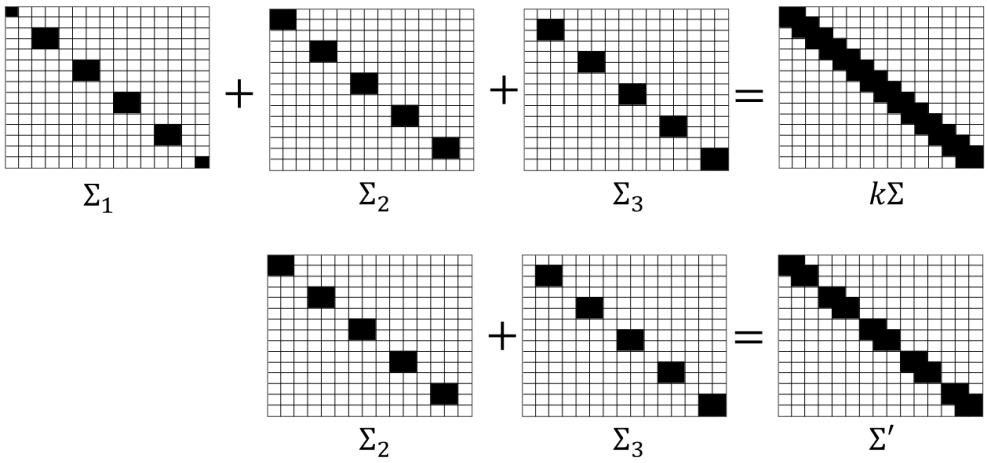

Figure 1: The sparsity pattern of Σ and Σ' when $U = \{2, 3\}$, $b = 15$ and $k = 3$.

4. The vector $w^* = \arg\min_w \bar{p}(w)$ is non-zero in all coordinates and, when b is sufficiently larger than t, $\|\hat{w} - w^*\|^2 \geq \Omega((1 - \frac{1}{\sqrt{\kappa'}})^{2t})$ for any $\hat{w} \in E_t$.

Because the gradient $\nabla\bar{p}(w) = (\Sigma + \mu' I)w + q$, where Σ is tridiagonal and $q \in E_1$, if a first-order method is applied to $\min_w \bar{p}(w)$ with an initial solution of $\mathbf{0}$, the solution will stay in E_t after t iterations. According to the 4th property above, a large number of iterations is needed for the first-order method to achieve an ϵ-optimal solution. We will use a similar argument to show that, when $\min_w \bar{p}(w)$ is solved by a distributed first-order method but each machine does not have a full collection of $\{p_s\}_{s\in[k]}$, a large number of rounds of communication is needed in order to find an ϵ-optimal solution.

Suppose a machine can only locally access the functions in $\{p_s\}_{s\in U}$ with $U \subsetneqq [k]$. By the sparsity patterns of Σ and Σ_s, $\Sigma' = \sum_{s\in U} \Sigma_s$ will be block diagonal with the size of each block being at most $(k + 1) \times (k + 1)$. The sparsity patterns of Σ' can be seen in Figure 1 for an example with $U = \{2, 3\}$, $b = 15$ and $k = 3$. Since $q, q_s \in E_1$, if the local computation of this machine starts with an initial solution in E_t and the solution is updated only using linear combinations of gradients $\nabla p_s(w) = (\Sigma_s + \mu' I)w + q_s$ for $s \in U$, this machine can only produce a solution in E_{t+k} no matter how many local iterations are performed. According to the 4th property above, this machine has to receive a solution outside E_{t+k} from other machines in order to make progress in approximting w^*. Hence, if no machine in a distributed first-order method has a full collection of $\{p_s\}_{s\in[k]}$, many rounds of communication is needed in order to generate an ϵ-optimal solution for $\min_w \bar{p}(w)$.

Since $\bar{p}$ is the average of only $k = \Theta(\log(m))$ functions while the communication lower bound is given for minimizing the average of N functions, we will make a few copies of $\{p_s\}_{s\in[k]}$ and add some zero functions to them. In particular, let $v = \frac{m}{k}$ and $\{g_i\}_{i\in[N]}$ be the multi-set of functions on $\mathbb{R}^b$ that consists of v copies of $\{p_s\}_{s\in[k]}$ and $N - vk$ zero functions. Therefore, $\bar{g}(w) \equiv \frac{1}{N}\sum_{i=1}^N g_i(w) = \frac{\bar{p}(w)}{n}$ so that $w^* = \arg\min_w \bar{g}(w)$. In addition, each g_i is L-smooth due to the 3rd property above and g is $\frac{\mu'}{n}$-strongly (i.e., μ-strongly) convex because m g_i's are μ'-strongly convex and $N - m$ g_i's are zero.

Suppose an algorithm in $\mathcal{F}_\alpha$ is applied to $\min_w \bar{g}(w)$ with the initial solution $\mathbf{0}$. Each machine will receives n functions from $\{g_i\}_{i\in[N]}$ by random partition and another αn func-

tions from $\{g_i\}_{i\in[N]}$ by random sampling with replacement. Since $k = \Theta(\log(m))$ and $\{g_i\}_{i\in[N]}$ contains only $vk = m$ functions from $\{p_s\}_{s\in[k]}$ and $N - m$ zero functions, we can show that, with a high probability, no machine will receive k or more functions from $\{p_s\}_{s\in[k]}$. According to the previous discussion, the number of non-zero coordinate in the solution generated by the algorithm will only increase by at most k between two consecutive rounds of communication. In other words, if $\hat{w}$ is the solution found by $\mathcal{F}_\alpha$ after H rounds of communication, we must have $\hat{w} \in E_t$ with $t = Hk$. By the 4th property above, the distance $\|\hat{w} - w^*\|^2 \geq \Omega((1 - \frac{1}{\sqrt{\kappa'}})^{2t})$. Since $\bar{g}$ is strongly convex, this distance implies an objective function gap of $\bar{g}(\hat{w}) - \bar{g}(w^*) \geq \Omega((1 - \frac{1}{\sqrt{\kappa'}})^{2t})$. Hence, in order to ensure $\bar{g}(\hat{w}) - \bar{g}(w^*) \leq \epsilon$, with a high probably, the rounds of communication must satisfy $H = \frac{t}{k} \geq \Omega(\sqrt{\kappa'}) \log\left(\frac{1}{\epsilon}\right) = \Omega(\sqrt{\frac{\kappa}{n}}) \log\left(\frac{1}{\epsilon}\right)$. This provides the lower bounds for the rounds of communication we found in this paper. Finally, we show that the same lower bound can be proved without having any zero function in the N functions. To do so, we raise the dimension of the problem from b to $d = nb$ by constructing a problem like (1), where $\{f_i\}_{i\in[N]}$ consists of vn copies of $\{p_s\}_{s\in[k]}$ with the first v copies defined on the first b coordinates of x, the second v copies defined on the second b coordinates of x and so on. Problem (1) with such $\{f_i\}_{i\in[N]}$ can be solved as n independent problems on $\mathbb{R}^b$ with each problem equivalent to $\min_w \bar{g}(w)$ so that the same communication lower bound can be obtained.

4.2 Some Definitions and Lemmas

Before we start the formal proof for Theorem 2, some definitions and technical lemmas are provided in this section. Given a vector $x \in \mathbb{R}^d$ and a set of indices $D \subset [d]$, we use x_D to represent the sub-vector of x that consists of the coordinates of x indexed by D.

Definition 4 *A function $f : \mathbb{R}^d \to \mathbb{R}$ is **decomposable** with respect to a partition $D_1, \ldots, D_r$ of coordinates $[d]$ if $f(x) = g^1(x_{D_1}) + \cdots + g^r(x_{D_r})$ with $g^l : \mathbb{R}^{|D_i|} \to \mathbb{R}$ for $l = 1, \ldots, r$. A set of functions $\{f_i\}_{i\in[N]}$ is **simultaneously decomposable** with respect to a partition $D_1, \ldots, D_r$ if each f_i is decomposable with respect to $D_1, \ldots, D_r$.*

It follows the Definition 1 and Definition 4 straightforwardly that:

Proposition 5 *Suppose the functions $\{f_i\}_{i\in[N]}$ in (1) are simultaneously decomposable with respect to a partition $D_1, \ldots, D_r$ so that $f_i(x) = \sum_{l=1}^r g_i^l(x_{D_l})$ with $g_i^l : \mathbb{R}^{|D_l|} \to \mathbb{R}$ for $i = 1, \ldots, N$ and $l = 1, \ldots, r$. Let x^* be the optimal solution of (1). We must have*

$$x^*_{D_l} \in \arg\min_{w\in\mathbb{R}^{|D_l|}} \left\{ \bar{g}^l(w) \equiv \frac{1}{N} \sum_{i=1}^N g_i^l(w) \right\}, \text{ for } l = 1, 2, \ldots, r. \tag{6}$$

Moreover, any algorithm $\mathcal{A} \in \mathcal{F}_\alpha$, when applied to $\{f_i\}_{i\in[N]}$, becomes decomposable with respect to the same partition $D_1, \ldots, D_r$ in the following sense: For $l = 1, \ldots, r$, there exists an algorithm $\mathcal{A}_l \in \mathcal{F}_\alpha$ such that, after any number of rounds H,

$$\mathbb{E}[f(\hat{x}) - f(x^*)] = \sum_{l=1}^r \mathbb{E}[\bar{g}^l(\hat{w}^l) - \bar{g}^l(x^*_{D_l})],$$

where $\hat{x} = \mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H) \in \mathbb{R}^d$ and $\hat{w}^l = \mathcal{A}_l(\{g_i^l\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H) \in \mathbb{R}^{|D_l|}$ for $l = 1, 2, \ldots, r$ and $\{S'_j\}_{j\in[m]}$ is generated as in Definition 1.

Proof The proof of this proposition is straightforward. Since $\{f_i\}_{i\in[N]}$ in (1) are simultaneously decomposable with respect to a partition $D_1, \ldots, D_r$, we have

$$f(x) = \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{r} g_i^l(x_{D_l}) = \sum_{l=1}^{r} \bar{g}^l(x_{D_l})$$

so that the problem (1) can be solved by solving (6) for each l separately and $x^*_{D_l}$ must be the solution of the l-th problem in (6).

In addition, the function F_j in Definition 1 is also decomposable with respect to the same partition $D_1, \ldots, D_r$. As a result, its gradient $\nabla F_j(x)$ also has a decomposed structure in the sense that the block $[\nabla F_j(x)]_{D_l}$ only depends on x_{D_l}. Similarly, its Hessian matrix $\nabla^2 F_j(x)$ is a block diagonal matrix with r blocks and the l-th block only depends on x_{D_l}. These properties ensure that each operation $\mathcal{A}$ is able to apply (as in Definition 1) to some $x \in \mathbb{R}^d$ can be decomposed into r independent operations applied on $x_{D_1}, x_{D_2}, \ldots, x_{D_r}$ separately. Hence, given the sets $\{S'_j\}_{j\in[m]}$, the sequence of operations conducted by $\mathcal{A}$ on x_{D_l} can be viewed as an algorithm $\mathcal{A}_l \in \mathcal{F}_\alpha$ applied to $\{g_i^l\}_{i\in[N]}$ so that $\hat{w}^l = \mathcal{A}_l(\{g_i^l\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$ is indeed the sub-vector $\hat{x}_{D_l}$ of the vector $\hat{x} = \mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$ for $l = 1, 2, \ldots, r$ and any H. ■

Because $m \geq (e + 4\max\{1,\alpha\})^2$ and $k \equiv \lceil (e + 4\max\{1,\alpha\}) \log m \rceil$, we can show that

$$\frac{m}{k} \geq \frac{m}{1.25(e + 4\max\{1,\alpha\})\log m} \geq \frac{e + 4\max\{1,\alpha\}}{2.5\log(e + 4\max\{1,\alpha\})} \geq \frac{e+4}{2.5\log(e+4)} > 3$$

for any $\alpha \geq 0$, where we use the fact that $1.25x \geq \lceil x \rceil$ for $x \geq 4$ and the fact that $\frac{x}{\log x}$ is monotonically increasing on $[e, +\infty)$. In the rest of the paper, we define

$$v \equiv \left\lfloor \frac{m}{k} \right\rfloor \text{ and } \theta \equiv \frac{vk}{m}.$$

It is easy to show that $\frac{2}{3} < 1 - \frac{k}{m} < \theta \leq 1$, where the first inequality is because $\frac{m}{k} > 3$ and the second and the last inequalities are by the definitions of v and θ.

4.3 Proof for Theorem 2

Now we are ready to give the proof for Theorem 2 in this subsection. Let

$$\mu' \equiv \frac{\mu n}{\theta}, \quad \kappa' \equiv \frac{L}{\mu'} = \frac{\theta\kappa}{n} > \frac{2\kappa}{3n} \geq 1. \tag{7}$$

We first generalize the machinery by Arjevani and Shamir (2015) to construct k functions on $\mathbb{R}^b$ where $b = uk$ for some integers $k, u \geq 1$. The values of k and u will be specified later. In particular, for $i, j = 1, \ldots, b$, let $\delta_{i,j}$ be an $b \times b$ matrix with its (i,j) entry being one and others being zeros. Let $M_0, M_1, \ldots, M_{b-1}$ be $b \times b$ matrices defined as

$$M_i = \begin{cases} \delta_{1,1} & \text{for } i = 0 \\ \delta_{i,i} - \delta_{i,i+1} - \delta_{i+1,i} + \delta_{i+1,i+1} & \text{for } 1 \leq i \leq b-2 \\ \delta_{b-1,b-1} - \delta_{b-1,b} - \delta_{b,b-1} + \frac{\sqrt{\kappa'+k-1}+3\sqrt{k}}{\sqrt{\kappa'+k-1}+\sqrt{k}}\delta_{b,b} & \text{for } i = b-1. \end{cases} \tag{8}$$

For $s \in [k]$, let

$$\Sigma_s = \sum_{i=0}^{u-1} M_{ik+s-1}. \tag{9}$$

For example, when $u = 2$ and $k = 3$ (so $b = 6$), the matrices Σ_s's are given as follows

$$\Sigma_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\Sigma_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & \mathcal{K} \end{bmatrix},$$

where $\mathcal{K} = \frac{\sqrt{\kappa'+k-1}+3\sqrt{k}}{\sqrt{\kappa'+k-1}+\sqrt{k}}$.

We define k functions $p_1, \ldots, p_k : \mathbb{R}^b \to \mathbb{R}$ as follows

$$p_s(w) = \begin{cases} \frac{L}{4}\left[\left(1-\frac{\mu'}{L}\right)\frac{w^T\Sigma_1 w}{2} - \left(1-\frac{\mu'}{L}\right)e_1^\top w\right] + \frac{\mu'}{2}\|w\|^2 & \text{for } s = 1 \\ \frac{L}{4}\left[\left(1-\frac{\mu'}{L}\right)\frac{w^T\Sigma_s w}{2}\right] + \frac{\mu'}{2}\|w\|^2 & \text{for } s = 2, \ldots, k, \end{cases} \tag{10}$$

where $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^b$, and denote their average by

$$\bar{p} \equiv \frac{1}{k}\sum_{s=1}^{k} p_s = \frac{L-\mu'}{4k}\left[\frac{1}{2}w^T\left(\sum_{s=1}^{k}\Sigma_s\right)w - e_1^T w\right] + \frac{\mu'}{2}\|w\|^2. \tag{11}$$

According to (7), we have $1 - \frac{\mu'}{L} > 0$ so that p_s for any $s \in [k]$ and $\bar{p}$ are all μ'-strongly convex functions. It is also easy to show that, for each s, $\lambda_{\max}(\Sigma_s) \leq 4$ so that ∇p_s has a Lipschitz continuity constant of $L\left(1-\frac{\mu'}{L}\right)+\mu' = L$ and p_s is L-smooth.

Next, we characterize the optimal solution of $\min_{w\in\mathbb{R}^b} \bar{p}(w)$.

Lemma 6 *Let $h \in \mathbb{R}$ be the smaller root of the equation $h^2 - 2\left(\frac{\kappa'-1+2k}{\kappa'-1}\right)h + 1 = 0$, namely, $h \equiv \frac{\sqrt{\kappa'+k-1}-\sqrt{k}}{\sqrt{\kappa'+k-1}+\sqrt{k}}$. Then, $w^* = (w_1^*, w_2^*, \ldots, w_b^*)^T \in \mathbb{R}^b$ with*

$$w_j^* = h^j, \text{ for } j = 1, 2, \ldots, b \tag{12}$$

is the optimal solutions of $\min_{w\in\mathbb{R}^b} \bar{p}(w)$.

Proof By (9), we have

$$\sum_{s=1}^{k} \Sigma_s = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & \frac{\sqrt{\kappa'+k-1}+3\sqrt{k}}{\sqrt{\kappa'+k-1}+\sqrt{k}} \end{bmatrix}.$$

With this tridiagonal structure in its Hessian matrix, $\bar{p}(w)$ is related to but different from the functions constructed by Nesterov (2013) and Lan and Zhou (2017) to establish the iteration lower bound for single-machine first-order algorithms. We can show that the optimal solution of $\min_{w\in\mathbb{R}^b} \bar{p}(w)$ must satisfy the following optimality conditions

$$\begin{aligned} w_2^* - 2\left(\frac{\kappa'-1+2k}{\kappa'-1}\right) w_1^* + 1 &= 0 \\ w_{j+1}^* - 2\left(\frac{\kappa'-1+2k}{\kappa'-1}\right) w_j^* + w_{j-1}^* &= 0, \text{ for } j = 2,3,\ldots,b-1 \\ -\left(\frac{\sqrt{\kappa'+k-1}+3\sqrt{k}}{\sqrt{\kappa'+k-1}+\sqrt{k}} + \frac{4k}{\kappa'-1}\right) w_b^* + w_{b-1}^* &= 0. \end{aligned} \tag{13}$$

We can easily verify that $w_j^* = h^j$ for $j = 1,2,\ldots,b$ satisfy all equations in (13) such that w^* is the optimal solution of $\min_{w\in\mathbb{R}^b} \bar{p}(w)$. ■

We claim that $\{p_s\}_{s\in[k]}$ in (10) has the following property according to our construction.

Lemma 7 *Suppose U is a* **strict** *subset of $\{p_s\}_{s\in[k]}$ defined in (10), and q is an arbitrary linear combination of some functions p_s in U. The Hessian of q is a block diagonal matrix where each block is a square matrix of a size at most k.*

Proof There must exist some $s' \in [k]$ with $p_{s'} \notin U$. Since q is a linear combination of p_s's in U, according to the construction in (10), the Hessian of q is a tridiagonal matrix because it is a linear combination of a diagonal matrix (the Hessian of $\frac{\mu'}{2}\|w\|^2$) and all the matrices Σ_s defined in (9) except $\Sigma_{s'}$. We note that $\Sigma_{s'}$ is the only matrix among all Σ_s's that has non-zero entries in the positions $(s'-1+ik, s'+ik)$ and $(s'+ik, s'-1+ik)$ for $i = 0,1,\ldots,u-1$ and both the row and column indices of these positions are equally spaced with a gap of k. Therefore, without $\Sigma_{s'}$ involved in the linear combination, the tridiagonal Hessian of q becomes block diagonal with each block of a size at most k. ■

To complete the proof of Theorem 2, the following lemma is critical. This lemma tells us that the property given by Lemma 7 forces the machines to perform a large number of rounds of communication in order to minimize $\bar{p}$ whenever $\{p_s\}_{s\in[k]}$ do not appear together in any machine.

Lemma 8 *For any $b = uk$ with integers $k, u \geq 1$, let $\{g_i\}_{i\in[N]}$ be the multi-set of functions on $\mathbb{R}^b$ that consists of v copies of $\{p_s\}_{s\in[k]}$ defined as (10) and $N - vk$ zero functions. More specifically,*

$$g_i(w) = \begin{cases} p_s(w) & \text{if } i = s, s+k, s+2k, \ldots, s+(v-1)k, \\ 0 & \text{if } i \geq vk+1. \end{cases} \tag{14}$$

Suppose an algorithm $\mathcal{A} \in \mathcal{F}_\alpha$ is applied to $\{g_i\}_{i\in[N]}$ to solve

$$\min_{w\in\mathbb{R}^b} \bar{g}(w) \equiv \frac{1}{N}\sum_{i=1}^{N} g_i \tag{15}$$

and the sets $\{S'_j\}_{j\in[m]}$ are generated as in Definition 1. Let $\mathcal{E}$ be the random event that none of the m machines has all functions in $\{p_s\}_{s\in[k]}$. Let $\hat{w} = \mathcal{A}(\{g_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$. Then, for any $\epsilon > 0$ and $k \geq 1$, there exist u and $b = uk$ such that $\mathbb{E}[g(\hat{w}) - g(w^*)|\mathcal{E}] \leq \epsilon$ only if $H \geq \left(\frac{\sqrt{\kappa'+k-1}-\sqrt{k}}{4k\sqrt{k}}\right)\log\left(\frac{\mu\|w^*\|^2}{4\epsilon}\right)$.

Proof Let $w^* \in \mathbb{R}^b$ defined as (12). Since $\bar{g} = \frac{1}{N}\sum_{i=1}^{N} g_i = \frac{v}{N}\sum_{s=1}^{k} p_s = \frac{\theta\bar{p}}{n}$, we have $w^* = \arg\min_{w\in\mathbb{R}^b} \bar{p}(w) = \arg\min_{w\in\mathbb{R}^b} \bar{g}(w)$ by Lemma 6.

Let $E_0 = \{\mathbf{0}\}$, where $\mathbf{0}$ is the all-zero vector in $\mathbb{R}^b$ and let E_t be the linear space in $\mathbb{R}^b$ spanned by the unit vectors $e_1, \ldots, e_t$ for $t = 1, \ldots, b$, where $e_s \in \mathbb{R}^b$ has one in its s-th component and zeros in other components.

Because $\mathcal{A}$ is applied to (15), by Definition 1, the working set W_j in machine j in each round can be updated for arbitrarily many times by including a vector w such that

$$\gamma w + \nu\nabla F_j(w) \in \text{span}\Big\{ w', \nabla F_j(w'), (\nabla^2 F_j(w') + D)w'', (\nabla^2 F_j(w') + D)^{-1}w'' \Big| \\ w', w'' \in W_j, D \text{ is diagonal}, \nabla^2 F_j(w') \text{ and } (\nabla^2 F_j(w') + D)^{-1} \text{ exist} \Big\} \tag{16}$$

for some constants $\gamma \geq 0$ and $\nu \geq 0$ with $\gamma\nu \neq 0$, where $F_j = \sum_{i\in U_j \subset S'_j} g_i$ with an arbitrary subset U_j of S'_j. Suppose event $\mathcal{E}$ happens such that S'_j does not contain all functions in $\{p_s\}_{s\in[k]}$ for any j. The set U_j in (16) will be a strict subset of $\{p_s\}_{s\in[k]}$ so that, according to Lemma 7, the Hessian matrix $\nabla^2 F_j$ in (16) is block-diagonal with a block size at most k. Because F_j is always a quadratic function, such a property of its Hessian matrix means, as long as W_j is contained by E_t, any vector w satisfying (16) will be in E_{t+k}. Therefore, we can show that, at the beginning of round ℓ in algorithm $\mathcal{A}$, if $\cup_j W_j \subset E_t$, then at the end of round ℓ (and at the beginning of round $\ell+1$), we have $\cup_j W_j \subset E_{t+k}$. Using this finding and the fact that $\cup_j W_j = \{\mathbf{0}\} = E_0$ initially, we conclude that, after H rounds in $\mathcal{A}$, we must have $\cup_j W_j \subset E_{Hk}$. Let $t = Hk$. Since $\hat{w} = \mathcal{A}(\{g_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$, we must have $\hat{w} \in E_t$ if $\mathcal{E}$ happens.

By (12), we can show that

$$\|w^*\|^2 = \sum_{j=1}^{b}(w_j^*)^2 = \sum_{j=1}^{b} h^{2j} = \frac{h^2(1-h^{2b})}{1-h^2}. \tag{17}$$

Because $\bar{p}$ defined in (11) is μ'-strong convex, we have

$$\frac{2}{\mu'}\mathbb{E}[\bar{p}(\hat{w})-\bar{p}(w^*)|\mathcal{E}] \geq \mathbb{E}[\|\hat{w}-w^*\|^2|\mathcal{E}] \geq \sum_{j=t+1}^{b}\mathbb{E}[(w_j^*)^2|\mathcal{E}] \geq \frac{h^{2t+2}(1-h^{2b-2t})}{1-h^2} \geq \frac{\|w^*\|^2(h^{2t}-h^{2b})}{1-h^{2b}}$$

where the second inequality is because $\hat{w} \in E_t$ and the third inequality is due to (17). For any $k \geq 1$, when u is large enough so $b = uk$ is large enough, the inequality above implies $\mathbb{E}[\bar{p}(\hat{w})-\bar{p}(w^*)|\mathcal{E}] \geq \frac{\mu'\|w^*\|^2h^{2t}}{4} = \frac{\mu'\|w^*\|^2}{4}\left(\frac{\sqrt{\kappa'+k-1}-\sqrt{k}}{\sqrt{\kappa'+k-1}+\sqrt{k}}\right)^{2t}$. Based on this inequality, when $\mathbb{E}[\bar{g}(\hat{w})-\bar{g}(w^*)|\mathcal{E}] \leq \epsilon$, or equivalently, when $\mathbb{E}[\bar{p}(\hat{w})-\bar{p}(w^*)|\mathcal{E}] \leq \frac{n\epsilon}{\theta}$, we must have $\log\left(\frac{\mu'\theta\|w^*\|^2}{4n\epsilon}\right) \leq 2t\log\left(1+\frac{2\sqrt{k}}{\sqrt{\kappa'+k-1}-\sqrt{k}}\right) \leq \frac{4t\sqrt{k}}{\sqrt{\kappa'+k-1}-\sqrt{k}}$, which further implies $H = \frac{t}{k} \geq \left(\frac{\sqrt{\kappa'+k-1}-\sqrt{k}}{4k\sqrt{k}}\right)\log\left(\frac{\mu\|w^*\|^2}{4\epsilon}\right)$. ■

In Lemma 8, the requirement of a large enough $b = uk$ is necessary since, if b is fixed and the number of rounds H satisfies $Hk \geq b$, the solution $\hat{w} = \mathcal{A}(\{g_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$ can be non-zero in all dimensions so that it may equal w^*. If this happens, $g(\hat{w})-g(w^*) \leq \epsilon$ for any ϵ so that the lower bound for H in Lemma 8 does not hold. Because the communication lower bound we study does not depend on the dimension of the problem, we construct the functions with a dimension higher than Hk so that there are dimensions that will not be reached by the generated solutions so that the optimality gap has to reduce to zero slowly.

We now complete the proof of Theorem 2 by constructing N special functions $\{f_i\}_{i\in[N]}$ on $\mathbb{R}^d$ with $d = nb$ and $b = uk$ for a sufficiently large u based on $\{p_s\}_{s\in[k]}$ defined as (10), so that any algorithm $\mathcal{A} \in \mathcal{F}_\alpha$, when applied to $\{f_i\}_{i\in[N]}$, will need at least the targeted amount of rounds of communication.

For a large enough u, we partition the set of indices $[d] = \{1,\ldots,d\}$ into n disjoint subsets $D_1, D_2, \ldots, D_n$ with $|D_l| = b$ and $D_l = \{b(l-1)+1, , b(l-1)+2, \ldots, bl\}$. For any $l \in [n]$ and $s \in [k]$, let $q_{l,s}(x)$ be a function on $\mathbb{R}^d$ defined as

$$q_{l,s}(x) \equiv p_s(x_{D_l}), \tag{18}$$

where $\{p_s\}_{s\in[k]}$ is defined as (10). By its definition, the function $q_{l,s}(x)$ only depends on the b coordinates of x indexed by D_l. Therefore, we obtain nk different functions $\{q_{l,s}\}_{l\in[n],s\in[k]}$. Finally, we define $\{f_i\}_{i\in[N]}$ to be a set that consists of v copies of $\{q_{l,s}\}_{l\in[n],s\in[k]}$ and $N-vkn$ zero functions. In other words, the non-zero functions in the set $\{f_i\}_{i\in[N]}$ we constructed in this way constitutes a multi-set as follows

$$\left\{\begin{matrix} p_1(x_{D_1}), & p_2(x_{D_1}), & \cdots, & p_k(x_{D_1}), & \cdots, & p_1(x_{D_1}), & p_2(x_{D_1}), & \cdots, & p_k(x_{D_1}) \\ p_1(x_{D_2}), & p_2(x_{D_2}), & \cdots, & p_k(x_{D_2}), & \cdots, & p_1(x_{D_2}), & p_2(x_{D_2}), & \cdots, & p_k(x_{D_2}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_1(x_{D_n}), & p_2(x_{D_n}), & \cdots, & p_k(x_{D_n}), & \ldots, & p_1(x_{D_n}), & p_2(x_{D_n}), & \cdots, & p_k(x_{D_n}) \end{matrix}\right\}, \tag{19}$$

where the l-th row consists of v copies of $\{p_s\}_{s\in[k]}$ applied on x_{D_l}.

Because of Lemma 6 and the fact that

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) = \frac{v}{N}\sum_{l=1}^{n}\sum_{s=1}^{k} q_{l,s}(x) = \frac{v}{N}\sum_{l=1}^{n}\sum_{s=1}^{k} p_s(x_{D_l}) = \frac{\theta}{n}\sum_{l=1}^{n}\bar{p}(x_{D_l}), \tag{20}$$

the optimal solution x^* for (1) with $\{f_i\}_{i\in[N]}$ constructed as (19) is $x^* = \mathbf{1}_n \otimes w^* = (w^*, w^*, \ldots, w^*)^T$ where $w^* \in \mathbb{R}^d$ is defined as (12) and repeated for n times in x^*.

Now, we want to verify that the functions $\{f_i\}_{i\in[N]}$ satisfy our assumptions. In fact, we have shown that p_s is L-smooth for each $s \in [k]$. Since f_i is either a zero function or equals $p_s(x_{D_l})$ for some $l \in [n]$ and $s \in [k]$, the function f_i is L-smooth for each $i \in [N]$ as well. Since $\bar{p}$ is μ'-strongly convex (on $\mathbb{R}^b$) and $\mu' = \frac{n\mu}{\theta}$, the function f defined in (1) must be μ-strongly convex (on $\mathbb{R}^d$) according to the relationship (20).

According to its construction, $\{f_i\}_{i\in[N]}$ are simultaneously decomposable with respect to the partition $D_1, \ldots, D_n$ with $D_l = \{b(l-1)+1, b(l-1)+2, \ldots, bl\}$ (see Definition 4). In fact, $f_i(x) = p_s(x_{D_l})$ for some $s \in [k]$ and $l \in [n]$ so that $f_i(x)$ can be represented as

$$f_i(x) = \sum_{l=1}^{n} g_i^l(x_{D_l}), \tag{21}$$

where $g_i^l \in \{p_s\}_{s\in[k]}$ for exactly one $l \in [n]$ and $g_i^l \equiv 0$ for other l's. Moreover, for any $l \in [n]$, the following equality holds between the two multi-sets

$$\{g_i^l\}_{i\in[N]} = \{g_i\}_{i\in[N]}, \tag{22}$$

where $\{g_i\}_{i\in[N]}$ are defined as (14).[4] By Proposition 5, $\mathcal{A}$ can be decomposed with respective to the same partition $D_1, D_2, \ldots, D_n$ into $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathcal{F}_\alpha$ and $\mathcal{A}_l$ is applied to

$$\min_{w\in\mathbb{R}^b} \left\{ \bar{g}^l \equiv \frac{1}{N}\sum_{i=1}^{N} g_i^l = \frac{1}{N}\sum_{i=1}^{N} g_i = \bar{g} \right\}, \tag{23}$$

which is exactly the problem (15) and has the form of (1). Recall Definition 1 for the procedure of generating $S_j' = S_j \cup R_j$, When $\{f_i | i \in S_j'\}$ is allocated to machine j for algorithm $\mathcal{A}$, what happens at the same time is that $\{g_i^l | i \in S_j'\}$ is allocated to machine j for algorithm $\mathcal{A}_l$ for $l = 1, 2, \ldots, n$.

We now focus on the solution generated by $\mathcal{A}_l$ for some l. For machine j in $\mathcal{A}_l$, let $Y_{1,j}$ be the number of functions in $\{p_s\}_{s\in[k]}$ (repetitions counted) that are contained in S_j and $Y_{2,j}$ be the number of functions in $\{p_s\}_{s\in[k]}$ (repetitions counted) that are contained in R_j.

Due to (14) and (22), exactly vk functions in $\{g_i^l\}_{i\in[N]}$ are from $\{p_s\}_{s\in[k]}$ and the other $N - vk$ functions are zero. Hence, $Y_{1,j}$ has a hypergeometric distribution where $\text{Prob}(Y_{1,j} = r)$ equals the probability of r successes in n draws, without replacement, from a population of size N that contains vk successes. Let $r = e \log m$. According to Chvatal (1979) and

4. The equation (22) does not means $g_i^l = g_i$ for any l and i. It only means, for any l and i, there exists an $i' \in [N]$ so that $g_i^l = g_{i'}$.

$\frac{vk}{N} \leq \frac{1}{n}$, we have $\text{Prob}(Y_{1,j} \geq r) \leq \binom{vkn}{Nr} \binom{n-vkn/N}{n-r}^{n-r} \leq \binom{1}{r}^{r} \binom{n-1}{n-r}^{n-r}$, which implies

$$\begin{aligned}
\text{Prob}(Y_{1,j} \geq e\log m) &\leq \left(\frac{1}{e\log m}\right)^{e\log m} \left(\frac{n-1}{n-e\log m}\right)^{n-e\log m} \\
&= \left(\frac{1}{e\log m}\right)^{e\log m} \left(1+\frac{e\log m-1}{n-e\log m}\right)^{n-e\log m} \\
&< \left(\frac{1}{e\log m}\right)^{e\log m} e^{e\log m-1} = \frac{1}{e}\left(\frac{1}{\log m}\right)^{e\log m} \\
&\leq \frac{1}{e}\left(\frac{1}{2\log(e+1)}\right)^{e\log m} \leq \frac{1}{em^2}, \qquad (24)
\end{aligned}$$

where the second inequality is because $(1+\frac{1}{x})^x < e$, the third inequality is because $m \geq (e+4\max\{1,\alpha\})^2 \geq (e+1)^2$, and the last inequality is because $(2\log(e+1))^e > e^2$.

On the other hand, we have $Y_{2,j} = 0$ when $\alpha = 0$ and $Y_{2,j} = \sum_{i\in R_j} \mathbf{1}_{g_i^l \in \{p_s\}_{s\in[k]}}$ when $\alpha > 0$, where $\mathbf{1}_{g_i^l \in \{p_s\}_{s\in[k]}}$ for $i \in R_j$ are αn i.i.d. binary random variables which equal one with a probability of $\frac{vk}{N}$ and zero with a probability of $1-\frac{vk}{N}$. Suppose $\alpha > 0$. The mean and the variance of $\mathbf{1}_{g_i^l \in \{p_s\}_{s\in[k]}}$ are $\frac{\theta}{n}$ and $\frac{\theta}{n}(1-\frac{\theta}{n})$, respectively. Hence, by Bernstein inequality (Uspensky, 1937), we have

$$\begin{aligned}
\text{Prob}(Y_{2,j} \geq 4\max\{1,\alpha\}\log m) &\leq \text{Prob}(Y_{2,j} > \alpha\theta + 3\max\{1,\alpha\}\log m) \\
&\leq \exp\left(\frac{-\frac{1}{2}(3\max\{1,\alpha\}\log m)^2}{\alpha\theta\left(1-\frac{\theta}{n}\right)+\max\{1,\alpha\}\log m}\right) \\
&\leq \exp\left(\frac{-\frac{1}{2}(3\max\{1,\alpha\}\log m)^2}{2\max\{1,\alpha\}\log m}\right) \\
&\leq \exp\left(-2\max\{1,\alpha\}\log m\right) \leq \frac{1}{m^2}, \qquad (25)
\end{aligned}$$

where the first inequality is because $\max\{1,\alpha\}\log m > \alpha\theta$ and the third inequality is because $\max\{1,\alpha\}\log m \geq \alpha\theta\left(1-\frac{\theta}{n}\right)$. Suppose $\alpha = 0$. (25) holds trivially.

Combining (24) and (25) for $j = 1, 2, \ldots, m$ and using the union bound, we have

$$\text{Prob}(Y_{1,j} + Y_{2,j} < (e+4\max\{1,\alpha\})\log m \text{ for } j = 1,2,\ldots,m) \geq 1 - \frac{2}{m}.$$

Therefore, we have shown that, with a probability of at least $1-\frac{2}{m}$, the multi-set $\{g_i^l | i \in S_j'\}$ contains fewer than $(e+4\max\{1,\alpha\})\log m \leq k$ functions from $\{p_s\}_{s\in[k]}$ (repetition counted) for any $j \in [m]$. In other words, with a probability of at least $1-\frac{2}{m}$, no machine in $\mathcal{A}_l$ has all of the functions in $\{p_s\}_{s\in[k]}$. If the event that "*none of the m machines has all functions in* $\{p_s\}_{s\in[k]}$" (same as the event $\mathcal{E}$ in Lemma 8) indeed happens in $\mathcal{A}_l$, we call $\mathcal{A}_l$ *bad*. Then, since $m \geq (e+4\max\{1,\alpha\})^2$, we have actually proved

$$\text{Prob}(\mathcal{A}_l \text{ is bad}) \geq 1 - \frac{2}{m} \geq 1 - \frac{1}{e}. \qquad (26)$$

By Proposition 5, after H rounds, the solutions $\hat{x} = \mathcal{A}(\{f_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H) \in \mathbb{R}^d$ and $\hat{w}^l = \mathcal{A}_l(\{g_i^l\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H) \in \mathbb{R}^{|D_l|}$ for $l = 1, 2, \ldots, n$ satisfy

$$\begin{aligned}
\mathbb{E}[f(\hat{x}) - f(x^*)] &= \sum_{l=1}^{n} \mathbb{E}[\bar{g}^l(\hat{w}^l) - \bar{g}^l(x^*_{D_l})] = \sum_{l=1}^{n} \mathbb{E}[\bar{g}(\hat{w}^l) - \bar{g}(x^*_{D_l})] \\
&\geq \sum_{l=1}^{n} \mathbb{E}[\bar{g}(\hat{w}^l) - \bar{g}(x^*_{D_l})|\mathcal{A}_l \text{ is bad}]\text{Prob}(\mathcal{A}_l \text{ is bad}) \\
&\geq \sum_{l=1}^{n} \mathbb{E}[\bar{g}(\hat{w}^l) - \bar{g}(x^*_{D_l})|\mathcal{A}_l \text{ is bad}] \left(1 - \frac{1}{e}\right).
\end{aligned}$$

Therefore, if $\mathbb{E}[f(\hat{x}) - f(x^*)] \leq \epsilon$, there must exist an $l \in [n]$ such that

$$\mathbb{E}[\bar{g}(\hat{w}^l) - \bar{g}(x^*_{D_l})|\mathcal{A}_l \text{ is bad}] \left(1 - \frac{1}{e}\right) \leq \frac{\epsilon}{n}. \tag{27}$$

When $\mathcal{A}_l$ is bad, none of the m machines in $\mathcal{A}_l$ has all functions in $\{p_s\}_{s\in[k]}$. By Lemma 8, there exist u and $b = uk$ such that (27) happens only if

$$\begin{aligned}
H &\geq \left(\frac{\sqrt{\kappa' + k - 1} - \sqrt{k}}{4k\sqrt{k}}\right) \log\left(\left(1 - \frac{1}{e}\right)\frac{\mu n \|w^*\|^2}{4\epsilon}\right) \\
&= \left(\frac{\sqrt{\kappa' + k - 1} - \sqrt{k}}{4k\sqrt{k}}\right) \log\left(\left(1 - \frac{1}{e}\right)\frac{\mu \|x^*\|^2}{4\epsilon}\right), \\
&\geq \left(\frac{\sqrt{\kappa' - 1}}{4\sqrt{2}k\sqrt{k}}\right) \log\left(\left(1 - \frac{1}{e}\right)\frac{\mu \|x^*\|^2}{4\epsilon}\right),
\end{aligned}$$

which is the desired lower bound after applying $\kappa' \geq \frac{2\kappa}{3n}$.

4.4 Proof for Theorem 3

We provide the proof for Theorem 3 in this subsection by following a similar argument to the proof of Theorem 2. Some notations in the previous sections will be used here but defined differently.

We first use the machinery developed in Section 2.1.1 by Nesterov (2013) to construct k functions on $\mathbb{R}^b$ where $b = uk$ for some integers $k, u \geq 1$. In particular, for $i, j = 1, \ldots, b$, let $\delta_{i,j}$ be an $b \times b$ matrix with its (i, j) entry being one and others being zeros. Let $M_0, M_1, \ldots, M_{b-1}$ be $b \times b$ matrices defined as

$$M_i = \begin{cases} \delta_{1,1} & \text{for } i = 0 \\ \delta_{i,i} - \delta_{i,i+1} - \delta_{i+1,i} + \delta_{i+1,i+1} & \text{for } 1 \leq i \leq b-1. \end{cases} \tag{28}$$

Here, only the M_{b-1} in (28) is different from the M_{b-1} in (8). For $s \in [k]$, let $\Sigma_s = \sum_{i=0}^{u-1} M_{ik+s-1}$ which has the same form as (9) but with a different M_{b-1}. For example,

when $u = 2$ and $k = 3$ (so $b = 6$), the matrices Σ_s's are given as follows.

$$\Sigma_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\Sigma_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

We define k functions $p_1, \ldots, p_k : \mathbb{R}^b \to \mathbb{R}$ as follows

$$p_s(w) = \begin{cases} \frac{L}{4}\left[\frac{w^T \Sigma_1 w}{2} - e_1^\top w\right] & \text{for } s = 1 \\ \frac{L}{4}\left[\frac{w^T \Sigma_s w}{2}\right] & \text{for } s = 2, \ldots, k, \end{cases} \tag{29}$$

where $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^b$, and denote their average by

$$\bar{p} \equiv \frac{1}{k}\sum_{s=1}^{k} p_s = \frac{L}{4k}\left[\frac{1}{2} w^T \left(\sum_{s=1}^{k} \Sigma_s\right) w - e_1^T w\right]. \tag{30}$$

It is also easy to show that $\lambda_{\max}(\Sigma_s) \leq 4$ so that ∇p_s has a Lipschitz continuity constant of L so that p_s is L-smooth.

Observing that

$$\sum_{s=1}^{k} \Sigma_s = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix},$$

we note that the function $\bar{p}(w)$ defined in (30) is the one studied in Section 2.1.2 by Nesterov (2013) for establishing the lower bound for the iteration complexity for single-machine first-order methods. Following Section 2.1.2 by Nesterov (2013), the optimal solution and the optimal value of $\min_{w \in \mathbb{R}^b} \bar{p}(w)$ can be characterized as follows.

Lemma 9 (Nesterov (2013)) *The vector $w^* = (w_1^*, w_2^*, \ldots, w_b^*)^T \in \mathbb{R}^b$ with*

$$w_j^* = 1 - \frac{j}{b+1}, \text{ for } j = 1, 2, \ldots, b \tag{31}$$

is the optimal solution of $\min_{w \in \mathbb{R}^b} \bar{p}(w)$. Moreover, $\|w^*\|^2 \leq \frac{b+1}{3}$ and $\bar{p}(w^*) = \frac{L}{8k}\left(-1 + \frac{1}{b+1}\right)$.

The following lemma can be extracted from Section 2.1.2 in Nesterov (2013) which characterizes the optimal value of $\bar{p}$ when restricted in the subspace of first t coordinates.

Lemma 10 (Nesterov (2013)) *Let $\bar{p}$ defined as (30). We have*

$$\min_{(w_1,\ldots,w_t)\in\mathbb{R}^t} \bar{p}(w_1,\ldots,w_t,0,\ldots,0) = \frac{L}{8k}\left(-1+\frac{1}{t+1}\right).$$

We can show that $\{p_s\}_{s\in[k]}$ defined in (29) has the following properties.

Lemma 11 *Suppose U is a* **strict** *subset of $\{p_s\}_{s\in[k]}$ defined in (10), and q is an arbitrary linear combination of some functions p_s in U. The Hessian of q is a block diagonal matrix where each block is a square matrix of a size at most k.*

Proof Same as the proof of Lemma 7. ■

Lemma 12 *For any $b = uk$ with integers $k, u \geq 1$, let $\{g_i\}_{i\in[N]}$ be the multi-set of functions on $\mathbb{R}^b$ that consists of v copies of $\{p_s\}_{s\in[k]}$ defined as (29) and $N - vk$ zero functions. More specifically, $\{g_i\}_{i\in[N]}$ is defined as (14) but with $\{p_s\}_{s\in[k]}$ defined as (29). Suppose an algorithm $\mathcal{A} \in \mathcal{F}_\alpha$ is applied to (15) and the sets $\{S'_j\}_{j\in[m]}$ are generated as in Definition 1. Let $\mathcal{E}$ be the random event that none of the m machines has all functions in $\{p_s\}_{s\in[k]}$. Let $\hat{w} = \mathcal{A}(\{g_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$. Then, for any $\epsilon > 0$ and $k \geq 1$, there exist u and $b = uk$ such that $\mathbb{E}[g(\hat{w}) - g(w^*)|\mathcal{E}] \leq \epsilon$ only if $H \geq \sqrt{\frac{3L\|w^*\|^2}{128k^3n\epsilon}}$.*

Proof Let $u = 2H + 1$, $b = 2Hk + k$ and $w^* \in \mathbb{R}^b$ defined as (31). Since $\bar{g} = \frac{1}{N}\sum_{i=1}^N g_i = \frac{v}{N}\sum_{s=1}^k p_s = \frac{\theta\bar{p}}{n}$, we have $w^* = \arg\min_{w\in\mathbb{R}^b} \bar{p}(w) = \arg\min_{w\in\mathbb{R}^b} \bar{g}(w)$ by Lemma 9.

Let E_t defined as in the proof of Lemma 8. Following the same argument as in the proof of Lemma 8, we can show that, if $\mathcal{E}$ happens, after H rounds in algorithm $\mathcal{A}$, we must have $\cup_j W_j \subset E_{Hk}$, where W_j is the working set of machine j by Definition 1. Let $t = Hk$. Since $\hat{w} = \mathcal{A}(\{g_i\}_{i\in[N]}, \{S'_j\}_{j\in[m]}, H)$, we must have $\hat{w} \in E_t$ if $\mathcal{E}$ happens. According to Lemma 10, we have

$$\bar{p}(\hat{w}) \geq \min_{(w_1,\ldots,w_t)\in\mathbb{R}^t} \bar{p}(w_1,\ldots,w_t,0,\ldots,0) = \frac{L}{8k}\left(-1+\frac{1}{Hk+1}\right),$$

which, together with Lemma 9, implies

$$\mathbb{E}\left[\frac{\bar{p}(\hat{w}) - \bar{p}(w^*)}{\|w^*\|}\Big|\mathcal{E}\right] \geq \frac{\frac{L}{8k}\left(-1+\frac{1}{Hk+1}+1-\frac{1}{b+1}\right)}{\frac{1}{3}(b+1)} \geq \frac{3L}{32k^3(H+1)^2} \geq \frac{3L}{128k^3H^2}.$$

Based on this inequality, when $\mathbb{E}[\bar{g}(\hat{w}) - \bar{g}(w^*)|\mathcal{E}] \leq \epsilon$, or equivalently, when $\mathbb{E}[\bar{p}(\hat{w}) - \bar{p}(w^*)|\mathcal{E}] \leq \frac{n\epsilon}{\theta} \leq \frac{3n\epsilon}{2}$, we must have $H \geq \sqrt{\frac{L\|w^*\|^2}{64k^3n\epsilon}}$. ■

The rest of the proof of Theorem 3 is almost identical to that of Theorem 2 so we will only provide a brief sketch. In particular, we will construct N special non-strongly convex functions $\{f_i\}_{i\in[N]}$ on $\mathbb{R}^d$ with $d = nb$ for a sufficiently large b based on $\{p_s\}_{s\in[k]}$ defined as (29), and show that any algorithm $\mathcal{A} \in \mathcal{F}_\alpha$ will need at least the targeted amount of rounds of communication when it is applied to $\{f_i\}_{i\in[N]}$.

For a large enough u and $b = uk$, we partition the set of indices $[d] = \{1,\ldots,d\}$ into n disjoint subsets $D_1, D_2, \ldots, D_n$ with $|D_l| = b$ and $D_l = \{b(l-1)+1, b(l-1)+2, \ldots, bl\}$.

For any $l \in [n]$ and $s \in [k]$, let $q_{l,s}(x)$ be a function on $\mathbb{R}^d$ defined as (18) where $\{p_s\}_{s\in[k]}$ is defined as (29). We define $\{f_i\}_{i\in[N]}$ to be a set that consists of v copies of $\{q_{l,s}\}_{l\in[n],s\in[k]}$ and $N - vkn$ zeros functions. In other words, the set of non-zero functions in $\{f_i\}_{i\in[N]}$ is defined as (19) with $\{p_s\}_{s\in[k]}$ given as (29). It is easy to verify that the each function in $\{f_i\}_{i\in[N]}$ is L-smooth.

Because of Lemma 9 and (20), the optimal solution x^* for (1) with $\{f_i\}_{i\in[N]}$ constructed as above is $x^* = \mathbf{1}_n \otimes w^* = (w^*, w^*, \ldots, w^*)^T$ where $w^* \in \mathbb{R}^d$ is defined as (31).

By its construction, $\{f_i\}_{i\in[N]}$ are simultaneously decomposable with respect to the partition $D_1, \ldots, D_n$ with $D_l = \{b(l-1)+1, b(l-1)+2, \ldots, bl\}$. In fact, $f_i(x) = p_s(x_{D_l})$ for some $s \in [k]$ and $l \in [n]$ with $\{p_s\}_{s\in[k]}$ defined as (29) so that $f_i(x)$ can be represented as (21) where $g_i^l \in \{p_s\}_{s\in[k]}$ for exactly one $l \in [n]$ and $g_i^l \equiv 0$ for other l's. Moreover, for any $l \in [n]$, the equality (22) holds with $\{g_i\}_{i\in[N]}$ defined as (14) and $\{p_s\}_{s\in[k]}$ defined as (29).

By Proposition 5, $\mathcal{A}$ can be decomposed with respective to the partition $D_1, D_2, \ldots, D_n$ into $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathcal{F}_\alpha$ and $\mathcal{A}_l$ is applied to (23) (the same as (15)). Let $S_j' = S_j \cup R_j$ generated as in Definition 1. When $\{f_i | i \in S_j'\}$ is allocated to machine j for algorithm $\mathcal{A}$, the set $\{g_i^l | i \in S_j'\}$ is allocated to machine j for algorithm $\mathcal{A}_l$ for $l = 1, 2, \ldots, n$. We still call $\mathcal{A}_l$ *bad* if none of the m machines in algorithm $\mathcal{A}_l$ has all functions in $\{p_s\}_{s\in[k]}$. Then, following exactly the same argument as in Theorem 2, we can obtain (26) which indicates that $\mathcal{A}_l$ can be *bad* with a constant probability for each l. As a result, if $\mathbb{E}[f(\hat{x}) - f(x^*)] \leq \epsilon$, there must exist an $l \in [n]$ such that (27) holds.

When $\mathcal{A}_l$ is bad, after the data distribution stage, none of the m machines in $\mathcal{A}_l$ has all functions in $\{p_s\}_{s\in[k]}$. According to Lemma 12, there exists b such that (27) happens only if $H \geq \sqrt{\left(1 - \frac{1}{e}\right) \frac{L\|x^*\|^2}{64k^3 n\epsilon}}$, which is the desired lower bound.

5. Distributed SVRG

In this section, we consider a *distributed stochastic variance reduced gradient* (DSVRG) method that is based on a parallelization of the stochastic variance reduced gradient (SVRG) method (Johnson and Zhang, 2013; Mahdavi et al., 2013; Xiao and Zhang, 2014; Konečnỳ and Richtárik, 2017). SVRG works in multiple stages and, in each stage, one batch gradient is computed using all N data points and $O(\kappa)$ iterative updates are performed with only one data point processed in each iterative update.

Our DSVRG algorithm randomly partitions the N data points into m subsets $\{S_j\}_{j\in[m]}$ with $|S_j| = n$ and S_j is allocated to machine j in order to parallelize the computation of the batch gradient in SVRG. Then, we let the m machines conduct the iterative update of SVRG in serial in a "round-robin" scheme, namely, let all machine stay idle except one machine that performs a certain steps of iterative updates of SVRG using its local data and pass the solution to the next machine. However, the only caveat in this idea is that the iterative update of SVRG requires an unbiased estimator of $\nabla f(x)$ which can be constructed by sampling over the whole data set. However, the unbiasedness will be lost if each machine can only sample over S_j. To address this issue, we let machine j sample with replacement from the whole data set to construct a second set R_j with $|R_j| = \alpha n$ for $\alpha > 0$ during the data allocation. In each iterative update, if each machine samples over R_j, the unbiased

Algorithm 1 Distributed SVRG (DSVRG)

Input: An initial solution $\tilde{x}^0 \in \mathbb{R}^d$, data $\{f_i\}_{i\in[N]}$, the number of machines m, a step length $\eta < \frac{1}{4L}$, the number of stages K, the number of iterations T in each stage, and a parameter $\alpha \geq \frac{TK}{N}$.

Output: $\tilde{x}^K$

1: Generate (a) a random partition $\{S_j\}_{j\in[m]}$ of $[N]$, (b) m multi-sets $\{R_j\}_{j\in[m]}$ with $|R_j| = \alpha n$ uniformly sampled with replacement from $[N]$, and (c) data $\{f_i \mid i \in S_j \cup R_j\}$ stored on machine j.
2: $k \leftarrow 1$
3: **for** $\ell = 0, 1, 2, \ldots, K-1$ **do**
4: Center sends $\tilde{x}^\ell$ to each machine
5: **for** machine $j = 1, 2, \ldots, m$ **in parallel do**
6: Compute $h_j^\ell = \sum_{i\in S_j} \nabla f_i(\tilde{x}^\ell)$ and send it to center
7: **end for**
8: Center computes $h^\ell = \frac{1}{N}\sum_{j=1}^m h_j^\ell$ and send it to machine k
9: $(\tilde{x}^{\ell+1}, \{R_j\}_{j\in[m]}, k) \leftarrow \text{SS-SVRG}(\tilde{x}^\ell, \{f_i\}_{i\in[N]}, h^\ell, \{R_j\}_{j\in[m]}, k, \eta, T)$
10: **end for**

estimator will be still available so that the convergence property can be inherited from the single-machine SVRG.

5.1 DSVRG Algorithm and Its Round Complexity

Let $\alpha > 0$. We first randomly partition $[N]$ into subsets $\{S_j\}_{j\in[m]}$ with $|S_j| = \frac{N}{m} = n$ and create m multi-sets $\{R_j\}_{j\in[m]}$ with $|R_j| = \alpha n$ by uniformly sampling with replacement from $[N]$. With the data $\{f_i | i \in S_j \cup R_j\}$ distributed on machine j for each $j \in [m]$, we describe formally each stage of this DSVRG in Algorithm 1 and the iterative update in Algorithm 2. From the definitions of S_j and R_j and the communication and computation performed, we can easily see that DSVRG belongs to the family $\mathcal{F}_\alpha$ ($\alpha > 0$) in Definition 1.

We start DSVRG in machine k with $k = 1$ initially at an initial solution $\tilde{x}^0 \in \mathbb{R}^d$. At the beginning of stage ℓ of DSVRG, all m machines participate in computing a batch gradient h^ℓ in parallel using the data indexed by $S_1, \ldots, S_m$. Within stage ℓ, in each iteration, machine k samples one data f_i from its local data indexed by R_k to construct a stochastic gradient $\nabla f_i(x_t) - \nabla f_i(\tilde{x}^\ell) + h^\ell$ and performs the *iterative update* (Line 3 of Algorithm 2). After this iteration, the index i is removed from R_k. Due to the construction of R_k, we can easily prove that this stochastic gradient is an unbiased estimation of the batch gradient $\nabla f(x_t)$.

Lemma 13 *Suppose $\tilde{x}$, x_t, $\{f_i\}_{i\in[N]}$, h, $\{R_j\}_{j\in[m]}$ and k are defined as in Algorithm 2 (SS-SVRG). In addition, suppose each element of R_j was i.i.d. sampled with replacement from $[N]$ and $h = \frac{1}{N}\sum_{i=1}^N \nabla f(\tilde{x})$. In the t-th iteration of Algorithm 2, we have*

$$\mathbb{E}[\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h | x_t, \tilde{x}] = \nabla f(x_t),$$

where the expectation is taken over i and $\{R_j\}_{j\in[m]}$.

Proof Since R_k is a multi-set that consists of indices uniformly sampled with replacement from $[N]$, the index i has a uniform distribution over $[N]$. Moreover, because each index

Algorithm 2 Single-Stage SVRG: SS-SVRG$(\tilde{x}, \{f_i\}_{i\in[N]}, h, \{R_j\}_{j\in[m]}, k, \eta, T)$

Input: A solution $\tilde{x} \in \mathbb{R}^d$, data $\{f_i\}_{i\in[N]}$, a batch gradient h, m multi-sets $\{R_j\}_{j\in[m]}$, the index of the active machine k, a step length $\eta < \frac{1}{4L}$, and the number of iterations T.

Output: The average solution $\bar{x}_T$, the updated multi-sets $\{R_j\}_{j\in[m]}$, and the updated index of active machine k.

1: $x_0 = \tilde{x}$ and $\bar{x}_0 = \mathbf{0}$
2: **for** $t = 0, 1, 2, \ldots, T-1$ **do**
3: Machine k uniformly samples an instance i from R_k and computes

$$x_{t+1} = x_t - \eta\left(\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h\right), \quad \bar{x}_{t+1} = \frac{x_{t+1} + t\bar{x}_t}{t+1}, \quad R_k \leftarrow R_k \backslash \{i\}$$

4: **if** $R_k = \emptyset$ **then**
5: x_{t+1} and $\bar{x}_{t+1}$ are sent to machine $k+1$
6: $k \leftarrow k+1$
7: **end if**
8: **end for**

sampled from R_k in the previous iteration has been removed from R_k, the set R_k at iteration t does not contain the indices in the previous iterations that determine x_t so that R_k is independent of x_t. With these facts, we can show that

$$\begin{aligned}
&\mathbb{E}[\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h | x_t, \tilde{x}] = \mathbb{E}\left[\mathbb{E}[\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h | R_k, x_t, \tilde{x}] \Big| x_t, \tilde{x}\right] \\
= \quad &\mathbb{E}\left[\frac{1}{|R_k|}\sum_{i\in R_k}(\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h) \Big| x_t, \tilde{x}\right] = \mathbb{E}_{i\sim N}[\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h | x_t, \tilde{x}] = \nabla f(x_t),
\end{aligned}$$

where the third equality is because R_k is independent of x_t and the marginal distribution of each i in R_k is uniform distribution on $[N]$. ■

The m machines do the iterative updates in the order from machine 1 to machine m. Once the current active machine, say, machine k, has removed all of its samples in R_k (so that $R_k = \emptyset$), then it must pass the current solution and the running average of all solutions generated in the current stage to machine $k+1$. At any time during the algorithm, there is only one machine updating the solution x_t and the other $m-1$ machines only contribute in computing the batch gradient h^ℓ. We want to emphasize that it is important that machines should never use any samples in R_j's more than once since, otherwise, the stochastic gradient $\nabla f_i(x_t) - \nabla f_i(\tilde{x}^\ell) + h^\ell$ will lose its unbiasedness. Because one element in R_k is removed in each iterative update (Line 3) of Algorithm 2, this update cannot be performed any longer once each R_k becomes empty. Hence, a requirement for using Algorithm 1 is $TK \le \sum_{j\in[m]} |R_j| = \alpha n m = \alpha N$.

The convergence of Algorithm 1 is established by the following theorem.

Theorem 14 *Suppose* $0 < \eta < \frac{1}{4L}$ *and* $\mu > 0$. *Algorithm 1 guarantees*

$$\mathbb{E}\left[f(\tilde{x}^K) - f(x^*)\right] \le \left(\frac{1}{\mu\eta(1-4L\eta)T} + \frac{4L\eta(T+1)}{(1-4L\eta)T}\right)^K \left[f(\tilde{x}^0) - f(x^*)\right]. \tag{32}$$

When $\eta = \frac{1}{16L}$, $T = 96\kappa$ *and* $\alpha \geq \frac{96\kappa}{\log(9/8)N}\log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right)$, *Algorithm 1 finds an* ϵ*-optimal solution with* $K \leq \frac{1}{\log(9/8)}\log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right)$ *stages and at most* $\frac{\frac{96\kappa}{\alpha n}+1}{\log(9/8)}\log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right)$ *rounds of communication.*

In particular, if $\frac{96\kappa}{\log(9/8)}\log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right) \leq O(N)$ *in addition to the previous conditions, Algorithm 1 with* $\alpha = \Theta(1)$ *finds an* ϵ*-optimal solution with at most* $O\left((\frac{\kappa}{n}+1)\log\left(\frac{1}{\epsilon}\right)\right)$ *rounds of communication.*

Proof In the iterative update given in Line 3 of Algorithm 2, a stochastic gradient $\nabla f_i(x_t) - \nabla f_i(\tilde{x}) + h$ is constructed with h being the batch gradient $\nabla f(\tilde{x})$ and i sampled from R_k in the active machine k. By Lemma 13, this stochastic gradient is unbiased estimator of $\nabla f(x_t)$. Therefore, the path of solutions $\tilde{x}^0, \tilde{x}^1, \tilde{x}^2, \ldots$ generated by Algorithm 1 has the same distribution as the ones generated by single-machine SVRG so that the convergence result for the single-machine SVRG can be directly applied to Algorithm 1. The inequality (32) has been shown in Theorem 1 in Xiao and Zhang (2014) for single-machine SVRG, which now also holds for Algorithm 1.

When $\eta = \frac{1}{16L}$ and $T = 96\kappa$, it is easy to show that

$$\frac{1}{\mu\eta(1-4L\eta)T} + \frac{4L\eta(T+1)}{(1-4L\eta)T} \leq \frac{1}{\mu\eta(1-4L\eta)T} + \frac{8L\eta}{(1-4L\eta)} = \frac{2}{9} + \frac{2}{3} = \frac{8}{9}$$

so that Algorithm 1 needs $K \leq \frac{1}{\log(9/8)}\log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right)$ stages to find an ϵ-optimal solution. Since Algorithm 1 performs TK iterative updates with one element of R_k removed after each update, it requires $TK \leq \sum_{j\in[m]}|R_j| = \alpha N$, which is satisfied by the above upper bound of K, the choice of T and the assumption on α.

Since Algorithm 1 needs one round of communication to compute a batch gradient at each stage (one call of SS-SVRG) and one round after every αn iterative updates (when $R_k = \emptyset$), it needs $K + \frac{TK}{\alpha n} \leq \left(\frac{96\kappa}{\alpha n}+1\right)\frac{1}{\log(9/8)}\log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right)$ rounds of communication in total to find an ϵ-optimal solution. The second part of the conclusion is a straightforward outcome from the first part. ■

By Theorem 14, to find an ϵ-optimal solution, DSVRG requires $\alpha \geq \Omega(\frac{\kappa}{N}\log(1/\epsilon))$ so that each machine must have a minimum memory space of $|S_j| + |R_j| = n + \alpha n = \Omega(n + \frac{\kappa}{m}\log(1/\epsilon))$. For ERM problem where $\kappa = \Theta(\sqrt{N})$ and ϵ is reasonably small, for example, $\epsilon = \frac{1}{n^s}$ for a constant s, the requirement becomes $\alpha \geq \Omega(\frac{\log(n)}{\sqrt{nm}})$, which can be easily satisfied with a constant α ($\alpha = \Theta(1)$). In this case, DSVRG only asks for $\Theta(n + \sqrt{\frac{n}{m}}\log(n)) = \Theta(n)$ memory space in each machine to store $S_j \cup R_j$. Note that a memory space of n is necessary in all distributed algorithms to store S_j so the space required by DSVRG is in the same order of magnitude as other distributed algorithms in this scenario. Although we use $\kappa = \Theta(\sqrt{N})$ as an example, DSVRG still applies with $\alpha = \Theta(1)$ and $\Theta(n)$ memory space when $\epsilon = \frac{1}{n^s}$ and $\kappa = \Theta(N^b)$ with $b \in (0,1)$.

Compared to other methods which use only S_j, DSVRG and DASVRG (introduced later) require additional $\Theta(\alpha N)$ amount of communication during the data allocation stage for transmitting the second data set R_j from the center to each machine. Although the

communication during the data allocation stage is not included in the total amount of communication in the performance metrics defined in Section 1.1, it will certainly affect the efficiency of an algorithm in practice. Hence, in Algorithm 4 in Appendix, we design an efficient data allocation scheme which reuses the randomness of first randomly partitioned data set S_j to construct R_j. We show in Lemma 18 that this method helps to increase the overlap between R_j and S_j so that it requires a smaller amount of communication to allocate $\{f_i | i \in S_j \cup R_j\}$ to machine j than the direct implementation. For example, when $\kappa = \Theta(\sqrt{N})$ in a typical ERM problem, with Algorithm 4, DSVRG only needs $O((\log(1/\epsilon))^2)$ more amount of communication during the data allocation stage than other methods.

5.2 Regime Where DSVRG is Optimal

In this subsection, we consider a scenario where $\kappa = O(n^{1-2\delta})$ with a constant $0 < \delta < \frac{1}{2}$ and $\epsilon = \frac{1}{n^s}$ with a constant $s > 0$. We show that with modified values of η, T and K, DSVRG can find an ϵ-optimal solution for (1) with the optimal parallel runtime, the optimal amount of communication, and the optimal number of rounds of communication *simultaneously*.

Proposition 15 *Suppose $0 < \kappa \leq \frac{n^{1-2\delta}}{32}$ with a constant $0 < \delta < \frac{1}{2}$. When $\eta = \frac{1}{16n^\delta L}$, $T = n$ and $\alpha \geq \frac{1}{\log(n^\delta/2)} \log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right)$, Algorithm 1 finds an ϵ-optimal solution for (1) with $K = O(\frac{\log(1/\epsilon)}{\log n})$ stages, $O(\left(\frac{1+\alpha}{\alpha}\right)\frac{\log(1/\epsilon)}{\log n})$ rounds of communications, $O(\frac{n\log(1/\epsilon)}{\log n})$ parallel runtime, and $O((m+\frac{1}{\alpha})\frac{\log(1/\epsilon)}{\log n})$ amount of communication.*

If $\epsilon = \frac{1}{n^s}$ with a positive constant s in addition to the conditions above, Algorithm 1 finds an ϵ-optimal solution for (1) with $\alpha = \Theta(1)$, $O(1)$ rounds of communications, $O(n)$ parallel runtime, and $O(m)$ amount of communication.

Proof Since $\eta = \frac{1}{16n^\delta L} < \frac{1}{4L}$, Algorithm 1 guarantees (32) according to Theorem 14. With $T = n$ and $\eta = \frac{1}{16n^\delta L}$, we have

$$\begin{aligned}\frac{1}{\mu\eta(1-4L\eta)T} + \frac{4L\eta(T+1)}{(1-4L\eta)T} &\leq \frac{1}{\mu\eta(1-4L\eta)T} + \frac{8L\eta}{(1-4L\eta)} = \frac{16n^\delta\kappa}{(1-1/(4n^\delta))n} + \frac{1}{2n^\delta(1-\frac{1}{4n^\delta})} \\ &\leq \frac{1}{2n^\delta(1-1/(4n^\delta))} + \frac{1}{2n^\delta(1-1/(4n^\delta))} \leq \frac{2}{n^\delta}.\end{aligned}$$

Hence, $\mathbb{E}\left[f(\tilde{x}^K) - f(x^*)\right] \leq \epsilon$ is implied by (32) with $K \leq \frac{1}{\log(n^\delta/2)} \log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right) = O(\frac{\log(1/\epsilon)}{\log n})$. The requirement $TK \leq \alpha N$ is satisfied by the above upper bound of K and the assumption on α.

Since Algorithm 1 needs one round of communication to compute a batch gradient at each stage and one round after every αn iterative updates, it needs $K + \frac{TK}{\alpha n} = O(\frac{(1+\alpha)\log(1/\epsilon)}{\alpha \log n})$ rounds of communication in total to find an ϵ-optimal solution. Since one gradient is computed in each iterative update and n gradients are computed in parallel in each stage, the parallel runtime of Algorithm 1 is $(n+T)K = O(\frac{n\log(1/\epsilon)}{\log(n)})$. Finally, each batch gradient computation requires machines sending m vectors of size $O(d)$ to the center and every αn iterative updates require sending two vectors of size $O(d)$ between machines. Hence, Algo-

rithm 1 requires $(mK + \frac{2TK}{\alpha n}) = O((m + \frac{1}{\alpha})\frac{\log(1/\epsilon)}{\log n})$ amount of communication to find an ϵ-optimal solution.[5]

We can easily obtain the second part of the conclusion after substituting ϵ in the three performance metrics above by $\frac{1}{n^s}$ with a positive constant s and using the fact that $\alpha \geq \frac{1}{\log(n^\delta/2)} \log\left(\frac{f(\tilde{x}^0)-f(x^*)}{\epsilon}\right) = \Theta(1)$ ■

We emphasize again that the amount of communication in Proposition 15 is measured by the number of vectors of size $O(d)$ transmitted through the network after the data distribution phase so it does not include the communication for sending S_j and R_j to machines. The justification for the scenario where $\kappa = O(n^{1-2\delta})$ and $\epsilon = \frac{1}{n^s}$ and why these performance guarantees are optimal have been discussed in Section 1.2. To apply DSVRG under this scenario, the required memory space in each machine is $O(n+\alpha n) = O(n)$ (since we are able to choose $\alpha = \Theta(1)$ in this case), which is in the same order of magnitude as the memory requirement of all distributed methods.

6. Accelerated Distributed SVRG

In this section, we use the generic acceleration techniques in Frostig et al. (2015) and Lin et al. (2015) to further improve the theoretical performance of DSVRG and obtain a *distributed accelerated stochastic variance reduced gradient* (DASVRG) method. DASVRG belongs to the family $\mathcal{F}_\alpha$ ($\alpha > 0$) in Definition 1. We will show that the lower bound given in Theorem 2 for the round complexity of $\mathcal{F}_\alpha$ is matched by the rounds needed by DASVRG with $\alpha = \Theta(1)$ up to logarithmic factors for a flexible and realistic regime of parameters.

6.1 DASVRG Algorithm and Its Round Complexity

We define the *proximal function* for $f(x)$ as

$$f_\sigma(x; y) \equiv f(x) + \frac{\sigma}{2}\|x - y\|^2 = \frac{1}{N}\sum_{i=1}^{N} f_{\sigma,i}(x; y), \tag{33}$$

where $f_{\sigma,i}(x; y) = f_i(x) + \frac{\sigma}{2}\|x - y\|^2$, σ is a positive constant to be determined later and $y \in \mathbb{R}^d$ is a *proximal point*. The condition number of this proximal function is $\kappa(f_\sigma) \equiv \frac{L+\sigma}{\mu+\sigma}$ which is always smaller than κ.

Given an algorithm, denoted by $\mathcal{A}$, that can be applied to (1), the acceleration scheme developed in Frostig et al. (2015) and Lin et al. (2015) is an iterative method that involves inner and outer loops and uses $\mathcal{A}$ as a sub-routine called in each outer loop. In particular, in the p-th outer iteration of this acceleration scheme, the algorithm $\mathcal{A}$ is applied to find a solution for the p-th proximal problem defined on a proximal point $y_p \in \mathbb{R}^d$, namely,

$$\min_{x\in\mathbb{R}^d} f_\sigma(x; y_p) \text{ for } p = 0, 1, 2, \ldots, P-1. \tag{34}$$

Since $\kappa(f_\sigma) < \kappa$, solving (34) is generally easier than solving (1). Moreover, the algorithm $\mathcal{A}$ does not need to solve (34) to optimality but only needs to generate an approximate

5. As we defined in Section 1.1, the communication used to distribute S_j and R_j to machine j is not included in the amount of communication given here.

solution $\hat{x}_{p+1}$ with an accuracy ϵ_p in the sense that

$$\mathbb{E}\left[f_\sigma(\hat{x}_{p+1}; y_p) - \min_{x\in\mathbb{R}^d} f_\sigma(x; y_p)\right] \leq \epsilon_p. \tag{35}$$

Then, the acceleration scheme uses the convex combination of $\hat{x}_{p+1}$ and another auxiliary solution z_{p+1} to construct a new proximal point y_{p+1}. After that, the $(p+1)$-th proximal problem is constructed based on y_{p+1} which will be solved in the next outer iteration while the auxiliary solution z_{p+1} will be updated by a momentum step. With an appropriately chosen value for σ, it is shown by Frostig et al. (2015) and Lin et al. (2015) that, for many existing $\mathcal{A}$ including SAG (Schmidt et al., 2017; Roux et al., 2012), SAGA (Defazio et al., 2014a), SDCA (Shalev-Shwartz and Zhang, 2013), SVRG (Johnson and Zhang, 2013) and Finito/MISO (Defazio et al., 2014b; Mairal, 2015), this acceleration scheme needs a smaller runtime for finding an ϵ-optimal solution than applying algorithm $\mathcal{A}$ directly to (1).

Given the success of this acceleration scheme in the single-machine setting, it will be promising to also apply this scheme to the DSVRG to further improve its theoretical performance. Indeed, this can be done by choosing $\mathcal{A}$ in this acceleration scheme to be DSVRG. In particular, we follow the scheme in Algorithm 2 in Frostig et al. (2015) and obtain the DASVRG method as in Algorithm 3. In the p-th outer iteration of DASVRG, DSVRG is used to solve the proximal problem (34) in a distributed way up to an accuracy ϵ_p.

Theorem 16 *When* $\mu > 0$, $\sigma = \frac{L}{n}$, $\eta = \frac{1}{16L}$, $T = 96\kappa(f_\sigma)$, $K = \frac{1}{\log(9/8)} \log\left(4\left(\frac{2\sigma+\mu}{\mu}\right)^{3/2}\right)$

$$\text{and} \quad \alpha \geq \frac{384}{\log(9/8)m}\sqrt{\frac{2\kappa}{n}+1}\log\left(4\left(\frac{2\kappa}{n}+1\right)^{3/2}\right)\log\left(\left(\frac{2\kappa}{n}+1\right)\frac{f(\hat{x}_0)-f(x^*)}{\epsilon}\right), \tag{36}$$

Algorithm 3 find an ϵ-optimal solution with $P \leq \sqrt{\frac{8\sigma+4\mu}{\mu}}\log\left(\frac{(2\sigma+2\mu)(f(\hat{x}_0)-f(x^*)}{\mu\epsilon}\right)$ outer loops and at most $\frac{\frac{384}{\alpha}+2}{\log(9/8)}\sqrt{\frac{2\kappa}{n}+1}\log\left(4\left(\frac{2\kappa}{n}+1\right)^{3/2}\right)\log\left(\left(\frac{2\kappa}{n}+1\right)\frac{f(\hat{x}_0)-f(x^*)}{\epsilon}\right)$ rounds of communication.

In particular, if the right hand side of (36) is at most $O(1)$ in addition to the previous conditions, Algorithm 1 with $\alpha = \Theta(1)$ finds an ϵ-optimal solution with at most $O\left((1+\sqrt{\frac{\kappa}{n}})\log(1+\frac{\kappa}{n})\log\frac{1}{\epsilon}\right)$ rounds of communication.

Proof Due to the way $\{R_j\}_{j\in[m]}$ is generated and Lemma 13, the solution path $\tilde{x}^0, \tilde{x}^1, \tilde{x}^2, \ldots$ generated within the p-th outer loop of Algorithm 3 has same distribution as the solution path generated by applying the single-machine SVRG to (34) with an initial solution of $\tilde{x}^0 = \hat{x}_p$. Hence, according to Theorem 1 in Xiao and Zhang (2014), the following inequality holds for the p-th outer loop of Algorithm 3

$$\mathbb{E}\left[f_\sigma(\hat{x}_{p+1}; y_p) - \min_{x\in\mathbb{R}^d} f_\sigma(x; y_p)\right] \leq \left(\frac{1}{\mu\eta(1-4L\eta)T} + \frac{4L\eta(T+1)}{(1-4L\eta)T}\right)^K \left[f_\sigma(\hat{x}_p; y_p) - \min_{x\in\mathbb{R}^d} f_\sigma(x; y_p)\right].$$

Applying the values of η and T to this inequality, the p-th outer loop of Algorithm 3 ensures

$$\mathbb{E}\left[f_\sigma(\hat{x}_{p+1}; y_p) - \min_{x\in\mathbb{R}^d} f_\sigma(x; y_p)\right] \leq \left(\frac{8}{9}\right)^K \left[f_\sigma(\hat{x}_p; y_p) - \min_{x\in\mathbb{R}^d} f_\sigma(x; y_p)\right].$$

Algorithm 3 Distributed Accelerated SVRG (DASVRG)

Input: An initial solution $\hat{x}_0 \in \mathbb{R}^d$, data $\{f_i\}_{i\in[N]}$, the number of machines m, a step length $\eta < \frac{1}{4L}$, the number of stages K in DSVRG, the number of iterations T in each stage of DSVRG, the number of outer iterations P in DASVRG, a parameter $\sigma \geq 0$, and a parameter $\alpha \geq \frac{TKP}{N}$.

Output: $\hat{x}_P$

Generate (a) a random partition $\{S_j\}_{j\in[m]}$ of $[N]$, (b) m multi-sets $\{R_j\}_{j\in[m]}$ with $|R_j| = \alpha n$ uniformly sampled with replacement from $[N]$, and (c) data $\{f_i \mid i \in S_j \cup R_j\}$ stored on machine j.
$k \leftarrow 1$
Initialize $\rho = \frac{\mu+2\sigma}{\mu}$ and $z_0 = \hat{x}_0$
for $p = 0, 1, 2, \ldots, P-1$ **do**
  Center computes $y_p = \frac{1}{1+\rho^{-1/2}}\hat{x}_p + \frac{\rho^{-1/2}}{1+\rho^{-1/2}} z_p$ and $\tilde{x}^0 = \hat{x}_p$ and sends y_p to each machine
  for $\ell = 0, 1, 2, \ldots, K-1$ **do**
    Center sends $\tilde{x}^\ell$ to each machine
    for machine $j = 1, 2, \ldots, m$ **in parallel do**
      Compute $h_j^\ell = \sum_{i\in S_j} \nabla \tilde{f}_i(\tilde{x}^\ell; y_p)$ and send it to center
    end for
    Center computes $h^\ell = \frac{1}{N}\sum_{j=1}^m h_j^\ell$ and sends it to machine k
    $(\tilde{x}^{\ell+1}, \{R_j\}_{j\in[m]}, k) \leftarrow \text{SS-SVRG}(\tilde{x}^\ell, \{f_{\sigma,i}(x; y_p)\}_{i\in[N]}, h^\ell, \{R_j\}_{j\in[m]}, k, \eta, T)$
  end for
  Machine k computes $\hat{x}_{p+1} = \tilde{x}^K$ and sends $\hat{x}_{p+1}$ to center
  Center computes $z_{p+1} = (1-\rho^{-1/2})z_p + \rho^{-1/2}[y_p - \rho(y_p - \hat{x}_{p+1})]$
end for

By this inequality and the choice of K, we can show that the solution $\hat{x}_{p+1}$ generated from the p-th outer loop of Algorithm 3 satisfies (35) with

$$\epsilon_p = \frac{1}{4}\left(\frac{\mu}{2\sigma+\mu}\right)^{3/2}\left[f_\sigma(\hat{x}_p; y_p) - \min_{x\in\mathbb{R}^d} f_\sigma(x; y_p)\right].$$

Using this result and Lemma 3.3, 3.4, 3.5 and 3.6 in Frostig et al. (2015), we can show that

$$\mathbb{E}\left[f(\hat{x}_P) - f(x^*)\right] \leq \frac{2\sigma+2\mu}{\mu}\left(1 - \frac{1}{2}\sqrt{\frac{\mu}{2\sigma+\mu}}\right)^P \left[f(\hat{x}_0) - f(x^*)\right].$$

This means Algorithm 3 finds an ϵ-optimal solution after $P \leq 2\sqrt{\frac{2\sigma+\mu}{\mu}}\log\left(\frac{(2\sigma+2\mu)(f(\hat{x}_0)-f(x^*)}{\mu\epsilon}\right)$ outer loops.

Since Algorithm 3 performs TKP iterative updates with one element of R_k removed after each update, it requires $TKP \leq \sum_{j\in[m]}|R_j| = \alpha N$, which is satisfied by the choices of T and K, the above upper bound of P and the assumption on α.

Since Algorithm 3 needs one round of communication to compute a batch gradient at each stage (one call of SS-SVRG) and one round after every αn iterative updates (when

$R_k = \emptyset$), it needs

$$\begin{aligned} & KP + \frac{TKP}{\alpha n} = \left(\frac{T}{\alpha n} + 1\right) KP \\ \leq & \left(\frac{96\kappa(f_\sigma)}{\alpha n} + 1\right) \frac{2}{\log(9/8)} \sqrt{\frac{2\sigma+\mu}{\mu}} \log\left(4\left(\frac{2\sigma+\mu}{\mu}\right)^{3/2}\right) \log\left(\frac{(2\sigma+2\mu)(f(\hat{x}_0)-f(x^*)}{\mu\epsilon}\right) \\ \leq & \left(\frac{192}{\alpha} + 1\right) \frac{2}{\log(9/8)} \sqrt{\frac{2\kappa}{n}+1} \log\left(4\left(\frac{2\kappa}{n}+1\right)^{3/2}\right) \log\left(\left(\frac{2\kappa}{n}+1\right)\frac{f(\hat{x}_0)-f(x^*)}{\epsilon}\right) \end{aligned}$$

rounds of communication in total to find an ϵ-optimal solution. In the second inequality above, we use the fact that $\sigma = \frac{L}{n}$ and $\frac{\kappa(f_\sigma)}{n} = \frac{L+\sigma}{n\mu+n\sigma} \leq \frac{n+1}{n} \leq 2$. The second conclusion is a straightforward outcome of the first conclusion. ■

According to Theorem 16, DASVRG requires $\alpha \geq \Omega(\frac{1}{m}\sqrt{\frac{\kappa}{n}+1}\log(\frac{\kappa}{n}+1)\log(1/\epsilon))$ so that each machine must have a memory space of $|S_j| + |R_j| = n + \alpha n = \Omega(n + \frac{n}{m}\sqrt{\frac{\kappa}{n}+1}\log(\frac{\kappa}{n}+1)\log(1/\epsilon))$. For ERM problem where $\kappa = \Theta(\sqrt{N})$ and $\epsilon = \frac{1}{n^s}$ for a constant s, the requirement becomes $\alpha \geq \Omega(\frac{1}{m}\sqrt{\sqrt{\frac{m}{n}}+1}\log(\sqrt{\frac{m}{n}}+1)\log(n))$, which can be easily satisfied with a constant α ($\alpha = \Theta(1)$). Moreover, in this case, DASVRG only asks for $\Theta(n + \frac{n}{m}\sqrt{\sqrt{\frac{m}{n}}+1}\log(\sqrt{\frac{m}{n}}+1)\log(n)) = \Theta(n)$ memory space in each machine to store $S_j \cup R_j$ which is in the same order of magnitude as the space required by other distributed algorithms. Although we use $\kappa = \Theta(\sqrt{N})$ as an example, DASVRG still applies with $\alpha = \Theta(1)$ and $\Theta(n)$ memory space when $\epsilon = \frac{1}{n^s}$ and $\kappa = \Theta(N^b)$ with $b \in (0,1)$.

Furthermore, another crucial insight we obtain here is that, although in the single-machine setting, the acceleration scheme of Frostig et al. (2015) and (Lin et al., 2015) only helps to reduce the time complexity when $\kappa \geq N$ by choosing $\sigma = \Theta(N)$, in the distributed setting, it helps to reduce the rounds as long as $\kappa \geq n$ by choosing $\sigma = \Theta(n)$. This shows an interesting and new application of this generic acceleration scheme.

In DSVRG and DASVRG, the subsets $\{S_j\}_{j\in[m]}$ are only used to compute the batch gradient $\nabla f(\tilde{x}^\ell)$. Therefore, both DSVRG and DASVRG can be applied with the same theoretical guarantees under the general setting where $\{S_j\}_{j\in[m]}$ is an arbitrary (not necessarily random) partition of $\{f_i\}_{i\in[N]}$ into equal-sized subsets. Because our lower bound for the round complexity is provided for the algorithms in $\mathcal{F}_\alpha$ where $\{S_j\}_{j\in[m]}$ is a random partition, we present both DSVRG and DASVRG in the same setting to show that the theoretical lower bound is almost reachable and DASVRG is nearly optimal within $\mathcal{F}_\alpha$.

7. DASVRG for Non-Strongly Convex Problem

In this section, we consider solving (1) when f is not strongly convex ($\mu = 0$) by the standard technique of applying DASVRG to the following regularized version of (1)

$$\min_{x\in\mathbb{R}^d} \left\{ f^\lambda(x) \equiv f(x) + \frac{\lambda}{2}\|x\|^2 = \frac{1}{N}\sum_{i=1}^N f_i^\lambda(x) \right\}, \tag{37}$$

where $f_i^\lambda(x) \equiv f_i(x) + \frac{\lambda}{2}\|x\|^2$ and λ is a positive constant. Note that problem (37) satisfies Assumption 1 with ∇f_i^λ being $(L+\lambda)$-Lipschitz continuous and f^λ being λ-convex. The

condition number of f^λ is $\kappa(f^\lambda) \equiv \frac{L+\lambda}{\lambda}$. With λ small enough, the ϵ-solution of (37) found by DASVRG can be an ϵ-solution for (1). Although this method is well-known, we present it here just to show that the lower bound of round complexity for non-strongly convex problems in Theorem 3 can be achieved up to logarithmic terms under some scenario.

Theorem 17 *Suppose Algorithm 3 is applied to (37) where* $\lambda = \frac{\epsilon}{D^*}$ *with* $D^* \geq \|x^*\|^2$. *When* $\sigma = \frac{L+\lambda}{n}$, $\eta = \frac{1}{16(L+\lambda)}$, $T = 96\kappa(f_\sigma^\lambda) = 96\frac{L+\lambda+\sigma}{\lambda+\sigma}$, $K = \frac{1}{\log(9/8)} \log\left(4\left(\frac{2\sigma+\lambda}{\lambda}\right)^{3/2}\right)$, *and*

$$\alpha \geq \frac{384}{\log(9/8)m}\sqrt{\frac{2D^*L}{n\epsilon}+3}\log\left(4\left(\frac{2D^*L}{n\epsilon}+3\right)^{3/2}\right)\log\left(\left(\frac{2D^*L}{n\epsilon}+3\right)\frac{f(\hat{x}_0)-f(x^*)}{\epsilon/2}\right). \quad (38)$$

Algorithm 3 find an ϵ-optimal solution of (1) with $P = \sqrt{\frac{8\sigma+4\lambda}{\lambda}}\log\left(\frac{(2\sigma+2\lambda)(f(\hat{x}_0)-f(x^*)}{\lambda\epsilon/2}\right)$ *and* $\left(\frac{192}{\alpha}+1\right)\frac{2}{\log(9/8)}\sqrt{\frac{2D^*L}{n\epsilon}+3}\log\left(4\left(\frac{2D^*L}{n\epsilon}+3\right)^{3/2}\right)\log\left(\left(\frac{2D^*L}{n\epsilon}+3\right)\frac{f(\hat{x}_0)-f(x^*)}{\epsilon/2}\right)$ *rounds of communication.*

In particular, if the right hand side of (38) is at most $O(1)$ in addition to the previous conditions, Algorithm 1 with $\alpha = \Theta(1)$ finds an ϵ-optimal solution of (1) with at most $\tilde{O}\left((\sqrt{\frac{L}{n\epsilon}})\log^2(\frac{1}{\epsilon})\right)$ *rounds of communication.*

Proof Let $\hat{x}_P$ be the output of Algorithm 3 when it is applied to (37) with the specific values of η, σ, T and K given in the statement of the theorem. Because ∇f_i^λ is $(L+\lambda)$-Lipschitz continuous and f^λ is λ-convex, according to Theorem 16, $\hat{x}_P$ is an $\frac{\epsilon}{2}$-optimal solution of (37) and it is found with the number of rounds of communication stated in the conclusion. Moreover, we can easily show that

$$\begin{aligned}\mathbb{E}[f(\hat{x}_P)-f(x^*)] &\leq f(\hat{x}_P)+\frac{\lambda}{2}\|\hat{x}_P\|^2-f(x^*)-\frac{\lambda}{2}\|x^*\|^2+\frac{\lambda}{2}\|x^*\|^2 \\ &\leq f^\lambda(\hat{x}_P)-\min_{x\in\mathbb{R}^d} f^\lambda(x)+\frac{\lambda}{2}\|x^*\|^2 \leq \frac{\epsilon}{2}+\frac{\epsilon}{2}=\epsilon,\end{aligned}$$

where the last inequality is because $\lambda = \frac{\epsilon}{D^*}$ with $D^* \geq \|x^*\|^2$. Hence, $\hat{x}_P$ is an ϵ-optimal solution of (1). Then, the rest of the conclusion will be proved by applying Theorem 20. ■

According to (38) in Theorem 17, we acknowledge here that DASVRG can be applied to non-strongly convex problem with $\alpha = \Theta(1)$ only under a restricted condition, for example, when $\epsilon = O(\frac{1}{N^b})$ with $b \in (0,1)$. The development in this section is mostly for the theoretical interest on the tightness of the lower bound by Theorem 3.

8. Numerical Experiments

This section presents the numerical performances of our methods in comparisons with some existing distributed optimization techniques on simulated and real data.

8.1 Experiments with Simulated Data

In this section, we conduct numerical experiments with simulated data to compare our DSVRG algorithm with DISCO (Zhang and Lin, 2015) and a distributed implementation of gradient descent (GD) method. We use a single machine to simulate the distributed environment and all methods are implemented in Matlab running on a 64-bit Microsoft Windows 10 machine with a 2.70Ghz Intel(R) i7-6820HQ CPU and 8GB of memory. We consider ridge regression, $\min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^N (a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2$, where $a_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ for $i \in [N]$ are the data points. We generate (a_i, b_i) following the experimental setup in Agarwal et al. (2012) by choosing a correlation parameter $\omega \in [0, 1]$. Let A be the $N \times d$ matrix whose rows are a_i's. According to Agarwal et al. (2012), the eigenvalues of $\frac{1}{N}\mathbb{E}[A^T A]$ lie within the interval $\left[\frac{1}{(1+\omega)^2}, \frac{2}{(1-\omega)^2(1+\omega)}\right]$ so that the condition number κ increases when we choose ω close to 1. To make comparisons under different settings, we choose $\lambda = 10^{-4}$, $d = 50$, $N = 10^4$, $\omega \in \{0, 0.3, 0.5\}$ and $m \in \{10, 20\}$. All generated instances satisfy $\kappa = \frac{n^{1-2\delta}}{32}$ with $0 < \delta < \frac{1}{2}$ which is in the optimal regime of DSVRG (see Section 5.2).

For each instance, we compute the largest and the smallest eigenvalues of $\frac{1}{N}A^T A$ so as to compute the condition number κ and the parameters δ, η and T as in Proposition 15 for implementing DSVRG. Note that the lower bound for α given in Proposition 15 can be conservative and depends on the unknown value $f(x^*)$. In this experiment, we choose $\alpha = 1$ so that DSVRG can run for at most N iterations, which corresponds to $\frac{N}{|R_j|} = m$ rounds because $|R_j| = T = n$. In each main iteration of DISCO, an inexact Newton direction v is computed by solving a preconditioned linear system $P^{-1}\nabla^2 f(x_k)v = P^{-1}\nabla f(x_k)$ using a distributed preconditioned conjugate gradient method, where x_k in the solution in the kth main iteration. We choose $P = \nabla^2 \bar{f}_1(x_k) + (\lambda + \sqrt{m} \times 0.001)I$ with $\bar{f}_1$ defined in (4) and compute P^{-1} directly. In the GD method, we use a step length of $\eta = \frac{1}{L}$.

Since DSVRG stores $|R_j| + |S_j| = (1+\alpha)n$ data points in machine j, it is meaningful to allow other methods to also store $(1+\alpha)n$ data points in each machine in the comparisons. With more local data, the round complexity of the distributed GD method is unchanged while that of DISCO will be reduced according to its round complexity $O((1 + \frac{\sqrt{\kappa}}{n^{.25}})\log\frac{1}{\epsilon})$. Therefore, when implementing DISCO, we use different numbers of machines which lead to different amounts of local data. In particular, we choose $m = 2, 5$ and 10 in DISCO when $m = 10$ in DSVRG and GD and choose $m = 4, 10$ and 20 in DISCO when $m = 20$ in DSVRG and GD. Since $\alpha = 1$, DSVRG with 10 (20) machines and DISCO with 5 (10) machines store the same amount of data points in each machine.

The performance of the three methods are shown in Figure 2, where the vertical axis represents the logarithm of optimality gap (i.e., $\log(f(x_k) - f(x^*))$) and the horizon axis represents the number of rounds of communication. The values of κ and δ for each instance are shown in the figures. According to Figure 2, when δ is relatively large, DSVRG is less efficient than GD because the step length $\eta = \frac{1}{16n^\delta L}$ in DSVRG is too small and the small κ leads to a low round complexity of GD. However, when δ is relatively small, DSVRG significantly outperforms GD, which is consistent with Proposition 15. The performance of DISCO is improved when it uses more machines (and a smaller n), which is also consistent with its round complexity $O((1 + \frac{\sqrt{\kappa}}{n^{.25}})\log\frac{1}{\epsilon})$. DSVRG with 10 machines is more efficient than DISCO with 10 and 5 machines for all instances here and is slightly worse than DISCO

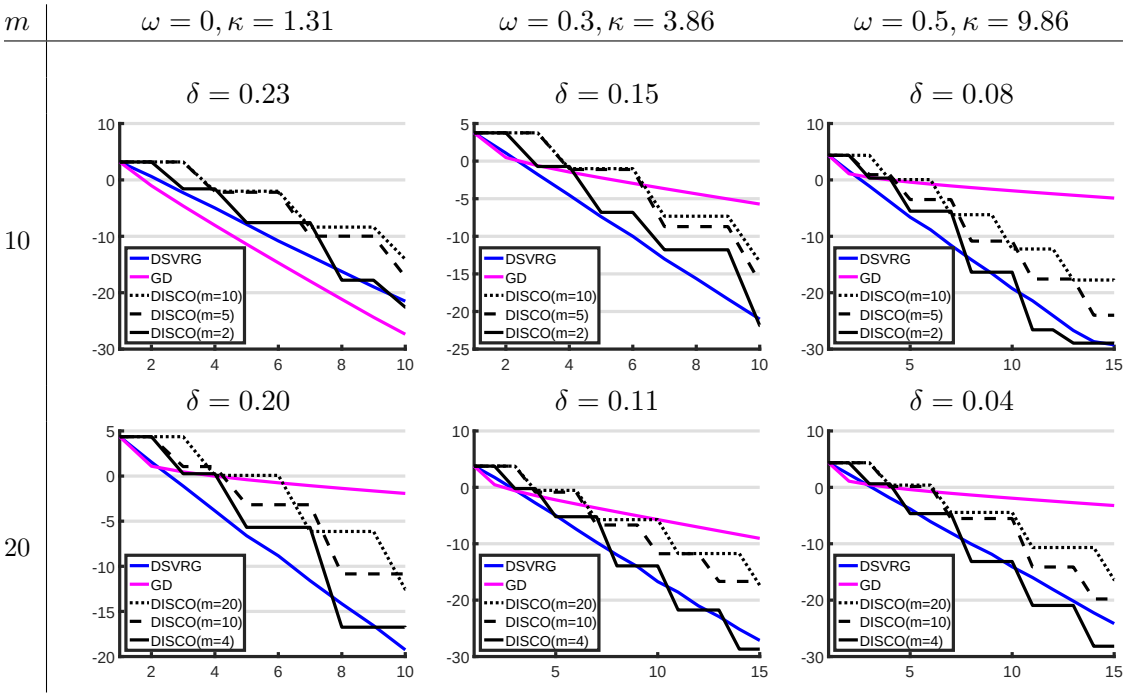

Figure 2: Comparing DSVRG implemented as in Proposition 15 with DISCO and the distributed gradient descent method (GD) in rounds. The condition number κ of each instance is in the regime where DSVRG is optimal (see Section 5.2).

with 2 machines. Recall that DSVRG with 10 machines stores the same amount of local data as DISCO with 5 machines. The results suggest that DSVRG with a small α can potentially outperform DISCO on the problems with $\kappa = O(n^{1-2\delta})$ even if the machines have access to the same amount of data in both algorithms. However, when α is large, DSVRG can be less efficient than DISCO because DISCO needs much fewer machines. The same conclusion can be made from the results where DSVRG uses 20 machines and DISCO uses 4, 10 and 20 machines.

8.2 Experiments with Real Data

In this section, we conduct numerical experiments with real data to compare our DSVRG and DASVRG algorithms with DisDSCA by Yang (2013) and a distributed implementation of the accelerated gradient method (Accel Grad) by Nesterov (2013). We apply these four algorithms to the ERM problem (2) with three data sets:[6] Covtype, Million Song and Epsilon. In particular, we solve ridge regression $\min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^N (a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|^2$ with Million Song data and regularized logistic regression $\min_{x\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^N \log(1+\exp(-b_i a_i^T x)) + \frac{\lambda}{2}\|x\|^2$ with Covtype and Epsilon data. Following the previous work, we map the target variable of year from $1922 \sim 2011$ into $[0,1]$ for the Million Song data. To compare these methods in a challenging setting, we conduct experiments using random Fourier features

6. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

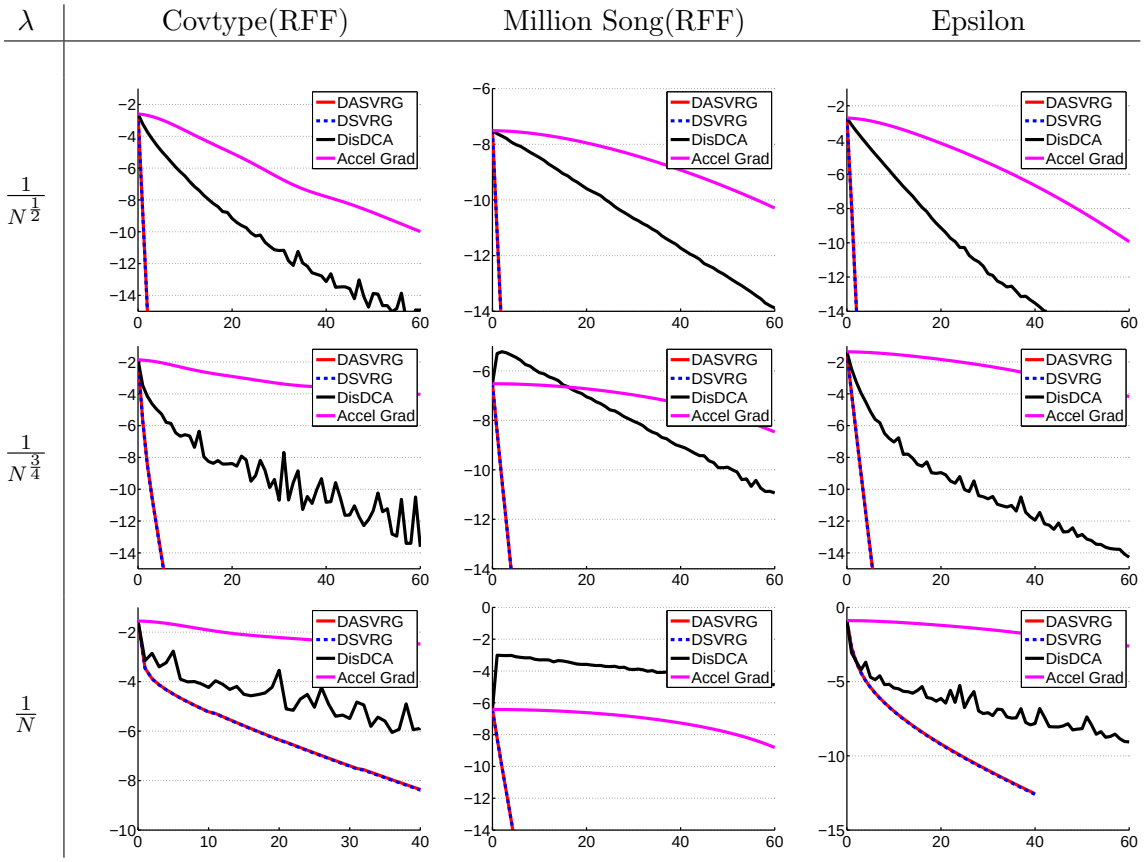

Figure 3: Comparing the DSVRG and DASVRG methods with DisDCA and the accelerated gradient method (Accel Grad) in rounds.

(RFF) (Rahimi and Recht, 2008) on Covtype and Million Song data sets. The RFF is a method for solving large-scale kernel methods by generating finite dimensional features, of which the inner product approximate the kernel similarity. We generate RFF corresponding to RBF kernel. After generating RFF, Covtype data has $N = 522,911$ examples $d = 1,000$ features and Million Song data has $N = 463,715$ examples and $d = 2,000$ features. Since the original Epsilon data is large enough (12G), we use its original features.

The experiments are conducted on a server (Intel(R) Xeon(R) CPU E5-2667 v2 3.30GHz) with multiple processes with each process simulating one machine. We choose the number of processes (machines) to be $m = 5$. To test the performances of algorithms for different condition numbers, we choose the regularization parameter λ to be $1/N^{0.5}$, $1/N^{0.75}$ and $1/N$. For each setting, L is computed as $\frac{\max_{i=1,\ldots,N}\|a_i\|^2}{\gamma} + \lambda$ where $\frac{1}{\gamma}$ is the Lipschiz continuous constant of $\nabla_x \phi(x, \xi)$ in (2), and μ is set to be λ. We implement DSVRG by choosing $\eta = \frac{1}{L}$, $T = 10,000$ and $K = \frac{N}{T}$. For DASVRG, we choose $\eta = \frac{1}{L}$, $T = 10,000$, $K = 1$ and $P = \frac{N}{T}$. In both DSVRG and DASVRG, we directly choose $R_j = S_j$ since, in practice, this implementation gives performances very similar to the performances when R_j is sampled separately. For DisDCA, we use SDCA (Shalev-Shwartz and Zhang, 2013) as

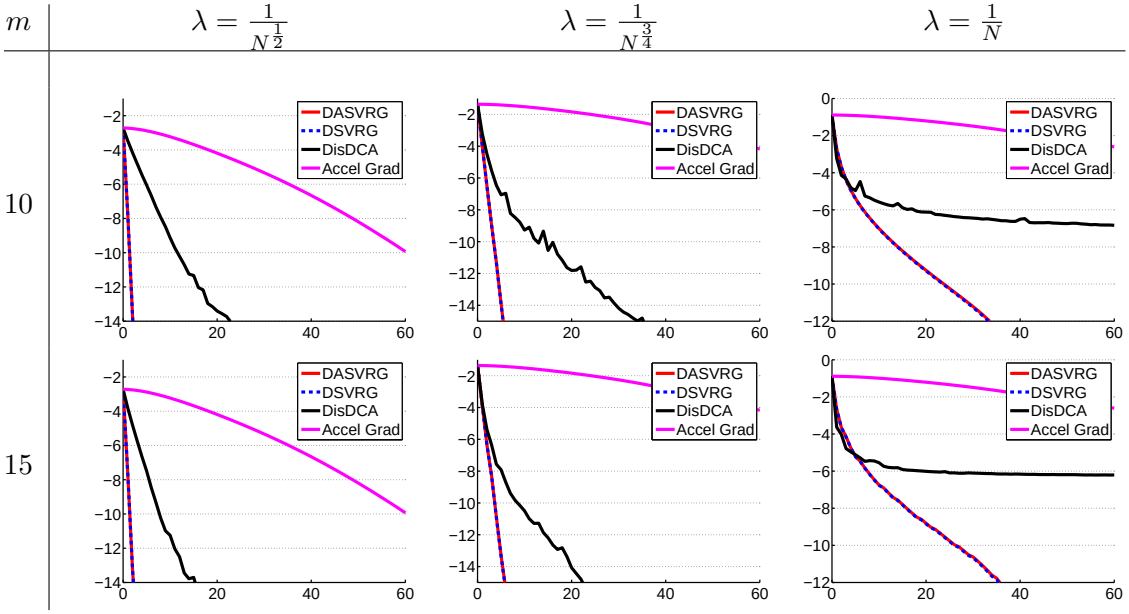

Figure 4: Comparing the DSVRG and DASVRG methods with DisDCA and the accelerated gradient method (Accel Grad) in rounds using Epsilon data.

the local solver so that it is equivalent to the implementation of CoCoA+ with $\sigma' = m$ and $\gamma = 1$ as in the experiments in Ma et al. (2015). We run SDCA for $T = 10,000$ iterations in each round of DisDCA with $\frac{N}{T}$ rounds in total.

The numerical results are presented in Figure 3, where the horizontal axis presents the number of rounds of communication conducted by algorithms and the vertical axis represents the logarithm of optimality gap. According to Figure 3, the performances of all algorithms get worse when λ decreases (so the condition number increases). We find that DSVRG and DASVRG have almost identical performances in rounds of communication and they both outperform the other two methods significantly. To compare the performances of algorithms under different values of m. We choose the $m = 10$ and 15 and repeat the same experiments on Epsilon data. The numerical results are shown in Figure 4. Similar to the case of $m = 5$, our DSVRG and DASVRG require fewer rounds to reach the same ϵ-optimal solution than the other two methods.

9. Conclusion

We study the round complexity for minimizing the average of N convex functions by distributed optimization with m machines under a new setting where each machine receives a subset of the N functions through both random partition and random sampling with replacement. Then, we propose a DSVRG algorithm under this setting which is a distributed extension of the existing SVRG algorithm. In DSVRG, the batch gradients are computed in parallel while the machines perform iterative updates in serial. DSVRG utilizes the local functions sampled with replacement to construct the unbiased stochastic gradient in each iterative update. We provide the theoretical analysis on the rounds of communica-

tion needed by DSVRG to find an ϵ-optimal solution, showing that DSVRG is optimal in terms of runtime, the amount of communication and the rounds of communication when the condition number is small. When the condition number is large, using an acceleration strategy by Frostig et al. (2015) and Lin et al. (2015), we proposed a DASVRG algorithm that requires even fewer rounds of communication than DSVRG and many existing methods that only store random partitioned data in machines, showing the advantage of the new distributed setting. Then, we define a broad family of algorithms which cover DSVRG, DASVRG and many existing methods in the literature. We provide the minimum number of rounds of communication needed by this family of algorithms for finding an ϵ-solution. The rounds of communication needed by DASVRG matches this lower bound up to logarithmic terms, and thus, is nearly optimal within this family.

Acknowledgments

We would like to thank Roy Frostig, Hongzhou Lin, Lin Xiao, and Yuchen Zhang for numerous helpful discussions throughout various stages of this work. Tianbao Yang is partially supported by National Science Foundation (IIS-1463988, IIS-1545995).

Appendix A. An Efficient Data Allocation Procedure

In this section, we design an efficient data allocation scheme which requires a smaller amount of communication to allocate $\{f_i|i \in S_j \cup R_j\}$ to machine j than the direct implementation.

We assume that a random partition $S_1, \ldots, S_m$ of $[N]$ can be constructed efficiently. Let Q be the number of samples in $\{R_j\}_{j\in[m]}$ needed for the algorithm to run the desired number of iterations. In other words, $Q = TK$ in DSVRG and $Q = TKP$ in DASVRG, both of which can be much smaller than αN, the total size $\{R_j\}_{j\in[m]}$. Therefore, in practical implementation, we only need to ensure the total size of $\{R_j\}_{j\in[m]}$ equal Q by allowing $|R_j| \leq \alpha n$ or even $R_j = \emptyset$ in order to save the communication amount.

A straightforward data allocation procedure is to prepare a partition $S_1, \ldots, S_m$ of $[N]$ and then sample a sequence of Q i.i.d. indices $r_1, \ldots, r_Q$ uniformly with replacement from $[N]$. After partitioning $r_1, \ldots, r_Q$ into m multi-sets $R_1, \ldots, R_m \subset [N]$, we allocate data $\{f_i|i \in S_j \cup R_j\}$ to machine j. Note that the amount of communication in distributing $S_1, \ldots, S_m$ is exactly N which is necessary for almost all distributed algorithms. However, in the worst case, this straightforward procedure requires an extra Q amount of communication for distributing $R_j \backslash S_j$.

To improve the efficiency of data allocation, we propose a procedure which reuses the randomness of $S_1, \ldots, S_m$ to generate the indices $r_1, \ldots, r_Q$ so that the overlap between S_j and R_j can be increased which helps reduce the additional amount of communication for distributing $R_1, \ldots, R_m$. The key observation is that the concatenation of $S_1, \ldots, S_m$ is a random permutation of $[N]$ which has already provided enough randomness needed by $R_1, \ldots, R_m$. Hence, it will be easy to build the i.i.d. samples $r_1, \ldots, r_Q$ by adding a little additional randomness on top of $S_1, \ldots, S_m$. With this observation in mind, we propose our data allocation procedure in Algorithm 4.

Algorithm 4 Data Allocation : DA(N, m, Q, α)

Input: Index set $[N]$, the number of machines m, and the constant α, and the length of target sequence Q.

Output: A random partition $S_1, \ldots, S_m$ of $[N]$, indices $r_1, \ldots, r_Q \in [N]$, multi-sets $R_1, \ldots, R_m \subset [N]$, and data $\{f_i \mid i \in S_j \cup R_j\}$ stored on machine j for each $j \in [m]$.

Center samples $r_1, \ldots, r_Q$ and $R_1, \ldots, R_m$ as follows:

1: Randomly partition $[N]$ into m disjoint sets $S_1, \ldots, S_m$ of the same size $n = \frac{N}{m}$.
2: Concatenate the subsets $S_1, \ldots, S_m$ into a random permutation $i_1, \ldots, i_N$ of $[N]$ so that $S_j = \{i_{(j-1)n+1}, \ldots, i_{jn}\}$.
3: **for** $\ell = 1$ to Q **do**
4: Let

$$r_\ell = \begin{cases} i_\ell & \text{with probability } 1 - \frac{\ell-1}{N} \\ i_{\ell'} & \text{with probability } \frac{1}{N} \text{ for } \ell' = 1, 2, \ldots, \ell - 1. \end{cases}$$

5: **end for**
6: Let

$$R_j = \begin{cases} \{r_{(j-1)\alpha n+1}, \ldots, r_{j\alpha n}\} & \text{if } j = 1, \ldots, \lceil \frac{Q}{\alpha n} \rceil - 1 \\ \{r_{(\lceil \frac{Q}{\alpha n} \rceil - 1)\alpha n+1}, \ldots, r_Q\} & \text{if } j = \lceil \frac{Q}{\alpha n} \rceil \\ \emptyset & \text{if } j = \lceil \frac{Q}{\alpha n} \rceil + 1, \ldots, m. \end{cases} \tag{39}$$

Distribute data points to machines:

7: Machine j acquires data points in $\{f_i | i \in S_j \cup R_j\}$ from the storage center.

The correctness and the expected amount of communication of Algorithm 4 are characterized as follows.

Lemma 18 *The sequence $r_1, \ldots, r_Q$ generated in Algorithm 4 has the same joint distribution as a sequence of i.i.d. indices uniformly sampled with replacement from $[N]$. Moreover, the expected amount of communication for distributing $\cup_{i=1}^m \{f_i | i \in R_j \backslash S_j\}$ is at most $\frac{Q^2}{2N}$.*

Proof Conditioned on $i_1, \ldots, i_{\ell-1}$ and $r_1, \ldots, r_{\ell-1}$, the random index i_ℓ has uniform distribution over $[N] \setminus \{i_1, \ldots, i_{\ell-1}\}$. Therefore, by Line 4 in Algorithm 4, the conditional distribution of the random index r_ℓ, conditioning on $i_1, \ldots, i_{\ell-1}$ and $r_1, \ldots, r_{\ell-1}$, is a uniform distribution over $[N]$. Hence, we complete the proof of the first claim of the lemma.

To analyze the amount of communication, we note that $i_\ell \neq r_\ell$ with probability $\frac{\ell-1}{N}$. Suppose $i_\ell \in S_j$ for some j. We know that the data point f_{r_ℓ} needs to be transmitted to machine j separately from S_j only if $i_\ell \neq r_\ell$. Therefore, the expected amount of communication for distributing $\cup_{i=1}^m \{f_i | i \in R_j \backslash S_j\}$ is upper bounded by $\sum_{\ell=1}^Q \frac{\ell-1}{N} \leq \frac{Q^2}{2N}$. ■

According to Lemma 18, besides the (necessary) N amount of communication to distribute $S_1, \ldots, S_m$, Algorithm 4 needs only $\frac{Q^2}{2N}$ additional amount of communication to distribute $R_1, \ldots, R_m$ thanks to the overlaps between S_j and R_j for each j. This additional amount is less than the Q amount required by the straightforward method when $Q \leq N$.

Note that we only need $Q = O(\kappa \log(1/\epsilon))$ in the DSVRG algorithm and $Q = O(n(1 + \sqrt{\frac{\kappa}{n}}) \log(1 + \frac{\kappa}{n}) \log \frac{1}{\epsilon})$ in the DASVRG algorithm. When $\kappa = \Theta(\sqrt{N})$ in a typical ERM problem, we have $\frac{Q^2}{2N} = O((\log(1/\epsilon))^2)$ in DSVRG and $\frac{Q^2}{2N} = O(\sqrt{\frac{n}{m}}(\log(\sqrt{\frac{m}{n}}) \log(1/\epsilon))^2)$ in DASVRG, both of which are much less than the N amount of communication for distributing $S_1, \ldots, S_m$. In other words, although DSVRG and DASVRG require additional amount of communication to allocate the data compared to other algorithms, this additional amount is marginal if κ is not too large.

Although Algorithm 4 is given for allocating S_j and R_j together to machine j, it can be used to only allocate R_j to machine j when S_j has already existed in machine j beforehand. The only requirements are that $\{S_j\}_{j\in[m]}$ still forms a random partition of $[N]$ and that, after generating R_j as (39), machine j knows which machine holds i for each $i \in R_j \backslash S_j$ so that machine j can acquire data in $R_j \backslash S_j$ from that machine in Line 7 of Algorithm 4.

References

Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. In *International Conference on Machine Learning (ICML)*, 2015.

Alekh Agarwal, Sahand Negahban, and Martin Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40 (5):2452–2482, 2012.

Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Necdet Aybat, Zi Wang, and Garud Iyengar. An asynchronous distributed proximal gradient method for composite convex optimization. In *International Conference on Machine Learning (ICML)*, 2015.

Amir Beck, Angelia Nedić, Asuman Ozdaglar, and Marc Teboulle. An $O(1/k)$ gradient method for network resource allocation problems. *Transactions on Control of Network Systems*, 1:64–73, 2014.

Stephen 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.

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 *The ACM Symposium on Theory of Computing*, pages 1011–1020, 2016.

Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed optimization via inexact consensus ADMM. *IEEE Transactions on Signal Processing*, 63:482 – 497, 2015.

Annie I. Chen and Asuman E. Ozdaglar. A fast distributed proximal-gradient method. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 601–608, 2012.

Zihao Chen, Luo Luo, and Zhihua Zhang. Communication lower bounds for distributed convex optimization: Partition data on features. In *AAAI*, pages 1812–1818, 2017.

Vaclav Chvatal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25: 285–287, 1979.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014a.

Aaron Defazio, Justin Domke, and Tiberio S. Caetano. Finito: A faster, permutable incremental gradient method for big data problems. In *International Conference on Machine Learning (ICML)*, 2014b.

Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66:889–916, 2016.

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.

Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning (ICML)*, pages 2540–2548, 2015.

William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

Martin Jaggi, Virginia Smith, Martin Takac, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Dusan Jakovetić, José M. F. Moura, and Joao Xavier. Distributed Nesterov-like gradient algorithms. In *IEEE Annual Conference on Decision and Control (CDC)*, 2012.

Dusan Jakovetić, Joao Xavier, and José M. F. Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59:1131–1146, 2014.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013.

Jakub Konečnỳ and Peter Richtárik. Semi-stochastic gradient descent methods. *Frontiers in Applied Mathematics and Statistics*, 3:9, 2017.

Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical Programming*, 12:1–49, 2017.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. In *International Conference on Machine Learning (ICML)*, 2015.

Mehrdad Mahdavi, Lijun Zhang, and Rong Jin. Mixed optimization for smooth functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25:829–855, 2015.

Ali Makhdoumi and Asuman Ozdaglar. Convergence rate of distributed ADMM over networks. *IEEE Transactions on Automatic Control*, 2017.

Aryan Mokhtari, Wei Shi, Qing Ling, and Alejandro Ribeiro. Dqm: Decentralized quadratically approximated alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 64:5158 –5173, 2016.

João F. C. Mota, João M. F. Xavier, Pedro M. Q. Aguiar, and Markus Püschel. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transaction on Signal Processing*, 61(10):2718–2723, 2013.

Angelia Nedić and Asuman Ozdaglar. On the rate of convergence of distributed subgradient methods for multi-agent optimization. In *IEEE Annual Conference on Decision and Control (CDC)*, 2007.

Angelia Nedić and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54:49–61, 2009.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2013.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2008.

Sashank J Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. AIDE: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2663–2671, 2012.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.

Shai Shalev-Shwartz, Ohad Shamir, Karthik Sridharan, and Nathan Srebro. Stochastic convex optimization. In *International Conference of Machine Learning (ICML)*, 2009.

Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Ohad Shamir and Nathan Srebro. On distributed stochastic optimization and learning. In *Annual Allerton Conference on Communication, Control, and Computing*, 2014.

Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate Newton-type method. In *International Conference on Machine Learning (ICML)*, pages 1000–1008, 2014.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62:1750–1761, 2014.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63:6013–6023, 2015a.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25:944–966, 2015b.

J. V. Uspensky. *Introduction to Mathematical Probability*. McGraw-Hill, 1937.

Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *IEEE Annual Conference on Decision and Control (CDC)*, pages 5445–5450, 2012.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Yuchen Zhang and Xiao Lin. Disco: Distributed optimization for self-concordant empirical loss. In *International Conference on Machine Learning (ICML)*, pages 362–370, 2015.