Communication-Efficient Distributed Optimization in Networks with Gradient Tracking and Variance Reduction

Boyue Li
Carnegie Mellon University

Shicong Cen
Carnegie Mellon University

Yuxin Chen
Princeton University

Yuejie Chi
Carnegie Mellon University

Abstract

Due to the imminent need to alleviate the communication burden in multi-agent and federated learning, the investigation of communication-efficient distributed optimization algorithms for empirical risk minimization has flourished recently. A large fraction of existing algorithms are developed for the *master/slave* setting, relying on the presence of a central parameter server.

This paper focuses on distributed optimization in the *network* setting (also known as the *decentralized* setting), where each agent is only allowed to aggregate information from its neighbors over a graph. By properly adjusting the global gradient estimate via a tracking term, we first develop a communication-efficient approximate Newton-type method, called `Network-DANE`, which generalizes the attractive DANE algorithm to decentralized networks. Our key algorithmic ideas can be applied, in a systematic manner, to obtain decentralized versions of other master/slave distributed algorithms. Notably, we develop `Network-SVRG/SARAH`, which employ stochastic variance reduction at each agent to accelerate local computations. We establish linear convergence of `Network-DANE` and `Network-SVRG` for strongly convex losses, and `Network-SARAH` for quadratic losses, which shed light on the impact of data homogeneity, network connectivity, and local averaging upon the rate of convergence. Numerical evidence is provided to demonstrate the appealing performance of our algorithms over competitive baselines, in terms of both communication and computation efficiency.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

1 Introduction

Distributed optimization has been a classic topic (Bertsekas and Tsitsiklis, 1989), yet is attracting significant attention recently in machine learning due to its numerous emerging applications such as distributed training (Boyd et al., 2011), multi-agent learning (Nedic et al., 2010), and federated learning (Konečnỳ et al., 2015, 2016; McMahan et al., 2017). Broadly speaking, there are two distributed settings that have received wide interest: 1) the *master/slave* setting, which assumes the existence of a central parameter server that can perform information aggregation and sharing with all agents; and 2) the *network* setting — also known as the *decentralized* setting — where each agent is only permitted to communicate with its neighbors over a locally connected network.

Many algorithms have been developed for the master/slave setting to improve communication efficiency, including deterministic algorithms such as one-shot parameter averaging (Zhang et al., 2012), CoCoA (Smith et al., 2018), DANE (Shamir et al., 2014), CEASE (Fan et al., 2019), and stochastic algorithms such as distributed SGD (Recht et al., 2011) and distributed SVRG (Lee et al., 2017; Konečnỳ et al., 2015; Cen et al., 2019). In comparison, the network setting is substantially less explored. It is therefore natural to ask whether one can adapt more appealing algorithmic ideas to the network setting — particularly for network topology with a high degree of locality — without compromising the convergence guarantees attainable in the master/slave counterparts.

1.1 Our Contributions

This paper investigates the problem of empirical risk minimization in the network setting, with the aim of achieving communication and computation efficiency simultaneously. We develop communication-efficient decentralized (stochastic) optimization algorithms with primal-only formulations, with the assistance of gradient tracking. The proposed algorithmic ideas accommodate both approximate Newton-type methods and stochastic variance-reduced methods with provable convergence guarantees.

We start by studying an approximate Newton-type method called DANE (Shamir et al., 2014), which is one of the most popular communication-efficient algorithms to solve empirical risk minimization. However, DANE was only designed for the master/slave setting in its original form. The main challenge in extending such an algorithm to the network setting is to track and adapt a faithful estimate of the global gradient at each agent, despite the lack of centralized information aggregation. Towards this end, we leverage the powerful idea of *dynamic average consensus* in control (Zhu and Martínez, 2010) to track and correct the locally aggregated gradients at each agent — a scheme commonly referred to as *gradient tracking*. We then employ the corrected gradient in local computation, according to the subroutine adapted from DANE. This simple idea leads to `Network-DANE`, which generalizes the approximate Newton-type method DANE to the network setting, without the need of communicating the Hessians.

Our ideas for designing `Network-DANE` can be extended, in a systematic manner, to obtain decentralized versions of other algorithms developed for the master/slave setting, by modifying the local computation step properly. As a notable example, we develop `Network-SVRG`, which performs variance-reduced stochastic optimization locally to enable further computational savings (Johnson and Zhang, 2013). The same approach can be applied to other distribute stochastic variance-reduced methods such as SARAH (Nguyen et al., 2017) to obtain `Network-SARAH`.

The proposed algorithms achieve an intriguing trade-off between communication and computation efficiency. During each iteration, each agent only communicates the parameter and the gradient estimate to its neighbors, and is therefore communication-efficient globally; moreover, the local subproblems at each agent can be solved efficiently with accelerated or variance-reduced gradient methods, and is thus computation-efficient locally. Theoretically, we establish the linear convergence of `Network-DANE` and `Network-SVRG` for smooth strongly convex losses, and the linear convergence of `Network-SARAH` for quadratic losses, using the method of Lyapunov functions to handle the tight coupling of optimization and network consensus errors.

Our results shed light on the impact of data homogeneity and network connectivity upon the rate of convergence; in particular, the proposed algorithms provably obtain faster convergence if the local data become more similar. When the network exhibits a high degree of locality, we show that by allowing multiple rounds of local mixing within each iteration, an improved overall communication complexity can be achieved as it accelerates the rate of convergence. Extensive numerical experiments are provided to corroborate our theoretical findings, and demonstrate the practical efficacy of the proposed algorithms over competitive baselines.

1.2 Related Work

First-order methods are of core interest to big data analytics due to their superior scalability. However, it is well-known that distributed gradient descent (DGD) suffers from a "speed" versus "accuracy" dilemma when naïvely implemented in a decentralized setting (Nedić et al., 2018). Various fixes have been proposed to address this issue, including the pioneering work such as EXTRA (Shi et al., 2015) and NEXT (Di Lorenzo and Scutari, 2016). Similar gradient tracking ideas (Zhu and Martínez, 2010) have been incorporated to adjust DGD to ensure its linear convergence using a constant step size (Nedić et al., 2017; Qu and Li, 2018; Li et al., 2019b; Scutari and Sun, 2019; Xin et al., 2019b). The current paper is inspired by the use of gradient tracking in these early results. Our paper implements and verifies the effectiveness of gradient tracking for algorithms that involve approximate Newton and variance reduction steps, which are far from straightforward and require significant efforts. During the preparation of this paper, it was brought to our attention that the SONATA algorithm (Sun et al., 2019b), which also applies gradient tracking with convergence guarantees, can be specialized to obtain the same local sub-problem studied in `Network-DANE`, up to different mixing approaches.

Scaman et al. (2017) proposed a multi-step dual accelerated (MSDA) method for network-distributed optimization, which is optimal within a class of black-box procedures that satisfy the span assumption — the parameter updates fall in the span of the previous estimates and their gradients. Further optimal algorithms are proposed by Uribe et al. (2017) and Scaman et al. (2018) for loss functions that are not necessarily convex or smooth. Their algorithms require knowledge of the dual formulation. In contrast, our algorithms are directly applied to the primal problem, which are more friendly for problems whose dual formulations are hard to obtain. Our algorithms also do not obey the span assumption and therefore do not fall into the class of procedures studied by Scaman et al. (2017).

Fan et al. (2019) recently proposed algorithm CEASE extended DANE with an additional proximal term in the objective function and strengthened its analysis. The connections between DANE and SVRG observed by Konečnỳ et al. (2015) motivate the development of `Network-SVRG` in this paper, which can be viewed as replacing the local optimization of `Network-DANE` with variance-reduced stochastic gradient methods. The same idea can be easily applied to obtain network-distributed versions of other algorithms such as Katyusha (Allen-Zhu, 2017). Compared with decentralized SGD (Lan et al., 2017; Lian et al., 2017), the proposed `Network-SVRG/SARAH` employ variance

reduction to achieve much faster convergence. We note that variance-reduced methods have been adapted to the network setting by Mokhtari and Ribeiro (2016); Yuan et al. (2018); Sun et al. (2019a); Xin et al. (2019a); however, they either have a large memory complexity or impose substantial communication burdens.

Paper organization and notations. Section 2 introduces the problem formulation and presents some preliminaries. Section 3 presents the proposed `network-DANE` together with its theoretical guarantees. Section 4 presents `network-SVRG/SARAH`, which apply variance reduction to further reduce local computation. We provide numerical experiments in Section 5 and conclude in Section 6. Extra experiments and all proofs are provided by Li et al. (2019a) due to space limits. Throughout this paper, we use boldface letters to represent vectors and matrices. In addition, $\|\boldsymbol{A}\|$ denotes the spectral norm of a matrix $\boldsymbol{A}$, $\|\boldsymbol{a}\|_2$ represents the ℓ_2 norm of a vector $\boldsymbol{a}$, $\otimes$ stands for the Kronecker product, and $\boldsymbol{I}_n$ denotes the identity matrix of dimension n.

2 Formulation and Preliminaries

Consider the empirical risk minimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \quad f(\boldsymbol{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N}\ell(\boldsymbol{x};\boldsymbol{z}_i), \tag{1}$$

where $\boldsymbol{x}\in\mathbb{R}^d$ represents the parameter to optimize, $\ell(\boldsymbol{x};\boldsymbol{z}_i)$ encodes certain empirical loss of $\boldsymbol{x}$ w.r.t. the ith sample $\boldsymbol{z}_i$, and N denotes the total number of samples. This paper primarily focuses on the case where the function $\ell(\cdot;\boldsymbol{z})$ is both convex and smooth for any given $\boldsymbol{z}$, although we shall also study nonconvex problems in numerical experiments.

In a distributed optimization framework, the data samples are distributed over n agents. For simplicity, we assume throughout that data samples are split into disjoint subsets of equal sizes. The jth local data set, represented by $\mathcal{M}_j$, thus contains $m \triangleq N/n$ samples. As such, the global loss function can alternatively be represented by

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} f_j(\boldsymbol{x}), \text{ with } f_j(\boldsymbol{x}) \triangleq \frac{1}{m}\sum_{\boldsymbol{z}\in\mathcal{M}_j}\ell(\boldsymbol{x};\boldsymbol{z}). \tag{2}$$

Here, $f_j(\boldsymbol{x})$ denotes the local loss function at the jth agent ($1\le j\le n$). In addition, there exists a network — represented by an undirected graph $\mathcal{G}$ of n nodes — that captures the local connectivity across all agents. More specifically, each node in $\mathcal{G}$ represents an agent, and two agents are allowed to exchange information only if there is an edge connecting them in $\mathcal{G}$. The set of neighbors of the jth agent over $\mathcal{G}$ is denoted by $\mathcal{N}_j$.

The goal is to minimize $f(\cdot)$ in a decentralized manner, subject to the aforementioned network-based communication constraints. Before continuing, we find it helpful to introduce and explain two important concepts.

Mixing. Mathematically, the mixing of information between neighboring nodes is often characterized by a mixing or gossiping matrix, denoted by $\boldsymbol{W} = [w_{ij}]_{1\le i,j\le n} \in \mathbb{R}^{n\times n}$. More specifically, this matrix satisfies

$$\boldsymbol{W}^\top\boldsymbol{1}_n = \boldsymbol{1}_n \qquad \text{and} \qquad \boldsymbol{W}\boldsymbol{1}_n = \boldsymbol{1}_n, \tag{3}$$

where $\boldsymbol{1}_n\in\mathbb{R}^n$ is the all-one vector. The spectral quantity, which we call the *mixing rate*,

$$\alpha_0 \triangleq \|\boldsymbol{W} - \tfrac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\| \in [0,1) \tag{4}$$

dictates how fast information mixes over the network. As an example, in a fully-connected network, one can attain $\alpha_0 = 0$ by setting $\boldsymbol{W} = \frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top$. Comprehensive bounds on $1/(1-\alpha_0)$ for various graphs are provided by Nedić et al. (2018). For instance, one has $1/(1-\alpha_0)\asymp 1$ with high probability for an Erdös-Rényi random graph, as long as the edge between each pair of nodes is connected independently with probability $p = O(\log n/n)$.

Dynamic average consensus. Assume that each agent generates some *time-varying* quantity $r_j^{(t)}$ (e.g. the current local parameter estimate). We are interested in tracking the dynamic average $\frac{1}{n}\sum_{j=1}^n r_j^{(t)} = \frac{1}{n}\boldsymbol{1}_n^\top\boldsymbol{r}^{(t)}$ at each of the agents, where $\boldsymbol{r}^{(t)} = [r_1^{(t)},\cdots,r_n^{(t)}]^\top$. To accomplish this, Zhu and Martínez (2010) proposed a simple tracking algorithm: suppose each agent maintains an estimate $q_j^{(t)}$ in the tth iteration, then the network collectively adopts the following update rule

$$\boldsymbol{q}^{(t)} = \boldsymbol{W}\boldsymbol{q}^{(t-1)} + \boldsymbol{r}^{(t)} - \boldsymbol{r}^{(t-1)}, \tag{5}$$

where $\boldsymbol{q}^{(t)} = [q_1^{(t)},\cdots,q_n^{(t)}]^\top$. The first term $\boldsymbol{W}\boldsymbol{q}^{(t-1)}$ represents the standard information mixing operation, whereas the term $\boldsymbol{r}^{(t)} - \boldsymbol{r}^{(t-1)}$ tracks the (time-varying) temporal difference. A crucial property of (5) is $\boldsymbol{1}_n^\top\boldsymbol{q}^{(t)} = \boldsymbol{1}_n^\top\boldsymbol{r}^{(t)}$, which indicates that the average of $\{q_i^{(t)}\}_{1\le i\le n}$ dynamically tracks the average of $\{r_i^{(t)}\}_{1\le i\le n}$. We shall adapt this procedure in our algorithmic development, in the hope of reliably tracking the global gradients (i.e. the average of the local, and often time-varying, gradients at all agents).

3 `Network-DANE`

In this section, we propose an algorithm called `Network-DANE` (cf. Alg. 1), which generalizes DANE (Shamir et al., 2014) to the network / decentralized setting. This is accomplished by carefully coordinating the information sharing mechanism and employing dynamic average consensus for gradient tracking.

Algorithm 1 `Network-DANE`

input: initial parameter estimate $\boldsymbol{x}_j^{(0)} \in \mathbb{R}^d$ $(1 \le j \le n)$, regularization parameter μ.
initialization: set $\boldsymbol{y}_j^{(0)} = \boldsymbol{x}_j^{(0)}$, $\boldsymbol{s}_j^{(0)} = \nabla f_j(\boldsymbol{y}_j^{(0)})$ for all agents $1 \le j \le n$.
for $t = 1, 2, \cdots$ **do**
  for Agents $1 \le j \le n$ in parallel **do**
    Set $\boldsymbol{y}_j^{(t),0} = \boldsymbol{x}_j^{(t-1)}$ and $\boldsymbol{s}_j^{(t),0} = \boldsymbol{s}_j^{(t-1)}$.
    for $k = 1, 2, \ldots, K$ **do**
      Receive information $\boldsymbol{y}_i^{(t),k-1}$ and $\boldsymbol{s}_i^{(t),k-1}$ from its neighbors $i \in \mathcal{N}_j$.
      Aggregate parameter estimates from neighbors:

$$\boldsymbol{y}_j^{(t),k} = \sum\nolimits_{i \in \mathcal{N}_j} w_{ji} \boldsymbol{y}_i^{(t),k-1}, \tag{6a}$$

$$\boldsymbol{s}_j^{(t),k} = \sum\nolimits_{i \in \mathcal{N}_j} w_{ji} \boldsymbol{s}_i^{(t),k-1}. \tag{6b}$$

    end for
    Set $\boldsymbol{y}_j^{(t)} = \boldsymbol{y}_j^{(t),K}$, and update the global gradient estimate by gradient tracking:

$$\boldsymbol{s}_j^{(t)} = \boldsymbol{s}_j^{(t),K} + \underbrace{\nabla f_j\big(\boldsymbol{y}_j^{(t)}\big) - \nabla f_j\big(\boldsymbol{y}_j^{(t-1)}\big)}_{\text{gradient tracking}}. \tag{7}$$

    Update the parameter estimate by solving:

$$\boldsymbol{x}_j^{(t)} = \operatorname*{argmin}_{\boldsymbol{z} \in \mathbb{R}^d} \Big\{ f_j(\boldsymbol{z}) - \big\langle \nabla f_j(\boldsymbol{y}_j^{(t)}) - \boldsymbol{s}_j^{(t)}, \boldsymbol{z} \big\rangle + \frac{\mu}{2} \big\| \boldsymbol{z} - \boldsymbol{y}_j^{(t)} \big\|_2^2 \Big\}. \tag{8}$$

  end for
end for

3.1 Algorithm Development

Recall the DANE algorithm Shamir et al. (2014) developed for the master/slave setting. In DANE, each agent performs an update using both the local loss function $f_j(\cdot)$ and the gradient $\nabla f(\cdot)$ of the global loss function. In the tth iteration, the jth agent solves the following Newton-type problem to update its local estimate $\boldsymbol{x}_j^{(t)}$:

$$\boldsymbol{x}_j^{(t)} = \operatorname*{arg\,min}_{\boldsymbol{z} \in \mathbb{R}^d} f_j(\boldsymbol{z}) - \Big\langle \nabla f_j\big(\overline{\boldsymbol{x}}^{(t)}\big) - \nabla f\big(\overline{\boldsymbol{x}}^{(t)}\big), \boldsymbol{z} \Big\rangle + \frac{\mu}{2} \big\| \boldsymbol{z} - \overline{\boldsymbol{x}}^{(t)} \big\|_2^2, \tag{9}$$

where $\mu > 0$ is a tuning parameter.[1] Here, $\overline{\boldsymbol{x}}^{(t)} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{x}_i^{(t-1)}$ and $\nabla f(\overline{\boldsymbol{x}}^{(t)}) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\overline{\boldsymbol{x}}^{(t)})$ are the global estimate of the parameter and the gradient, respectively, which can be obtained via the assistance of a parameter server. In the network setting, however, the agents can no longer compute (9) locally, due to the absence of centralization enabled by the parameter server. More specifically, the agents do not have access to either $\overline{\boldsymbol{x}}^{(t)}$ or $\nabla f(\overline{\boldsymbol{x}}^{(t)})$, both of which are required when solving (9). To address this lack of global information, one might naturally wonder whether we can simply replace global averaging by local averaging; that is, replacing $\overline{\boldsymbol{x}}^{(t)}$ and $\nabla f(\overline{\boldsymbol{x}}^{(t)})$ by $\frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \boldsymbol{x}_i^{(t-1)}$ and $\frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \nabla f_i(\boldsymbol{x}_i^{(t-1)})$, respectively, at the jth agent. However, this simple idea fails to guarantee convergence at local agents. For instance, the local estimation errors may stay unchanged (but nonvanishing) — as opposed to converging to zero — as the iterations progress, primarily due to imperfect information sharing.

With this convergence issue in mind, our key idea is to maintain an additional estimate of the global gradient at each agent — denoted by $\boldsymbol{s}_j^{(t)}$ at the jth agent. This additional gradient estimate is updated via dynamic average consensus (7), in the hope of tracking the global gradient at $\boldsymbol{y}_j^{(t)}$, i.e. $\nabla f(\boldsymbol{y}_j^{(t)})$. Here, $\boldsymbol{y}_j^{(t)}$ stands for the parameter estimate obtained after local mixing in the tth iteration (see Alg. 1 for details). As the algorithm converges, $\boldsymbol{y}_j^{(t)}$ is expected to reach consensus, allowing $\boldsymbol{s}_j^{(t)}$ to converge to the true global gradient.

In addition, we also allow multiple rounds of mixing within each iteration, i.e. (6), which is particularly helpful to accelerate the convergence when the network exhibits a high degree of locality. In effect, by applying K rounds of mixing, we improve the mixing rate to

$$\alpha = \alpha_0^K. \tag{10}$$

It reduces the network setting to the master/slave setting by setting $K \to \infty$. As we shall see later, choosing a proper (but not too large) K leads to a desirable trade-off between consensus and optimization, which helps improve the overall communication cost.

Armed with such improved global gradient estimates, we propose to solve a modified local optimization subproblem (8) in `Network-DANE`, which approximates the original Newton-type problem (9) by replacing $\nabla f(\overline{\boldsymbol{x}}^{(t)})$ with the local surrogate $\boldsymbol{s}_j^{(t)}$. The proposed local subproblem (8) is convex and can be solved efficiently via, say, Nesterov's accelerated gradient methods. The whole algorithm is presented in Alg. 1.

3.2 Convergence Guarantees

In this subsection, we provide theoretical guarantees for the convergence of `Network-DANE` under one or more of the following assumptions.

Assumption 1 (strongly convex loss)**.** *The loss function $f_j(\boldsymbol{x})$ at each agent is strongly convex and smooth,*

[1] In Shamir et al. (2014), the second term in (9) contains an extra tuning parameter $\tilde{\eta}$ as $\nabla f_j(\overline{\boldsymbol{x}}^{(t)}) - \tilde{\eta} \nabla f(\overline{\boldsymbol{x}}^{(t)})$. We set $\tilde{\eta} = 1$ without loss of generality following Fan et al. (2019).

namely, $\sigma \boldsymbol{I} \preceq \nabla^2 f_j(\boldsymbol{x}) \preceq L\boldsymbol{I}$ $(1 \le j \le n)$ *for some quantities* $0 < \sigma \le L$*, where* $\kappa = L/\sigma$ *is the condition number.*

Assumption 2 (quadratic loss)**.** *The loss function* $f_j(\boldsymbol{x})$ *at each agent is quadratic w.r.t.* $\boldsymbol{x}$*.*

We further introduce a key quantity (Cen et al., 2019; Fan et al., 2019), called the homogeneity parameter,

$$\beta := \max_{1\le j\le n} \beta_j,\ \beta_j := \sup_{\boldsymbol{x}\in\mathbb{R}^d} \left\|\nabla^2 f_j(\boldsymbol{x}) - \nabla^2 f(\boldsymbol{x})\right\|, \quad (11)$$

which measures the similarity of data across agents. Let the global optimizer of $f(\boldsymbol{x})$ be

$$\boldsymbol{y}^{\mathsf{opt}} := \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}). \quad (12)$$

We define the (nd)-dimensional vector $\boldsymbol{x}^{(t)}$ by

$$\boldsymbol{x}^{(t)} := \left[\boldsymbol{x}_1^{(t)\top}, \cdots, \boldsymbol{x}_n^{(t)\top}\right]^\top, \quad (13)$$

and similarly define $\boldsymbol{y}^{(t)}$ and $\boldsymbol{s}^{(t)}$. The average of each (nd)-dimensional vector $\boldsymbol{x} = [\boldsymbol{x}_1^\top, \cdots, \boldsymbol{x}_n^\top]^\top$ is defined by $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_j \boldsymbol{x}_j \in \mathbb{R}^d$. In addition, we introduce the distributed gradient $\nabla F(\boldsymbol{x}) \in \mathbb{R}^{nd}$ and the full gradient $\nabla f(\boldsymbol{x}) \in \mathbb{R}^{nd}$ of an (nd)-dimensional vector $\boldsymbol{x}$ as

$$\nabla F(\boldsymbol{x}) := [\nabla f_1(\boldsymbol{x}_1)^\top, \cdots, \nabla f_n(\boldsymbol{x}_n)^\top]^\top, \quad (14a)$$

$$\nabla f(\boldsymbol{x}) := [\nabla f(\boldsymbol{x}_1)^\top, \cdots, \nabla f(\boldsymbol{x}_n)^\top]^\top. \quad (14b)$$

To characterize the convergence behavior of our algorithm, we need to simultaneously track several interrelated error metrics: (1) the convergence error: $\left\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\right\|_2$; (2) the parameter consensus error: $\left\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\right\|_2$; and (3) the gradient estimation error: $\left\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\right\|_2$. The `Network-DANE` algorithm is said to converge linearly with a rate $\rho \in (0,1)$ if there exists some constant $C > 0$ such that for all $t \ge 1$,

$$\max\Big\{\sqrt{n}\left\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\right\|_2, \left\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\right\|_2, L^{-1}\left\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\right\|_2\Big\} \le C\rho^t$$

holds. By properly setting the tuning parameter μ, we can guarantee the linear convergence of `Network-DANE`. This is formally supplied in Theorem 1 for quadratic objective functions, and in Theorem 2 for general smooth and strongly convex objective functions.

Theorem 1 (`Network-DANE` for quadratic losses)**.** *Suppose Assumptions 1 and 2 hold. Set* μ *such that* $\sigma + \mu \ge \frac{140L}{(1-\alpha)^2}\left(\frac{\beta}{\sigma}+1\right)$*, where* $\alpha = \alpha_0^K$*.* ***`Network-DANE`*** *converges linearly with a rate* ρ_1 *upper bounded by*

$$\max\left\{\frac{1+\theta_1}{2}, \alpha + \frac{140\kappa}{1-\alpha}\left(\frac{\sigma+\beta}{\sigma+\mu}\right), \frac{1+\alpha}{2} + \frac{2\beta}{\sigma+\mu}\right\},$$

where θ_1 *given by*

$$\theta_1 := 1 - \frac{\sigma}{\sigma+\mu} + \frac{L}{L+\mu}\frac{\beta^2}{(\sigma+\mu)(\sigma+\mu-\beta)}. \quad (15)$$

Remark 1. θ_1 *is the convergence rate of DANE in the master/slave setting (Shamir et al., 2014, Theorem 1).*

We have spent no efforts to optimize the constants in the theorem. In view of Theorem 1, if the network is sufficiently connected ($\alpha = \alpha_0^K$ is small), or the data are sufficiently homogeneous across the agents (β is small), we can use a smaller parameter μ, which in turn makes θ_1 (defined in (15)) smaller and results in a faster convergence rate. In summary, `Network-DANE` takes more iterations to converge when α and β are large. This is formalized by the following corollary.

Corollary 1. *Instate the assumptions of Theorem 1, and set* $\mu + \sigma = \frac{180L}{(1-\alpha)^2}(\frac{\beta}{\sigma}+1)$*. To reach* ε*-accuracy,* ***`Network-DANE`*** *takes at most* $O(\kappa(\beta/\sigma+1)\log(1/\varepsilon)/(1-\alpha)^2)$ *iterations and* $O(K\cdot\kappa(\beta/\sigma+1)\log(1/\varepsilon)/(1-\alpha)^2)$ *communication rounds.*

By Corollary 1, if we set the number of local averaging rounds as $K = 1$ and $\alpha = \alpha_0$, to reach ε-accuracy, `Network-DANE` takes no more than $O\big(\kappa(1+\beta/\sigma)\log(1/\varepsilon)/(1-\alpha_0)^2\big)$ iterations/communication rounds. If the homogeneous parameter $\beta \asymp \sigma$ is on the order of σ, this improves to $O\big(\kappa\log(1/\varepsilon)/(1-\alpha_0)^2\big)$, which is much faster than the corrected DGD (Qu and Li, 2018) with gradient tracking, which converges in $O(\kappa^2\log(1/\varepsilon)/(1-\alpha_0)^2)$ iterations. The convergence rate of `Network-DANE` degenerates to that of DGD (Qu and Li, 2018) with gradient tracking under the worst condition $\beta \asymp O(L)$. This observation highlights the communication efficiency of `Network-DANE` by harnessing the homogeneity of data across different agents.

More interestingly, consider the case where $K > 1$, where `Network-DANE` performs K rounds of communications per iteration. The total communication cost to reach ε-accuracy, in terms of the native network parameter α_0, is $O\big(K\cdot\kappa(1+\beta/\sigma)\log(1/\varepsilon)/(1-\alpha_0^K)^2\big)$. Therefore, by judiciously choosing K, it is possible to significantly improve the overall communication complexity, especially when α_0 is close to 1. For example, by setting $K \asymp 1/\log(1/\alpha_0) = O(1/(1-\alpha_0))$, we can ensure $\alpha_0^K \asymp 1/2$ and reduce the communication complexity to $O\big(\kappa\cdot(\beta/\sigma+1)\log(1/\varepsilon)/(1-\alpha_0)\big)$, thus improving the dependence with the graph topology.

With more refined analysis, we can show that by setting $K = O(\log\kappa/(1-\alpha_0))$ so that $\alpha = \alpha_0^K \asymp 1/(2\kappa)$, `Network-DANE` takes no more than $O\left((\beta^2/\sigma^2+1)\log(1/\varepsilon)\right)$ iterations, and $O\big(\log\kappa\cdot(\beta^2/\sigma^2+1)\log(1/\varepsilon)/(1-\alpha_0)\big)$ communications rounds to reach ε-accuracy for quadratic losses, which are almost independent of κ when $\beta = O(\sigma)$.

We now state the convergence rate of `Network-DANE` for the strongly convex setting, which is weaker than the rate for quadratic loss functions.

Theorem 2. *Suppose Assumption 1 holds. Set* μ *such that* $\sigma + \mu \ge \frac{170\kappa L}{(1-\alpha)^2}$*, where* $\alpha = \alpha_0^K$*.* ***`Network-DANE`***

converges linearly with a rate ρ_2 upper bounded by

$$\max\left\{\frac{1+\theta_2}{2}, \alpha+\frac{170\kappa}{1-\alpha}\left(\frac{L}{\sigma+\mu}\right), \frac{1+\alpha}{2}+\frac{2\beta}{\sigma+\mu}\right\},$$

where θ_2 is given by

$$\theta_2 := 1-\frac{\sigma}{\sigma+\mu}+\frac{\beta}{\sigma+\mu}\sqrt{1-\left(\frac{\mu}{\sigma+\mu}\right)^2}. \quad (16)$$

Remark 2. *θ_2 is the convergence rate of DANE in the master/slave setting (Fan et al., 2019, Theorem 3.1).*

Comparing the expressions of θ_1 and θ_2, we clearly see that one pays a price for covering more general loss functions. The complexity of `Network-DNAE` is summarized in Corollary 2, which is worse than Corollary 1.

Corollary 2. *Under the same assumptions of Theorem 2, set $\sigma+\mu = 180\kappa L/(1-\alpha)^2$. To reach ε-accuracy, `Network-DANE` takes no more than $O\big(\kappa^2\log(1/\varepsilon)/(1-\alpha)^2\big)$ iterations and $O\big(K\cdot\kappa^2\log(1/\varepsilon)/(1-\alpha)^2\big)$ communication rounds.*

The above complexity of `Network-DANE` is quite pessimistic; numerical experiments in Section 5 have suggested that the performance of `Network-DANE` is rather insensitive to the condition number κ. Similar to the quadratic case, by setting K appropriately so that $\alpha = \alpha_0^K \asymp 1/(2\kappa)$, `Network-DANE` takes no more than $O(\kappa(\beta/\sigma+1)\log(1/\varepsilon))$ iterations and $O\big(\log\kappa\cdot\kappa(\beta/\sigma+1)\log(1/\varepsilon)/(1-\alpha_0)\big)$ communication rounds to reach ε-accuracy for general strongly convex losses. This again outperforms DGD with gradient tracking when $\beta = O(\sigma)$, highlighting the benefits of extra averaging.

Remark 3. *The quantity β can be fairly small if the data sets across different agents are sufficiently similar. Shamir et al. (2014) provided bounds on β with respect to the sample size m if the data samples at all agents are i.i.d., with $\ell(\boldsymbol{x};\boldsymbol{z})$ in (2) satisfying $0 \preceq \nabla^2\ell(\boldsymbol{x};\boldsymbol{z}) \preceq L$ for all $\boldsymbol{z}$. With probability at least $1-\delta$ over the samples, we have $\beta < \sqrt{\frac{32L^2}{m}\log\frac{nd}{\delta}}$. Therefore, the convergence of `Network-DANE` is better than DGD if the local data size m is sufficiently large.*

Remark 4. *The homogeneity parameter β (11) measures the largest deviation of local Hessian from the global Hessian. A refined analysis using local deviation β_j is possible by permitting different regularization parameters μ_j in (8) for each agent.*

4 Generalizing the Algorithm Design with Variance Reduction

The design of `Network-DANE` suggests a systematic approach to obtain decentralized versions of other algorithms, which we illustrate by reducing local computation of `Network-DANE` using stochastic variance reduction methods. Stochastic variance reduction methods are a popular class of stochastic optimization algorithms, developed to allow for constant step sizes and faster convergence in finite-sum optimization (Johnson and Zhang, 2013; Xiao and Zhang, 2014; Nguyen et al., 2017). It is therefore natural to ask whether such variance reduction techniques can be leveraged in a network setting to further save local computation without compromising communication.

Algorithm 2 `Network-SVRG/SARAH`

1: Replace the local optimization (8) of `Network-DANE` by the following:
2: **Input:** $\boldsymbol{y}_j^{(t)}$, $\boldsymbol{s}_j^{(t)}$, step size δ, number of local iterations S.
3: **Initialization:** set $\boldsymbol{u}_j^{(t),0} = \boldsymbol{y}_j^{(t)}$, $\boldsymbol{v}_j^{(t),0} = \boldsymbol{s}_j^{(t)}$.
4: **for** $s = 1, ..., S$ **do**
5: $\quad\boldsymbol{u}_j^{(t),s} = \boldsymbol{u}_j^{(t),s-1} - \delta\boldsymbol{v}_j^{(t),s-1}$.
6: $\quad$Sample $\boldsymbol{z}$ from $\mathcal{M}_j$ uniformly at random, then,

$$\boldsymbol{v}_j^{(t),s} = \nabla\ell(\boldsymbol{u}_j^{(t),s};\boldsymbol{z}) - \nabla\ell(\boldsymbol{u}_j^{(t),0};\boldsymbol{z}) + \boldsymbol{v}_j^{(t),0}; \quad \texttt{(SVRG)} \quad (17)$$

$$\boldsymbol{v}_j^{(t),s} = \nabla\ell(\boldsymbol{u}_j^{(t),s};\boldsymbol{z}) - \nabla\ell(\boldsymbol{u}_j^{(t),s-1};\boldsymbol{z}) + \boldsymbol{v}_j^{(t),s-1}.\texttt{(SARAH)} \quad (18)$$

7: **end for**
8: Choose the new parameter estimate $\boldsymbol{x}_j^{(t)}$ from $\{\boldsymbol{u}_j^{(t),1}, \cdots, \boldsymbol{u}_j^{(t),S}\}$ uniformly at random.

Inspired by the connection between DANE and SVRG , we introduce `Network-SVRG/SARAH` in Alg. 2, a distributed variant of SVRG and SARAH tailored to the decentralized setting, with the assistance of proper gradient tracking. In particular, the inner loop of SVRG (Johnson and Zhang, 2013) or SARAH (Nguyen et al., 2017) is adopted in place of the local computation step (8) of `Network-DANE`, where the reference to global gradient is replaced by $\boldsymbol{s}_j^{(t)}$ to calculate the variance-reduced stochastic gradient. The convergence analysis of Alg. 2 is more challenging due to the biased stochastic gradient involved in each local iteration. Encouragingly, the theorem below establishes the linear convergence of `Network-SVRG` for strongly convex losses, and of `Network-SARAH` for quadratic losses, as long as β is sufficiently small and the number of mixing rounds K is sufficiently large. Again, the constants are not optimized in the theorem.

Theorem 3. *Suppose that the sample loss $\ell(\boldsymbol{x};\boldsymbol{z})$ is convex and L-smooth w.r.t. $\boldsymbol{x}$ for all $\boldsymbol{z}$. and Assumption 1 holds. If $\beta/\sigma \le 1/200$, then by setting K such that $\alpha = \alpha_0^K \asymp 1/\kappa$ and S large enough, `Network-SVRG` converges linearly. If Assumption 2 further holds, `Network-SARAH` also converges linearly. To reach ε-accuracy, `Network-SVRG/SARAH` take no more than $O\big(\log(1/\varepsilon)\big)$ iterations and $O\big(\log\kappa\log(1/\varepsilon)/(1-\alpha_0)\big)$*

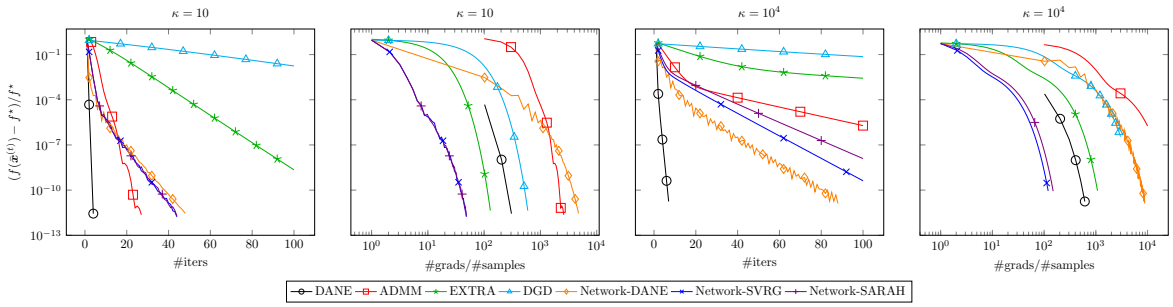

Figure 1: The relative optimality gap with respect to the number of iterations and gradient evaluations under different conditioning $\kappa = 10$ (left two panels) and $\kappa = 10^4$ (right two panels) for logistic regression.

communication rounds.

It is straightforward to extend this idea to obtain decentralized variants of other stochastic variance reduced algorithms such as Katyusha (Allen-Zhu, 2017), by replacing the local computation step (8) with the inner loop update rules of the stochastic methods of interest. For the sake of brevity, this paper does not pursue such "plug-and-play" extensions.

5 Numerical Experiments

In this section, we evaluate the performance of `Network-DANE` and `Network-SVRG/SARAH`[2] for solving both strongly convex and nonconvex problems, in order to demonstrate its appealing performance in terms of communication-computation trade-offs. Throughout, we set the number of agents to be $n = 20$. We use a symmetric fastest distributed linear averaging (FDLA) matrix (Xiao and Boyd, 2004) generated according to the communication graph as the mixing matrix $\boldsymbol{W}$ for aggregating $\boldsymbol{x}_j^{(t)}$ in (6). For aggregating $\boldsymbol{s}_j^{(t)}$ in (7), we use a convex combination of $\boldsymbol{I}$ and $\boldsymbol{W}$ which leads to more stable performance in practice.

Experiments for the strongly convex setting. We conduct a synthetic numerical experiment on logistic regression. We generate connected random communication graphs using an Erdös-Rènyi graph with $p = 0.3$. The same random starting point $\boldsymbol{x}^{(0)}$ and mixing matrix $\boldsymbol{W}$ are used for all algorithms.

The loss function of each agent is given as

$$f_i(\boldsymbol{x}) = -\frac{1}{m}\sum_{j=1}^{m}\left[b_i^{(j)}\log\left(\frac{1}{1+\exp(\boldsymbol{x}^\top \boldsymbol{a}_i^{(j)})}\right) + (1-b_i^{(j)})\log\left(\frac{\exp(\boldsymbol{x}^\top \boldsymbol{a}_i^{(j)})}{1+\exp(\boldsymbol{x}^\top \boldsymbol{a}_i^{(j)})}\right)\right] + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2,$$

where $\boldsymbol{a}_i^{(j)} \in \mathbb{R}^d$ and $b_i^{(j)} \in \{0,1\}$ are samples stored at agent i. We generate $m = 1000$ i.i.d. random samples from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for each agent, and change the condition number κ by changing λ. We generate data according to $b_i^{(j)} = \mathbb{I}(1 + \exp(-\boldsymbol{x}_0{}^\top \boldsymbol{a}_i^{(j)})) < 0.5)$ with a random signal $\boldsymbol{x}_0$, then flip 5% of labels uniformly at random, where $\mathbb{I}(\cdot)$ denotes the indicator function.

We compute the relative optimality gap, given as $(f(\overline{\boldsymbol{x}}^{(t)}) - f^\star)/f^\star$, where $\overline{\boldsymbol{x}}^{(t)}$ is the average parameter of all agents at the tth iteration, and $f^\star$ denotes the optimal value. We compare `Network-DANE` (Alg. 1) and `Network-SVRG/SARAH` (Alg. 2) with DANE (Shamir et al., 2014) and ADMM (Boyd et al., 2011),[3] and two decentralized gradient descent algorithms DGD (Qu and Li, 2018) and EXTRA (Shi et al., 2015).

Fig. 1 shows the relative optimality gap with respect to the number of iterations as well as the number of gradient evaluations under different conditioning $\kappa = 10$ and $\kappa = 10^4$ for logistic regression. In both experiments, `Network-DANE` and `Network-SVRG/SARAH` significantly outperform DGD and EXTRA in terms of the numbers of communication rounds. `Network-SVRG/SARAH` has similar communication rounds with ADMM but only communicates locally. In both experiments, `Network-DANE` is quite insensitive to the condition number, performing nearly as well as the centralized DANE algorithm, but operates in a fully decentralized setting. On the other hand, `Network-SVRG/SARAH` further outperforms other algorithms in terms of gradient evaluations in most settings, especially for well-conditioned cases. The performances of `Network-SVRG` and `Network-SARAH` are almost indistinguishable.

Benefits of extra mixing. We conduct synthetic experiments to investigate the communication-computation trade-off observed in Section 3.2 when employing multiple rounds of mixing within every it-

[2] In our experiments, we use the last iterate $\boldsymbol{u}_j^{(t),S}$ as the new parameter estimate, which is more practical. Code is available at github.com/liboyue/Network-Distributed-Algorithm.

[3] We apply ADMM to the constrained optimization problem, which amounts to the master/slave setting, $\min_{\boldsymbol{x}_i, \boldsymbol{x}} \frac{1}{n}\sum f_i(\boldsymbol{x}_i)$ s.t. $\boldsymbol{x}_i = \boldsymbol{x}$. Note that ADMM can also be applied to the network setting, which is not shown here since our network algorithms already outperform ADMM in the master/slave setting.

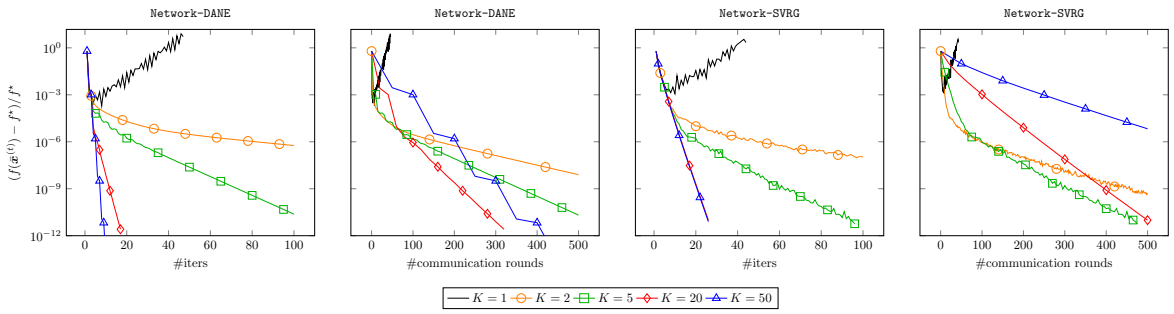

Figure 2: The relative optimality gap with respect to the number of iterations and communication rounds under different rounds of mixing K for `Network-DANE` and `Network-SVRG`. The mixing rate of the graph is $\alpha_0 = 0.922$.

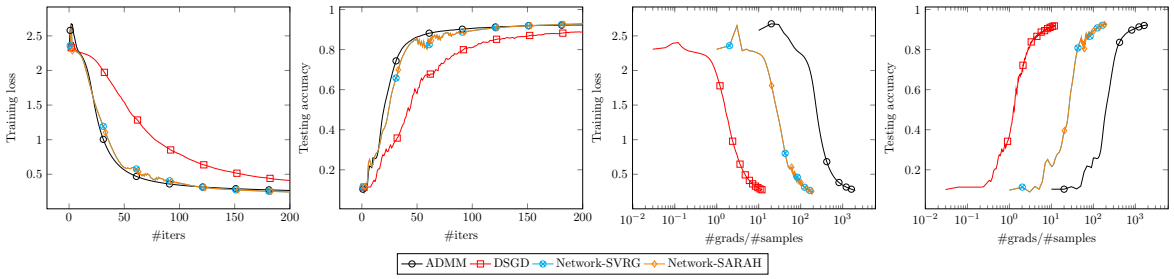

Figure 3: The training loss and testing accuracy with respect to the number of iterations (left two panels) and gradient evaluations (right two panels) for different algorithms on the MNIST dataset.

eration, over a poorly-connected network with mixing rate $\alpha_0 = 0.907$. We plot the relative optimality gap for a linear regression problem with $\kappa = 10$, with respect to the number of iterations and communication rounds for `Network-DANE` and `Network-SVRG`, under different values of K. Fig. 2 shows the relative optimality gap with respect to the number of iterations and communication rounds for `Network-DANE` and `Network-SVRG`. Due to poor connectivity, `Network-DANE` and `Network-SVRG` fail to converge when $K = 1$ using moderate parameters. However, by using a larger K, due to improvement in consensus, both algorithms converge faster in terms of the number of iterations. Notice that after a certain threshold, further increasing K will not improve the convergence rate in terms of communications. It is clear there is a trade-off between convergence speed and communication rounds, where a properly selected K will lead to an overall performance gain.

Experiments on neural network training. Though our theory only applies to the strongly convex case, we examine `Network-SVRG/SARAH` in the nonconvex case, by training a one-hidden-layer neural network with 64 hidden neurons and sigmoid activations for a classification task using the MNIST dataset. Training samples are split evenly to all agents. Fig. 3 plots the training loss and testing accuracy against the numbers of iterations and gradient evaluations for different algorithms, communicated over an an Erdös-Rènyi graph with $p = 0.3$, where centralized ADMM and decentralized stochastic algorithm (DSGD) are plotted as baselines. Being more communication-efficient than DSGD, and more computation-efficient than ADMM, `Network-SVRG/SARAH` reach a desirable balance between computation and communication efficacies.

6 Conclusions

This paper proposes decentralized (stochastic) optimization algorithms that are communication-efficient over a network: `Network-DANE` based on the approximate Newton-type update, and `Network-SVRG/SARAH` based on stochastic variance-reduced gradient updates. Theoretical convergence guarantees are developed for `Network-DANE/SVRG` for strongly convex losses, and `Network-SARAH` for quadratic losses, highlighting the impact of network topology and data homogeneity across agents. Moreover, extensive numerical experiments are conducted to verify the excellent performance of the proposed algorithms. This work opens up many exciting directions for future investigation, including but not limited to establishing the convergence for a broader family of `Network-DANE/SVRG` type algorithms under general loss functions for both convex and nonconvex settings, with the possibility of asynchronous updates across agents.

Acknowledgments

The work of B. Li, S. Cen and Y. Chi is supported in part by the grants ONR N00014-18-1-2142 and N00014-19-1-2404, ARO W911NF-18-1-0303, and NSF CAREER ECCS-1818571, CCF-1806154 and CCF-1901199. The work of Y. Chen is supported in part by the grants AFOSR YIP award FA9550-19-1-0030, ONR N00014-19-1-2120, ARO W911NF-18-1-0303, NSF CCF-1907661 and IIS-1900140.

References

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205. ACM, 2017.

Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

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.

Shicong Cen, Huishuai Zhang, Yuejie Chi, Wei Chen, and Tie-Yan Liu. Convergence of distributed stochastic variance reduced methods without sampling extra data. *arXiv preprint arXiv:1905.12648*, 2019.

Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

Jianqing Fan, Yongyi Guo, and Kaizheng Wang. Communication-efficient accurate statistical estimation. *arXiv preprint arXiv:1906.04870*, 2019.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

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

Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, pages 1–48, 2017.

Jason D Lee, Qihang Lin, Tengyu Ma, and Tianbao Yang. Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *The Journal of Machine Learning Research*, 18(1):4404–4446, 2017.

Boyue Li, Shicong Cen, Yuxin Chen, and Yuejie Chi. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. *arXiv preprint arXiv:1909.05844*, 2019a.

Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019b.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

Aryan Mokhtari and Alejandro Ribeiro. DSA: Decentralized double stochastic averaging gradient algorithm. *The Journal of Machine Learning Research*, 17(1):2165–2199, 2016.

Angelia Nedic, Asuman Ozdaglar, and Pablo A Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

Angelia Nedić, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621, 2017.

Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pages 3027–3036, 2017.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

Gesualdo Scutari and Ying Sun. Distributed nonconvex constrained optimization over time-varying digraphs. *Mathematical Programming*, 176(1-2):497–544, 2019.

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

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Virginia Smith, Simone Forte, Chenxin Ma, Martin Takáč, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:230, 2018.

Haoran Sun, Songtao Lu, and Mingyi Hong. Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach. *arXiv preprint arXiv:1910.05857*, 2019a.

Ying Sun, Amir Daneshmand, and Gesualdo Scutari. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv preprint arXiv:1905.02637*, 2019b.

César A Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. Optimal algorithms for distributed optimization. *arXiv preprint arXiv:1712.00232*, 2017.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, 2004. ISSN 01676911. doi: 10.1016/j.sysconle.2004.02.022.

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

Ran Xin, Usman A Khan, and Soummya Kar. Variance-reduced decentralized stochastic optimization with gradient tracking. *arXiv preprint arXiv:1909.11774*, 2019a.

Ran Xin, Anit Kumar Sahu, Usman A Khan, and Soummya Kar. Distributed stochastic optimization with gradient tracking over strongly-connected networks. *arXiv preprint arXiv:1903.07266*, 2019b.

Kun Yuan, Bicheng Ying, Jiageng Liu, and Ali H Sayed. Variance-reduced stochastic learning by networked agents under random reshuffling. *IEEE Transactions on Signal Processing*, 67(2):351–366, 2018.

Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.