ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096994

COMPARING DECENTRALIZED GRADIENT DESCENT APPROACHES AND GUARANTEES

Shana Moothedath and Namrata Vaswani

Iowa State University, Ames, IA, USA

ABSTRACT

This work studies our recently developed decentralized algorithm, decentralized alternating projected gradient descent algorithm, called Dec-AltProjGDmin, for solving the following low-rank (LR) matrix recovery problem: recover an LR matrix from independent column-wise linear projections (LR column-wise Compressive Sensing). In recent work, we presented constructive convergence guarantees for Dec-AltProjGDmin under simple assumptions. By "constructive", we mean that the convergence time lower bound is provided for achieving any error level ϵ. However, our guarantee was stated for the equal neighbor consensus algorithm (at each iteration, each node computes the average of the data of all its neighbors) while most existing results do not assume the use of a specific consensus algorithm, but instead state guarantees in terms of the weights matrix eigenvalues. In order to compare with these results, we first modify our result to be in this form. Our second and main contribution is a theoretical and experimental comparison of our new result with the best existing one from the decentralized GD literature that also provides a convergence time bound for values of ϵ that are large enough. The existing guarantee is for a different problem setting and holds under different assumptions than ours and hence the comparison is not very clear cut. However, we are not aware of any other provably correct algorithms for decentralized LR matrix recovery in any other settings either.

***Index Terms*—** Low Rank matrix recovery, compressed sensing, decentralized algorithms

1. INTRODUCTION

This work studies our recently developed decentralized algorithm, decentralized alternating projected gradient descent algorithm (Dec-AltProjGDmin) [1], for solving the following low-rank (LR) matrix recovery problem: recover an LR matrix from independent column-wise linear projections (LR column-wise Compressive Sensing (LR-cCS)) [2, 3, 4]. One application of our setting is in federated sketching where the goal is to reconstruct the data (images or videos) from their compressed signals acquired at geographically distributed nodes (mobile phones or IoT devices). Typically there is no central server and the agents/sensors communicate among themselves for reconstructing the data. In recent work [5, 6], we presented a constructive convergence guarantee for Dec-AltProjGDmin under simple assumptions. By "constructive", we mean that the convergence time lower bound is provided for achieving any error level ϵ. One important application where this problem occurs is in decentralized federated sketching. Sketching refers to lossy data compression where the compression step is a very fast operation, typically a linear projection, but the decompression can be more complex.

Our guarantee from [5, 6] was stated for the equal neighbor consensus algorithm (at each iteration, each node computes the average of the data of all its neighbors) while most existing results do not assume use of a specific consensus algorithm, but instead state guarantees in terms of the weights' (mixing) matrix eigenvalues. In order to compare with these results, we first modify our result to also be in this form. Our second and main contribution is a theoretical and experimental comparison of our new result with the best existing one from the decentralized GD literature that also provides a convergence time bound for values of ϵ that are large enough. We also provide experimental comparisons with using the well known Decentralized GD (DecGD) algorithm [7] for our problem and explain why it does not work.

The guarantee for DecGD is for a different problem setting and hold under different assumptions than ours and hence the comparison is not very clear cut. However, we are not aware of any other provably correct algorithms for decentralized LR matrix recovery in any other settings either. To the best of our knowledge, there is only one work on non-convex optimization that provides a randomized algorithm with a non-asymptotic guarantee: the work of [20] (given below). The setting in [20] is very different than ours and it requires the Polyak Łojasiewicz (PL) condition to hold.

1.1. Problem setting and notation

LRcCS problem aims to recover a set of q n-dimensional vectors/signals $\mathbf{X}^\star := [\mathbf{x}_1^\star, \mathbf{x}_2^\star, \ldots, \mathbf{x}_q^\star]$ such that the $n \times q$ matrix $\mathbf{X}^\star$ has rank $r \ll \min(n,q)$, from m-length projections $\mathbf{y}_k$ given by

$$\mathbf{y}_k := \mathbf{A}_k \, \mathbf{x}_k^\star, \; k = 1, 2, \ldots, q. \tag{1}$$

The $m \times n$ matrices $\mathbf{A}_k$ are known and mutually independent for different k. We consider $\mathbf{y}_k$'s are low-dimensional and hence $m < n$ and the goal is to have to use as few number of samples m as possible. The total sample complexity is mq.

We assume that there is a set of L distributed nodes/sensors, each of which obtains sketches (linear projections) of a disjoint subset of columns of $\mathbf{X}^\star$. We denote the set of columns sketched at node g by $\mathcal{S}_g$. The sets $\mathcal{S}_g$ form a partition of $[q] := \{1, 2, \ldots, q\}$, i.e., they are mutually disjoint and $\cup_{g=1}^{L} \mathcal{S}_g = [q]$. The communication network is specified by an undirected graph $\mathcal{G} = (V, E)$, where V denotes the set of nodes, with $|V| = L$, and E denotes the set of undirected edges. The neighbor set of the g^{th} node (sensor) is defined as by $\mathcal{N}_g := \{j : (g, j) \in E\}$. We consider a decentralized setting where there is no central coordinating node, each node of the network can only communicate with its neighboring nodes.

Let us denote the reduced (rank r) Singular Value Decomposition (SVD) of the rank-r matrix $\mathbf{X}^\star$ as $\mathbf{X}^\star \overset{\text{SVD}}{=} \mathbf{U}^\star \, \Sigma^\star \, \mathbf{V}^{\star\top}$. Let κ be the condition number of $\Sigma^\star$. We define $\mathbf{B} := \mathbf{V}^\top$ and $\tilde{\mathbf{B}} := \Sigma \mathbf{V}^\top$. Thus $\mathbf{X}^\star \overset{\text{SVD}}{=} \mathbf{U}^\star \, \Sigma^\star \, \mathbf{B}^\star = \mathbf{U}^\star \, \tilde{\mathbf{B}}^\star$. For a matrix $\tilde{\mathbf{Z}}$, we use $\mathbf{Z}$ to denote orthonormal basis. When computed using QR decomposition, $\tilde{\mathbf{Z}} \overset{\text{QR}}{=} \boldsymbol{Z}\boldsymbol{R}$. We denote the Frobenius norm as $\|\cdot\|_F$ and the induced ℓ_2 norm as $\|\cdot\|$. We use $\mathbf{e}_k$ to denote the k^{th} canonical basis vector and $h \in [d]$ for $h \in \{1, 2, \ldots, d\}$ for some integer d. We define the Subspace Distance (SD) measure between two matrices $\mathbf{U}_1$ and $\mathbf{U}_2$

as $SD(\mathbf{U}_1,\mathbf{U}_2) := \left\|(I-\mathbf{U}_1\mathbf{U}_1^\top)\mathbf{U}_2\right\|_F$, where I is the identity matrix. Further, $^\top$ denotes matrix or vector transpose and $|\boldsymbol{z}|$ for a vector $\boldsymbol{z}$ denotes element-wise absolute values. We use $\mathbb{1}_{\text{statement}}$ to denote an indicator function that takes the value 1 if statement is true and zero otherwise. We use $\circ$ to denote component-wise multiplication (Hadamard product). We reuse c, C to denote different numerical constants in each use with $c < 1$ and $C \geqslant 1$.

Assumption 1 (Right singular vectors' incoherence)**.** *Assume that* $\max_k \|\mathbf{x}_k^\star\| = \max_k \|\mathbf{b}_k^\star\| \leq \sigma_{\max}^\star \mu\sqrt{r/q}$ *for a constant* $\mu \geq 1$ *(μ does not grow with n, q, r). This further implies that* $\max_k \|\mathbf{x}_k^\star\| \leq \kappa\mu\|\mathbf{X}^\star\|_F/\sqrt{q}$.

1.2. Related work and contributions

Related work: The LRcCS problem has been studied in the centralized setting and a GD-based solution was proposed in [3]. Recently, LRcCS has been studied in the decentralized setting in our prior works [5, 1, 6]. In [1] we presented the Dec-AltProjGDmin algorithm and an empirical validation using generated data. In [5, 6], we presented the theoretical guarantee and and its proof. The algorithm and guarantee given in papers [5, 1, 6] are for an equal neighbor model communication network [8].

Projected GD is a GD-based solution approach for solving constrained optimization problems. It involves projecting the output of each GD step onto the constraint set. In the last decade, the design of decentralized GD and projected GD algorithms has received a lot of attention [7, 9, 10, 11, 12, 13, 14, 15], starting with the seminal work of Nedić *et al.* [7]. There is also some recent work on projected GD for constrained optimization problems [9, 12, 13] or for imposing the consensus constraint [12]. However, all existing approaches that come with guarantees assume convex cost functions and either no constraints or convex constraint sets. The works of [9, 13] study projected GD approaches to solve a decentralized convex optimization with convex constraint sets. Both use projection onto convex sets to impose the constraint after each GD iteration. The work of [12] considers the unconstrained optimization problem, and uses projection onto an appropriately defined subspace to impose the consensus constraint at each algorithm iteration.

Contributions: In this paper we focus on comparing our fully-decentralized GD algorithm (Dec-AltProjGDmin) with the decentralized GD algorithm in [15], both theorems and numerical experiments. The theoretical guarantees presented in this paper use fundamental theorem of calculus [16], sub-exponential Bernstein inequality [17], Markov chain convergence results [18], average consensus algorithms and results [8], and analysis of perturbed QR decomposition [19].

2. THE PROPOSED ALGORITHM AND GUARANTEE

2.1. Dec-AltProjGDmin: Decentralized Alternating Projected GD and minimization

Pseudocode of Dec-AltProjGDmin is presented in Algorithm 1. There are two key steps. We decompose $\mathbf{X} = \mathbf{UB}$ and define

$$f(\mathbf{U},\mathbf{B}) = \sum_{g=1}^{L} f_g(\mathbf{U},\mathbf{B}), \text{ where } f_g(\mathbf{U},\mathbf{B}) = \sum_{k\in\mathcal{S}_g} \|\mathbf{y}_k - \mathbf{A}_k\mathbf{U}\mathbf{b}_k\|^2. \quad (2)$$

At each GD iteration, for each new estimate of $\mathbf{U}$, we first solve for $\mathbf{B}$ by minimizing $f(\mathbf{U},\mathbf{B})$ over it while keeping $\mathbf{U}$ fixed at its current value. Then we compute $\tilde{\mathbf{U}} = \mathbf{U} - \eta\nabla_U f(\mathbf{U},\mathbf{B})$ and orthonormalize it using QR decomposition as $\mathbf{U} = \text{QR}(\tilde{\mathbf{U}})$. Here $\nabla_U f(\mathbf{U},\mathbf{B}) = \sum_{k=1}^{q} \mathbf{A}_k^\top(\mathbf{A}_k\mathbf{U}\mathbf{b}_k - \mathbf{y}_k)\mathbf{b}_k^\top$.

To initialize, we compute $\mathbf{U}_0$ as the top r left singular vectors of $\mathbf{X}_0 = (1/m)\sum_{k\in[q]} \mathbf{A}_k^\top \mathbf{y}_{k,\text{trunc}}(\alpha) e_k^\top$ with $\alpha := \tilde{C}\frac{\sum_{ki}(\mathbf{y}_{ki})^2}{mq}$ and $\mathbf{y}_{k,\text{trunc}}(\alpha) := \mathbf{y}_k \circ \mathbb{1}_{\{\mathbf{y}_{ki}^2 \leqslant \alpha\}}$. This is $\mathbf{y}_k$ with large magnitude entries zeroed out. To compute $\mathbf{X}_0$, we use an average consensus algorithm. Let T_{con} denote the number of iterations of the consensus algorithm and $\mathbf{Z}_{\text{in}}^{(g)}$, for $g \in [L]$, be the input. For $\mathbf{Z}_0^{(g)} := \mathbf{Z}_{\text{in}}^{(g)}$ average consensus updates as

$$\mathbf{Z}_{t+1}^{(g)} = \mathbf{Z}_t^{(g)} + \sum_{j\in\mathcal{N}_g} \mathbf{W}_{gj}\left(\mathbf{Z}_t^{(j)} - \mathbf{Z}_t^{(g)}\right), \text{ for } g \in [L] \quad (3)$$

and outputs $\text{AvgCon}_{(g)}(\{\mathbf{Z}_{\text{in}}^{(g)}\}_{g=1}^L, \mathcal{G}, T_{\text{con}}) := \mathbf{Z}_{\text{out}}^{(g)} = L\cdot\mathbf{Z}_{T_{\text{con}}}^{(g)}$, for all $g \in [L]$. Here the mixing matrix $\mathbf{W}$ is symmetric and doubly stochastic. In Algorithm 1, we use average consensus at three places. In the initialization step, to approximate the threshold α and the top r singular vectors of $\mathbf{X}_0$ (which are equal to those of $\mathbf{X}_0\mathbf{X}_0^\top$) computed using the PM, and in the ProjGD iterations to approximate the sum of the individual gradients at all the nodes, i.e. compute $\nabla_U f(\mathbf{U},\mathbf{B}) = \sum_{k\in[q]} \mathbf{A}_k^\top(\mathbf{A}_k\mathbf{U}(\mathbf{b}_k) - \mathbf{y}_k)(\mathbf{b}_k)^\top$.

2.2. Convergence, sample, time, & communication complexities

We first present the convergence result for the consensus update and then present the main result.

Proposition 2.1. *Consider the average-consensus update in* (3) *and let the mixing matrix* $\mathbf{W}$ *be symmetric and doubly stochastic with eigenvalues ordered as* $1 = \lambda_1 \geqslant \lambda_2 \geqslant \ldots, \lambda_L \geqslant -1$. *Define* $\rho = \max\{|\lambda_2|, |\lambda_L|\}$. *Let* $z_{true} := \sum_{g=1}^L z_{in}^{(g)}$ *be the true sum. Pick an* $\varepsilon_{\text{con}} < 1$. *If the undirected graph is connected, if* $T_{\text{con}} \geqslant \frac{\log(L^{1.5}/\varepsilon_{\text{con}})}{\log(1/\rho)}$, *then*

$$\max_g |z_{out}^{(g)} - z_{true}| \leqslant \varepsilon_{\text{con}} \max_g |z_{in}^{(g)} - z_{true}|.$$

Proposition 2.1 is an easy consequence of Eq. (1.9) in Proposition 3 of [18]. We use Proposition 2.1 for PM and for GDmin iterations, to show that, for any g, $\tilde{\mathbf{U}}_\tau^{(g)}$ and $\tilde{\mathbf{U}}_\tau^{(1)}$ are close after a sufficient number of consensus iterations, i.e., estimates of any agent g and the first agent (or any arbitrary agent other than the g^{th} agent) are close.

Theorem 2.2. *Consider Algorithm 1. Assume that the network is connected, Assumption 1 on* $\mathbf{X}^\star$ *holds, and that the* $\mathbf{A}_k$*s are i.i.d. with each containing i.i.d. standard Gaussian entries. For final desired error* $\varepsilon < 1$, *set* $\tilde{C} = 9\kappa^2\mu^2$, $\eta = 0.8/{\sigma_{\max}^\star}^2$, $T = C\kappa^2\log(\frac{1}{\varepsilon})$, $T_{PM} = C\kappa^2(\log n + \log\kappa)$, $T_{\text{con},\alpha} = C\log(L)/\log(\frac{1}{\rho})$, $T_{\text{con},PM} = C(\kappa^2\log(\frac{1}{\varepsilon}) + \log L + \log n + \log\kappa)/\log(\frac{1}{\rho})$, $T_{\text{con},GD} = C(\kappa^2\log(\frac{1}{\varepsilon}) + \log L + \log n)/\log(\frac{1}{\rho})$.

If m *satisfies* $mq \geqslant C\kappa^6\mu^2(n+q)r^2 := m_{\text{init}}q$ *for the initialization step, and* $mq \geqslant C\kappa^4\mu^2(n+q)r^2\log\kappa := m_{\text{GD}}q$ *and* $m \geqslant C\max(\log L, \log q, \log n, r)$ *for each GD iteration, then, with probability (w.p.) at least* $1 - n^{-10}$, *for all* $g \in [L]$,

$$\text{SD}(\mathbf{U}_T^{(g)}, \mathbf{U}^\star) \leqslant \varepsilon, \text{ and}$$
$$\max_k \frac{\|(\mathbf{x}_k^{(g)})_T - \mathbf{x}_k^{\star(g)}\|}{\|\mathbf{x}_k^{\star(g)}\|} \leqslant 1.4\varepsilon, \ \|\mathbf{X}_T^{(g)} - \mathbf{X}^\star\|_F \leqslant 1.4\varepsilon\|\mathbf{X}^\star\|.$$

Sample, Time and Communication complexity. The above result implies that the total number of samples per column $m_{tot} := m_{\text{init}} + T\cdot m_{\text{GD}}$ needs to satisfy $m_{tot}q \geqslant C\kappa^6\mu^2(n+q)r^2\log(1/\varepsilon)\log\kappa$ along with needing $m_{tot} \geqslant C\max(\log L, \log q, \log n, r)$.

The time complexity of our algorithm is the time needed per iteration times the total number of consensus iterations. For one inner loop (consensus) iteration, our algorithm needs to (i) compute

Algorithm 1 Pseudocode of the Dec-AltProjGDmin algorithm for agent $g \in [L]$. We have omitted $^{(g)}$ in most places except where it is needed to make things clear (only for input to Avgcons algorithm).

Input: $\mathbf{A}_k, \mathbf{y}_k$, for all $k \in [q]$, set $\mathcal{S}_g$ of all agents $g \in [L]$, graph $\mathcal{G}$, consensus iteration T_{con}

Output: $\mathbf{U}^{(g)}$, $\mathbf{B}^{(g)}$ and $\mathbf{X}^{(g)} = \mathbf{U}^{(g)}\mathbf{B}^{(g)}$.

Parameters: Multiplier in specifying α for init step, $\tilde{C}$; GD step size, η; Number of consensus iterations, $T_{\text{con},\alpha}, T_{\text{con},PM}, T_{\text{con},GD}$, number of PM iterations, T_{PM}, and number of PM iterations, T.

Sample-split: Partition the measurements and measurement matrices into $2T+2$ equal-sized disjoint sets: two sets for initialization and $2T$ sets for the iterations. Denote these by $\mathbf{y}_k^{(\ell)}, \mathbf{A}_k^{(\ell)}, \ell = 00, 0, 1, \ldots 2T$.

Initialization:

Let $\mathbf{y}_k \equiv \mathbf{y}_k^{(00)}, \mathbf{A}_k \equiv \mathbf{A}_k^{(00)}$ for all $k \in [q]$. Set $\alpha^{(g)} \leftarrow \text{AvgCon}_g(\{\alpha_{\text{in}}^{(g)}\}_{g=1}^L, \mathcal{G}, T_{\text{con},\alpha})$ with $\alpha_{\text{in}}^{(g)} \leftarrow \tilde{C}\frac{1}{mq}\sum_{k\in\mathcal{S}_g}\sum_{i=1}^m \mathbf{y}_{ki}^2$

Let $\mathbf{y}_k \equiv \mathbf{y}_k^{(0)}, \mathbf{A}_k \equiv \mathbf{A}_k^{(0)}$ for all $k \in [q]$. Define $\mathbf{y}_{k,trunc}(\alpha^{(g)}) := \mathbf{y}_k \circ \mathbb{1}\{|\mathbf{y}_{ki}| \le \sqrt{\alpha^{(g)}}\}$ for $k \in \mathcal{S}_g$, $g \in [L]$

Generate $\mathbf{U}_0^{(g)} = \mathbf{U}_0$, for $g \in [L]$; the same $n \times r$ matrix with i.i.d. standard Gaussian entries (use the same random seed for all g)

PM iterations:

for $\tau = 1$ to T_{PM} and for all $g \in [L]$ **do**

$\tilde{\mathbf{U}}^{(g)} \leftarrow \text{AvgCon}_g(\{\tilde{\mathbf{U}}_{\text{in}}^{(g)}\}_{g=1}^L, \mathcal{G}, T_{\text{con},PM})$ with $\tilde{\mathbf{U}}_{\text{in}}^{(g)} = ((1/m)\sum_{k\in\mathcal{S}_g}(\mathbf{A}_k^\top \mathbf{y}_{k,\text{trunc}}(\alpha^{(g)}))e_k^\top)(\cdot)^\top \mathbf{U}_{\tau-1}^{(g)}$

$[\mathbf{U}^{(g)}, \mathbf{R}^{(g)}] \leftarrow \text{QR}(\tilde{\mathbf{U}}^{(g)})$

Set $\mathbf{U}_\tau^{(g)} \leftarrow \mathbf{U}^{(g)}$

end for

AltGDmin iterations:

Initialize $\mathbf{U}_0^{(g)} \leftarrow \mathbf{U}_{T_{PM}}^{(g)}$

for $t = 1$ **to** T and for all $g \in [L]$ **do**

Update $\mathbf{b}_k^{(g)}, \mathbf{x}_k^{(g)}$: Let $\mathbf{y}_k = \mathbf{y}_k^{(t)}, \mathbf{A}_k = \mathbf{A}_k^{(t)}$ for all $k \in [q]$. Set $\mathbf{b}_k^{(g)} \leftarrow (\mathbf{A}_k \mathbf{U}_{t-1}^{(g)})^\dagger \mathbf{y}_k$, $\mathbf{x}_k^{(g)} \leftarrow \mathbf{U}_{t-1}^{(g)} \mathbf{b}_k^{(g)}$, for all $k \in [q]$ with $k \in \mathcal{S}_g$

Gradient w.r.t. $\mathbf{U}_{t-1}^{(g)}$: Let $\mathbf{y}_k = \mathbf{y}_k^{(T+t)}, \mathbf{A}_k = \mathbf{A}_k^{(T+t)}$. Compute $\text{GradU}^{(g)} = \text{AvgCon}_g(\{\nabla f_g(\mathbf{U}^{(g)}, \mathbf{B}^{(g)})\}_{g=1}^L, \mathcal{G}, T_{\text{con},GD})$ with $\nabla f_g(\mathbf{U}^{(g)}, \mathbf{B}^{(g)}) := \sum_{k=1}^q \mathbf{A}_k^\top(\mathbf{A}_k \mathbf{U}^{(g)} \mathbf{b}_k^{(g)} - \mathbf{y}_k)\mathbf{b}_k^{(g)\top}$

GD step: Set $\tilde{\mathbf{U}}^{(g)} \leftarrow \mathbf{U}_{t-1}^{(g)} - (\eta/m)\text{GradU}^{(g)}$

Projection step: Compute $[\mathbf{U}^{(g)}, \mathbf{R}^{(g)}] \leftarrow \text{QR}(\tilde{\mathbf{U}}^{(g)})$

Set $\mathbf{U}_t^{(g)} \leftarrow \mathbf{U}^{(g)}$

end for

return $\mathbf{U}^{(g)}$, $\mathbf{B}^{(g)}$ and $\mathbf{X}^{(g)} = \mathbf{U}^{(g)}\mathbf{B}^{(g)}$

$\mathbf{A}_k\mathbf{U}$ for all $k \in [q]$, (ii) solve the LS problem for updating $\mathbf{b}_k$ for all $k \in [q]$, and (iii) compute the gradient w.r.t. $\mathbf{U}$ of $f_k(\mathbf{U}, \mathbf{B})$, i.e. compute $\mathbf{A}_k^\top(\mathbf{A}_k\mathbf{U}\mathbf{b}_k - \mathbf{y}_k)\mathbf{b}_k^\top$. Thus, order-wise, the time taken per iteration is $\max(q \cdot mnr, q \cdot mr^2, q \cdot mnr) = mqnr$. The total number of consensus iterations for initialization is $T_{\text{con},\alpha} + T_{\text{con},PM} \cdot T_{PM}$; this number for GD is $T_{\text{con},GD} \cdot T$. Thus, the total number of consensus iterations needed is $T_{\text{con}} \cdot (T_{PM} + T)$. Since $T > T_{PM}$, This simplifies to $O(\kappa^4 \log^2(1/\varepsilon)\log(Ln\kappa)/\log(1/\rho))$. Thus, the time complexity of our approach is $O(mqnr \cdot \kappa^4 \log^2(1/\varepsilon)\log(Ln\kappa)/\log(1/\rho))$. Treating κ as a numerical constant, this simplifies to $O(mqnr \cdot \log^2(1/\varepsilon)\log(Ln)/\log(1/\rho))$.

For communication complexity, notice that, in all consensus iterations, the nodes are exchanging approximations to $\nabla_U f(\mathbf{U}, \mathbf{B})$ which is a matrix of size $n \times r$. In one such iteration, each node receives nr scalars from its neighbors. Thus the cost per iteration per node is $nr \cdot \deg$ where deg is the maximum degree of any node. The cost per iteration for all the nodes is $nr \cdot deg \cdot L$.

2.3. Theoretical comparisons

There are many results that analyze the decentralized GD algorithm of [7]. However most provide asymptotic convergence guarantees, but do not provide an explicit bound on how the cost function decays over iterations. We use the result of Yuan et al. [15] for our comparison because this provides such a bound at least in a certain regime. Yuan et al. [15] analyzed the decentralized GD algorithm of [7] which solves the problem of minimizing $f(x) = \sum_{g=1}^L f_g(x)$, where $x \in \mathbb{R}^n$ and each f_g is only known to the agent g in a connected network of L agents. DecGD proceeds as follows: each agent g updates its local variable $x^{(g)} \in \mathbb{R}^n$ by combining the average of its neighbors' with a local negative-gradient step as

$$x^{(g)}(t+1) = \sum_{j=1}^g \mathbf{W}_{gj} x^{(j)}(t) - \eta \nabla f_g(x^{(g)}(t)), \text{ for each agent.}$$

Here $\mathbf{W}$ is the mixing matrix and the $(g,j)^{\text{th}}$ entry is non-zero only if g and j are neighbors or $g = j$ and $\mathbf{W}$ is symmetric and doubly stochastic. The paper of Yuan et al. [15] provides an asymptotic convergence guarantee for this algorithm under certain convexity and Lipschitz differentiability assumptions. In addition, it bounds the deviation of the cost function from its minimum value at each iteration of this algorithm until the deviation reaches order $O(\eta/(1-\rho))$.

Proposition 2.3 (Theorem 2, [15])**.** *Consider a connected network and assume that the mixing matrix $\mathbf{W}$ is symmetric and doubly stochastic with $\rho < 1$. For each $g = 1, 2, \ldots, L$, assumes that the functions f_g are proper closed convex, lower bounded, and Lipschitz differentiable with constant $\mathscr{L}_g > 0$. Define $\mathscr{L}_{\max} := \max_g\{\mathscr{L}_g\}$. Then, if the step size $\eta \le (1 + \lambda_L(\mathbf{W}))/\mathscr{L}_{\max}$, then*

$$f(\bar{x}(t)) - f^\star \leqslant \max\{\frac{1}{\eta t}, \frac{\eta}{(1-\rho)}\},$$

Here $\bar{x}(t) = \sum_g x_{(g)}(t)/L$ and $f^* = \min_x f(x)$ is the minimum cost function value. Thus, for $\varepsilon > \frac{\eta}{(1-\rho)}$, the required number of iterations to reach ε deviation is $1/(\eta\varepsilon)$. By setting η equal to its allowed upper bound, this simplifies to $\frac{\mathscr{L}_{\max}}{(1+\lambda_L)\varepsilon}$.

Now consider what our guarantee states. From our bound on $\text{SD}(\mathbf{U}^\star, \mathbf{U}_t^{(g)})$ it is easy to derive a bound on $f(\mathbf{U}, \mathbf{B}) - f^*$ where $f^* = f(\mathbf{U}^\star, \mathbf{B}^\star)$ and $f(\cdot,\cdot)$ is defined in (2). Clearly, in our case $f^* = 0$. Using the sub-exponential Bernstein inequality and linear algebra tricks similar to those used in our proof, it is possible to show that

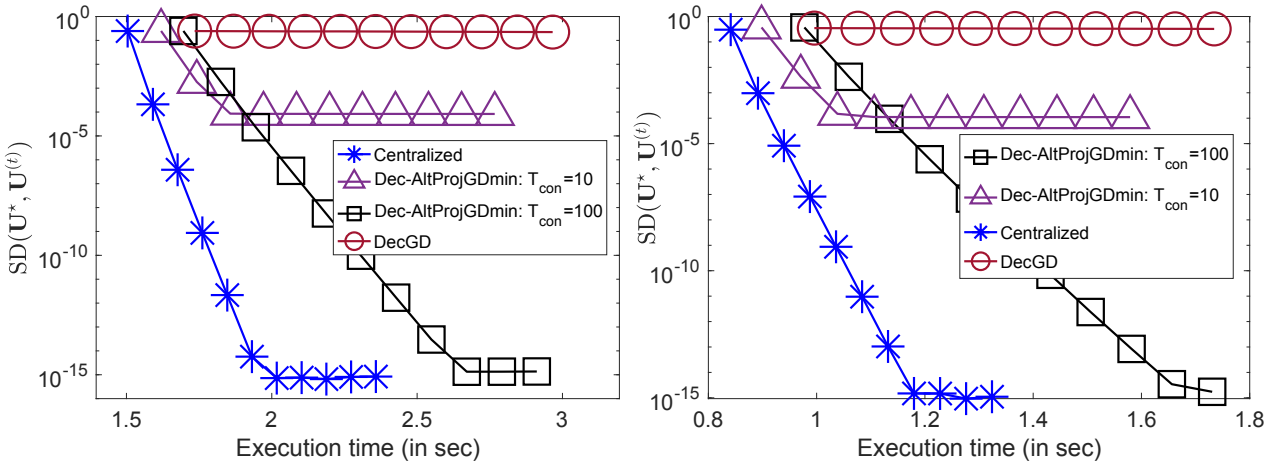
$$f(\mathbf{U}, \mathbf{B}) - f^* = f(\mathbf{U}, \mathbf{B}) \leqslant C \cdot {\sigma_{\max}^\star}^2 \cdot \text{SD}(\mathbf{U}^\star, \mathbf{U}_t^{(g)}).$$

Thus the cost function decays at the same rate as the estimates. Combining this with our main result, $f(\mathbf{U}, \mathbf{B}) - f^* \le \varepsilon {\sigma_{\max}^\star}^2$ after T_{total} iterations with

$$T_{total} := (T_{PM} + T) \cdot T_{T_{\text{con}}} = C\kappa^4 \frac{1}{\log(1/\rho)} \log^2(1/\varepsilon)\log(Ln).$$

To compare our guarantee with that of Yuan et al. given above, the following can be said.

- Assumptions made by both results are different. Their result, and most other previous results that provide an asymptotic analysis of DecGD [7, 9, 10, 15], assume convexity of the cost function and Lipschitz differentiability of the cost function at



(a) $q = 400, L = 20, p = 0.5$ (b) $q = 200, L = 20, p = 0.5$

Fig. 1: Error versus execution time plot with time. We compare performance our fully decentralized algorithm (Dec-AltProjGDmin) with the centralized AltProjGDmin algorithm in [3] (which is the memory efficient existing approach with guarantees when there is a central server and this is equivalent to the decentralized setting with $T_{\text{con}} = \infty$) and the DecGD algorithm in [15] (which computes average of $\mathbf{U}^{(g)}$'s using consensus algorithm followed by a local GD for GD step on $\mathbf{U}$). In Figure (1a), $n = 100$, $r = 4$, $q = 400$, $m = 40$, $p = 0.5$, and $L = 20$. In Figure (1b), $n = 100$, $r = 4$, $q = 200$, $m = 40$, $p = 0.5$, and $L = 20$.

each node. On the other hand, our result is for a specific non-convex optimization problem: the cost function is a specific one given earlier and we are assuming that our unknown matrix $\mathbf{X}$ is LR. In addition, we need the right singular vectors' incoherence assumption stated earlier. One cannot say which set of assumptions is stronger.

- Our result is what can be called a constructive convergence result, it provides the number of iterations needed to reach ε error for any value of $\varepsilon > 0$. The result of Yuan et al. either provides an asymptotic convergence guarantee or provides the explicit number of iterations only for $\varepsilon > \eta/(1-\rho)$.
- In the regime of $\varepsilon > \eta/(1-\rho)$, DecGD deviation from optimum decays as $1/t$ while ours decays exponentially. Thus, our T_{total} depends on $\log^2(1/\varepsilon)$ while theirs depends on $1/\varepsilon$. The latter number of iterations is much larger.
- On first glance, it seems that the guarantee of DecGD does not depend on the network connectivity at all as long as it is connected; however this is not true. The second term in the max expression is $\frac{\eta}{(1-\rho)}$ and thus, their result provides an iteration count only for $\varepsilon > \eta/(1-\rho)$, for this to work for small enough values of ε, one needs η small and a network with ρ small (network with fast mixing properties).

3. SIMULATIONS

In this section, we present the numerical experiments of the Dec-AltProjGDmin algorithm. We used MATLAB for these experiments. First we generated the dataset, i.e., $\mathbf{A}_k$'s and $\mathbf{y}_k$'s, in a random fashion. Then we simulated the a communication network $\mathcal{G}$ as an Erdős Rényi (ER) graph with L vertices and with probability of an edge between any pair of nodes being p. For an ER graph, if $p > (1+\zeta)\log L/L$, then, for large values of L, with high probability (w.h.p.), the graph is connected. Also, if $L < (1+\zeta)\log L/L$, then, for large values of L, w.h.p., the graph is not connected. For a particular simulated graph, we used the *conncomp* function in MATLAB to verify that the graph is connected. The data for our experiment was generated as follows. We know $\mathbf{X}^\star = \mathbf{U}^\star\mathbf{B}^\star$, where $\mathbf{U}^\star$ is an $n \times r$ orthonormal matrix. We generated the entries of $\mathbf{U}^\star$ by orthonormalizing an i.i.d standard Gaussian matrix and the entries of $\mathbf{B}^\star \in \mathbb{R}^{r\times q}$ from a different i.i.d Gaussian distribution. The matrices $\mathbf{A}_k$s were i.i.d. standard Gaussian. We set the step size of GD as $\eta^{(g)} = 0.8/\lambda_{\max}(\mathbf{R}^{(g)}_{T_{PM}})$, where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue. We performed two experiments on the generated dataset which is detailed below. We compared performance our fully decentralized algorithm (Dec-AltProjGDmin) with the centralized AltProjGDmin algorithm in [3] and the decentralized GD algorithm (DecGD) in [7]. The approach of [7] and the follow-up works designed for standard GD cannot be used for updating $\mathbf{U}$ because it involves averaging the partial estimates $\mathbf{U}^{(g)}$, $g \in [L]$, obtained locally at the different nodes. However, since $\mathbf{U}^{(g)}$'s are subspace basis matrices, their numerical average will not provide a valid "subspace mean" [1]. We validate this through our experiments.

We plot the variation of the matrix estimation error (at the end of the iteration) $SD(\mathbf{U}^\star, \mathbf{U}^{(t)})$ and the execution time-taken (until the end of that iteration) on the y-axis and x-axis, respectively. The parameters are $p = 0.5$, $n = 100$, $r = 4$, $m = 40$, and $L = 20$. We implemented the Dec-AltProjGDmin algorithm for three values of T_{con}, $T_{\text{con}} = 1, 10, 100$ for $q = 400$ (Fig.1a) and $q = 200$ (Fig.1b). To compare with we also implement the centralized algorithm in [3] and decentralized algorithm in [7]. The experimental results shows the trade-off between convergence rate and execution time with different number of consensus iterations and the effectiveness of our approach as compared to [3] and [7].

4. CONCLUSION

In this paper we studied the recently developed Dec-AltProjGDmin algorithm and presented a constructive convergence guarantee in terms of the eigenvalues of the weight (mixing) matrix.We compared our results and algorithm, both theoretically and experimentally, with the best existing result from the decentralized GD literature that also provided a convergence time bound for values of ε that are *large enough*. While our guarantees hold for any $\varepsilon > 0$, to the best of our knowledge, there is no convergence guarantee result for decentralized GD available in the literature for any given $\varepsilon > 0$, and hence a clear cut comparison is not possible.

[1] To compute the subspace mean of $\mathbf{U}^{(g)}$'s w.r.t. the subspace distance $SD(\cdot,\cdot)$, one would need to solve $\min_{\bar{\mathbf{U}}} \sum_g SD^2(\mathbf{U}^{(g)}, \bar{\mathbf{U}})$. This cannot be done in closed form and will require an expensive iterative algorithm.

5. REFERENCES

[1] Shana Moothedath and Namrata Vaswani, "Fully decentralized and federated low rank compressive sensing," *in American Control Conference (ACC)*, 2022.

[2] Rakshith Sharma Srinivasa, Kiryung Lee, Marius Junge, and Justin Romberg, "Decentralized sketching of low rank matrices," in *Neural Information Processing Systems (NeurIPS)*, 2019, pp. 10101–10110.

[3] S. Nayer and N. Vaswani, "Fast low rank column-wise compressive sensing," *IEEE Transactions on Information Theory*, 2022, to appear. Also at arXiv:2102.10217.

[4] S. Nayer and N. Vaswani, "Sample-efficient low rank phase retrieval," *IEEE Transactions on Information Theory*, 2021.

[5] Shana Moothedath and Namrata Vaswani, "Fast, communication-efficient, and provable decentralized low rank matrix recovery," *submitted to IEEE Transactions on Signal Processing*, 2022.

[6] Shana Moothedath and Namrata Vaswani, "Dec-AltProjGD: Fully-decentralized alternating projected gradient descent for low rank column-wise compressive sensing," *Conference on Decision and Control (CDC)*, 2022.

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

[8] Alex Olshevsky and John N Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM journal on control and optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[9] Angelia Nedić, Asuman Ozdaglar, and Pablo A Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[10] Angelia Nedić, "Convergence rate of distributed averaging dynamics and optimization in networks," *Foundations and Trends® in Systems and Control*, vol. 2, no. 1, pp. 1–100, 2015.

[11] Soomin Lee and Angelia Nedić, "Distributed random projection algorithm for convex optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 221–229, 2013.

[12] Alexander Rogozin and Alexander Gasnikov, "Projected gradient method for decentralized optimization over time-varying networks," *ArXiv preprint arXiv:1911.08527*, 2019.

[13] Firooz Shahriari-Mehr, David Bosch, and Ashkan Panahi, "Decentralized constrained optimization: Double averaging and gradient projection," *arXiv preprint arXiv:2106.11408*, 2021.

[14] Ilan Lobel and Asuman Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1291–1306, 2010.

[15] Kun Yuan, Qing Ling, and Wotao Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[16] S. Lang, *Real and Functional Analysis*, Springer-Verlag, New York 10:11–13, 1993.

[17] Roman Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47, Cambridge University Press, 2018.

[18] Persi Diaconis and Daniel Stroock, "Geometric bounds for eigenvalues of markov chains," *The Annals of Applied Probability*, pp. 36–61, 1991.

[19] GW Stewart, "Perturbation bounds for the QR factorization of a matrix," *SIAM Journal on Numerical Analysis*, vol. 14, no. 3, pp. 509–518, 1977.

[20] Ran Xin, Usman A Khan, and Soummya Kar, "A fast randomized incremental gradient method for decentralized non-convex optimization," *IEEE Transactions on Automatic Control*, 2021.