EFFICIENT VARIANCE-REDUCED LEARNING OVER MULTI-AGENT NETWORKS

Kun Yuan[⋆] *Bicheng Ying*[⋆] *Ali H. Sayed*[†]

[⋆]Department of Electrical and Computer Engineering, University of California, Los Angeles
[†]School of Engineering, Ecole Polytechnique Federale de Lausanne, Switzerland

ABSTRACT

This work develops a fully decentralized variance-reduced learning algorithm for multi-agent networks where nodes store and process the data locally and are only allowed to communicate with their immediate neighbors. In the proposed algorithm, there is no need for a central or master unit while the objective is to enable the dispersed nodes to learn the *exact* global model despite their limited localized interactions. The resulting algorithm is shown to have low memory requirement, guaranteed linear convergence, robustness to failure of links or nodes and scalability to the network size. Moreover, the decentralized nature of the solution makes large-scale machine learning problems more tractable and also scalable since data is stored and processed locally at the nodes.

Index Terms— diffusion strategy, variance-reduction, stochastic gradient descent, memory efficiency, SVRG, SAGA, AVRG.

1. INTRODUCTION

This paper considers the empirical risk minimization (ERM) problem over a network. Consider a connected network with K nodes. If agent k stores local data samples $\{x_{k,n}\}_{n=1}^{N_k}$, where N_k is the size of the local samples, then the data stored by the entire network are:

$$\{x_n\}_{n=1}^{N} \;\overset{\Delta}{=}\; \Big\{\{x_{1,n}\}_{n=1}^{N_1},\cdots,\{x_{K,n}\}_{n=1}^{N_K}\Big\}, \qquad (1)$$

where $N=\sum_{k=1}^{K} N_k$ is the size of all data within the network. Note that we are allowing for different (uneven) amount of samples at various nodes. We then consider minimizing an empirical risk function, $J(w)$, which is defined as the sample average of loss values over *all* observed data samples, i.e.,

$$w^\star \;\overset{\Delta}{=}\; \operatorname*{arg\,min}_{w\in\mathbb{R}^M}\; J(w)=\frac{1}{N}\sum_{n=1}^{N} Q(w;x_n) \qquad (2)$$

Here, $Q(w;x_n)$ denotes the loss value evaluated at w and the n-th sample, x_n. For convenience, we introduce the local empirical risk function, $J_k(w)$, which is the sample average of loss values over the *local* data samples stored at node k, i.e., over $\{x_{k,n}\}_{n=1}^{N_k}$:

$$J_k(w) \;\overset{\Delta}{=}\; \frac{1}{N_k}\sum_{n=1}^{N_k} Q(w;x_{k,n}). \qquad (3)$$

Using the local empirical risk functions, it can be verified that

$$\frac{1}{N}\sum_{n=1}^{N} Q(w;x_n)\overset{(1)}{=}\frac{1}{N}\sum_{k=1}^{K}\sum_{n=1}^{N_k} Q(w;x_{k,n})\overset{(3)}{=}\sum_{k=1}^{K}\frac{N_k}{N}J_k(w), \qquad (4)$$

and hence the original global optimization problem (2) can be reformulated as the equivalent problem of minimizing the weighted aggregation of K local empirical risk functions:

$$w^\star \;\overset{\Delta}{=}\; \operatorname*{arg\,min}_{w\in\mathbb{R}^M}\;\; J(w)=\sum_{k=1}^{K} q_k J_k(w), \qquad (5)$$

where $q_k \overset{\Delta}{=} N_k/N$. Problem (5) are typical in multi-agent settings, where dispersed agents collect its own data and are interested in learning $w^\star$ in a decentralized manner. Each node is assigned a local computation task and the objective is to enable the nodes to learn the global solution $w^\star$. In this work we develop a fully decentralized algorithm where nodes process the data locally and are allowed to communicate only with their *immediate* neighbors. The following assumptions are standard in the decentralized optimization literature, and some common risk functions such as linear regression, ℓ_2-induced logistic regression satisfy these assumptions.

Assumption 1 *The loss function, $Q(w;x_n)$, is convex, twice-differentiable, and has a δ-Lipschitz continuous gradient, i.e., for any $w_1,w_2\in\mathbb{R}^M$ and $1\le n\le N$:*

$$\|\nabla_w Q(w_1;x_n)-\nabla_w Q(w_2;x_n)\|\le\delta\|w_1-w_2\| \qquad (6)$$

where $\delta>0$. Moreover, there exists at least one loss function $Q(w;x_{n_o})$ that is strongly convex, i.e.,

$$\nabla_w^2 Q(w;x_{n_o})\ge\nu I_M>0,\quad \text{for some } n_o. \qquad (7)$$

1.1. Related Work

There has been an extensive body of research on solving optimization problems of the form (5) in a fully decentralized manner. Some recent works include techniques such as EXTRA [1], DIGing [2] and Exact diffusion [3, 4]. These methods provide linear convergence to the exact minimizer, $w^\star$. However, all these methods require the evaluation of the true gradient $\nabla J_k(w)$ at each iteration. It is seen from definition (3) that this computation can be prohibitive for large-data scenarios where N_k can be large.

One can replace the true gradient by a stochastic gradient approximation, as is commonplace in traditional diffusion or consensus algorithms [5–7]. While this solution method is efficient, it converges linearly only to a small $O(\mu)-$neighborhood around the exact solution $w^\star$ where μ is the constant step-size. However, one can employ *variance-reduced* techniques to enable convergence to the *exact* minimizer. One such proposal is the DSA method [8], which is based on the SAGA method [9]. However, similar to SAGA, the DSA method suffers from the same huge memory requirement since each node k will need to store an estimate for each possible gradient $\{\nabla Q(w;x_{k,n})\}_{n=1}^{N_k}$. This requirement is a burden when N_k is large, as happens in applications involving large data sets.

1.2. Contribution

This paper derives a fully-decentralized variance-reduced stochastic-gradient algorithm with both linear convergence guarantees and sig-

This work was supported in part by NSF grant CCF-1524250. Emails: {kunyuan, ybc}@ucla.edu, and ali.sayed@epfl.ch

nificantly reduced memory requirements. We refer to the technique as the Diffusion-AVRG method (where AVRG stands for "amortized variance-reduced gradient" technique [10, 11]). Unlike DSA and SAGA, this method does not require extra memory to store gradient estimates. The proposed method also has balanced computations for each iteration, which is different from the well-known alternative to SAGA known as SVRG [12]. The SVRG method has an inner loop to perform stochastic variance-reduced gradient descent and an outer loop to calculate the true local gradient. These two loops complicate decentralized implementations when data is *unevenly* distributed across the nodes. In comparison, the proposed AVRG construction works efficiently for such scenario. Moreover, the proposed method applies to any connected network while other stochastic implementations [14,15] are limited to star network architectures.

2. DIFFUSION–AVRG ALGORITHM FOR BALANCED DATA DISTRIBUTIONS

2.1. Exact Diffusion Optimization

One effective decentralized method to solve problem (5) is the Exact diffusion strategy [3, 4]. To implement this algorithm, we need to associate a combination matrix $A = [a_{\ell k}]_{\ell,k=1}^{K}$ with the network graph, where a positive weight $a_{\ell k}$ is used to scale data that flows from node ℓ to k if both nodes happen to be neighbors. In this paper we assume A is symmetric and doubly stochastic, i.e.,

$$a_{\ell k} = a_{k\ell}, \quad A = A^{\mathsf{T}} \text{ and } A\mathbb{1}_K = \mathbb{1}_K \tag{8}$$

where $\mathbb{1}$ is a vector with all unit entries. We further introduce μ as the step-size parameter for all nodes, and let $\mathcal{N}_k$ denote the set of neighbors of node k (including node k itself).

The exact diffusion algorithm [3, 4] is listed in (9)–(11). The subscript k refers to the node while the subscript i refers to the iteration. It is proved in [4] that the local variables $w_{k,i}$ converge to the exact minimizer of problem (5), $w^\star$, at a linear convergence rate under relatively mild conditions.

Algorithm 1 (Exact diffusion strategy for each node k) [3, 4]

```
Let $\overline{A} = (I_N + A)/2$ and $\overline{a}_{\ell k} = [\overline{A}]_{\ell k}$. Initialize $w_{k,0}$ arbitrarily, and let $\psi_{k,0} = w_{k,0}$.
Repeat iteration $i = 1, 2, 3 \cdots$ until convergence
    $\psi_{k,i+1} = w_{k,i} - \mu\, q_k \nabla J_k(w_{k,i}),$                    (9)
    $\phi_{k,i+1} = \psi_{k,i+1} + w_{k,i} - \psi_{k,i},$                         (10)
    $w_{k,i+1} = \sum_{\ell \in \mathcal{N}_k} \overline{a}_{\ell k} \phi_{\ell,i+1}.$   (11)
```

2.2. Diffusion-AVRG

Note that in step (9) each agent needs to evaluate its $\nabla J_k(w_{k,i})$, which can be expensive when N_k is huge. To save computations, one can select a random data sample $x_{\boldsymbol{n}_i,k}$ and approximate $\nabla J_k(w_{k,i})$ by $\widehat{\nabla J}_k(w_{k,i}) = \nabla Q(w_{k,i}; x_{\boldsymbol{n}_i,k})$ as shown in [5, 6]. However, the presence of the gradient noise variance $\mathbb{E}\|\widehat{\nabla J}_k(\boldsymbol{w}_{k,i}) - \nabla J_k(\boldsymbol{w}_{k,i})\|^2$ drives convergence towards an $O(\mu)$-neighborhood around $w^\star$.

Algorithm 2 (Diffusion-AVRG at node k for balanced data)

```
Initialize $\boldsymbol{w}^0_{k,0}$ arbitrarily; let $\boldsymbol{\psi}^0_{k,0} = \boldsymbol{w}^0_{k,0}$, $\boldsymbol{g}^0_k = 0$, and $\nabla Q(\boldsymbol{w}^0_0; x_{k,n}) \leftarrow 0,\ 1 \le n \le \bar{N}$
Repeat epoch $t = 0, 1, 2, \cdots$
    generate a random permutation function $\boldsymbol{\sigma}^t_k$ and set $\boldsymbol{g}^{t+1}_k = 0$.
    Repeat iteration $i = 0, 1, \cdots, \bar{N} - 1$:
        $\boldsymbol{n}^t_i = \boldsymbol{\sigma}^t_k(i + 1),$                                        (15)
        $\widehat{\nabla J}_k(\boldsymbol{w}^t_{k,i}) = \nabla Q(\boldsymbol{w}^t_{k,i}; x_{k,\boldsymbol{n}^t_i}) - \nabla Q(\boldsymbol{w}^t_{k,0}; x_{k,\boldsymbol{n}^t_i}) + \boldsymbol{g}^t_k,$   (16)
        $\boldsymbol{g}^{t+1}_k \leftarrow \boldsymbol{g}^{t+1}_k + \frac{1}{\bar{N}} \nabla Q(\boldsymbol{w}^t_{k,i}; x_{k,\boldsymbol{n}^t_i}),$   (17)
        update $\boldsymbol{w}^t_{k,i+1}$ with exact diffusion:
        $\boldsymbol{\psi}^t_{k,i+1} = \boldsymbol{w}^t_{k,i} - \mu \widehat{\nabla J}_k(\boldsymbol{w}^t_{k,i}),$   (18)
        $\boldsymbol{\phi}^t_{k,i+1} = \boldsymbol{\psi}^t_{k,i+1} + \boldsymbol{w}^t_{k,i} - \boldsymbol{\psi}^t_{k,i},$   (19)
        $\boldsymbol{w}^t_{k,i+1} = \sum_{\ell \in \mathcal{N}_k} \overline{a}_{\ell k} \boldsymbol{\phi}^t_{\ell,i+1}.$   (20)
    End
    set $\boldsymbol{w}^{t+1}_{k,0} = \boldsymbol{w}^t_{k,\bar{N}}$ and $\boldsymbol{\psi}^{t+1}_{k,0} = \boldsymbol{\psi}^t_{k,\bar{N}}$
End
```

To correct the $O(\mu)$-bias, we propose to approximate $\nabla J_k(w)$ with a variance-reduced stochastic gradient. We first consider the scenario in which all nodes store the same amount of local data, i.e., $N_1 = \cdots = N_K = \bar{N} = N/K$. To reduce variance, we approximate $\nabla J_k(w)$ in the form of

$$\widehat{\nabla J}_k(\boldsymbol{w}^t_{k,i}) = \nabla Q(\boldsymbol{w}^t_{k,i}; x_{\boldsymbol{n}^t_i}) - \nabla Q(\boldsymbol{w}^t_{k,0}; x_{\boldsymbol{n}^t_i}) + \boldsymbol{g}^t_k, \tag{12}$$

where the superscript t is the epoch index, subscript i is the inner recursion index, and the auxiliary variable $\boldsymbol{g}^t_k$ is used to help reduce the variance. Inspired by our recent work on an amortized variance-reduced gradient method (AVRG) [10, 11], the variable $\boldsymbol{g}^{t+1}_k$ can be updated in a *recursive* manner at each inner iteration i:

$$\boldsymbol{g}^{t+1}_k \leftarrow \boldsymbol{g}^{t+1}_k + \frac{1}{\bar{N}} \nabla Q(\boldsymbol{w}^t_{k,i}; x_{k,\boldsymbol{n}^t_i}). \tag{13}$$

It can be proved that that when $\boldsymbol{n}^t_i$ is sampled without replacement, the stochastic gradient shown in (12) has vanishing variance. Specifically, it is proved in the extended version [13] of this paper that

$$\mathbb{E}\|\widehat{\nabla J}_k(\boldsymbol{w}^t_{k,i}) - \nabla J(\boldsymbol{w}^t_{k,i})\|^2 \le 6\delta^2 \mathbb{E}\|\boldsymbol{w}^t_{k,i} - \boldsymbol{w}^t_{k,0}\|^2 + \frac{3\delta^2}{\bar{N}} \sum_{j=0}^{\bar{N}-1} \mathbb{E}\|\boldsymbol{w}^{t-1}_{k,j} - \boldsymbol{w}^{t-1}_{k,\bar{N}}\|. \tag{14}$$

Suppose it holds for any $i \in [0, \bar{N}]$ that $\mathbb{E}\|\boldsymbol{w}^t_{k,i} - w^\star\|^2 \to 0$ when $t \to \infty$. It then follows that $\mathbb{E}\|\boldsymbol{w}^t_{k,i} - \boldsymbol{w}^t_{k,0}\| \to 0$ and $\mathbb{E}\|\boldsymbol{w}^{t-1}_{k,j} - \boldsymbol{w}^{t-1}_{k,\bar{N}}\| \to 0$, which implies $\mathbb{E}\|\widehat{\nabla J}_k(\boldsymbol{w}^t_{k,i}) - \nabla J(\boldsymbol{w}^t_{k,i})\|^2 \to 0$ by (14). In other words, the stochastic gradient (12) will perform as true gradient as $\boldsymbol{w}_{k,i}$ converges to $w^\star$, which is the intuition why Diffusion-AVRG will converge to the exact solution of problem (5).

Diffusion-AVRG is listed in Algorithm 2. At inner iteration i, each node k will first generate an amortized variance-reduced gradient $\widehat{\nabla J}_k(\boldsymbol{w}^t_{k,i})$ via (15)–(17), and then apply exact diffusion (18)–(20) to update $\boldsymbol{w}^t_{k,i+1}$. Unlike DSA [8], this algorithm does not require extra memory to store gradient estimates.

Remark 1. It is also possible to use SVRG [12] rather than AVRG to generate the variance-reduced gradient $\widehat{\nabla J}(\boldsymbol{w}_{k,i})$. In SVRG, it is suggested to set $\boldsymbol{g}^t_k$ as

$$\boldsymbol{g}^t_k = \frac{1}{\bar{N}} \sum_{n=1}^{\bar{N}} \nabla Q(\boldsymbol{w}^t_{k,0}; x_n), \tag{21}$$

where all $\nabla Q(\,\cdot\,; x_n)$'s are evaluated at the *same* point $\boldsymbol{w}_{k,0}$. The construction of $\boldsymbol{g}_k^t$ in (21) indicates that SVRG has to maintain an outer loop to calculate $\boldsymbol{g}_k^t$ in advance, which is different from AVRG in which the calculation of $\boldsymbol{g}_k^t$ is amortized into each inner iteration. The fact that $\boldsymbol{g}_k^t$ has to be updated in an offline manner (21) rather than the online manner (13) in SVRG will cause issues for scenarios in which local data sizes N_k differ from each other, as discussed in Sec. 3. ■

Remark 2. Diffusion-AVRG is more computation efficient than exact diffusion, but this computational advantage comes with extra communication costs. Note that Diffusion-AVRG will communicate after calculating only one stochastic gradient. Thus, in order to reach the same accuracy, Diffusion-AVRG will generally need more iterations than exact diffusion, which results in more communications. However, the computation-communication issue can be leveraged by the mini-batch strategy. From the simulations in [13], it is observed that Diffusion-AVRG with proper mini-batch size can be more computation efficient while maintaining almost the same communication efficiency as exact diffusion. ■

2.3. Convergence

Recursions (18)–(20) of Algorithm 2 only involve local variables $\boldsymbol{w}_{k,i}^t$, $\boldsymbol{\phi}_{k,i}^t$ and $\boldsymbol{\psi}_{k,i}^t$. To analyze the convergence of all $\{\boldsymbol{w}_{k,i}^t\}_{k=1}^K$, we need to combine all iterates from across the network into extended vectors. To do so, we introduce

$$\boldsymbol{\mathcal{W}}_i^t = \mathrm{col}\{\boldsymbol{w}_{1,i}^t, \cdots, \boldsymbol{w}_{K,i}^t\}, \quad \boldsymbol{\phi}_i^t = \mathrm{col}\{\boldsymbol{\phi}_{1,i}^t, \cdots, \boldsymbol{\phi}_{K,i}^t\} \tag{22}$$

$$\boldsymbol{\psi}_i^t = \mathrm{col}\{\boldsymbol{\psi}_{1,i}^t, \cdots, \boldsymbol{\psi}_{K,i}^t\}, \qquad \overline{\mathcal{A}} = \overline{A} \otimes I_M \tag{23}$$

$$\nabla \mathcal{J}(\boldsymbol{\mathcal{W}}_i^t) = \mathrm{col}\{\nabla J_1(\boldsymbol{w}_{1,i}^t), \cdots, \nabla J_K(\boldsymbol{w}_{K,i}^t)\} \tag{24}$$

$$\widehat{\nabla \mathcal{J}}(\boldsymbol{\mathcal{W}}_i^t) = \mathrm{col}\{\widehat{\nabla J}_1(\boldsymbol{w}_{1,i}^t), \cdots, \widehat{\nabla J}_K(\boldsymbol{w}_{K,i}^t)\} \tag{25}$$

where $\otimes$ is the Kronecker product. With the above notation, for $0 \le i \le \bar{N} - 1$ and $t \ge 0$, recursions (18)–(20) can be rewritten as

$$\begin{cases} \boldsymbol{\psi}_{i+1}^t = \boldsymbol{\mathcal{W}}_i^t - \mu \widehat{\nabla \mathcal{J}}(\boldsymbol{\mathcal{W}}_i^t), \\ \boldsymbol{\phi}_{i+1}^t = \boldsymbol{\psi}_{i+1}^t + \boldsymbol{\mathcal{W}}_i^t - \boldsymbol{\psi}_i^t, \\ \boldsymbol{\mathcal{W}}_{i+1}^t = \overline{\mathcal{A}}\, \boldsymbol{\phi}_{i+1}^t, \end{cases} \tag{26}$$

where we let $\boldsymbol{\psi}_0^{t+1} = \boldsymbol{\psi}_{\bar{N}}^t$ and $\boldsymbol{\mathcal{W}}_0^{t+1} = \boldsymbol{\mathcal{W}}_{\bar{N}}^t$ for new epoch $t+1$. By substituting the first and second equations of (26) into the third one, it holds for $1 \le i \le \bar{N}$ and $t \ge 0$ that:

$$\boldsymbol{\mathcal{W}}_{i+1}^t = \overline{\mathcal{A}}\Big(2\boldsymbol{\mathcal{W}}_i^t - \boldsymbol{\mathcal{W}}_{i-1}^t - \mu[\widehat{\nabla \mathcal{J}}(\boldsymbol{\mathcal{W}}_i^t) - \widehat{\nabla \mathcal{J}}(\boldsymbol{\mathcal{W}}_{i-1}^t)]\Big), \tag{27}$$

where we let $\boldsymbol{\mathcal{W}}_0^{t+1} = \boldsymbol{\mathcal{W}}_{\bar{N}}^t$ and $\boldsymbol{\mathcal{W}}_1^{t+1} = \boldsymbol{\mathcal{W}}_{\bar{N}+1}^t$ for epoch $t+1$. It is observed that recursion (27) involves two consecutive variables $\boldsymbol{\mathcal{W}}_i^t$ and $\boldsymbol{\mathcal{W}}_{i-1}^t$, which complicates the analysis. To deal with this issue, we introduce the eigen-decomposition $\frac{1}{2K}(I_K - A) = U \Sigma U^{\mathsf{T}}$ where Σ is a nonnegative diagonal matrix (note that $I_K - A$ is positive semi-definite because A is doubly stochastic), and U is an orthonormal matrix. We also define $V = U \Sigma^{1/2} U^{\mathsf{T}}$ and $\mathcal{V} = V \otimes I_M$. Note that V and $\mathcal{V}$ are symmetric matrices. It can be verified (see Appendix A in [13]) that recursion (27) is equivalent to

$$\begin{cases} \boldsymbol{\mathcal{W}}_{i+1}^t = \overline{\mathcal{A}}\Big(\boldsymbol{\mathcal{W}}_i^t - \mu \widehat{\nabla \mathcal{J}}(\boldsymbol{\mathcal{W}}_i^t)\Big) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_i^t \\ \boldsymbol{\mathcal{Y}}_{i+1}^t = \boldsymbol{\mathcal{Y}}_i^t + \mathcal{V}\boldsymbol{\mathcal{W}}_{i+1}^t \end{cases} \tag{28}$$

where $\boldsymbol{\mathcal{Y}}_i^t \in \mathbb{R}^{KM}$ is the auxiliary variable with initialization $\boldsymbol{\mathcal{Y}}_0^0 = 0$. We denote the gradient noise by

$$\boldsymbol{s}(\boldsymbol{\mathcal{W}}_i^t) = \widehat{\nabla \mathcal{J}}(\boldsymbol{\mathcal{W}}_i^t) - \nabla \mathcal{J}(\boldsymbol{\mathcal{W}}_i^t). \tag{29}$$

Substituting into (28), we get

$$\begin{cases} \boldsymbol{\mathcal{W}}_{i+1}^t = \overline{\mathcal{A}}\Big(\boldsymbol{\mathcal{W}}_i^t - \mu \nabla \mathcal{J}(\boldsymbol{\mathcal{W}}_i^t)\Big) - K\mathcal{V}\boldsymbol{\mathcal{Y}}_i^t - \mu\overline{\mathcal{A}}\,\boldsymbol{s}(\boldsymbol{\mathcal{W}}_i^t) \\ \boldsymbol{\mathcal{Y}}_{i+1}^t = \boldsymbol{\mathcal{Y}}_i^t + \mathcal{V}\boldsymbol{\mathcal{W}}_{i+1}^t \end{cases} \tag{30}$$

In summary, recursions (18)–(20) are equivalent to (30). Let $\widetilde{\boldsymbol{\mathcal{W}}}_i^t = \mathcal{W}^\star - \boldsymbol{\mathcal{W}}_i^t$ and $\widetilde{\boldsymbol{\mathcal{Y}}}_i^t = \mathcal{Y}^\star - \boldsymbol{\mathcal{Y}}_i^t$ denote error vectors relative to the solution pair $(\mathcal{W}^\star, \mathcal{Y}^\star)$. It is proved in Appendix B from [13] that recursion (30), under Assumption 1, can be transformed into the following linear recursion perturbed by a gradient noise term:

$$\begin{bmatrix} \widetilde{\boldsymbol{\mathcal{W}}}_{i+1}^t \\ \widetilde{\boldsymbol{\mathcal{Y}}}_{i+1}^t \end{bmatrix} = (\mathcal{B} - \mu \boldsymbol{\mathcal{T}}_i^t) \begin{bmatrix} \widetilde{\boldsymbol{\mathcal{W}}}_i^t \\ \widetilde{\boldsymbol{\mathcal{Y}}}_i^t \end{bmatrix} + \mu \mathcal{B}_l \boldsymbol{s}(\boldsymbol{\mathcal{W}}_i^t), \tag{31}$$

where $0 \le i \le \bar{N} - 1$, $t \ge 0$, and $\widetilde{\boldsymbol{\mathcal{W}}}_0^{t+1} = \widetilde{\boldsymbol{\mathcal{W}}}_{\bar{N}}^t$, $\widetilde{\boldsymbol{\mathcal{Y}}}_0^{t+1} = \widetilde{\boldsymbol{\mathcal{Y}}}_{\bar{N}}^t$ after epoch t. Moreover, $\mathcal{B}, \mathcal{B}_l$ and $\boldsymbol{\mathcal{T}}_i^t$ are defined as

$$\mathcal{B} \triangleq \begin{bmatrix} \overline{\mathcal{A}} & -K\mathcal{V} \\ \mathcal{V}\overline{\mathcal{A}} & \overline{\mathcal{A}} \end{bmatrix}, \; \mathcal{B}_l \triangleq \begin{bmatrix} \overline{\mathcal{A}} \\ \mathcal{V}\overline{\mathcal{A}} \end{bmatrix}, \; \boldsymbol{\mathcal{T}}_i^t \triangleq \begin{bmatrix} \overline{\mathcal{A}}\boldsymbol{\mathcal{H}}_i^t & 0 \\ \mathcal{V}\overline{\mathcal{A}}\boldsymbol{\mathcal{H}}_i^t & 0 \end{bmatrix}, \tag{32}$$

where

$$\boldsymbol{\mathcal{H}}_i^t = \mathrm{diag}\{\boldsymbol{H}_{1,i}^t, \cdots, \boldsymbol{H}_{K,i}^t\} \in \mathbb{R}^{KM \times KM}, \tag{33}$$

$$\boldsymbol{H}_{k,i}^t \triangleq \int_0^1 \nabla^2 J_k\big(w^\star - r\widetilde{\boldsymbol{w}}_{k,i}^t\big) dr \in \mathbb{R}^{M \times M}. \tag{34}$$

To facilitate the convergence analysis of recursion (31), we diagonalize $\mathcal{B}$ and transform (31) into an equivalent error dynamics. From equations (64)–(67) in [4], we know that $\mathcal{B}$ admits the decomposition

$$\mathcal{B} \;\triangleq\; \mathcal{X}\mathcal{D}\mathcal{X}^{-1}, \tag{35}$$

where $\mathcal{X}, \mathcal{D}$ and $\mathcal{X}^{-1}$ are matrices defined as

$$\mathcal{D} \;\triangleq\; \begin{bmatrix} I_M & 0 & 0 \\ 0 & I_M & 0 \\ 0 & 0 & \mathcal{D}_1 \end{bmatrix} \in \mathbb{R}^{2KM \times 2KM}, \tag{36}$$

$$\mathcal{X} \;\triangleq\; \begin{bmatrix} \mathcal{R}_1 & \mathcal{R}_2 & \mathcal{X}_R \end{bmatrix} \in \mathbb{R}^{2KM \times 2KM}, \tag{37}$$

$$\mathcal{X}^{-1} \;\triangleq\; \begin{bmatrix} \mathcal{L}_1^{\mathsf{T}} \\ \mathcal{L}_2^{\mathsf{T}} \\ \mathcal{X}_L \end{bmatrix} \in \mathbb{R}^{2KM \times 2KM}. \tag{38}$$

In (36), matrix $\mathcal{D}_1 = D_1 \otimes I_M$ and $D_1 \in \mathbb{R}^{2(K-1) \times 2(K-1)}$ is a diagonal matrix with $\|D_1\| = \lambda_2(A) \;\triangleq\; \lambda < 1$. In (37) and (38), matrices $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{L}_1$ and $\mathcal{L}_2$ take the form

$$\mathcal{R}_1 = \begin{bmatrix} \mathbb{1}_K \\ 0_K \end{bmatrix} \otimes I_M, \qquad \mathcal{R}_2 = \begin{bmatrix} 0_K \\ \mathbb{1}_K \end{bmatrix} \otimes I_M \tag{39}$$

$$\mathcal{L}_1 = \begin{bmatrix} \frac{1}{K}\mathbb{1}_K \\ 0_K \end{bmatrix} \otimes I_M, \quad \mathcal{L}_2 = \begin{bmatrix} 0_K \\ \frac{1}{K}\mathbb{1}_K \end{bmatrix} \otimes I_M \tag{40}$$

Moreover, $\mathcal{X}_R \in \mathbb{R}^{2KM \times 2(K-1)M}$ and $\mathcal{X}_L \in \mathbb{R}^{2(K-1)M \times 2KM}$ are some constant matrices. Since $\mathcal{B}$ is independent of $\bar{N}$, δ and ν, all matrices appearing in (35)–(38) are independent of these variables as well. By multiplying $\mathcal{X}^{-1}$ to both sides of recursion (31), we have

$$\mathcal{X}^{-1}\begin{bmatrix} \widetilde{\boldsymbol{\mathcal{W}}}_{i+1}^t \\ \widetilde{\boldsymbol{\mathcal{Y}}}_{i+1}^t \end{bmatrix}$$

$$\overset{(35)}{=} \Big(\mathcal{D} - \mu\mathcal{X}^{-1}\boldsymbol{\mathcal{T}}_i^t\mathcal{X}\Big)\left(\mathcal{X}^{-1}\begin{bmatrix} \widetilde{\boldsymbol{\mathcal{W}}}_i^t \\ \widetilde{\boldsymbol{\mathcal{Y}}}_i^t \end{bmatrix}\right) + \mu\mathcal{X}^{-1}\mathcal{B}_l\boldsymbol{s}(\boldsymbol{\mathcal{W}}_i^t) \tag{41}$$

Now we define

$$\begin{bmatrix} \bar{\boldsymbol{x}}_i^t \\ \widehat{\boldsymbol{x}}_i^t \\ \check{\boldsymbol{x}}_i^t \end{bmatrix} \;\triangleq\; \mathcal{X}^{-1}\begin{bmatrix} \widetilde{\boldsymbol{\mathcal{W}}}_i^t \\ \widetilde{\boldsymbol{\mathcal{Y}}}_i^t \end{bmatrix} \overset{(38)}{=} \begin{bmatrix} \mathcal{L}_1^{\mathsf{T}} \\ \mathcal{L}_2^{\mathsf{T}} \\ \mathcal{X}_L \end{bmatrix}\begin{bmatrix} \widetilde{\boldsymbol{\mathcal{W}}}_i^t \\ \widetilde{\boldsymbol{\mathcal{Y}}}_i^t \end{bmatrix}, \tag{42}$$

as transformed errors. Moreover, we partition $\mathcal{X}_R$ as

$$\mathcal{X}_R = \begin{bmatrix} \mathcal{X}_{R,u} \\ \mathcal{X}_{R,d} \end{bmatrix}, \quad \text{where} \quad \mathcal{X}_{R,u} \in \mathbb{R}^{KM \times 2(K-1)M}. \tag{43}$$

With recursion (41), we can establish the following lemma.

Lemma 1 (Useful Transformation) *When $\boldsymbol{\mathcal{y}}_0^0$ is initialized at* 0, *recursion* (31) *can be transformed into*

$$\begin{bmatrix}\bar{\boldsymbol{\mathcal{x}}}_{i+1}^t\\ \check{\boldsymbol{\mathcal{x}}}_{i+1}^t\end{bmatrix}=\begin{bmatrix}I_M-\frac{\mu}{K}\mathcal{I}^{\mathsf{T}}\boldsymbol{\mathcal{H}}_i^t\mathcal{I} & -\frac{\mu}{K}\mathcal{I}^{\mathsf{T}}\boldsymbol{\mathcal{H}}_i^t\mathcal{X}_{R,u}\\ -\mu\mathcal{X}_L\boldsymbol{\mathcal{T}}_i^t\mathcal{R}_1 & \mathcal{D}_1-\mu\mathcal{X}_L\boldsymbol{\mathcal{T}}_i^t\mathcal{X}_R\end{bmatrix}\begin{bmatrix}\bar{\boldsymbol{\mathcal{x}}}_i^t\\ \check{\boldsymbol{\mathcal{x}}}_i^t\end{bmatrix}+\mu\begin{bmatrix}\frac{1}{K}\mathcal{I}^{\mathsf{T}}\\ \mathcal{X}_L\mathcal{B}_l\end{bmatrix}\boldsymbol{s}(\boldsymbol{\mathcal{w}}_i^t) \tag{44}$$

where $\mathcal{I}=\mathbb{1}_K\otimes I_M$. Moreover, the relation between $\widetilde{\boldsymbol{\mathcal{w}}}_i^t,\widetilde{\boldsymbol{\mathcal{y}}}_i^t$ and $\bar{\boldsymbol{\mathcal{x}}}_i^t,\check{\boldsymbol{\mathcal{x}}}_i^t$ in (41) *reduces to*

$$\begin{bmatrix}\widetilde{\boldsymbol{\mathcal{w}}}_i^t\\ \widetilde{\boldsymbol{\mathcal{y}}}_i^t\end{bmatrix}=\mathcal{X}\begin{bmatrix}\bar{\boldsymbol{\mathcal{x}}}_i^t\\ 0_M\\ \check{\boldsymbol{\mathcal{x}}}_i^t\end{bmatrix}. \tag{45}$$

Notice that $\mathcal{X}_L$, $\mathcal{X}_R$, $\mathcal{X}_{R,u}$ and $\mathcal{X}$ are all constant matrices and independent of $\bar{N}$, δ and ν.

Proof. See Appendix C in [13]. ■

In the next lemma we bound the gradient noise $\mathbb{E}\|\boldsymbol{s}(\boldsymbol{\mathcal{w}}_i^t)\|^2$.

Lemma 2 (Gradient Noise) *Under Assumption 1, the second moment of the gradient noise term satisfies:*

$$\begin{aligned}&\mathbb{E}\|\boldsymbol{s}(\boldsymbol{\mathcal{w}}_i^t)\|^2\\ &\le 6b\delta^2\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_i^t-\bar{\boldsymbol{\mathcal{x}}}_0^t\|^2+12b\delta^2\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_i^t\|^2+18b\delta^2\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_0^t\|^2\\ &\quad+\frac{3b\delta^2}{\bar{N}}\sum_{j=0}^{\bar{N}-1}\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_j^{t-1}-\bar{\boldsymbol{\mathcal{x}}}_{\bar{N}}^{t-1}\|^2+\frac{6b\delta^2}{\bar{N}}\sum_{j=0}^{\bar{N}-1}\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_j^{t-1}\|^2,\end{aligned} \tag{46}$$

where b is a positive constant that is independent of $\bar{N}$, ν and δ.

Proof. See Appendix E in [13]. ■

In the following, we will exploit the error dynamic (44) and the upper bound (46) to establish the convergence of $\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_i^t\|^2$ and $\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_i^t\|^2$. To simplify the notation, we define

$$\begin{aligned}&\mathbf{A}^t\triangleq\frac{1}{\bar{N}}\sum_{j=0}^{\bar{N}-1}\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_j^t-\bar{\boldsymbol{\mathcal{x}}}_0^t\|^2,\ \mathbf{B}^t\triangleq\frac{1}{\bar{N}}\sum_{j=0}^{\bar{N}-1}\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_j^t-\bar{\boldsymbol{\mathcal{x}}}_{\bar{N}}^t\|^2,\\ &\mathbf{C}^t\ \triangleq\ \frac{1}{\bar{N}}\sum_{j=0}^{\bar{N}-1}\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_j^t\|^2.\end{aligned} \tag{47}$$

All these quantities appear in the upper bound in (46).

Theorem 1 (Linear Convergence) *Under Assumption 1, if the step-size μ satisfies*

$$\mu\le C\left(\frac{\nu(1-\lambda)}{\delta^2\bar{N}}\right), \tag{48}$$

where $C>0$ is a constant independent of $\bar{N}$, ν and δ, and $\lambda=\lambda_2(A)<1$ is the second largest eigenvalue of the combination matrix A, it then holds that

$$\begin{aligned}&\left(\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_0^{t+1}\|^2+\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_0^{t+1}\|^2\right)+\frac{\gamma}{2}\left(\mathbf{A}^{t+1}+\mathbf{B}^t+\mathbf{C}^t\right)\\ &\le\rho\left\{\left(\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_0^t\|^2+\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_0^t\|^2\right)+\frac{\gamma}{2}(\mathbf{A}^t+\mathbf{B}^{t-1}+\mathbf{C}^{t-1})\right\}\end{aligned} \tag{49}$$

where $\gamma>0$ is a constant, and

$$\rho=\frac{1-\frac{\bar{N}}{8}a\mu\nu}{1-8b\mu^3\delta^4\bar{N}^3/\nu}<1. \tag{50}$$

The positive constants a and b are independent of $\bar{N}$, ν and δ.

Proof. See Appendix K in [13]. ■

From Theorem 1 and the fact that $\mathbf{A}^t,\mathbf{B}^t$ and $\mathbf{C}^t$ are all positive constants, we get

$$\begin{aligned}\mathbb{E}\|\widetilde{\boldsymbol{\mathcal{w}}}_0^{t+1}\|^2&\le\mathbb{E}\left(\|\widetilde{\boldsymbol{\mathcal{w}}}_0^{t+1}\|^2+\|\widetilde{\boldsymbol{\mathcal{y}}}_0^{t+1}\|^2\right)\\ &\overset{(45)}{\le}\|\mathcal{X}\|^2\left(\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_0^{t+1}\|^2+\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_0^{t+1}\|^2\right)\overset{(49)}{\le}\rho^t\|\mathcal{X}\|^2C_0\end{aligned} \tag{51}$$

where $C_0=\left(\mathbb{E}\|\bar{\boldsymbol{\mathcal{x}}}_0^0\|^2+\mathbb{E}\|\check{\boldsymbol{\mathcal{x}}}_0^0\|^2\right)+\frac{\gamma}{2}(\mathbf{A}^1+\mathbf{B}^0+\mathbf{C}^0)$. The above inequality implies that $\mathbb{E}\|\boldsymbol{w}_{k,0}^t-w^\star\|\to 0$ for any agent k.

Algorithm 3 (Diffusion-AVRG at node k for unbalanced data)

```
Initialize $\boldsymbol{w}_{k,0}$ arbitrarily; let $q_k=N_k/N$, $\boldsymbol{\psi}_{k,0}=\boldsymbol{w}_{k,0}$, $\boldsymbol{g}_k^0=0$, and $\nabla Q(\boldsymbol{\theta}_{k,0}^0;x_{k,n})\leftarrow 0,\ 1\le n\le N_k$
Repeat $i=0,1,2,\cdots$
  calculate $t$ and $s$ such that $i=tN_k+s$, where $t\in\mathbb{Z}_+$ and $s=\mathrm{mod}(i,N_k)$;
  If $s=0$:
    generate a random permutation $\boldsymbol{\sigma}_k^t$; let $\boldsymbol{g}_k^{t+1}=0,\ \boldsymbol{\theta}_{k,0}^t=\boldsymbol{w}_{k,i}$;
  End
  generate the local stochastic gradient:
      $\boldsymbol{n}_s^t=\boldsymbol{\sigma}_k^t(s+1),$   (52)
      $\widehat{\nabla J}_k(\boldsymbol{w}_{k,i})=\nabla Q(\boldsymbol{w}_{k,i};x_{k,\boldsymbol{n}_s^t})-\nabla Q(\boldsymbol{\theta}_{k,0}^t;x_{k,\boldsymbol{n}_s^t})+\boldsymbol{g}_k^t,$   (53)
      $\boldsymbol{g}_k^{t+1}\leftarrow\boldsymbol{g}_k^{t+1}+\frac{1}{N_k}\nabla Q(\boldsymbol{w}_{k,i};x_{k,\boldsymbol{n}_s^t}),$   (54)
  update $\boldsymbol{w}_{k,i+1}$ with exact diffusion:
      $\boldsymbol{\psi}_{k,i+1}=\boldsymbol{w}_{k,i}-\mu q_k\widehat{\nabla J}_k(\boldsymbol{w}_{k,i}),$   (55)
      $\boldsymbol{\phi}_{k,i+1}=\boldsymbol{\psi}_{k,i+1}+\boldsymbol{w}_{k,i}-\boldsymbol{\psi}_{k,i},$   (56)
      $\boldsymbol{w}_{k,i+1}=\sum_{\ell\in\mathcal{N}_k}\bar{a}_{\ell k}\boldsymbol{\phi}_{\ell,i+1}.$   (57)
End
```

3. DIFFUSION–AVRG ALGORITHM UNDER UNBALANCED DATA DISTRIBUTIONS

When the size of the data collected at the nodes may vary, some challenges arise. For example, assume we select $\widehat{N}=\max_k\{N_k\}$ as the epoch size for all nodes. When node k with a smaller N_k finishes its epoch, it will stop and wait for the other nodes to finish their epochs. Such an implementation is inefficient because nodes will be idle while they could be assisting in improving the convergence performance. We instead assume that nodes will continue updating without any idle time. If a particular node k finishes running over all its data samples during an epoch, it will then continue into its next epoch right away. In this way, there is no need to introduce a uniform epoch. We list the method in Algorithm 3; this listing includes the case of balanced data as a special case.

In Algorithm 3, to generate the local gradient $\widehat{\nabla J}_k(\boldsymbol{w}_{k,i})$, node k will transform the *global* iteration index i to its own *local* epoch index t and *local* inner iteration s. With t and s determined, node k is able to generate $\widehat{\nabla J}_k(\boldsymbol{w}_{k,i})$ with the AVRG recursions (52)–(54). It is worth noting that the local update (52)–(56) for each node k at each iteration requires the same amount of computations no matter how different the sample sizes $\{N_k\}$ are. This balanced computation feature guarantees the efficiency of Diffusion-AVRG. Figure 1 illustrates the operation of Algorithm 4 for a two-node network with $N_1=2$ and $N_2=3$. Observe that the local computations has similar widths because each node has a balanced computation cost per iteration. Note that $\boldsymbol{\mathcal{w}}_i=[\boldsymbol{w}_{1,i};\boldsymbol{w}_{2,i}]$ in Figure 1.

3.1. Comparison with Decentralized SVRG

SVRG has two-loop structures, which is not suitable for decentralized setting, especially when data can be distributed unevenly. To

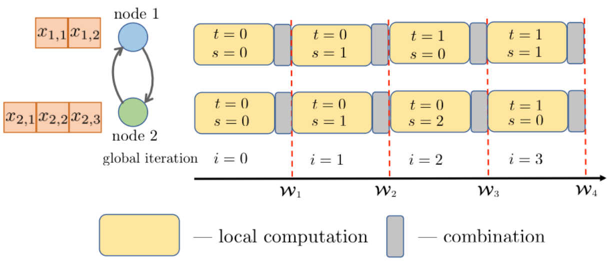

Fig. 1: Illustration of the operation of Diffusion-AVRG.

illustrate this fact assume, for the sake of argument, that we combine exact diffusion with SVRG to obtain a Diffusion-SVRG variant, which we list in Algorithm 5 in [13]. Similar to Diffusion-AVRG, each node k will transform the global iteration index i into a local epoch index t and a local inner iteration s, which are then used to generate $\widehat{\nabla J}(\boldsymbol{w}_{k,i})$ through SVRG. At the very beginning of each local epoch t, a true local gradient has to be calculated in advance; this step causes a pause before the update of $\boldsymbol{\phi}_{k,i+1}$. Now since the neighbors of node k will be waiting for $\boldsymbol{\phi}_{k,i+1}$ in order to update their own $\boldsymbol{w}_{\ell,i+1}$, the pause by node k will cause all its neighbors to wait. These waits reduce the efficiency of this decentralized implementation, which explains why the Diffusion-AVRG algorithm is preferred. Figure 2 in the extended version [13] illustrates the Diffusion-SVRG strategy with $N_1 = 2$ and $N_2 = 3$, from which we can observe that the balanced calculation resulting from AVRG effectively reduces idle times and enhances the efficiency of the decentralized implementation.

4. SIMULATIONS

In this section, we illustrate the convergence performance of Diffusion-AVRG. We consider problem (5) in which $J_k(w)$ takes the form of regularized logistic regression loss function:

$$J_k(w) \triangleq \frac{1}{N_k}\sum_{n=1}^{N_k}\left(\frac{\rho}{2}\|w\|^2+\ln\left(1+\exp(-\gamma_k(n)h_{k,n}^{\mathsf{T}}w)\right)\right)$$

The vector $h_{k,n}$ is the n-th feature vector kept by node k and $\gamma_k(n) \in \{\pm 1\}$ is the corresponding label. In all experiments, the factor ρ is set to $1/N$, and the solution $w^\star$ to (5) is computed by using the Scikit-Learn Package. All experiments are run over four datasets: covtype.binary, rcv1.binary, and MNIST. The last dataset has been transformed into binary classification problems by considering data with labels 2 and 4. We generate a network with $K = 20$ nodes, the topology of which is shown in Figure 3 in [13].

In our experiments, we test the convergence performance of Diffusion-AVRG with both even and uneven data distribution. We compare Diffusion-AVRG with DSA [8]. The experimental results for even data are shown in the top 3 plots of Figure 2, and the results for uneven data are shown in bottom 3 plots. To enable fair comparisons, we tune the step-size parameter of each algorithm for fastest convergence in each case. The plots are based on measuring the averaged relative square-error, $\frac{1}{K}\sum_{k=1}^{K}\|\boldsymbol{w}_{k,0}^{t} - \boldsymbol{w}^\star\|^2/\|\boldsymbol{w}^\star\|^2$. It is observed that both algorithms converge linearly to $w^\star$, while Diffusion-AVRG converges faster (especially on Covtype).

5. REFERENCES

[1] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, May 2015.

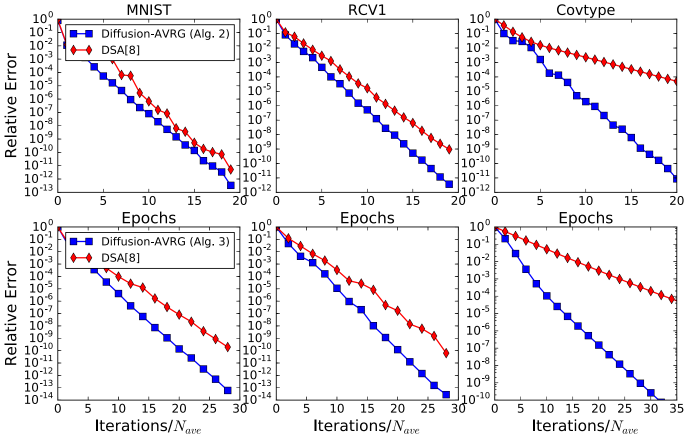

Fig. 2: Comparison between Diffusion-AVRG and DSA. Top: data are evenly distributed over the nodes; Bottom: data are unevenly distributed over the nodes. The average sample size is $N_{\text{ave}} = \sum_{k=1}^{K} N_k/K$.

[2] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *arXiv:1607.03218*, Jul. 2016.

[3] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact dffusion for distributed optimization and learning – Part I: Algorithm development," *submitted for publication,* and available as *arXiv:1702.05122*, Feb. 2017.

[4] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact dffusion for distributed optimization and learning – Part II: Convergence analysis," *submitted for publication,* and available as *arXiv:1702.05122*, Feb. 2017.

[5] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[6] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, Jul. 2014.

[7] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[8] A. Mokhtari and A. Ribeiro, "DSA: Decentralized double stochastic averaging gradient algorithm," *Journal of Machine Learning Research*, vol. 17, no. 61, pp. 1–35, Mar. 2016.

[9] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. NIPS*, Montréal, Canada, 2014, pp. 1646–1654.

[10] B. Ying, K. Yuan, and A. H. Sayed, "Variance-reduced stochastic learning under random reshuffling," *Submitted for publication,* and available as arXiv:1708.01383, 2017.

[11] B. Ying, K. Yuan, and A. H. Sayed, "Convergence of variance-reduced stochastic learning under random reshuffling," in *Proc. IEEE ICASSP*, Calgary, Canada, Apr. 2018., pp. 1 – 5.

[12] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Advances in Neural Information Processing Systems* (NIPS), Lake Tahoe, 2013, pp. 315–323.

[13] K. Yuan, B. Ying, J. Liu, and A. H. Sayed, "Variance-reduced stochastic learning by networked agents under random reshuffling," *Submitted for publication.* Also available as arXiv 1708.01384, Aug. 2017.

[14] J. D. Lee, Q. Lin, T. Ma, and T. Yang, "Distributed stochastic variance reduced gradient methods by sampling extra data with replacement," *Journal of Machine Learning Research*, vol. 18, pp. 1–43, 2017.

[15] J. Wang, W. Wang, and N. Srebro, "Memory and communication efficient distributed stochastic optimization with minibatch prox," in *Proc. Machine Learning Research*, vol. 65, pp. 1–37, 2017.