Shi Pu, Alex Olshevsky, and Ioannis Ch. Paschalidis

Asymptotic Network Independence in Distributed Stochastic Optimization for Machine Learning

Examining distributed and centralized stochastic gradient descent

Digital Object Identifier 10.1109/MSP.2020.2975212
Date of current version: 28 April 2020

We provide a discussion of several recent results which, in certain scenarios, are able to overcome a barrier in distributed stochastic optimization for machine learning (ML). Our focus is the so-called asymptotic network independence property, which is achieved whenever a distributed method executed over a network of n nodes asymptotically converges to the optimal solution at a comparable rate to a centralized method with the same computational power as the entire network. We explain this property through an example involving the training of ML models and sketch a short mathematical analysis for comparing the performance of distributed stochastic gradient descent (DSGD) with centralized SGD.

Introduction: Distributed optimization and its limitations

First-order optimization methods, ranging from vanilla gradient descent to Nesterov acceleration and its many variants, have emerged over the past decade as the principal way to train ML models. There is a great need for techniques that train such models quickly and reliably in a distributed fashion over networks where the individual processors or GPUs may be scattered across the globe and communicate over an unreliable network, which may suffer from message losses, delays, and asynchrony (see [1], [2], [29], and [33]).

Unfortunately, what often happens is that the gains achieved from having many different processors running an optimization algorithm are squandered by the cost of coordination, shared memory, message losses, and latency. This effect is especially pronounced when there are many processors and they are spread across geographically distributed data centers. As is widely recognized by the distributed systems community, "throwing" more processors at a problem will not, after a certain point, result in better performance.

This is typically reflected in the convergence time bounds obtained for distributed optimization in the literature. The problem formulation is that one must solve

$$z^* \in \operatorname*{argmin}_{z \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(z), \tag{1}$$

over a network of n nodes (see Figure 1 for an example). Only node i has knowledge of the function $f_i(z)$, and the standard assumption is that, at every step when it is awake, node i can compute the (stochastic) gradient of its own local function $f_i(z)$. These functions $f_i(z)$ are assumed to be convex. The problem is how to compute this minimum in a distributed manner over the network based on peer-to-peer communication, possible message losses, delays, and asynchrony.

This article is concerned with the current limitations of distributed optimization and how to overcome them in certain scenarios.

This relatively simple formulation captures a large variety of learning problems. Suppose each agent i stores training data points $\mathcal{X}_i = \{(x_j, y_j)\}$, where $x_j \in \mathbb{R}^p$ are vectors of features and $y_j \in \mathbb{R}$ are the associated responses (either discrete or continuous). We are interested in learning a predictive model $h(x;\theta)$, parameterized by parameters $\theta \in \mathbb{R}^d$, so that $h(x_j;\theta) \approx y_j$ for all j. In other words, we are looking for a model that fits all of the data throughout the network. This can be accomplished by empirical risk minimization:

$$\theta^* \in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} c_i(\theta, \mathcal{X}_i), \tag{2}$$

where

$$c_i(\theta, \mathcal{X}_i) = \sum_{(x_j, y_j) \in \mathcal{X}_i} \ell(h(x_j;\theta), y_j)$$

measures how well the parameter θ fits the data at node i, with $\ell(h(x_j;\theta), y_j)$ being a loss function measuring the difference between $h(x_j;\theta)$ and y_j. Much of modern ML is built around such a formulation, including regression, classification, and regularized variants [7].

It is also possible that each agent i does not have a static data set, but instead collects streaming data points $(x_i, y_i) \sim \mathbb{P}_i$ repetitively over time, where $\mathbb{P}_i$ represents an unknown distribution of (x_i, y_i). In this case, we can find θ^* through expected risk minimization:

$$\theta^* \in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta), \tag{3}$$

where

$$f_i(\theta) = \mathbb{E}_{(x_i, y_i) \sim \mathbb{P}_i} \ell(h(x_i;\theta), y_i).$$

This article is concerned with the current limitations of distributed optimization and how to overcome them in certain scenarios. To illustrate our main concern, let us consider the distributed subgradient method in the simplest possible setting, namely, the problem of computing the median of a collection of numbers in a distributed manner over a fixed graph. Each agent i in the network holds value $m_i > 0$, and the global objective is to find the median of $m_1, m_2, \ldots, m_n$. This can be incorporated in the framework of (1) by choosing

$$f_i(z) = |z - m_i|, \quad \forall i.$$

The distributed subgradient method (see [18]) uses subgradients $s_i(z)$ of $f_i(z)$ at any point z to have agent i update as

$$z_i(k+1) = \sum_{j=1}^{n} w_{ij} z_j(k) - \alpha_k s_i(z_i(k)), \tag{4}$$

where $\alpha_k > 0$ denotes the step size at iteration k, and $w_{ij} \in [0,1]$ are the weights agent i assigns to agent j's solutions: two agents i and j are able to exchange information if $w_{ij}, w_{ji} > 0$ ($w_{ij} = w_{ji} = 0$ otherwise). The weights w_{ij} are assumed to be symmetric. For comparison, the centralized subgradient method updates the solution at iteration k according to

$$z(k+1) = z(k) - \alpha_k \frac{1}{n} \sum_{j=1}^{n} s_j(z(k)). \tag{5}$$

In Figure 2, we show the performance of algorithm (4) as a function of the network size n, assuming the agents communicate over a ring network. As can be clearly seen, when the network size grows, it takes a longer time for the algorithm to reach a certain performance threshold.

Clearly this is an undesirable property. Glancing at the figure, we see that distributing computation over 50 nodes can result in a convergence time on the order of 10^7 iterations. Few practitioners will be enthusiastic about distributed optimization if the final effect is vastly increased convergence time.

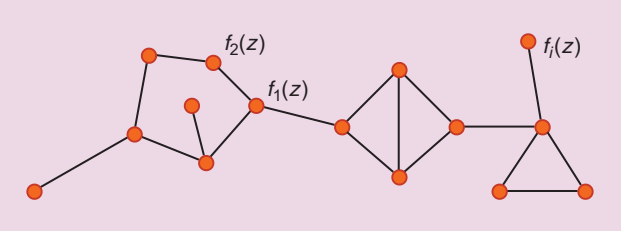

FIGURE 1. An example of a network. Two nodes are connected if there is an edge between them.

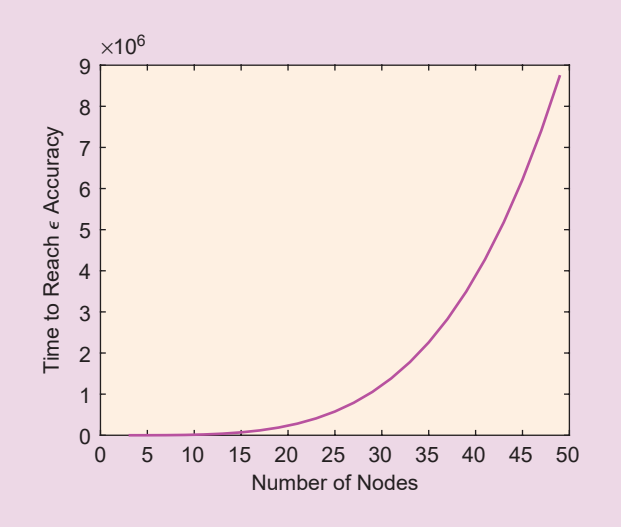

FIGURE 2. The performance of algorithm (4) as a function of the network size n. The agents communicate over a ring network [see Figure 4(b)] and choose the Metropolis weights (see the "Setup" section for the definition). Step sizes $\alpha_k = 1/\sqrt{k}$, and m_i are evenly distributed in $[-10, 10]$. The time k to reach $(1/n)\sum_{i=1}^{n} |y_i(k)| < \epsilon$ is plotted, where $y_i(k) = (1/k)\sum_{\ell=0}^{k-1} z_i(\ell)$ and $\epsilon = 0.1$.

One might hope that this phenomenon, demonstrated for the problem of median computation and considered here because it is arguably the simplest problem to which one can apply the subgradient method, will not hold for the more sophisticated optimization problems in ML literature. Unfortunately, most work in distributed optimization replicates this undesirable phenomenon. Next we give an extremely brief discussion of known convergence times in the distributed setting (for a much more extended discussion, we refer the reader to the recent survey in [17]).

This relatively simple formulation captures a large variety of learning problems.

We confine our discussion to the following point: most known convergence times in the distributed optimization literature imply bounds of the form

$$\text{Time}_{n,\epsilon}(\text{decentralized}) \leq p(\mathcal{G})\,\text{Time}_{n,\epsilon}(\text{centralized}), \quad (6)$$

where $\text{Time}_{n,\epsilon}(\text{decentralized})$ denotes the time for the decentralized algorithm on n nodes to reach ϵ accuracy (error $< \epsilon$), and $\text{Time}_{n,\epsilon}(\text{centralized})$ is the time for the centralized algorithm, which can query n gradients per time step to reach the same level of accuracy. The graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ consists of the set of nodes and edges in the network, denoted by $\mathcal{N}$ and $\mathcal{E}$, respectively. The function $p(\mathcal{G})$ can usually be bounded in terms of some polynomial in the number of nodes n.

For instance, in the subgradient methods, [17, Corollary 9] implies that

$$\text{Time}_{n,\epsilon}(\text{decentralized}) = \mathcal{O}\left(\frac{\max\left\{\left\|\frac{1}{n}\sum_{i=1}^{n} z_i(0) - z^*\right\|^2, G^4 h(\mathcal{G})\right\}}{\epsilon^2}\right),$$

$$\text{Time}_{n,\epsilon}(\text{centralized}) = \mathcal{O}\left(\frac{\max\left\{\left\|z(0) - z^*\right\|^2, G^4\right\}}{\epsilon^2}\right),$$

where $z(0), z_i(0)$ are initial estimates, z^* denotes the optimal solution, and G bounds the ℓ_2-norm of the subgradients. Function $h(\mathcal{G})$ is the inverse of the spectral gap corresponding to the graph and will typically grow with n; hence, when n is large, $p(\mathcal{G}) \simeq h(\mathcal{G})$. In particular, if the communication graphs are 1) path graphs, then $p(\mathcal{G}) = \mathcal{O}(n^2)$; 2) star graphs, then $p(\mathcal{G}) = \mathcal{O}(n^2)$; or 3) geometric random graphs, then $p(\mathcal{G}) = \mathcal{O}(n \log n)$. The method developed in [20] achieves $p(\mathcal{G}) = n$ but, typically, $p(\mathcal{G})$ is at least n^2.

By comparing $\text{Time}_{n,\epsilon}(\text{decentralized})$ and $\text{Time}_{n,\epsilon}(\text{centralized})$, we are keeping the computational power the same in both cases. Naturally, centralized is always better: anything that can be done in a decentralized way could be done in a centralized way. The question though, is: How much better?

Framed in this way, the polynomial scaling in the quantity $p(\mathcal{G})$ is extremely disconcerting. It is, for example, hard to argue that an algorithm should be run in a distributed manner with say, $n = 100$, if the quantity $p(\mathcal{G})$ in (6) satisfies $p(\mathcal{G}) = n^2$; that would imply that the distributed variant would be 10,000-times slower than the centralized one with the same computational power.

Sometimes $p(\mathcal{G})$ is written as the inverse spectral gap $1/(1-\lambda_2)$ in terms of the second eigenvalue of some matrix. Because the second-smallest eigenvalue of an undirected graph Laplacian is approximately $\sim 1/n^2$ away from zero, such bounds will translate into at least quadratic scalings with n in the worst case. Over time-varying B-connected graphs, the best-known bounds on $p(\mathcal{G})$ will be cubic in n using the results in [16].

There are a number of caveats to the pessimistic argument outlined previously in this section. For example, in a multiagent scenario where data sharing is not desirable or feasible, decentralized computation might be the only option available. Generally speaking, however, fast-growing $p(\mathcal{G})$ will preclude the widespread applicability of distributed optimization. Indeed, returning to the back-of-the-envelope calculation mentioned previously, if a user has to pay a multiplicative factor of 10,000 in convergence speed to use an algorithm, the most likely scenario is that the algorithm will not be used.

There are some scenarios that avoid the pessimistic discussion mentioned previously: for example, when the underlying graph is an expander, the associated spectral gap is constant (see [8, Ch. 6] for a definition of these terms as well as an explanation), and likewise when the graph is a star graph. In particular, on a random Erdős–Rényi random graph, the quantity $p(\mathcal{G})$ is constant with high probability [17, Corollary 9, Part 9] . Unfortunately, these are very special cases and may not always be realistic. A star graph requires a single node to have the ability to receive and broadcast messages to all other nodes in the system. On the other hand, an expander graph may not occur in geographically distributed systems. By way of comparison, a random graph where nodes are associated with random locations, with links between nodes close together, will not have constant spectral gap and will thus have $p(\mathcal{G})$, which grows with n [17, Corollary 9, Part 10]. The Erdős–Rényi graph escapes this because, if we again associate nodes with locations, the average link in such a graph is a "long-range" one, connecting nodes that are geographically far apart. It is a consequence of Cheeger's inequality, that graphs based on connecting nearest neighbors (i.e., where nodes are regularly spaced in $\mathbb{R}^d$ and each node is connected to a constant number of closest neighbors) will not have constant spectral gap.

Asymptotic network independence in distributed stochastic optimization

In this article, we provide a discussion of several recent papers which have obtained that, for a number of settings involving distributed stochastic optimization, $p(\mathcal{G}) = 1$ as long as k is large enough. In other words, asymptotically, the distributed stochastic gradient algorithm converges to the optimal solution at a comparable rate to a centralized algorithm with the same computational power.

We call this property *asymptotic network independence*: it is as if the network is not even there. Asymptotic network

independence provides an answer to the concerns raised in the previous section.

We begin by illustrating these results with a simulation from [21], shown in Figure 3. Here the problem to be solved is classification using a smooth support vector machine (SVM) between overlapping clusters of points. The performance of the centralized algorithm is shown in orange, and the performance of the decentralized algorithm is shown in dark blue. The graph is a ring of 50 nodes, and the problem being solved is the search for a support vector classifier. The graph illustrates the main result, which is that a network of 50 nodes performs as well in the limit as does a centralized method with 50 times the computational power of one node. Indeed, after ~8,000 iterations, the orange and dark blue lines are nearly indistinguishable.

We call this property *asymptotic network independence*: it is as if the network is not even there.

We note that similar simulations are available for other ML methods (training neural networks, logistic regression, elastic net regression, and so on). The asymptotic network independence property enables us to efficiently distribute the training process for a variety of existing learning methods.

The name *asymptotic network independence* is a slight misnomer, as we actually do not care whether the asymptotic performance depends in some complicated way on the network. All we want is for the decentralized convergence rate to be bounded by $O(1)$ times the convergence rate of the centralized method.

The authors in [4]–[6] and [31] gave the first crisp statement of the relationship between centralized and distributed methods in the setting of distributed optimization of smooth, strongly convex functions in the presence of noise. Under constant step sizes, the authors in [4]–[6] were the first to show that, when the step size is sufficiently small, a distributed stochastic gradient method achieves a performance comparable to that of the centralized method in terms of the steady-state mean-square error. The step size has to be small enough as a function of the network topology for this to hold. In [31], the authors showed that the distributed stochastic gradient algorithm asymptotically achieves a convergence rate comparable to that of the centralized method, but assuming that all of the local functions f_i have the same minimum. This gives the first "asymptotic network independence" result.

The work in [22] approximated distributed stochastic gradient algorithm by stochastic differential equations in continuous time by assuming a sufficiently small constant step size. It was shown that the distributed method outperforms a centralized scheme with synchronization overhead; however, it did not lead to straightforward algorithmic bounds. In our recent work [21], we generalized the results to graphs that are time varying, with delays, message losses, and asynchrony. In a parallel recent work [9], a similar result was demonstrated using a further compression technique, which allowed nodes to save on communication.

When the objective functions are not assumed to be convex, several recent works have obtained asymptotic network independence for distributed stochastic gradient methods. In [13] and [14], a general stochastic approximation setting was considered with decaying step sizes, and the convergence rates of centralized and distributed methods were shown to be asymptotically the same; the proof proceeded based on certain technical properties of stochastic approximation methods. The work in [12] was the first to show that distributed algorithms could achieve a speedup like that of a centralized method when the number of computing steps is large enough. Such a result was generalized to the setting of the directed communication networks in [1] for training deep neural networks, where the push-sum technique was combined with the standard distributed stochastic gradient scheme.

We remark that in this survey, all of the previously mentioned algorithms that enjoy the asymptotic network independence property assume smooth objective functions, i.e., functions with Lipschitz continuous gradients.

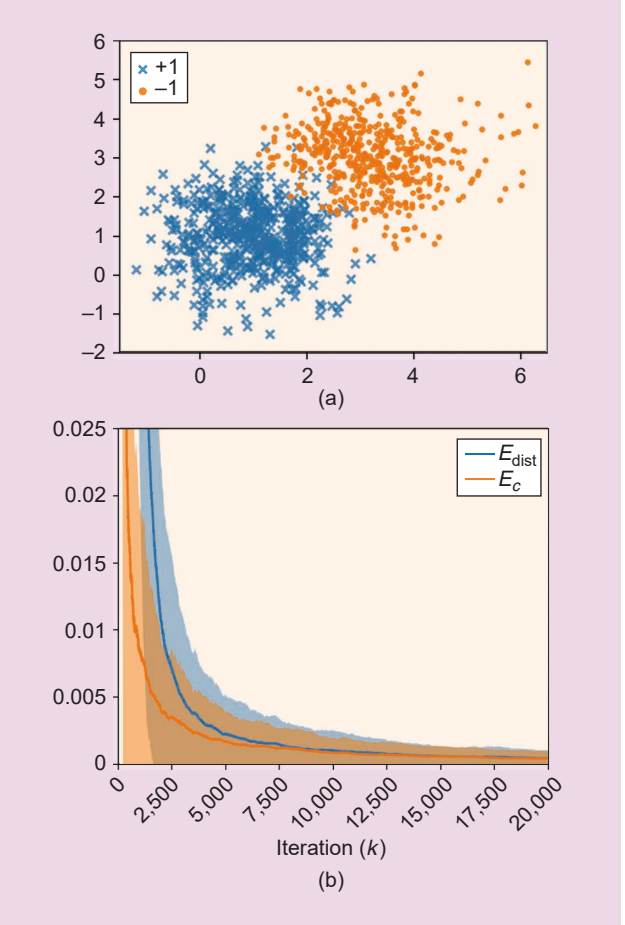

FIGURE 3. A comparison of DSGD and centralized SGD for training an SVM. (a) A total of 1,000 data points and their labels for SVM classification. The data points are randomly generated around 50 cluster centers. (b) The squared errors and one standard-deviation band for DSGD and centralized SGD. The performance of the centralized algorithm is shown in orange, and the performance of the decentralized algorithm is shown in dark blue. A total of 1,000 Monte Carlo simulations are conducted for estimating the average performance.

In the next sections, we provide a simple and readable explanation of the asymptotic network independence phenomenon in the context of distributed stochastic optimization over smooth and strongly convex objective functions. For more information on the topic of distributed stochastic optimization, the reader is referred to [10], [15], [23], [24], [28], [30], and [32] and the references therein.

Setup

We are interested in minimizing (1) over a network of n communicating agents. Regarding the objective functions f_i, we make the following standing assumption.

Assumption 1

Each $f_i: \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex with L-Lipschitz continuous gradients, i.e., for any $z, z' \in \mathbb{R}^d$,

$$\begin{aligned} &\langle \nabla f_i(z) - \nabla f_i(z'), z - z' \rangle \geq \mu \| z - z' \|^2, \\ &\| \nabla f_i(z) - \nabla f_i(z') \| \leq L \| z - z' \|. \end{aligned} \tag{7}$$

Under "Assumption 1," (1) has a unique optimal solution, z^*, and the function $f(z)$ defined as $f(z) = (1/n)\sum_{i=1}^n f_i(z)$ has the following contraction property presented in the next section [26, Lemma 10].

Lemma 1

For any $z \in \mathbb{R}^d$ and $\alpha \in (0, 1/L)$, we have $\| z - \alpha \nabla f(z) - z^* \| \leq (1 - \alpha\mu) \| z - z^* \|$.

In other words, gradient descent with a small step size reduces the distance between the current solution and z^*.

In the stochastic optimization setting, we assume that at each iteration k of the algorithm, $z_i(k)$ being the input for agent i, each agent is able to obtain noisy gradient estimates $g_i(z_i(k), \xi_i(k))$, which satisfy the following condition.

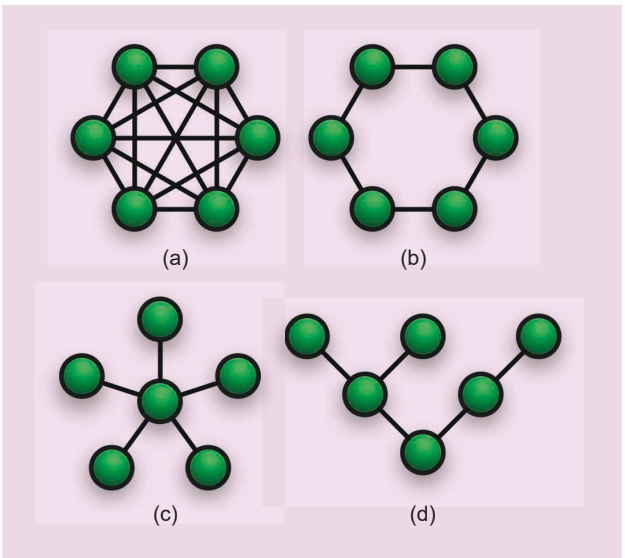

FIGURE 4. Examples of undirected connected graphs. (a) A fully connected graph and (b) ring, (c) star, and (d) tree networks.

Assumption 2

For all $i \in \{1, 2, \ldots, n\}$ and $k \geq 1$, each random vector $\xi_i(k) \in \mathbb{R}^m$ is independent, and

$$\begin{aligned} &\mathbb{E}_{\xi_{i,k}}[g_i(z_i(k), \xi_i(k)) | z_i(k)] = \nabla f_i(z_i(k)), \\ &\mathbb{E}_{\xi_{i,k}}\left[\| g_i(z_i(k), \xi_i(k)) - \nabla f_i(z_i(k)) \|^2 | z_i(k) \right] \leq \sigma^2, \text{ for some } \sigma > 0. \end{aligned} \tag{8}$$

Stochastic gradients appear, for instance, when the gradient estimation of $c_i(\theta, \mathcal{X}_i)$ in empirical risk minimization (2) introduces noise from various sources, such as sampling and quantization errors. For another example, when minimizing the expected risk in (3), where independent data points (x_i, y_i) are gathered over time, $g_i(z, (x_i, y_i)) = \nabla_z \ell(h(x_i; z), y_i)$ is a stochastic, unbiased estimator of $\nabla f_i(z)$, satisfying the first condition in (8). The second condition holds for popular problems such as smooth SVMs, logistic regression, and softmax regression, assuming the domain of (x_i, y_i) is bounded.

The algorithm we discuss is the DSGD method adapted from distributed gradient descent and the diffusion strategy [3]; note that in [3] this method was called *adapt-then-combine*. We let each agent i in the network hold a local copy of the decision vector denoted by $z_i \in \mathbb{R}^d$, and its value at iteration/time k is written as $z_i(k)$. Denote $g_i(k) = g_i(z_i(k), \xi_i(k))$ for short. At each step $k \geq 0$, every agent i performs the following update:

$$z_i(k+1) = \sum_{j=1}^n w_{ij} (z_j(k) - \alpha_k g_j(k)), \tag{9}$$

where $\{\alpha_k\}$ is a sequence of nonnegative nonincreasing step sizes. The initial vectors $z_i(0)$ are arbitrary for all i, and $\mathbf{W} = [w_{ij}]$ is a mixing matrix.

DSGD belongs to the class of so-called consensus-based distributed optimization methods, where different agents mix their estimates at each iteration to reach a consensus of the solutions, i.e., $z_i(k) \approx z_j(k)$, for all i and j in the long run. To achieve consensus, the following condition is assumed on the mixing matrix and the communication topology among agents.

Assumption 3

The graph $\mathcal{G}$ of agents is undirected and connected (there exists a path between any two agents). The mixing matrix $\mathbf{W}$ is nonnegative, symmetric, and doubly stochastic, i.e., $\mathbf{W1} = \mathbf{1}$ and $\mathbf{1}^\mathsf{T}\mathbf{W} = \mathbf{1}^\mathsf{T}$, where $\mathbf{1}$ is the all-one vector. In addition, $w_{ii} > 0$ for some $i \in \{1, 2, \ldots, n\}$.

Some examples of undirected connected graphs are presented in Figure 4. Because of "Assumption 3," the mixing matrix $\mathbf{W}$ has an important contraction property.

Lemma 2

Let "Assumption 3" hold, and let $1 = \lambda_1 \geq \lambda_2 \geq \cdots \lambda_n$ denote the eigenvalues of the matrix $\mathbf{W}$. Then, $\lambda = \max(|\lambda_2|, |\lambda_n|) < 1$ and

$$\| \mathbf{W}\boldsymbol{\omega} - \mathbf{1}\bar{\omega} \| \leq \lambda \| \boldsymbol{\omega} - \mathbf{1}\bar{\omega} \|$$

for all $\boldsymbol{\omega} \in \mathbb{R}^{n\times d}$, where $\bar{\omega} = (1/n)\mathbf{1}^{\mathsf{T}}\boldsymbol{\omega}$. As a result, when running a consensus algorithm (which is just (9) without gradient descent)

$$z_i(k+1) = \sum_{j=1}^{n} w_{ij} z_j(k), \tag{10}$$

the speed of reaching consensus is determined by $\lambda = \max(|\lambda_2|, |\lambda_n|)$. In particular, if we adopt the so-called lazy Metropolis rule for defining the weights, the dependency of λ on the network size n is upper bounded by $1 - c/n^2$ for some constant c [20] (See "Lazy Metropolis Rule for Constructing **W**.").

Despite the fact that λ may be very close to 1 with large n, the consensus algorithm (10) enjoys geometric convergence speed, i.e.,

$$\sum_{i=1}^{n} \left\| z_i(k) - \frac{1}{n}\sum_{j=1}^{n} z_j(k) \right\|^2 \le \lambda^k \sum_{i=1}^{n} \left\| z_i(0) - \frac{1}{n}\sum_{j=1}^{n} z_j(0) \right\|^2.$$

By contrast, the optimal rate of convergence for any stochastic gradient methods is sublinear, asymptotically $O(1/k)$ (see [19]). This difference suggests that a consensus based distributed algorithm for stochastic optimization may match the centralized methods in the long term: any errors due to consensus will decay at a fast-enough rate so that they ultimately do not matter.

In the next sections, we discuss and compare the performance of the centralized SGD method and DSGD. We show that both methods asymptotically converge at the rate $\sigma^2/(n\mu^2 k)$. Furthermore, the time needed for DSGD to approach the asymptotic convergence rate turns out to scale as $O(n/(1-\lambda)^2)$.

Centralized SGD

The benchmark for evaluating the performance of DSGD is the centralized SGD method, which we describe in this section. At each iteration k, the following update is executed:

$$z(k+1) = z(k) - \alpha_k \bar{g}(k), \tag{11}$$

where step sizes satisfy $\alpha_k = 1/(\mu k)$ and $\bar{g}(k) = (1/n)\Sigma_{i=1}^{n} g_i(z(k), \xi_i(k))$, i.e., $\bar{g}(k)$ is the average of n noisy gradients evaluated at $z(k)$ (by utilizing n gradients at each iteration, we are keeping the computational power the same for SGD and DSGD). As a result, the gradient estimation is more accurate than using just one gradient. Indeed, from "Assumption 2" we have

$$\mathbb{E}\left[\left\| \bar{g}(k) - \nabla f(z(k)) \right\|^2\right] = \frac{1}{n^2}\sum_{i=1}^{n} \mathbb{E}\left[\left\| g_i(z(k), \xi_i(k)) - \nabla f_i(z(k)) \right\|^2\right] \le \frac{\sigma^2}{n}. \tag{12}$$

We measure the performance of SGD by $R(k) = \mathbb{E}\left[\left\| z(k) - z^* \right\|^2\right]$, the expected squared distance between the solution at time k and the optimal solution. "Theorem 1" characterizes the convergence rate of $R(k)$, which is optimal for such stochastic gradient methods (see [19] and [27]).

Lazy Metropolis Rule for Constructing W

$$w_{ij} = \begin{cases} \dfrac{1}{2\max\{\deg(i), \deg(j)\}}, & \text{if } i \in \mathcal{N}_i, \\ 1 - \displaystyle\sum_{j\in\mathcal{N}_i} w_{ij}, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Notation: $\deg(i)$ denotes the degree (number of "neighbors") of node i. Correspondingly, $\mathcal{N}_i$ is the set of "neighbors" for agent i.

Theorem 1

Under SGD (11), supposing "Assumption 1," "Assumption 2," and "Assumption 3" hold, we have

$$R(k) \le \frac{\sigma^2}{n\mu^2 k} + O_k\left(\frac{1}{k^2}\right). \tag{13}$$

To compare with the analysis for DSGD later, we briefly describe how to obtain (13). Note that

$$\begin{aligned} R(k+1) &= \mathbb{E}\left[\left\| z(k) - \alpha_k \bar{g}(k) - z^* \right\|^2\right] \\ &= \mathbb{E}\left[\left\| z(k) - \alpha_k \nabla f(z(k)) - z^* \right\|^2\right] \\ &\quad + \alpha_k^2 \mathbb{E}\left[\left\| \nabla f(z(k)) - \bar{g}(k) \right\|^2\right]. \end{aligned}$$

For large k, in light of "Lemma 1" and relation (12), we have the following inequality that relates $R(k+1)$ to $R(k)$:

$$R(k+1) \le (1-\alpha_k\mu)^2 R(k) + \frac{\alpha_k^2\sigma^2}{n} = \left(1 - \frac{1}{k}\right)^2 R(k) + \frac{\sigma^2}{n\mu^2}\frac{1}{k^2}. \tag{14}$$

A simple induction then gives (13).

DSGD

We assume the same step-size policy for DSGD and SGD. To analyze DSGD starting from (9), define

$$\bar{z}(k) = \frac{1}{n}\sum_{i=1}^{n} z_i(k), \tag{15}$$

as the average of all of the iterates in the network. Different from the analysis for SGD, we are concerned with two error terms. The first term $\mathbb{E}\left[\left\| \bar{z}(k) - z^* \right\|^2\right]$, called the *expected optimization error*, defines the expected squared distance between $\bar{z}(k)$ and z^*. The second term $\Sigma_{i=1}^{n}\mathbb{E}\left[\left\| z_i(k) - \bar{z}(k) \right\|^2\right]$, called the *expected consensus error*, measures the dissimilarities of individual estimates among all the agents. The average squared distance between individual iterate $z_i(k)$ and the optimum z^* is given by

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|z_i(k)-z^*\right\|^2\right]=\mathbb{E}\left[\left\|\bar{z}(k)-z^*\right\|^2\right]+\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|z_i(k)-\bar{z}(k)\right\|^2\right]. \quad (16)$$

Exploring the two terms will provide us with insights into the performance of DSGD. To simplify notation, denote $U(k)=\mathbb{E}\left[\left\|\bar{z}(k)-z^*\right\|^2\right]$, $V(k)=\Sigma_{i=1}^{n}\mathbb{E}\left[\left\|z_i(k)-\bar{z}(k)\right\|^2\right]$, $\forall k$.

Inspired by the analysis for SGD, we first look for an inequality that bounds $U(k)$, which is analogous to $\mathbb{E}\left[\left\|z(k)-z^*\right\|^2\right]$ in SGD. One such relation turns out to be [25]:

$$U(k+1)\le\left(1-\frac{1}{k}\right)^2U(k)+\frac{2L}{\sqrt{n}\,\mu}\frac{\sqrt{U(k)V(k)}}{k}+\frac{L^2}{n\mu^2}\frac{V(k)}{k^2}+\frac{\sigma^2}{n\mu^2}\frac{1}{k^2}. \quad (17)$$

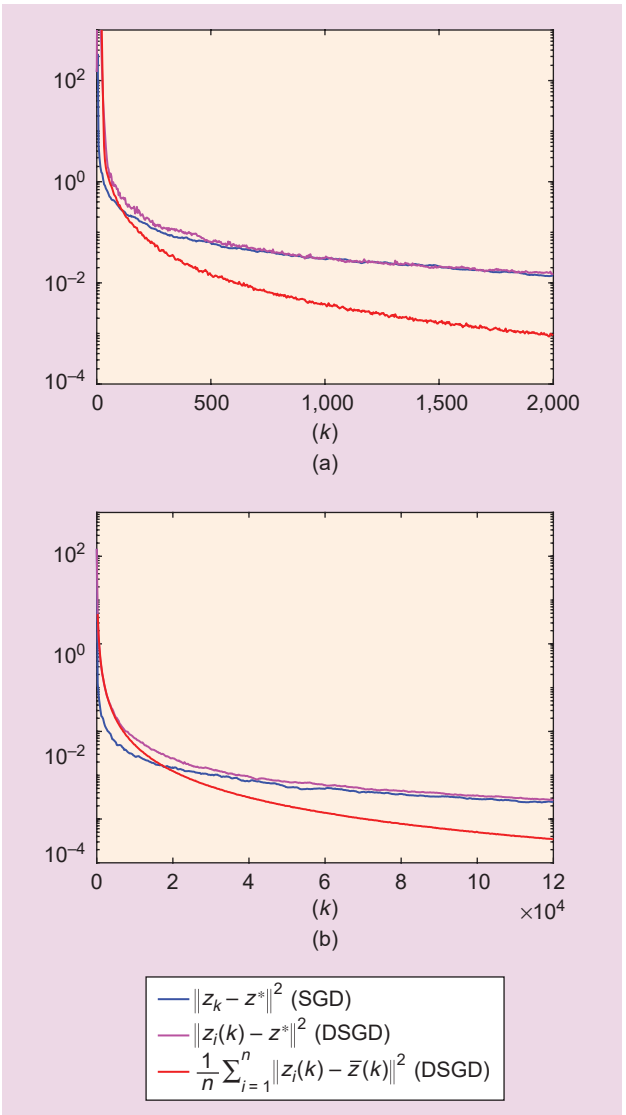

FIGURE 5. The performance comparison between DSGD and SGD for online Ridge regression. For DSGD, the plots show the iterates generated by a randomly selected node i from the set $\{1, 2, \ldots, n\}$. The results are averaged over 100 Monte Carlo simulations. (a) Instance 1 (a random network used for DSGD) and (b) instance 2 (a grid network used for DSGD).

Comparing (17) to (14), we find two additional terms on the right-hand side of the inequality. Both terms involve the expected consensus error $V(k)$, thus reflecting the additional disturbances caused by the dissimilarities of solutions. Equation (17) also suggests that the convergence rate of $U(k)$ cannot be better than $R(k)$ for SGD, which is expected. Nevertheless, if $V(k)$ decays fast enough compared to $U(k)$, it is likely that the two additional terms are negligible in the long run, and we deduce that the convergence rate of $U(k)$ is comparable to that of $R(k)$ for SGD.

This indeed turns out to be the case, as shown in [25], that $V(k)\le O(n/(1-\lambda)^2)(1/k^2)$ for $k\ge O(1/(1-\lambda))$. Plugging this into (17) leads to the inequality $U(k)\le\theta^2\sigma^2/((1.5\theta-1)n\mu^2k)+O(1/(1-\lambda)^2)(1/k^2)$. Therefore, when $k\ge O(n/(1-\lambda)^2)$, we have

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|z_i(k)-z^*\right\|^2\right]\le\frac{\sigma^2}{n\mu^2k}O(1).$$

In other words, we have the asymptotic network independence phenomenon: after a transient, DSGD performs comparably to a centralized SGD method with the same computational power (e.g., which can query the same number of gradients per step as that of the entire network).

Numerical illustration

We provide a numerical example to illustrate the asymptotic network independence property of DSGD. Consider the online Ridge regression problem

$$z^*=\operatorname{argmin}_{z\in\mathbb{R}^d}\sum_{i=1}^{n}f_i(z)\left(=\mathbb{E}_{u_i,v_i}\left[(u_i^{\mathsf{T}}z-v_i)^2+\rho\left\|z\right\|^2\right]\right), \quad (18)$$

where $\rho>0$ is a penalty parameter. Each agent i collects data points in the form of (u_i, v_i) continuously over time, with $u_i\in\mathbb{R}^d$ representing the features and $v_i\in\mathbb{R}$ being the observed outputs. Suppose each $u_i\in[-1,1]^d$ is uniformly distributed and v_i is drawn according to $v_i=u_i^{\mathsf{T}}\tilde{z}_i+\varepsilon_i$, where $\tilde{z}_i$ are predefined parameters uniformly situated in $[0,10]^d$ and ε_i are independent Gaussian random variables with mean 0 and variance 1. Given a pair (u_i, v_i), agent i can compute an estimated gradient of $f_i(z)$: $g_i(z,u_i,v_i)=2(u_i^{\mathsf{T}}z-v_i)u_i+2\rho z$, which is unbiased. Equation (18) has a unique solution z^* given by

$$z^*=\left(\sum_{i=1}^{n}\mathbb{E}_{u_i}[u_iu_i^{\mathsf{T}}]+n\rho\mathbf{I}\right)^{-1}\sum_{i=1}^{n}\mathbb{E}_{u_i}[u_iu_i^{\mathsf{T}}]\tilde{z}_i.$$

In the experiments, we consider two instances. In the first instance, we assume $n=50$ agents constitute a random network for DSGD, where every two agents are linked with probability 0.2. In the second instance, we let $n=49$ agents form a 7×7 grid network. We use Metropolis weights in both instances. The problem dimension is set to $d=10$ and $z_i(0)=\mathbf{0}$, the zero vector for all i. The penalty parameter is set to $\rho=0.1$ and the step sizes $\alpha_k=(5/k)$. For both SGD and DSGD, we run the simulations 100 times and average the results to approximate the expected errors.

The performance of SGD and DSGD is shown in Figure 5. We notice that in both instances the expected consensus error for DSGD converges to 0 faster than the expected optimization error, as predicted from our previous discussion. Regarding the expected optimization error, DSGD is slower than SGD in the first ~800 (respectively, $\sim 4 \times 10^4$) iterations for random network (respectively, the grid network). But after that, their performance is almost indistinguishable. The difference in the transient times is due to the stronger connectivity (or smaller λ) of the random network compared to that of the grid network.

Few practitioners will be enthusiastic about distributed optimization if the final effect is vastly increased convergence time.

Conclusions

In this article, we provided a discussion of recent results that have overcome a barrier in distributed stochastic optimization methods for ML under certain scenarios. These results established an asymptotic network independence property, that is, asymptotically, the distributed algorithm achieves a convergence rate comparable to that of a centralized algorithm with the same computational power. We explained the property using examples of training ML models and provided a short mathematical analysis.

Along the line of achieving asymptotic network independence in distributed optimization, there are various future research directions, including considering nonconvex objective functions, reducing communication costs and transient time, and using exact gradient information. In this section, we briefly describe these directions.

First, the distributed training of deep neural networks, the state-of-the-art ML approach in many application areas, involves minimizing nonconvex objective functions, which are different from the main objectives considered in this article. This area is largely unexplored with a few recent works in [1], [12], [14] and [29].

In distributed algorithms, the costs associated with communication among the agents are often nonnegligible and may become the main burden for large networks. It is therefore important to explore communication-reduction techniques that do not sacrifice the asymptotic network independence property. Recent works [1], [9] touched on this point.

When considering asymptotic network independence for distributed optimization, an important factor is the transient time needed to reach the asymptotic convergence rate, as it may take a long time before the distributed implementation catches up with the corresponding centralized method. In fact, as we have shown in the "Setup" section, this transient time can be a function of the network topology and grows with the network size. Reducing the transient time is thus a key future objective.

Finally, although several recent works have established the asymptotic network independence property in distributed optimization, they are mainly constrained to using stochastic gradient information. If the exact gradient is available, can distributed methods compete with the centralized ones? As we know, centralized algorithms typically enjoy a faster convergence speed with exact gradients. For example, plain gradient descent achieves linear convergence for strongly convex and smooth objective functions. To the best of the authors' knowledge, as of the writing of this article, with the exception of [11] and [29], the results on asymptotic network independence in this setting are currently lacking.

Acknowledgments

We would like to thank Artin Spiridonoff from Boston University for his kind help in providing Figure 3. The research was partially supported by the NSF under grants ECCS 1933027, IIS 1914792, DMS 1664644, and CNS 1645681, the U.S. Office of Naval Research under grant N00014-19-1-2571, the National Institutes of Health under grant 1R01GM135930, and the Shenzhen Research Institute of Big Data Startup Fund JCYJ-SP2019090001.

Authors

Shi Pu (pushi@cuhk.edu.cn) received his B.S. degree in engineering mechanics from Peking University, Beijing, China, and his Ph.D. degree in systems engineering from the University of Virginia, Charlottesville, in 2012 and 2016, respectively. He is currently an assistant professor with the Institute for Data and Decision Analytics, The Chinese University of Hong Kong, Shenzhen, China. He was a postdoctoral associate at the University of Florida, Gainesville, from 2016 to 2017, a postdoctoral scholar at Arizona State University, Tempe, from 2017 to 2018, and a postdoctoral associate at Boston University, Massachusetts, from 2018 to 2019. His research interests include distributed optimization, network science, machine learning, and game theory.

Alex Olshevsky (alexols@bu.edu) received his B.S. degree in applied mathematics and electrical engineering from the Georgia Institute of Technology, Atlanta, and his Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge. He is currently an associate professor in the Department of Electrical and Computer Engineering at Boston University, Massachusetts. He is a recipient of the National Science Foundation CAREER Award, the Air Force Office of Scientific Research Young Investigator Award, the INFORMS Prize for the best paper on the interface of operations research and computer science, a SIAM Award for best annual paper from *SIAM Journal on Control and Optimization* chosen to be reprinted in *SIAM Review*, and an IMIA Award for best paper on clinical informatics.

Ioannis Ch. Paschalidis (yannisp@bu.edu) received his Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a professor and data science

fellow at Boston University, Massachusetts, and the director of the Center for Information and Systems Engineering. He is a recipient of the National Science Foundation CAREER Award and several best paper awards. From 2013 to 2019, he was the founding editor-in-chief of *IEEE Transactions on Control of Network Systems*. His research interests lie in the fields of systems and control, networks, applied probability, optimization, operations research, computational biology, and medical informatics. He is a Fellow of the IEEE.

References

[1] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proc. Int. Conf. Machine Learning*, 2019, pp. 344–353.

[2] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Inform.*, vol. 112, pp. 59–67, Apr. 2018. doi: 10.1016/j.ijmedinf.2018.01.007.

[3] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012. doi: 10.1109/TSP.2012.2198470.

[4] J. Chen and A. H. Sayed, "On the limiting behavior of distributed optimization strategies," in *Proc. 50th IEEE Annu. Allerton Conf. Communication, Control, and Computing (Allerton)*, 2012, pp. 1535–1542. doi: 10.1109/Allerton.2012.6483402.

[5] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks–Part I: Transient analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3487–3517, 2015. doi: 10.1109/TIT.2015.2427360.

[6] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks–Part II: Performance analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3518–3548, 2015. doi: 10.1109/TIT.2015.2427352.

[7] R. Chen and I. C. Paschalidis, "A robust learning approach for regression models based on distributionally robust optimization," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 517–564, 2018.

[8] R. Durrett, Ed., *Random Graph Dynamics*, vol. 200. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[9] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. Int. Conf. Machine Learning,* pp. 3478–3487, 2019.

[10] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *Math. Program.*, vol. 180, nos. 1–2, pp. 1–48, 2017. doi: 10.1007/s10107-018-1355-4.

[11] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4494–4506, 2019. doi: 10.1109/TSP.2019.2926022.

[12] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5336–5346.

[13] G. Morral, P. Bianchi, and G. Fort, "Success and failure of adaptation-diffusion algorithms for consensus in multi-agent networks," in *Proc. 53rd IEEE Conf. Decision and Control*, 2014, pp. 1476–1481. doi: 10.1109/CDC.2014.7039609.

[14] G. Morral, P. Bianchi, and G. Fort, "Success and failure of adaptation-diffusion algorithms with decaying step size in multiagent networks," *IEEE Trans. Signal Process.*, vol. 65, no. 11, pp. 2798–2813, 2017. doi: 10.1109/TSP.2017.2666771.

[15] A. Nedić and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3936–3947, 2016. doi: 10.1109/TAC.2016.2529285.

[16] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2506–2517, 2009. doi: 10.1109/TAC.2009.2031203.

[17] A. Nedić, A. Olshevsky, and M. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proc. IEEE*, vol. 106, no. 5, pp. 953–976, 2018. doi: 10.1109/JPROC.2018.2817461.

[18] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009. doi: 10.1109/TAC.2008.2009515.

[19] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. Optim.*, vol. 19, no. 4, pp. 1574–1609, 2009. doi: 10.1137/070704277.

[20] A. Olshevsky, "Linear time average consensus and distributed optimization on fixed graphs," *SIAM J. Control Optim.*, vol. 55, no. 6, pp. 3990–4014, 2017. doi: 10.1137/16M1076629.

[21] A. Olshevsky, I. C. Paschalidis, and A. Spiridonoff, Robust asynchronous stochastic gradient-push: Asymptotically optimal and network-independent performance for strongly convex functions. 2018. [Online]. Available: arXiv: 1811.03982

[22] S. Pu and A. Garcia, "A flocking-based approach for distributed stochastic optimization," *Oper. Res.*, vol. 66, no. 1, pp. 267–281, 2017. doi: 10.1287/opre.2017.1666.

[23] S. Pu and A. Garcia, "Swarming for faster convergence in stochastic optimization," *SIAM J. Control Optim.*, vol. 56, no. 4, pp. 2997–3020, 2018. doi: 10.1137/17M1111085.

[24] S. Pu and A. Nedić, Distributed stochastic gradient tracking methods. 2018. [Online]. Available: arXiv:1805.11454

[25] S. Pu, A. Olshevsky, and I. C. Paschalidis, A sharp estimate on the transient time of distributed stochastic gradient descent. 2019. [Online]. Available: arXiv:1906.02702

[26] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1245–1260, 2018. doi: 10.1109/TCNS.2017.2698261.

[27] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proc. 29th Int. Conf. Machine Learning*, 2012, pp. 1571–1578.

[28] M. O. Sayin, N. D. Vanli, S. S. Kozat, and T. Basar, "Stochastic subgradient algorithms for strongly convex optimization over distributed networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 4, no. 4, pp. 248–260, 2017. doi: 10.1109/TNSE.2017.2713396.

[29] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal convergence rates for convex distributed optimization in networks," *J. Mach. Learn. Res.*, vol. 20, no. 159, pp. 1–31, 2019.

[30] B. Sirb and X. Ye, "Decentralized consensus algorithm with delayed and stochastic gradients," *SIAM J. Optim.*, vol. 28, no. 2, pp. 1232–1254, 2018. doi: 10.1137/16M1081257.

[31] Z. J. Towfic, J. Chen, and A. H. Sayed, "Excess-risk of distributed stochastic learners," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5753–5785, 2016. doi: 10.1109/TIT.2016.2593769.

[32] R. Xin, U. A. Khan, and S. Kar, Variance-reduced decentralized stochastic optimization with gradient tracking. 2019. [Online]. Available: arXiv:1909.11774

[33] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 977–992, 2018. doi: 10.1109/TSP.2018.2881661.

SP