Towards More Efficient Stochastic Decentralized Learning: Faster Convergence and Sparse Communication

Zebang Shen [1 2] **Aryan Mokhtari** [3] **Tengfei Zhou** [1] **Peilin Zhao** [4] **Hui Qian** [1]

Abstract

Recently, the decentralized optimization problem is attracting growing attention. Most existing methods are deterministic with high per-iteration cost and have a convergence rate quadratically depending on the problem condition number. Besides, the dense communication is necessary to ensure the convergence even if the dataset is sparse. In this paper, we generalize the decentralized optimization problem to a monotone operator root finding problem, and propose a stochastic algorithm named DSBA that (i) converges geometrically with a rate linearly depending on the problem condition number, and (ii) can be implemented using sparse communication only. Additionally, DSBA handles learning problems like AUC-maximization which cannot be tackled efficiently in the decentralized setting. Experiments on convex minimization and AUC-maximization validate the efficiency of our method.

1. Introduction

Over the last decades, decentralized learning has received a lot of attention in the machine learning community due to the rise of distributed high-dimensional datasets. This paper focuses on finding a global solution to learning problems in the setting where each node merely has access to a subset of data and are allowed to exchange information with their neighboring nodes only. Specifically, consider a connected network with N nodes where each node n has access to a local function $f_n : \mathbb{R}^d \to \mathbb{R}$ which is the average of q component functions $f_{n,i} : \mathbb{R}^d \to \mathbb{R}$, i.e. $f_n(\mathbf{x}) = (1/q)\sum_{i=1}^q f_{n,i}(\mathbf{x})$. Considering $\mathbf{x}_n$ as the local variable of node n, the problem of interest is

$$\min_{\substack{\{\mathbf{x}_n\}_{n=1}^N \\ \text{s.t. } \mathbf{x}_1=\ldots=\mathbf{x}_N}} \sum_{n=1}^N \left\{ f_n(\mathbf{x}_n) \triangleq \frac{1}{q}\sum_{i=1}^q f_{n,i}(\mathbf{x}_n) \right\}. \quad (1)$$

The formulation (1) captures problems in sensor network, mobile computation, and multi-agent control, where either efficiently centralizing data or globally aggregate intermediate results is unfeasible (Johansson, 2008; Bullo et al., 2009; Forero et al., 2010; Ribeiro, 2010).

Developing efficient methods for such problem has been one of the major efforts in the machine learning community. While early work dates back to 1980's (Tsitsiklis et al., 1986; Bertsekas & Tsitsiklis, 1989), consensus based gradient descent and dual averaging methods with sublinear convergence have made their debut (Nedic & Ozdaglar, 2009; Duchi et al., 2012), which consist of two steps: all nodes (i) gather the (usually dense) iterates from theirs neighbors via communication to compute a weighted average, and (ii) update the average by the full gradient of the local f_n to obtain new iterates. Following such protocol, successors with linear convergence have been proposed recently (Shi et al., 2015a; Mokhtari et al., 2016; Scaman et al., 2017).

Despite the progress, two entangled challenges, realized by the above interlacing steps, still remain. The first challenge is the *high computation complexity* of existing methods. Real world tasks commonly suffer from the *ill-conditionness* of the underlying problem, which deteriorates the performance of existing methods due to their heavy dependence on the problem condition number (Shi et al., 2015a; Mokhtari et al., 2016). Besides, even a single node could contain a plethora of data points, which impedes the *full local gradient* evaluation required by most existing methods. The second challenge is the high *communication overhead*. The existing linear convergent methods overlooked such practical issue and simply adopt the *dense communication* strategy, which restrains their applications.

Furthermore, important problems like AUC maximization involves pairwise component functions which take input outside the local nodes. Multiple rounds of communications are necessary to estimate the gradient, which precludes the direct application of existing linear convergent algorithms.

[1]Zhejiang University [2]Tencent AI Lab [3]Massachusetts Institute of Technology [4]South China University of Technology. Correspondence to: Hui Qian <qianhui@zju.edu.cn>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Only sublinear convergent algorithm exists (Colin et al., 2016; Ying et al., 2016).

To bridge these gaps, we rephrase the problem (1) under the monotone operator framework and propose an efficient algorithm named Decentralized Stochastic Backward Aggregation (DSBA). In the computation step of DSBA, each node computes the resolvent of a stochastically approximated monotone operator to reduce the dependence on the problem condition number. Such resolvent admits closed form solutions in problems like Ridge Regression. In the communication step of DSBA, each node receives the nonzero components of the difference between consecutive iterates to reconstruct the neighbors' iterates. Since the ℓ_2-relaxed AUC maximization problem is equivalent to the minimax problem of a convex-concave function, whose differential is a monotone operator, fitting it into our formulation is seamless. More specifically, our contributions are as follows:

1. DSBA accesses a single data point in each iteration and converges linearly with a fast rate. The number of steps required to ϵ accurate solution is $\mathcal{O}((\kappa+\kappa_g+q)\log\frac{1}{\epsilon})$, where κ is the condition number of the problem and κ_g is the condition number of the graph. This rate significantly improves over the existing stochastic decentralized solvers and most deterministic ones, which also holds for the ℓ_2-relaxed AUC maximization.
2. In contrast to the dense vector transmission in existing methods, the inter-node communication is sparse in DSBA. Specifically, the per-iteration communication complexity is $\mathcal{O}(\rho d)$ for DSBA and $\mathcal{O}(d)$ for all the other linear convergent methods, where ρ is the sparsity of the dataset and d is the problem dimension. When communication is a critical factor, our sparse communication scheme is more favorable.

Empirical studies on convex minimization and AUC maximization problems are conducted to validate the efficiency of our algorithm. Improvements are observed in both computation and communication.

Notations

We use the bold uppercase letters to denote matrices and bold lowercase letters to denote vectors. We refer the i^{th} row of matrix $\mathbf{W}$ by $[\mathbf{W}]_i$ and refer the element in the i^{th} row and j^{th} column by $[\mathbf{W}]_{i,j}$. $\mathbf{W}^k$ denotes the k^{th} power of $\mathbf{W}$. $\mathrm{Proj}_{\mathbf{U}}$ is the projection operator to the range of $\mathbf{U}$.

2. Related Work

Deterministic Methods: Directly solving the primal objective, the consensus-based Decentralized Gradient Descent (DGD) method (Nedic & Ozdaglar, 2009; Yuan et al., 2016) has been proposed, yielding sublinear convergence rate. EXTRA (Shi et al., 2015a) improves over DGD by incorporating information from the last two iterates and is shown to converge linearly. Alternatively, D-ADMM (Shi et al., 2014) directly applies ADMM method to problem (1) and achieves linear convergence. However, D-ADMM computes the proximal operator of f_n in each iteration. To avoid such expensive proximal operator computation, Ling et al. propose a linearized variant of D-ADMM named DLM (Ling et al., 2015). There also have been some efforts to exploit second-order information for accelerating convergence in ill-condition problems (Mokhtari et al., 2017; Eisen et al., 2017). From the dual perspective, (Duchi et al., 2012) uses the dual averaging method and obtains a sublinear convergent algorithm. The work in (Necoara et al., 2017) applies the random block coordinate gradient descent on the dual objective to obtain linear convergence with a rate that depends on τ, the number of blocks being selected per iteration. When $\tau > 2$, multiple rounds of communications are needed to implement the method. Recently, (Scaman et al., 2017) applies the accelerated gradient descent methods on the dual problem of (1) to give a method named SSDA and its multi-step communication variant MSDA and shows that the proposed methods are optimal. However, both SSDA and MSDA require computing the gradient of the conjugate function f_n^*. All the above methods access the whole dataset in each iteration without exploiting the finite sum structure.

Stochastic Methods: By incorporating the SAGA approximation technique, Mokhtari & Ribeiro recently proposed a method named DSA to handle Problem (1) in a stochastic manner. In each iteration, it only computes the gradient of a single component function $f_{n,i}$, which is significantly cheaper than the full gradient evaluation (Mokhtari & Ribeiro, 2016). DSA converges linearly, while the overall required complexity heavily depends on function and graph condition numbers.

We summarize the convergence rate, computation and communication cost of the aforementioned methods in Table 1.

3. Preliminary

3.1. Monotone Operator

Monotone operator is a tool for modeling optimization problems including convex minimization (Rockafellar et al., 1970) and minimax problem of convex-concave functions (Rockafellar, 1970). A relation $\mathcal{B}$ is a monotone operator if

$$(u-v)^\top(x-y)\geq 0, \forall (x,u),(y,v)\in\mathcal{B}. \tag{2}$$

$\mathcal{B}$ is maximal monotone if there is no monotone operator that properly contains it. We say an operator $\mathcal{B}$ is μ-strongly monotone if

$$\langle \mathcal{B}(\mathbf{x})-\mathcal{B}(\mathbf{y}),\mathbf{x}-\mathbf{y}\rangle \geq \mu\|\mathbf{x}-\mathbf{y}\|^2 \tag{3}$$

Table 1. κ is the condition number of the problem and κ_g be the condition number of the network graph, defined in section 6. $\Delta(\mathcal{G})$ is the max degree of the graph $\mathcal{G}$. ρ is the sparsity of the dataset, i.e. the ratio of nonzero elements. τ is the complexity of solving a 1-dimensional equation, and is $\mathcal{O}(1)$ in problems like Ridge Regression. All the complexity are derived for problems with linear predictor.

Method	Convergence Rate	Per-iteration Cost	Communication Cost
EXTRA (Shi et al., 2015a)	$\mathcal{O}((\kappa^2+\kappa_g)\log\frac{1}{\epsilon})$	$\mathcal{O}(\rho qd+\Delta(\mathcal{G})d)$	$\mathcal{O}(\Delta(\mathcal{G})d)$
DLM (Ling et al., 2015)	$\mathcal{O}((\kappa^2+\kappa_g\kappa)\log\frac{1}{\epsilon})$	$\mathcal{O}(\rho qd+\Delta(\mathcal{G})d)$	$\mathcal{O}(\Delta(\mathcal{G})d)$
SSDA (Scaman et al., 2017)	$\mathcal{O}(\sqrt{\kappa\kappa_g}\log\frac{1}{\epsilon})$	$\mathcal{O}(\rho qd+q\tau+\Delta(\mathcal{G})d)$	$\mathcal{O}(\Delta(\mathcal{G})d)$
DSA (Mokhtari & Ribeiro, 2016)	$\mathcal{O}((\kappa^4\kappa_g+\kappa_g^2+\kappa q)\log\frac{1}{\epsilon})$	$\mathcal{O}(\rho d+\Delta(\mathcal{G})d)$	$\mathcal{O}(\Delta(\mathcal{G})d)$
DSBA (this paper)	$\mathcal{O}((\kappa+\kappa_g+q)\log\frac{1}{\epsilon})$	$\mathcal{O}(\rho d+\tau+\Delta(\mathcal{G})d)$	$\mathcal{O}(\Delta(\mathcal{G})d)$
DSBA-sparse (this paper)	$\mathcal{O}((\kappa+\kappa_g+q)\log\frac{1}{\epsilon})$	$\mathcal{O}(\rho d+\tau+N^2d)$	$\mathcal{O}(N\rho d)$

and is $\frac{1}{L}$-cocoercive if

$$\langle\mathcal{B}(\mathbf{x})-\mathcal{B}(\mathbf{y}),\mathbf{x}-\mathbf{y}\rangle\geq\frac{1}{L}\|\mathcal{B}(\mathbf{x})-\mathcal{B}(\mathbf{y})\|^2. \quad (4)$$

The cocoercive property implies the maximality and the Lipschitz continuity of $\mathcal{B}$,

$$\|\mathcal{B}(\mathbf{x})-\mathcal{B}(\mathbf{y})\|\leq L\|\mathbf{x}-\mathbf{y}\|, \quad (5)$$

but not vise versa (Bauschke et al., 2017). However, if $\mathcal{B}$ is both Lipschitz continuous and strongly monotone, it is cocoercive. We denote the identity operator by $\mathcal{I}$ and define the resolvent $\mathcal{J}_{\mathcal{B}}$ of a maximal monotone operator $\mathcal{B}$ as

$$\mathcal{J}_{\mathcal{B}}\triangleq(\mathcal{I}+\mathcal{B})^{-1}. \quad (6)$$

Finding the root of a maximal monotone operator is equivalent to find the fixed point of its resolvent:

$$\mathbf{z}^*=\mathcal{J}_{\mathcal{B}}(\mathbf{z}^*)\Leftrightarrow\mathbf{z}^*+\mathcal{B}(\mathbf{z}^*)=\mathbf{z}^*\Leftrightarrow\mathcal{B}(\mathbf{z}^*)=\mathbf{0}, \quad (7)$$

and when $\mathcal{B}=\nabla f_{n,i}$, $\mathcal{J}_{\mathcal{B}}$ is equivalent to the proximal operator of function f.

3.2. Convex-concave Formulation of AUC Maximization

Area Under the ROC Curve (AUC) (Hanley & McNeil, 1982) is a widely used metric for measuring performance of classification, defined as

$$\sum_{i,j=1}^{q}1\{h(\mathbf{w};\mathbf{a}_i)\geq h((\mathbf{w};\mathbf{a}_j)|y_i=+1,y_j=-1\}, \quad (8)$$

where $\{(\mathbf{a}_j,y_j)\}_{i=j}^q$ is the set of samples, and $h(\cdot)$ is some scoring function. However, directly maximizing AUC is NP-hard as it is equivalent to a combinatorial optimization problem (Gao et al., 2013). Practical implementations take $h(\mathbf{w};\mathbf{a})=\mathbf{a}_i^\top\mathbf{w}$ and replace the discontinuous indicator function 1 with its convex surrogates, e.g. the ℓ_2-loss

$$F(\mathbf{w})=\frac{1}{q^+q^-}\sum_{y_i=+1,y_j=-1}(1-\mathbf{w}^\top(\mathbf{a}_i-\mathbf{a}_j))^2, \quad (9)$$

where q^+ and q^- are the numbers of positive and negative instances. However, $F(\cdot)$ comprises of pairwise losses

$$f_{i,j}(\mathbf{w})=(1-\mathbf{w}^\top(\mathbf{a}_i-\mathbf{a}_j))^2 1\{y_i=+1,y_j=-1\}, \quad (10)$$

each of which depends on two data points. As discussed in (Colin et al., 2016), minimizing (9) in a decentralized manner remains a challenging task.

For $a,b\in\mathbb{R}$, define $\bar{\mathbf{w}}=[\mathbf{w},a,b]\in\mathbb{R}^{d+2}$. (Ying et al., 2016) reformulates the maximization of function (9) as

$$\min_{\bar{\mathbf{w}}\in\mathbb{R}^{d+2}}\max_{\theta\in\mathbb{R}}F(\bar{\mathbf{w}},\theta)=\frac{1}{q}\sum_{i=1}^{q}f(\bar{\mathbf{w}},\theta;\mathbf{a}_i), \quad (11)$$

where, for $p=q^+/q$ the function $f(\bar{\mathbf{w}},\theta;\mathbf{a}_i)$ is given by

$$\begin{aligned}f(\bar{\mathbf{w}},\theta;\mathbf{a}_i)&=-p(1-p)\theta^2+\frac{\lambda}{2}\|\mathbf{w}\|^2 \quad (12)\\&+(1-p)(\mathbf{w}^\top\mathbf{a}_i-a)^2 1_{\{y_i=1\}}+p(\mathbf{w}^\top\mathbf{a}_i-b)^2 1_{\{y_i=-1\}}\\&+2(1+\theta)(p\mathbf{w}^\top\mathbf{a}_i 1_{\{y_i=-1\}}-(1-p)\mathbf{w}^\top\mathbf{a}_i 1_{\{y_i=1\}}).\end{aligned}$$

Such singleton formulation is amenable to decentralized framework because f only depends on a single data point.

4. Problem Formulation

Consider a set of N nodes which create a connected graph $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$ with the node set $\mathcal{V}=\{1,\dots,N\}$ and the edge set $\mathcal{E}=\{(i,j)\mid\text{if } i,j \text{ are connected}\}$. We assume that the edges are reciprocal, i.e., $(i,j)\in\mathcal{E}$ iff $(j,i)\in\mathcal{E}$ and denote $\mathcal{N}_n$ as the neighborhood of node n, i.e. $\mathcal{N}_n=\{m:(m,n)\in\mathcal{E}\}$.

For the decision variable $\mathbf{z}\in\mathbb{R}^d$, consider the problem of finding the root of the operator $\sum_{n=1}^N\mathcal{B}_n(\mathbf{z})$, where the operator $\mathcal{B}_n:\mathbb{R}^d\mapsto\mathbb{R}^d$ is only available at node n and is defined as the sum of q Lipschitz continuous strongly monotone operators $\mathcal{B}_{n,i}:\mathbb{R}^d\mapsto\mathbb{R}^d$.

To handle this problem in a decentralized fashion we define $\mathbf{z}_n$ as the local copy of $\mathbf{z}$ at node n and solve the program

$$\underset{\substack{\{\mathbf{z}_n\}_{n=1}^N\\\mathbf{z}_1=\dots=\mathbf{z}_N}}{\text{find}}\sum_{n=1}^{N}\mathcal{B}_n(\mathbf{z}_n)=\sum_{n=1}^{N}\frac{1}{q}\sum_{i=1}^{q}\mathcal{B}_{n,i}(\mathbf{z}_n)=\mathbf{0}. \quad (13)$$

The finite sum minimization problem (1) is a special case of (13) by setting $\mathcal{B}_{n,i}=\nabla f_{n,i}$, and the ℓ_2-relaxed AUC maximization (11) is captured by choosing $\mathcal{B}_{n,i}(\mathbf{z})=[\frac{\partial f}{\partial\bar{\mathbf{w}}};-\frac{\partial f}{\partial\theta}]$

with $\mathbf{z} = [\bar{\mathbf{w}}; \theta]$. Since $\mathcal{B}_{n,i}$ is strongly monotone and Lipschitz continuous, it is cocoercive (Bauschke et al., 2017).

To have a more concrete understanding of the problem, we first introduce an equivalent formulation of Problem (13). Define the matrix $\mathbf{Z} = [\mathbf{z}_1^\top; \ldots; \mathbf{z}_n^\top] \in \mathbb{R}^{N\times d}$ as the concatenation of the local iterates $\mathbf{z}_n$ and the operator $\mathcal{B}(\mathbf{Z}) : \mathbb{R}^{N\times d} \mapsto \mathbb{R}^{N\times d}$ as $\mathcal{B}(\mathbf{Z}) := [\mathcal{B}_1(\mathbf{z}_1)^\top; \ldots; \mathcal{B}_n(\mathbf{z}_n)^\top]$. Consider the mixing matrix $\mathbf{W} = [w_{m,l}] \in \mathbb{R}^{N\times N}$ satisfying the following conditions, which satisfies

(i) (Graph sparsity) If $m \notin \mathcal{N}_l$, then $w_{m,l} = 0$;

(ii) (Symmetry) $\mathbf{W} = \mathbf{W}^\top$;

(iii) (Null space property) $\mathrm{null}(\mathbf{I} - \mathbf{W}) = \mathrm{span}\{\mathbf{1}_N\}$;

(iv) (Spectral property) $0 \preccurlyeq \mathbf{W} \preccurlyeq \mathbf{I}_N$.

It can be shown that Problem (13) is equivalent to

$$\begin{aligned} &\underset{\mathbf{Z}\in\mathbb{R}^{N\times d}}{\text{find}} \quad \mathcal{B}(\mathbf{Z})^\top \mathbf{1}_N = \mathbf{0}_d \\ &\text{subject to} \quad (\mathbf{I}_N - \mathbf{W})\mathbf{Z} = \mathbf{0}_{N\times d}. \end{aligned} \tag{14}$$

This is true since $\mathrm{null}(\mathbf{I} - \mathbf{W}) = \mathrm{span}(\mathbf{1}_N)$ and therefore the condition $(\mathbf{I} - \mathbf{W})\mathbf{Z} = \mathbf{0}$ implies that a matrix $\mathbf{Z}$ is feasible iff $\mathbf{z}_1 = \ldots = \mathbf{z}_N$.

If we define $\mathbf{U} \triangleq (\mathbf{I} - \mathbf{W})^{1/2}$, the optimality conditions of Problem (14) imply that there exists some $\mathbf{P}^* \in \mathbb{R}^{N\times d}$, such that for $\mathbf{Q}^* = \mathbf{U}\mathbf{P}^*$ and $\alpha > 0$

$$\mathbf{U}\mathbf{Q}^* + \alpha\mathcal{B}(\mathbf{Z}^*) = \mathbf{0} \text{ and } -\mathbf{U}\mathbf{Z}^* = \mathbf{0}, \tag{15}$$

where $\mathbf{Z}^* \in \mathbb{R}^{N\times d}$ is a solution of Problem (14). Note that $\mathrm{span}(\mathbf{I}-\mathbf{W}) = \mathrm{span}(\mathbf{U})$. The first equation of (15) depicts the optimality of $\mathbf{Z}^*$: if $\mathbf{Z}^*$ is a solution, every column of $\mathcal{B}(\mathbf{Z}^*)$ is in $\mathrm{span}\{\mathbf{1}_N\}^\perp = \mathrm{span}(\mathbf{U})$ and hence there exists $\mathbf{P} \in \mathbb{R}^{N\times d}$ such that $\mathbf{U}\mathbf{P} + \alpha\mathcal{B}(\mathbf{Z}^*) = \mathbf{0}$. We can simply take $\mathbf{Q}^* = \mathrm{Proj}_{\mathbf{U}}\mathbf{P}$ which gives $\mathbf{U}\mathbf{Q}^* = \mathbf{U}\mathbf{P}$. The second equation of (15) describes the consensus property of $\mathbf{Z}^*$ and is equivalent to the constraint of Problem (14).

Using (15), we formulate Problem (13) as finding the root of the following operator

$$\mathcal{T}(\mathbf{A}) = \Bigg(\underbrace{\begin{bmatrix}\mathcal{B} & 0\\ 0 & 0\end{bmatrix}}_{\mathcal{T}_1} + \underbrace{\frac{1}{\alpha}\begin{bmatrix}\mathbf{0} & \mathbf{U}\\ -\mathbf{U} & \mathbf{0}\end{bmatrix}}_{\mathcal{T}_2}\Bigg)\underbrace{\begin{bmatrix}\mathbf{Z}\\ \mathbf{Y}\end{bmatrix}}_{\mathbf{A}}, \tag{16}$$

where the augmented variable matrix $\mathbf{A} \in \mathbb{R}^{2N\times d}$ is obtain by concatenating $\mathbf{Z}$ with $\mathbf{Y} \in \mathbb{R}^{N\times d}$. Using the result in (Davis, 2015), it can be shown that $\mathcal{T}$ is a maximally monotone operator, and hence its resolvent $\mathcal{J}_{\mathcal{T}}$ is well defined. Unfortunately, directly implementing the fixed point iteration $\mathbf{A}^{t+1} = \mathcal{J}_{\mathcal{T}}(\mathbf{A}^t)$ requires access to global information which is infeasible in decentralized settings. Inspired by (Wu et al., 2016), we introduce the positive definite matrix

$$\mathbf{D} \triangleq \frac{1}{\alpha}\begin{bmatrix}\mathbf{I} & \mathbf{U}\\ \mathbf{U} & \mathbf{I}\end{bmatrix}, \tag{17}$$

and use the fixed point iteration of the resolvent of $\mathbf{D}^{-1}\mathcal{T}$ to find the root of (16) according to the recursion

$$\mathbf{A}^{t+1} = \mathcal{J}_{\mathbf{D}^{-1}\mathcal{T}}(\mathbf{A}^t). \tag{18}$$

Note that since $\mathbf{D}$ is positive definite, $\mathbf{D}^{-1}\mathcal{T}$ shares the same roots with $\mathcal{T}$, therefore the solutions of the fixed point updates of $\mathcal{J}_{\mathbf{D}^{-1}\mathcal{T}}$ and $\mathcal{J}_{\mathcal{T}}$ are identical.

The main advantage of the recursion in (18) is that it can be implemented with a single round of local communication only. However, (18) is usually computationally expensive to evaluate. For instance, when $\mathcal{B}_{n,i} = \nabla f_{n,i}$, (18) degenerates to the update of P-EXTRA (Shi et al., 2015b), which computes the proximal operator of $f_n = \frac{1}{q}\sum_{i=1}^q f_{n,i}$ in each iteration. The evaluation of such proximal operator is considered computational costly in general, especially for large-scale optimization.

In the following section, we introduce an alternative approach that improves the update in (18) in terms of both computation and communication cost by stochastically approximating $\mathcal{T}$.

5. Decentralized Stochastic Backward Aggregation

In this section, we propose the Decentralized Stochastic Backward Aggregation (DSBA) algorithm for Problem (13). By exploiting the finite sum structure of each $\mathcal{B}_n$ and the sparsity pattern in component operator $\mathcal{B}_{n,i}$, DSBA yields lower per-iteration computation and communication cost.

Let i_n^t be a random sample, approximate $\mathcal{B}_n(\mathbf{z})$ by

$$\hat{\mathcal{B}}_n^t(\mathbf{z}) = \mathcal{B}_{n,i_n^t}(\mathbf{z}) - \phi_{n,i_n^t}^t + \frac{1}{q}\sum_{i=1}^q \phi_{n,i}^t, \tag{19}$$

where $\phi_{n,i}^t = \mathcal{B}_{n,i}(\mathbf{y}_{n,i}^t)$ is the history operator output maintained in the same manner with SAGA (Defazio et al., 2014). We further denote $\bar{\phi}_n^t = \frac{1}{q}\sum_{i=1}^q \phi_{n,i}^t$. Using such definition $\hat{\mathcal{B}}_n^t(\mathbf{z})$, we replace the operators $\mathcal{B}$ and $\mathcal{T}_1$ in (16) by their approximate versions, defined as

$$\hat{\mathcal{B}}^t(\mathbf{Z}) = \begin{bmatrix}\hat{\mathcal{B}}_1^t(\mathbf{z}_1)\\ \vdots \\ \hat{\mathcal{B}}_N^t(\mathbf{z}_N)\end{bmatrix} \text{ and } \hat{\mathcal{T}}_1^t = \begin{bmatrix}\hat{\mathcal{B}}^t & 0\\ 0 & 0\end{bmatrix}. \tag{20}$$

Hence, the fixed point update (18) is changed to

$$\mathbf{A}^{t+1} = (\mathcal{I} + \mathbf{D}^{-1}(\hat{\mathcal{T}}_1^t + \mathcal{T}_2))^{-1}(\mathbf{A}^t), \tag{21}$$

which by plugging in the definitions of $\mathbf{A}$, $\mathbf{D}$, $\hat{\mathcal{T}}_1^t$, and $\mathcal{T}_2$ can be written as

$$\mathbf{Z}^{t+1} + 2\mathbf{U}\mathbf{Y}^{t+1} = \mathbf{Z}^t - \alpha\hat{\mathcal{B}}^t(\mathbf{Z}^{t+1}) + \mathbf{U}\mathbf{Y}^t, \tag{22}$$

$$\mathbf{Y}^{t+1} = \mathbf{U}\mathbf{Z}^t + \mathbf{Y}^t. \tag{23}$$

Algorithm 1 DSBA for node n

Input: consensus initializer $\mathbf{z}^0$, step size α, $\mathbf{W}$, $\tilde{\mathbf{W}}$;
For all $i \in [q]$, initialize $\phi_{n,i}^0 = \mathcal{B}_{n,i}(\mathbf{z}^0)$, set $\delta_n^0 = 0$;
for $t = 0, 1, 2, \ldots$ **do**
  Gather the iterates $\mathbf{z}_m^t$ from neighbors $m \in \mathcal{N}_n$;
  Choose i_n^t uniformly at random from the set $[q]$;
  Update ψ_n^t according to (31) ($t = 0$) or (29) ($t > 0$)
  Compute $\mathbf{z}_n^{t+1}$from (30);
  Compute $\delta_n^t = \mathcal{B}_{n,i_n^t}(\mathbf{z}_n^{t+1}) - \phi_{n,i_n^t}^t$;
  Set $\phi_{i_n^t}^{t+1} = \mathcal{B}_{n,i_n^t}(\mathbf{z}_n^{t+1})$ and $\phi_i^{t+1} = \phi_i^t$ for $i \neq i_n^t$;
end for

Computing the difference between two consecutive iterations of (22) and using (23) lead to the update of the proposed DSBA algorithm, for $t > 1$,

$$\mathbf{Z}^{t+1} \triangleq 2\tilde{\mathbf{W}}\mathbf{Z}^t - \tilde{\mathbf{W}}\mathbf{Z}^{t-1} - \alpha(\hat{\mathcal{B}}^t(\mathbf{Z}^{t+1}) - \hat{\mathcal{B}}^{t-1}(\mathbf{Z}^t)), \quad (24)$$

where $\tilde{\mathbf{W}} = (\mathbf{W} + \mathbf{I})/2 = [\tilde{w}_{m,n}] \in \mathbb{R}^{N \times N}$. By setting $\mathbf{Y}^0 = \mathbf{0}$, the update for step $t = 0$ is given by

$$\mathbf{Z}^1 \triangleq \mathbf{W}\mathbf{Z}^0 - \alpha\hat{\mathcal{B}}^0(\mathbf{Z}^1). \quad (25)$$

Implementation on Node n

We now focus on the detailed implementation on a single node n. The local version of the update (24) writes

$$\mathbf{z}_n^{t+1} \triangleq \sum_{m \in \mathcal{N}_n} \tilde{w}_{n,m}(2\mathbf{z}_m^t - \mathbf{z}_m^{t-1}) - \alpha(\hat{\mathcal{B}}_n^t(\mathbf{z}_n^{t+1}) - \hat{\mathcal{B}}_n^{t-1}(\mathbf{z}_n^t)). \quad (26)$$

This update can be further simplified. Using the definition

$$\delta_n^t \triangleq \mathcal{B}_{n,i_n^t}(\mathbf{z}_n^{t+1}) - \phi_{n,i_n^t}^t, \quad (27)$$

we have $\hat{\mathcal{B}}_n^{t+1} - \hat{\mathcal{B}}_n^t = \delta_n^t - \frac{q-1}{q}\delta_n^{t-1}$, and therefore the update in (26) can be simplified to

$$\mathbf{z}_n^{t+1} \triangleq \sum_{m \in \mathcal{N}_n} \tilde{w}_{n,m}(2\mathbf{z}_m^t - \mathbf{z}_m^{t-1}) + \alpha(\frac{q-1}{q}\delta_n^{t-1} - \delta_n^t). \quad (28)$$

Note that δ_n^t shares the same nonzero pattern as the dataset and is usually sparse. For the initial step $t = 0$, since $\hat{\mathcal{B}}_n^1 = \delta_n^0 + \bar{\phi}_n^0$, we have $\mathbf{z}_n^1 = \sum_{m \in \mathcal{N}_n} w_{n,m}\mathbf{z}_m^0 - \alpha(\delta_n^0 + \bar{\phi}_n^0)$. However, we cannot directly carry out (28) since δ_n^t involves the unknown $\mathbf{z}_n^{t+1}$. To resolve this issue we define for $t \geq 1$

$$\psi_n^t \triangleq \sum_{m \in \mathcal{N}_n} \tilde{w}_{n,m}(2\mathbf{z}_m^t - \mathbf{z}_m^{t-1}) + \alpha(\frac{q-1}{q}\delta_n^{t-1} + \phi_{n,i_n^t}^t). \quad (29)$$

Using (29) and (28), it can be easily verified that $\mathbf{z}_n^{t+1} + \alpha\mathcal{B}_{n,i_n^t}(\mathbf{z}_n^{t+1}) = \psi_n^t$, therefore $\mathbf{z}_n^{t+1}$ can be computed as

$$\mathbf{z}_n^{t+1} = \mathcal{J}_{\alpha\mathcal{B}_{n,i}}(\psi_n^t) = (I + \alpha\mathcal{B}_{n,i_n^t})^{-1}(\psi_n^t). \quad (30)$$

Indeed, the outcome of the updates in (29)-(30) is equivalent to the update in (28), and they can be computed in a decentralized manner. Also, for the initial step $t = 0$, the variable $\mathbf{z}_n^1$ can be computed according to (30) with

$$\psi_n^0 := \sum_{m \in \mathcal{N}_n} w_{n,m}\mathbf{z}_m^0 + \alpha(\phi_{n,i_n^0}^0 - \bar{\phi}_n^0). \quad (31)$$

The resolvent (30) can be obtained by solving a one dimensional equation for learning problems like Logistic Regression, and admits closed form solution for problems like least square and ℓ_2-relaxed AUC maximization. DSBA is summarized in Algorithm 1.

Remark 5.1. *DSBA is related to DSA in the case that $\mathcal{B}_{n,i}$ is the gradient of a function, i.e. $\mathcal{B}_{n,i} = \nabla f_{n,i}$. In each iteration, if we compute δ_n^t with*

$$\delta_n^t = \mathcal{B}_{n,i_n^t}(\mathbf{z}_n^t) - \phi_{n,i_n^t}^t, \quad (32)$$

i.e. we evaluate $\mathcal{B}_{n,i_n^t}$ at the $\mathbf{z}_n^t$ instead of $\mathbf{z}_n^{t+1}$, we recover the DSA method (Mokhtari & Ribeiro, 2016). In such gradient operator setting, when the is only a single node, DSBA degenerates to the Point-SAGA method (Defazio, 2016).

5.1. Implementation with Sparse Communication

In existing decentralized methods, nodes need to compute the weighted averages of their neighbors' iterates, which are dense in general. Therefore a d-dimensional full vector must be transmitted via every edge $(m, l) \in \mathcal{E}$ in each iteration.

In this section, we assume the output of every component operator $\mathcal{B}_{n,i}$ is ρ-sparse, i.e. $nnz(\mathcal{B}_{n,i}(\mathbf{z}))/d \leq \rho$ for all $\mathbf{z} \in \mathbb{R}^d$ and show that DSBA can be implemented by only transmitting the usually sparse vector δ_n^t (27).

WLOG, we take the perspective of node 0 to describe the communication and computation strategies. First, we define the topological distance ξ_i from node i to node 0 by

$$\operatorname*{argmin}_{k \in \mathbb{N}} \text{ s.t. } [W^k]_{0,i} \neq 0, \quad (33)$$

and we have, $[W^k]_{0,i} = 0$ for all node i with distance $\xi_i > k$. Let the diameter of the network be $E = \max_{i \in \mathcal{V}} \xi_i$.

All the communication in the network happens when computing ψ_0^t. For $n = 0$ and we unfold the iteration (29) by the definition of $\mathbf{Z}^t$ in (24),

$$\begin{aligned}\psi_0^t &= 2^E[W^E]_0\mathbf{Z}^{t-E} - \sum_{\tau=1}^{E} 2^{\tau-1}[W^\tau]_0\mathbf{Z}^{t-\tau-1} \\ &\quad + \sum_{\tau=1}^{E} 2^\tau[W^\tau]_0\mathbf{\Delta}^{t-\tau} + \alpha(\frac{1-q}{q}\delta_0^{t-1} + \phi_{0,i_0^t}^t), \\ &= ① + ② + ③ + ④ \end{aligned} \quad (34)$$

where $\mathbf{\Delta}^t = [(\delta_0^t - \delta_0^{t-1})^\top; \ldots; (\delta_{N-1}^t - \delta_{N-1}^{t-1})^\top]$. Suppose that we have a communication strategy that satisfies the following assumption: before evaluating (34), node 0 has the set $\mathcal{Q}_t = \{\delta_n^\tau : \tau + \xi_n \leq t, n \neq 0\}$. ③ can be computed because computing each term of ③, $[W^\tau]_0\mathbf{\Delta}^{t-\tau}$,

only needs $\{\delta_i^{t-\tau} : \xi_i \le \tau\}$. Further, if we can inductively ensure that ① and every term ② are in the memory of node 0 before computing (34), ψ_0^t can be computed since ④ is local information.

In the following, we introduce a communication strategy that satisfies the assumption and show that the inductions on ① and ② holds.

Communication: We group the nodes based on the distance: $\mathcal{V}_j = \{n \in \{0,\dots,N-1\} : \xi_n = j\}$. Define the set $\mathcal{G}_j^t \triangleq \{\delta_n^t : n \in \mathcal{V}_j\}$. Let $\mathcal{F}_E^t \triangleq \{\mathcal{G}_E^t\}$, we recursively define $\mathcal{F}_j^t \triangleq \mathcal{F}_{j+1}^{t-1} \cup \{\mathcal{G}_j^t\}$. Our communication strategy is, in the t^{th} iteration, $\mathcal{V}_j$ sends the set $\mathcal{F}_j^t = \mathcal{F}_{j+1}^{t-1} \cup \{\mathcal{G}_j^t\} = \{\mathcal{G}_i^\tau : i+\tau = t+j, i \ge j\}$ to $\mathcal{V}_{j-1}$. From such strategy, in iteration t, node 0 receives from $\mathcal{V}_j$ the set $\mathcal{F}_1^t = \{\mathcal{G}_i^\tau : i+\tau = t, i \ge 1\}$. Note that if δ_n^τ appears in multiple neighbors of node 0, only the one with the minimum node index sends it to node 0. Since $\mathcal{Q}_t = \cup_{\tau \le t} \mathcal{F}_1^\tau$, the desired set is obtained.

Computation: We now inductively show that ① and ② can be computed. At the beginning of iteration t, assume that $\{[\tilde{\mathbf{W}}^\tau]_0 \mathbf{Z}^{t-\tau}, \tau \in [E]\}$, $\mathbf{Z}^{t-E-1}$, and $\mathbf{Z}^{t-E-2}$ are maintained in memory. According to the above discussion, ψ_0^t can be computed and hence $\mathbf{z}_0^{t+1}$ and δ_0^t can be obtained. To maintain the induction, we compute $\mathbf{Z}^{t-E}$ by its definition (28) where $\{\delta_n^{t-E}, n = 0,\dots,N-1\}$ (by the communication strategy), $\mathbf{Z}^{t-E-1}$, and $\mathbf{Z}^{t-E-2}$ (by induction) are already available to node 0. To obtain $\{[W^\tau]_0 \mathbf{Z}^{t-\tau}, \tau \in [E-1]\}$, we compute $[W^E]_0 \mathbf{Z}^{t-E}$ first and then compute recursively

$$[W^{\tau-1}]_0 \mathbf{Z}^{t-\tau+1} = 2[W^\tau]_0 \mathbf{Z}^{t-\tau} - [W^\tau]_0 \mathbf{Z}^{t-\tau-1} + [W^{\tau-1}]_0 \mathbf{\Delta}^{t-\tau+1}, \quad (35)$$

for $\tau = E,\dots,2$, where the first term is from induction, the second term is in memory, and the last term is computed in ③. We summarize our strategy in Algorithm 2.

As the choice of node 0 is arbitrary, we use the aforementioned communication and computation strategies for all nodes. By induction, if each node n generates δ_n^{t-1} correctly at iteration $t-1$, we can show that ψ_n^t and hence δ_n^t can also be correctly computed in the same manner. The computation complexity at each node is $\mathcal{O}(dN^2)$, dominated by step 1 in Algorithm 2.

The average communication complexity is of $\mathcal{O}(Nd\rho)$. WLOG, use node 0 as a proxy of all nodes. The computation part requires the set $\mathcal{F}_1^t$ to be received by node 0 at time t. Removing the duplicate, we have $|\mathcal{F}_1^t| \le N$. Hence, the number of DOUBLEs received by node 0 is of $\mathcal{O}(Nd\rho)$. Further, since that of data sent by all nodes equals to the amount of data received by all nodes, we have the result.

The local storage requirement of DSBA is $O(qd\rho + Nd)$. Aside from the $O(\rho qd)$ storage for the dataset, a node stores a delayed copy of other nodes which costs a memory of $\mathcal{O}(Nd)$, and due to the use of linear predictor the cost of storing gradient information at each node is $\mathcal{O}(q)$, (Schmidt et al., 2017). Hence, the overall required storage is $\mathcal{O}(qd\rho + Nd + q)$. As $\rho \cdot d$ is the number of nonzero elements in the vector it follows that $\rho \cdot d >= 1$ and hence $\mathcal{O}(q) \le \mathcal{O}(qd\rho)$. Further, if we assume every sample has more than N nonzero entries, $\mathcal{O}(qd\rho)$ dominates $\mathcal{O}(Nd)$ as well, we need a memory of $\mathcal{O}(qd\rho)$.

Algorithm 2 Computation on node 0 at iteration t

Require: $\{[\tilde{\mathbf{W}}^\tau]_0 \mathbf{Z}^{t-\tau}, \tau \in [E]\}, \mathbf{Z}^{t-E-1}, \mathbf{Z}^{t-E-2}$
1: Compute $\mathbf{Z}^{t-E}$ from its definition;
2: Compute ψ_0^t from (34) and $\mathbf{z}_0^{t+1}$ from (30);
3: $\delta_0^t = B_{0,i_0^t}(\mathbf{z}_0^{t+1}) - \phi_{0,i_0^t}^t$, update the gradient table;
4: Compute $\{[\tilde{\mathbf{W}}^\tau]_0 \mathbf{Z}^{t-\tau+1}, \tau \in [E]\}$ from (35);
Ensure: $\mathbf{Z}^{t-E}, \mathbf{z}_0^{t+1}, \delta_0^t, \{[\tilde{\mathbf{W}}^\tau]_0 \mathbf{Z}^{t-\tau+1}, \tau \in [E]\}$

6. Convergence Analysis

In this section, we study the convergence properties of the proposed DSBA method. To achieve this goal, we define a proper Lyapunov function for DSBA and prove its linear convergence to zero which leads to linear convergence of the iterates $\mathbf{z}_n^t$ to the optimal solution $\mathbf{z}^*$. To do so, first we define $\mathbf{M}$ and the sequence of matrices $\mathbf{Q}^t$ and $\mathbf{X}^t$ as

$$\mathbf{M} \triangleq \begin{bmatrix} \tilde{\mathbf{W}} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix}, \quad \mathbf{Q}^t \triangleq \sum_{k=0}^{t} \mathbf{U}\mathbf{Z}^k, \quad \mathbf{X}^t \triangleq \begin{bmatrix} \mathbf{Z}^t \\ \mathbf{U}\mathbf{Q}^t \end{bmatrix}. \quad (36)$$

Recall the definition of $\mathbf{Q}^*$ in (15) and define $\mathbf{X}^*$ as the concatenation of $\mathbf{Z}^*$ and $\mathbf{U}\mathbf{Q}^*$, i.e., $\mathbf{X}^* = [\mathbf{Z}^*; \mathbf{U}\mathbf{Q}^*]$.

Lemma 6.1. *Consider the proposed DSBA method defined in Algorithm 1. By incorporating the definitions of the matrices $\mathbf{Q}^t$ and $\mathbf{X}^t$ in (36), it can be shown that*

$$\alpha[\hat{\mathcal{B}}^t(\mathbf{Z}^{t+1}) - \mathcal{B}(\mathbf{Z}^*)] = \tilde{\mathbf{W}}(\mathbf{Z}^t - \mathbf{Z}^{t+1}) - \mathbf{U}(\mathbf{Q}^{t+1} - \mathbf{Q}^*), \quad (37)$$

and

$$2\langle \mathbf{Z}^{t+1} - \mathbf{Z}^*, \alpha[\mathcal{B}(\mathbf{Z}^*) - \hat{\mathcal{B}}^t(\mathbf{Z}^{t+1})] \rangle \quad (38)$$
$$= \|\mathbf{X}^{t+1} - \mathbf{X}^*\|_{\mathbf{M}}^2 + \|\mathbf{X}^{t+1} - \mathbf{X}^t\|_{\mathbf{M}}^2 - \|\mathbf{X}^t - \mathbf{X}^*\|_{\mathbf{M}}^2.$$

Proof. See Section 9.1 in the supplementary material. □

The result in Lemma 6.1 shows the relation between the norm $\|\mathbf{X}^{t+1} - \mathbf{X}^*\|_{\mathbf{M}}^2$ and its previous iterate $\|\mathbf{X}^t - \mathbf{X}^*\|_{\mathbf{M}}^2$. Therefore, to analyze the speed of convergence for $\|\mathbf{X}^t - \mathbf{X}^*\|_{\mathbf{M}}^2$ we first need to derive bounds for the remaining terms in (38). To do so, we need to define a few more terms. By selecting component operator i on node n in the t^{th} iteration, we define $\mathbf{z}_{n,i}^{t+1}$ for all $i \in [q]$ as

$$\mathbf{z}_{n,i}^{t+1} \triangleq \sum_{m \in \mathcal{N}_n} \tilde{w}_{n,m}(2\mathbf{z}_m^t - \mathbf{z}_m^{t-1}) - \alpha(\hat{\mathcal{B}}_n^t(\mathbf{z}_{n,i}^{t+1}) - \hat{\mathcal{B}}^{t-1}(\mathbf{z}_n^t)).$$

Computing $\mathbf{z}_{n,i}^{t+1}$ requires to evaluate the resolvent of $\mathcal{B}_{n,i}$, but here we only define it for the analysis. In the actual procedure, we only select $i = i_n^t$, compute $\mathbf{z}_{n,i_n^t}^{t+1}$, and set $\mathbf{z}_n^{t+1} = \mathbf{z}_{n,i_n^t}^{t+1}$. Having such definition, we define two nonnegative sequences that are crucial to our analysis:

$$S^t \triangleq \sum_{n=1}^{N} \frac{2}{q} \sum_{i=1}^{q} \|\mathcal{B}_{n,i}(\mathbf{z}_{n,i}^t) - \mathcal{B}_{n,i}(\mathbf{z}^*)\|^2, \tag{39}$$

$$T^t \triangleq \sum_{n=1}^{N} \frac{2}{q} \sum_{i=1}^{q} \langle \mathbf{z}_{n,i}^t - \mathbf{z}^*, \mathcal{B}_{n,i}(\mathbf{z}_{n,i}^t) - \mathcal{B}_{n,i}(\mathbf{z}^*) \rangle, \tag{40}$$

where the nonnegativity of the sequence T^t is due to the monotonicity of each component operator $\mathcal{B}_{n,i}$. Define D^t as the component-wise discrepancy between the historically evaluated stochastic gradients and gradients at the optimum

$$D^t = \sum_{n=1}^{N} \frac{2}{q} \sum_{i=1}^{q} \|\mathcal{B}_{n,i}(\mathbf{y}_{n,i}^t) - \mathcal{B}_{n,i}(\mathbf{z}^*)\|^2, \tag{41}$$

where $\mathcal{B}_{n,i}(\mathbf{y}_{n,i}^t)$ is maintained by the SAGA strategy. In the following lemma, we derive an upper bound on the expected inner product $\mathbb{E}\langle \mathbf{Z}^{t+1} - \mathbf{Z}^*, \mathcal{B}(\mathbf{Z}^*) - \hat{\mathcal{B}}^t(\mathbf{Z}^{t+1}) \rangle$.

Lemma 6.2. *Consider the proposed DSBA method defined in Algorithm 1. Further, recall the defintions of the sequences S^t, T^t, and D^t in (39), (40), and (41), respectively. If each operator $\mathcal{B}_{n,i}$ is $(1/L)$-cocoercive, it holds for any $0 \leq \theta \leq 1$ and $\eta > 0$ that*

$$\begin{aligned} &\mathbb{E}\langle \mathbf{Z}^{t+1} - \mathbf{Z}^*, \mathcal{B}(\mathbf{Z}^*) - \hat{\mathcal{B}}^t(\mathbf{Z}^{t+1}) \rangle \qquad (42) \\ \leq &\frac{1}{2\eta} \mathbb{E}\|\mathbf{Z}^{t+1} - \mathbf{Z}^t\|^2 + \frac{\eta}{4} D^t - \frac{\theta}{2L} S^{t+1} - \frac{1-\theta}{2} T^{t+1}. \end{aligned}$$

Proof. See Section 9.2 in the supplementary material. □

The next lemma bounds the discrepancy between the average of the historically evaluated stochastic gradients and the gradients at the optimal point.

Lemma 6.3. *Consider the DSBA method outlined in Algorithm 1. From the construction of $\hat{\mathcal{B}}^t$ and the definitions of S^t and D^t in (39) and (41), respectively, we have for $t \geq 0$,*

$$\mathbb{E}\|\hat{\mathcal{B}}^t(\mathbf{Z}^{t+1}) - \mathcal{B}(\mathbf{Z}^*)\|^2 \leq S^{t+1} + D^t. \tag{43}$$

Proof. See Section 9.3 in the supplementary material. □

Lemma 6.4. *Consider the update rule of Algorithm 1 and the definition $\tilde{\mathbf{W}} = (\mathbf{I} + \mathbf{W})/2$. Further, recall the definitions of the sequences S^t, T^t, and D^t in (39), (40), and (41), respectively. If each component operator $\mathcal{B}_{n,i}$ is μ-strongly monotone, $\mathbb{E}\|\mathbf{X}^t - \mathbf{X}^*\|_M^2$ is upper bounded by*

$$\begin{aligned} \mathbb{E}\|\mathbf{X}^t - \mathbf{X}^*\|_M^2 \leq &(2 + \frac{4}{\gamma}) \mathbb{E}\|\mathbf{Z}^{t+1} - \mathbf{Z}^t\|_{\tilde{\mathbf{W}}}^2 + \frac{1}{\mu} T^{t+1} \\ &+ 2\mathbb{E}\|\mathbf{Q}^{t+1} - \mathbf{Q}^t\|^2 + \frac{4\alpha^2}{\gamma}(S^{t+1} + D^t). \end{aligned} \tag{44}$$

Proof. See Section 9.4 in the supplementary material. □

Having the above lemmas, we now are ready to state the main theorem. We proceed to show that the Lyapunov function H^t defined as

$$H^t := \|\mathbf{X}^t - \mathbf{X}^*\|_{\mathbf{M}}^2 + cD^t \tag{45}$$

converges to zero linearly, where c is a positive constant formally defined in the following theorem.

Theorem 6.1. *Consider the proposed DSBA method defined in Algorithm 1. If each component operator $\mathcal{B}_{n,i}$ is $1/L$-cocoersive and μ-strongly monotone, by taking the step size $\alpha \leq \frac{1}{24L}$ and $c = \frac{q}{96L^2}$, it holds that*

$$\mathbb{E}[H^{t+1}] \leq \left(1 - \min\left\{\frac{\gamma}{12}, \frac{\mu}{48L}, \frac{1}{3q}, \frac{1}{4}\right\}\right) \mathbb{E}[H^t]. \tag{46}$$

Proof. See Section 9.5 in the supplementary material. □

The result in (46) indicates that the Lyapunov function H^t converges to zero Q-linearly in expectation where the coefficient of the linear convergence is a function of graph condition number $\kappa_g \triangleq 1/\gamma$, operator condition number $\kappa \triangleq L/\mu$, and number of samples at each node q. Indeed, using the definition of H^t in (45), the result in Theorem 6.1 implies R-linear convergence of $\mathbb{E}[\|\mathbf{Z}^t - \mathbf{Z}^*\|^2]$ to zero, i.e.,

$$\mathbb{E}[\|\mathbf{Z}^t - \mathbf{Z}^*\|_{\tilde{\mathbf{W}}}^2] \leq \delta^t(\|\mathbf{Z}^0 - \mathbf{Z}^*\|_{\tilde{\mathbf{W}}}^2 + \|\mathbf{Q}^0 - \mathbf{Q}^*\|^2 + cD^0),$$

where $\delta := 1 - \min\{\frac{\gamma}{12}, \frac{\mu}{48L}, \frac{1}{3q}, \frac{1}{4}\}$. Note that this result indicates that to obtain an ϵ accurate solution the number of required iterations is of $\mathcal{O}(\kappa_g + \kappa + q)\log(1/\epsilon)$.

7. Numerical Experiments

In this section, we evaluate the empirical performance of DSBA and compare it with several state-of-the-art methods including: DSA, EXTRA, SSDA, and DLM. (Colin et al., 2016) is excluded in comparison since it is sublinear convergent. Additionally, DSA is implemented using the sparse communication technique developed in Section 5.2.

In all experiments, we set $N = 10$ and generate the edges with probability 0.4. As to dataset, we use News20-binary, RCV1, and Sector from LIBSVM dataset and randomly split the them into N partitions with equal sizes. Further we normalize each data point $\mathbf{a}_{n,i}$ such that $\|\mathbf{a}_{n,i}\| = 1$. We tune the step size of all algorithms and select the ones that give the best performance. We set $\mathbf{W}$ to be the Laplacian-based constant edge weight matrix: $\mathbf{W} = \mathbf{I} - \frac{\mathbf{L}}{\tau}$, where $\mathbf{L}$ is the Laplacian and $\tau \geq (\lambda_{max}(\mathbf{L}))/2$ is a scaling parameter.

We use the *effective pass* over the dataset to measure the cost of computation, which is a common practice in stochastic optimization literature (Johnson & Zhang, 2013; Defazio et al., 2014) and is also the one adopted in DSA (Mokhtari & Ribeiro, 2016). To measure the cost of communication, we let $\mathcal{C}_n^t$ be the number of DOUBLEs received by node

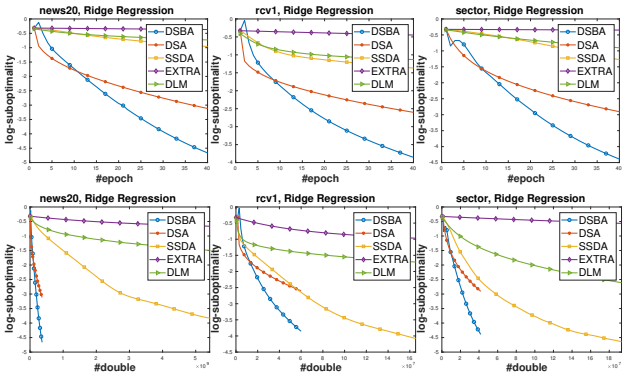

Figure 1. Ridge Regression

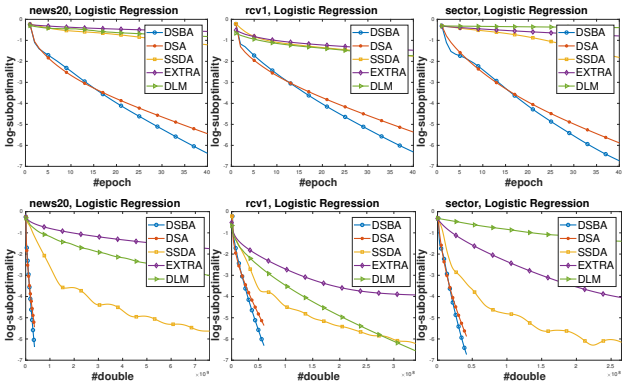

Figure 2. Logistic Regression

n until iterate t and use $\mathcal{C}^t_{max} = \max_n \mathcal{C}^t_n$ as our metric. Such value $\mathcal{C}^t_{max}$ captures the communication traffic on the hottest node in the network, which usually is the bottleneck of the learning procedure.

To avoid overfitting and to ensure the strongly monotonicity of an operator $\mathcal{B}$, we add an ℓ_2 regularization to all experiments. Let $\mathcal{B}^\lambda = \mathcal{B} + \lambda\mathcal{I}$, then the resolvent of $\mathcal{B}^\lambda$ is closely related to that of $\mathcal{B}$, $\mathcal{J}_{\alpha\mathcal{B}^\lambda}(\mathbf{Z}) = \mathcal{J}_{\rho\alpha\mathcal{B}}(\rho\mathbf{Z})$, where $\rho = 1 - (\lambda\alpha)/(1+\lambda\alpha)$ is a scaling factor. The ℓ_2-regularization parameter λ is set to $1/(10Q)$ in all cases.

7.1. Ridge Regression

We define $\mathcal{B}_{n,i} = (\mathbf{a}_{n,i}^\top \mathbf{z} - y_{n,i})\mathbf{a}_{n,i}$, where $\mathbf{a}_{n,i} \in \mathbb{R}^d$ is the feature vector of a sample in node n and $y_{n,i} \in \mathbb{R}$ is its response. The resolvent of $\alpha\mathcal{B}_{n,i}$ admits closed form solution: let $z = \frac{\alpha y_{n,i} + \mathbf{a}_{n,i}^\top \mathbf{z}}{\alpha+1} \in \mathbb{R}$, then $\mathcal{J}_{\alpha\mathcal{B}_{n,i}}(\mathbf{z}) = \mathbf{z} - \alpha(z - y_{n,i})\mathbf{a}_{n,i}$. The results are given in Figure 1. It can be seen that the stochastic methods (DSA and DSBA) have the better performance of the deterministic ones. And DSBA always outperform DSA after several iterations.

7.2. Logistic Regression

We define $\mathcal{B}_{n,i}(\mathbf{z}) = \frac{-y_{n,i}}{1+\exp(y_{n,i}\cdot\mathbf{a}_{n,i}^\top \mathbf{z})}\mathbf{a}_{n,i}$, where $\mathbf{a}_{n,i} \in \mathbb{R}^d$ is the feature vector of a sample and $y_{n,i} \in \{-1, +1\}$ is its class label. The resolvent $\mathcal{J}_{\alpha\mathcal{B}_{n,i}}(\mathbf{z})$ does not admit a closed form solution, but can be computed efficiently using a one dimensional newton iteration. The details are given in the appendix. We list the experiment results in Figure 2. DSBA has the best performance among all the compared methods, and is able to converge quickly with low communication cost.

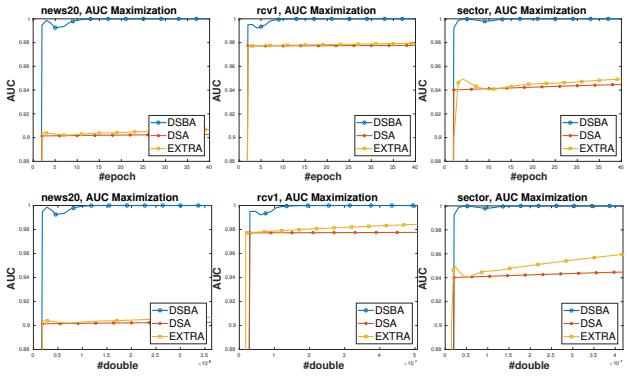

Figure 3. ℓ_2-relaxed AUC maximization

7.3. AUC maximization

In the ℓ_2-relaxed AUC maximization, we only compare with DSA and EXTRA because SSDA does not apply and DLM does not converge. The variable $\mathbf{z} \in \mathbb{R}^{d+3}$ is a $(d+3)$-dimensional augmented vector, where d is the dimension of the dataset. The component monotone operator $\mathcal{B}_{n,i}$ is defined in (75) and (76) in the appendix for positive and negative samples respectively. Similar to Ridge Regression, the resolvent of $\mathcal{B}_{n,i}$ also admits a closed form solution, which is explicitly given in the appendix. The results are given in Figure 3, where DSBA quickly achieves high AUC after a few epochs over the dataset.

8. Conclusion

In this paper, we studied the root finding problem of a monotone operator in a decentralized setting. At a low computation cost, a stochastic algorithm named DSBA is proposed to solve such problem with provably better convergence rate. By exploiting the dataset sparsity, a sparse communication scheme for implementing DBSA is derived to reduce the communication overhead. Our theoretical and numerical results demonstrate the superiority of DSBA over stat-of-the-art deterministic and stochastic decentralized methods.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No: 61472347, 61672376, 61751209), and Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ18F020002.

References

Bauschke, H. H., Combettes, P. L., et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 2011. Springer, 2017.

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

Bullo, F., Cortes, J., and Martinez, S. *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

Colin, I., Bellet, A., Salmon, J., and Clémençon, S. Gossip dual averaging for decentralized optimization of pairwise functions. In *International Conference on Machine Learning*, pp. 1388–1396, 2016.

Davis, D. Convergence rate analysis of primal-dual splitting schemes. *SIAM Journal on Optimization*, 25(3):1912–1943, 2015.

Defazio, A. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pp. 676–684, 2016.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pp. 1646–1654, 2014.

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

Eisen, M., Mokhtari, A., and Ribeiro, A. Decentralized quasi-Newton methods. *IEEE Transactions on Signal Processing*, 65(10):2613–2628, 2017.

Forero, P. A., Cano, A., and Giannakis, G. B. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11(May):1663–1707, 2010.

Gao, W., Jin, R., Zhu, S., and Zhou, Z.-H. One-pass auc optimization. In *International Conference on Machine Learning*, pp. 906–914, 2013.

Hanley, J. A. and McNeil, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Johansson, B. *On distributed optimization in networked systems*. PhD thesis, KTH, 2008.

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

Ling, Q., Shi, W., Wu, G., and Ribeiro, A. DLM: Decentralized linearized alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(15):4051–4064, 2015.

Mokhtari, A. and Ribeiro, A. DSA: Decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 17(61):1–35, 2016.

Mokhtari, A., Shi, W., Ling, Q., and Ribeiro, A. DQM: Decentralized quadratically approximated alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 64(19):5158–5173, 2016.

Mokhtari, A., Ling, Q., and Ribeiro, A. Network Newton distributed optimization methods. *IEEE Transactions on Signal Processing*, 65(1):146–161, 2017.

Necoara, I., Nesterov, Y., and Glineur, F. Random block coordinate descent methods for linearly constrained optimization over networks. *Journal of Optimization Theory and Applications*, 173(1):227–254, 2017.

Nedic, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Ribeiro, A. Ergodic stochastic optimization algorithms for wireless communication and networking. *IEEE Transactions on Signal Processing*, 58(12):6369–6386, 2010.

Rockafellar, R. et al. On the maximal monotonicity of subdifferential mappings. *Pacific J. Math*, 33(1):209–216, 1970.

Rockafellar, R. T. Monotone operators associated with saddle-functions and minimax problems. *Nonlinear functional analysis*, 18(part 1):397–407, 1970.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pp. 3027–3036, 2017.

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

Shi, W., Ling, Q., Yuan, K., Wu, G., and Yin, W. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7): 1750–1761, 2014.

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

Shi, W., Ling, Q., Wu, G., and Yin, W. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015b.

Tsitsiklis, J., Bertsekas, D., and Athans, M. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

Wu, T., Yuan, K., Ling, Q., Yin, W., and Sayed, A. H. Decentralized consensus optimization with asynchrony and delays. In *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pp. 992–996. IEEE, 2016.

Ying, Y., Wen, L., and Lyu, S. Stochastic online auc maximization. In *Advances in neural information processing systems*, pp. 451–459, 2016.

Yuan, K., Ling, Q., and Yin, W. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.