Dec-AltProjGDmin: Fully-Decentralized Alternating Projected Gradient Descent for Low Rank Column-wise Compressive Sensing

Shana Moothedath and Namrata Vaswani

***Abstract*— This work develops a fully-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 columnwise linear projections (LR column-wise Compressive Sensing). We prove its correctness under simple assumptions and argue that Dec-AltProjGDmin is both faster and more communication-efficient than various other potential solution approaches, in addition to also having one of the best sample complexity guarantees. To our best knowledge, this work is the first attempt to develop a provably correct fully-decentralized algorithm for any problem involving the use of an alternating projected GD algorithm when the constraint set (the set to be projected onto) is non-convex.**

I. Introduction

In this work we develop a fully-decentralized gradient descent (GD) based algorithm for solving the *low rank (LR) column-wise Compressive Sensing (LRcCS)* problem [1], [2], [3] defined below and prove its correctness under simple assumptions. LRcCS is a lesser known LR matrix recovery problem that involves recovering an LR matrix from mutually independent dense linear projections of each of its columns. One important application where LRcCS is useful is for 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. The term 'federated' sketching means that the data to be compressed, e.g., images or videos, is acquired at geographically distributed nodes, e.g. mobile phones or IoT devices [1], [4]. For networks of such devices, it is often impractical to assume that a centralized coordinating node exists. A fully-decentralized network where each node (phone or IoT device) can only exchange information with its neighbors is more common. Moreover, when the distributed nodes are far from many others, this setting also implies much lesser communication power usage.

A. Problem setting and notation

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

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

The $m \times n$ matrices $\mathbf{A}_k$ are known and mutually independent for different k. The regime of interest is $m < n$ and the goal is to have to use as few number of samples m as possible. The total sample complexity is mq.

We consider a fully-decentralized setting, i.e., there is *no central node* to aggregate the summaries computed by the individual nodes. *Henceforth, in the paper below, the term "decentralized" also refers to this fully-decentralized setting.* 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 a graph $\mathcal{G} = (V, E)$, where V, with $|V| = L$, denotes the set of nodes and E denotes the set of undirected edges. The neighbor set of the g^{th} node (sensor) is given by $\mathcal{N}_g$, i.e., $\mathcal{N}_g := \{j : (g, j) \in E\}$.

At various places in the paper, for tall $n \times r$ matrices, we are only interested in the orthonormal basis spanned by the matrix. For a matrix $\tilde{\mathbf{Z}}$, we use $\mathbf{Z} = \text{Orth}(\tilde{\mathbf{Z}})$ to denote this. When needed, we compute it using QR decomposition, i.e. $\tilde{\mathbf{Z}} \overset{\text{QR}}{=} \mathbf{Z}\boldsymbol{R}$. We denote the Frobenius norm as $\|\cdot\|_F$, the induced ℓ_2 norm (often called the operator norm or spectral norm) as $\|\cdot\|$, and the (conjugate) transpose of a matrix $\mathbf{Z}$ as $\mathbf{Z}^\top$. We use $\mathbf{e}_k$ to denote the k^{th} canonical basis vector. Also, $[d] := \{1, 2, \ldots, d\}$. We say $\mathbf{U}$ is a *basis matrix* if it contains orthonormal columns. For basis matrices $\mathbf{U}_1, \mathbf{U}_2$, the default measure of Subspace Distance (SD) used in this work is $\text{SD}(\mathbf{U}_1, \mathbf{U}_2) := \left\|(I - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{U}_2\right\|_F$. Here I is the identity matrix. At certain places, we also use $\text{SD}_2(\mathbf{U}_1, \mathbf{U}_2) := \left\|(I - \mathbf{U}_1\mathbf{U}_1^\top)\mathbf{U}_2\right\|$. $\text{SD}(\cdot,\cdot)$ is the ℓ_2 norm of the sines of the r principal angles between $\text{span}(\mathbf{U}_1)$ and $\text{span}(\mathbf{U}_2)$ while $\text{SD}_2(\cdot,\cdot)$ is their infinity norm (equivalently, size of largest principal angle). Further, $^\top$ denotes matrix or vector transpose, $\otimes$ denotes Kronecker product, and $|\boldsymbol{z}|$ for a vector $\boldsymbol{z}$ denotes element-wise absolute values. We define the max norm of a $u \times v$ matrix as the maximum absolute entry, and denoted it as $\|\mathbf{Z}\|_{\max}$. 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 $C \geqslant 1$. We let

$$\mathbf{X}^\star \overset{\text{SVD}}{=} \mathbf{U}^\star \boldsymbol{\Sigma}^\star \mathbf{V}^\star := \mathbf{U}^\star \mathbf{B}^\star$$

denote its reduced (rank r) SVD, i.e., $\mathbf{U}^\star$ and $\mathbf{V}^{\star\top}$ are matrices with orthonormal columns *(basis matrices)*, $\mathbf{U}^\star$ is $n \times r$, $\mathbf{V}^\star$ is $r \times q$, and $\boldsymbol{\Sigma}^\star$ is an $r \times r$ diagonal matrix with non-negative entries. We let $\mathbf{B}^\star := \boldsymbol{\Sigma}^\star \mathbf{V}^\star$. We use $\kappa := \sigma_{\max}^\star / \sigma_{\min}^\star$ for the condition number of $\boldsymbol{\Sigma}^\star$. *We note that we have omitted $^{(g)}$ in most places except where it is needed to make things clear for notational brevity.*

Another way to understand (1) is as follows: each scalar measurement $\mathbf{y}_{ki}$ (i-th entry of $\mathbf{y}_k$) satisfies $\mathbf{y}_{ki} := \langle \mathbf{a}_{ki}, \mathbf{x}_k^\star \rangle, \; i \in [m], \; k \in [q]$ with $\mathbf{a}_{ki}{}^\top$ being the i-th row of $\mathbf{A}_k$. The measurements are not global: no $\mathbf{y}_{ki}$ is a function of the entire matrix $\mathbf{X}^\star$. They are global for each column, but not across the different columns. We thus need an assumption that enables correct interpolation across the different columns. The following incoherence (w.r.t. the canonical basis) assumption on the right singular vectors suffices for this purpose [5]. Such an assumption on both left and right singular vectors was first introduced in [6] for making the low-rank matrix completion (LRMC) problem well posed. In case of LRMC the measurements are both row- and column-wise local while for LRcCS they are only column-wise local.

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* $\max_k \|\mathbf{x}_k^\star\| \leq \kappa\mu \|\mathbf{X}^\star\|_F / \sqrt{q}$.

TABLE I: Existing work versus our work. Table treats κ, μ as numerical constants. Let *deg* denote the maximum degree of any node. All approaches also need $m \geqslant \max(\log q, \log n)$. The approaches projGD-X and altGDnormbal cannot be analyzed for reasons explained in the footnote below the table and in [2] and hence we are only providing a guess for their required number of iterations. This guess is based on the guarantees for solving LRMC which is a different but related problem. Also, the other approaches have not been studied in decentralized settings, hence the parameter T_{con} (number of consensus loop iterations) cannot be not specified.

	Time Comp.	Communic Comp per iter.	Sample Comp. $mq \gtrsim$	Provable exponential error decay in decentralized case?
Convex [1] (mixed norm min)	$mqnr\frac{1}{\sqrt{\varepsilon_{fin}}} \cdot T_{\text{con}}$	Not clear	$(n+q)r\frac{1}{\varepsilon_{fin}^2}$	Not studied
AltMin [3]	$mqnr\log^2(1/\varepsilon_{fin}) \cdot T_{\text{con}}$	$(nr)^2 \cdot L \cdot deg$ (guessed)	$(n+q)r^3 \log \frac{1}{\varepsilon_{fin}}$	Not studied
Dec-AltProjGDmin (proposed)	$mqnr \log \frac{1}{\varepsilon_{fin}} \cdot T_{\text{con}}$ $T_{\text{con}} = L^3 \log \frac{1}{\varepsilon_{fin}} \log(Ln)$	$nr \cdot L \cdot deg$	$(n+q)r^2 \log \frac{1}{\varepsilon_{fin}}$	**Yes**
ProjGD-X [7] for LRMC	$mqnr \log \frac{1}{\varepsilon_{fin}} \cdot T_{\text{con}}$ (guessed)	$nq \cdot L \cdot deg$	Not clear studied only for LRMC**	Not clear
AltGDnormbal [8], [9] for LRMC	$mqnr^2 \log \frac{1}{\varepsilon_{fin}} \cdot T_{\text{con}}$ (guessed)	$nr \cdot L \cdot deg$	Not clear studied only for LRMC**	Not clear
Decentral. Projected GD [10], [11]			Not clear studied only for convex costs and constraint sets	Not clear

**It is not clear how to analyze either of ProjGD-X or AltGDnormbal for LRcCS because in both cases the estimates of $\hat{\mathbf{x}}_k$ are coupled across different columns (i.e., $\hat{\mathbf{x}}_k$ is a function of not just $\mathbf{x}_k^\star$ but also of the other columns). Because of this it is not possible to get a tight enough bound on $\max_k ||\hat{\mathbf{x}}_k - \mathbf{x}_k^\star||$. Such a bound is needed to bound $||\nabla_U f(\mathbf{U},\mathbf{B})||$ and show that the gradient norm decays to zero with iteration count; for details, see [2].

B. Related work and our contribution

LRcCS has not been studied in the decentralized setting so far, except in our preliminary work [12]. LRMC is the closest problem to LRcCS that has been extensively studied in the centralized case [6], [8], [9], [13], [14] and there has been some work also in the (fully-)decentralized setting [15], [16], [17]. The work of [15], [16] designs a decentralized Gauss-Seidel method, while [17] develops a decentralized approach to solve the convex relaxation of robust LRMC. Both these approaches are also slower than GD based solutions. Moreover, neither comes with guarantees for the decentralized implementation to converge.

For the LRcCS problem that we consider here, in the centralized setting, a convex relaxation (mixed norm minimization) was proposed in [1], and an alternating minimization (AltMin) solution for its generalization, LR phase retrieval, was developed and studied in [3], [5]. Solving convex relaxation is very slow (both theoretically and empirically). In recent work [2], a GD-based solution was proposed. This was shown to be faster than the AltMin solution (both converge geometrically, but the per-iteration cost of the GD approach is lower). In this work, we develop a fully-decentralized version of this solution.

There are two common approaches to designing GD algorithms for LR recovery problems. The first, projected GD on $\mathbf{X}$ (projGD-X) involves running one step of GD on $\mathbf{X}$ for minimizing the cost function $f(\mathbf{X})$; followed by projecting the output onto the space of rank r matrices [7], [18], and repeating this at each iteration. For large matrices, this is communication-inefficient in a distributed or decentralized setting, since the gradients to be shared are of size $n \times q$. The second approach, alternating GD, is much more communication-efficient: it has a per-iteration communication, and memory, cost proportional to $(n+q)r$. It factorizes $\mathbf{X}$ as $\mathbf{X} = \mathbf{UB}$ and alternatively updates estimates of $\mathbf{U}$ and $\mathbf{B}$ using GD w.r.t. one keeping the other fixed (or modifications). One important point here is that the decomposition of $\mathbf{X}$ into $\mathbf{UB}$ is not unique since $\mathbf{UB} = (\mathbf{U}\boldsymbol{R}^{-1}\boldsymbol{R}\mathbf{B})$ for any $r \times r$ invertible matrix $\boldsymbol{R}$. This means that $\mathbf{U}$ uniquely specifies only the column span of $\mathbf{X}$ and the norm of one of $\mathbf{U}$ or $\mathbf{B}$ can keep increasing in an unbounded fashion, while that of the other decreases. To prevent this, either norm balancing terms are added to the cost function as in [8], [9] (AltGDnormbal), or, one projects one of them onto the space of matrices with orthonormal columns after each GD step (AltProjGD) [2].

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 [10], [11], [19]-[20], starting with the seminal work of Nedić *et al.* [19]. There is also some recent work on projected GD for constrained optimization problems [10], [21], [22] or for imposing the consensus constraint [21]. However, all existing approaches that come with guarantees assume convex cost functions and either no constraints or convex constraint sets. The works of [10], [22] 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 [21] considers the unconstrained optimization problem, and uses projection onto an appropriately defined subspace to impose the consensus constraint at each algorithm iteration.

Our Contribution. In this work we develop a provably accurate fast and fully-decentralized alternating projected gradient descent (Dec-AltProjGDmin) algorithm for solving the LRcCS problem described above. We factor the unknown $n \times q$ rank-r matrix $\mathbf{X}$ as $\mathbf{X} = \mathbf{UB}$, where $\mathbf{U}$ and $\mathbf{B}$ are matrices with r columns and rows respectively, and consider the squared loss cost function

$$f(\mathbf{U},\mathbf{B}) := \sum_k \|\mathbf{y}_k - \mathbf{A}_k\mathbf{U}\mathbf{b}_k\|^2.$$

Here $\mathbf{b}_k$ is the k-th column of $\mathbf{B}$. Starting with a careful spectral initialization for $\mathbf{U}$, AltProjGD alternatively updates $\mathbf{U}$ and $\mathbf{B}$ by (a) one step of ProjGD for $\mathbf{U}$ (GD step followed by projecting onto space of basis matrices) keeping $\mathbf{B}$ fixed at its previous value,

and (b) minimizing $f(\mathbf{U},\mathbf{B})$ over $\mathbf{B}$ keeping $\mathbf{U}$ fixed at its most recently updated value. For our specific problem, because of the column-wise decoupled form of the measurement model, step (b) can be done locally at each node and it is as fast a GD step. For reasons explained above, the projection in ProjGD is critical for ensuring that the norms of $\mathbf{U}$ and $\mathbf{B}$ remain bounded. As we show in Table I, AltProjGD is faster than all competing approaches except projGD-X and it is as fast as projGD-X. Communication cost wise, it is significantly better than projGD-X as well as AltMin (which requires a communication cost per edge per iteration of $(nr)^2$; this is needed to approximate an $nr \times nr$ matrix which involves a summation over all $k \in [q]$).

To our best knowledge, this work is the first attempt to develop a provably correct decentralized algorithm for any problem involving the use of an alternating projected GD algorithm when the constraint set (the set to be projected onto) is non-convex. We expect to be able to extend the ideas developed here to also try to develop a similar fast and communication-efficient decentralized solution to LR matrix completion (LRMC). We are unable to borrow ideas from the existing literature on efficient consensus algorithms for decentralized (projected) GD because (i) the cost function $f(\mathbf{U},\mathbf{B})$ is not a convex function of the unknowns $\{\mathbf{U},\mathbf{B}\}$; and (ii) the constraint set (set of $n \times r$ matrices with orthonormal columns) is not a convex set either. In particular, (ii) means that the matrices $\mathbf{U}^g$ estimated at the various nodes g cannot be averaged to get a matrix whose column span is close to that of the matrices being averaged; while (i) means that our algorithm needs a careful initialization (which also needs to be computed in a decentralized fashion).

In our case, we have to use a scalar average/sum consensus algorithm both to approximate each entry of the gradient at each iteration as well as for the spectral initialization. The spectral initialization needs to be computed using a decentralized approximation to the power method (PM). Our proof strategy consists of interpreting this approximation as an instance of the noisy PM studied in [23] and using the consensus algorithm guarantees [24] to ensure that the "noise" is small enough to satisfy the assumptions required by the noisy PM guarantee from [23]. We then combine this guarantee with that used to analyze the initialization in case of centralized LRcCS in [2] to get our initialization guarantee. A similar strategy is used for the rest of the algorithm as well.

II. The Proposed Algorithm and Guarantee

A. Dec-AltProjGDmin: Decentralized Alternating projected GD and minimization

We first explain the main steps of AltProjGD and then explain how to develop its decentralized version. At each GD iteration, we have the following two steps

- *Minimization over* $\mathbf{B}$*:* For each new estimate of $\mathbf{U}$, solve for $\mathbf{B}$ by minimizing $f(\mathbf{U},\mathbf{B})$ over it while keeping $\mathbf{U}$ fixed at its current value. Because of the LRcCS measurement model, this becomes a decoupled column-wise least squares (LS) step and the solution can be written in closed form as $\mathbf{b}_k = (\mathbf{A}_k\mathbf{U})^\dagger \mathbf{y}_k$ for each k. In this step, the most expensive part is the computation of the matrices $\mathbf{A}_k\mathbf{U}$ for all k, this takes time of order $mnr \cdot q$. Each LS computation only needs time of order mr^2.
- *Projected-GD (ProjGD) for* $\mathbf{U}$*:* compute $\tilde{\mathbf{U}} \leftarrow \mathbf{U} - \eta \nabla_U f(\mathbf{U},\mathbf{B})$ and orthonormalize it using QR decomposition: $\mathbf{U} \leftarrow \mathrm{QR}(\tilde{\mathbf{U}})$. Here $\nabla_U f(\mathbf{U},\mathbf{B}) = \sum_{k=1}^q \mathbf{A}_k^\top(\mathbf{A}_k\mathbf{U}\mathbf{b}_k - \mathbf{y}_k)\mathbf{b}_k^\top$. The gradient computation takes time $mqnr$. The QR decomposition only needs time nr^2.

Notice that $f(\mathbf{U},\mathbf{B})$ is not a convex function of the unknowns $\{\mathbf{U},\mathbf{B}\}$ and hence a careful initialization which can be shown to be a good approximation of the true $\mathbf{U}^\star$ (desired minimizer). Here and below "approximation" of $\mathbf{U}^\star$ means the subspace spanned by the columns of the approximation is close to that spanned by columns of $\mathbf{U}^\star$ in $\mathrm{SD}_F(\cdot,\cdot)$ or $\mathrm{SD}_2(\cdot,\cdot)$ distance. We initialize as follows. Compute $\mathbf{U}_0$ as the top r left singular vectors of

$$\mathbf{X}_0 = (1/m)\sum_{k\in[q]} \mathbf{A}_k^\top \mathbf{y}_{k,\mathrm{trunc}}(\alpha) e_k^\top$$

with $\alpha := \tilde{C}\frac{\sum_{ki}(\mathbf{y}_{ki})^2}{mq}$ and $\mathbf{y}_{k,\mathrm{trunc}}(\alpha) := \mathbf{y}_k \circ \mathbb{1}_{\{\mathbf{y}_{ki}^2 \leqslant \alpha\}}$. This is $\mathbf{y}_k$ with large magnitude entries zeroed out, i.e., $(\mathbf{y}_{k,trunc})_i = \mathbf{y}_{ki}$ if $\mathbf{y}_{ki}^2 \leqslant \alpha$ and it equals zero otherwise. Observe that we are summing $\mathbf{a}_{ki}\mathbf{y}_{ki}\mathbf{e}_k{}^\top$ over only those values i,k for which $\mathbf{y}_{ki}^2$ is not too large (is not much larger than its empirically computed average value). This *truncation* filters out the too large (outlier-like) measurements and sums over the rest. Theoretically, this converts the summands into sub-Gaussian r.v.s which have lighter tails than the un-truncated ones which are sub-exponential.

To approximate the above approach in a decentralized fashion (where each node can only exchange data summaries with its neighbors), the use of a average (sum) consensus approach is required. This is needed at three places. In the initialization step, we need it to approximate (i) the threshold α; and (ii) 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 power method (PM). (iii) In the ProjGD iterations, we need it to approximate the sum of the individual gradients at all the nodes, i.e. compute $\nabla_U f(\mathbf{U},\mathbf{B}) = \sum_{k\in[q]} \mathbf{A}_k^\top(\mathbf{A}_k\mathbf{U}(\mathbf{b}_k) - \mathbf{y}_k)(\mathbf{b}_k)^\top$. Each of these three steps requires computing a sum (equivalently an average). Since we are unable to use the vector decentralized GD approaches, we use simple scalar average-consensus [24]. This is summarized in Algorithm 1. In each of its iterations, nodes update their values by taking a weighted sum of their own and their neighbors' partial sums. By doing this enough number of times, if the graph is connected, and the weights are equal to $1/d_g$ where d_g is the degree of node g, one can show that one finally computes an estimate of the true sum at each node [24].

We specify the scalar consensus algorithm in Algorithm 1 and the Dec-AltProjGDmin algorithm in Algorithm 2.

Algorithm 1 Pseudo-code for distributed average consensus $\mathrm{AvgCons}(\mathbf{Z},\mathcal{G},T_{\mathrm{con}})$

Input: $\mathbf{Z}_{\mathrm{in}}^{(\mathrm{g})}$, for all $g \in [L]$ ($\mathbf{Z}_{\mathrm{in}}^{(\mathrm{g})}$ can be a scalar or an $n \times r$ matrix)
Parameters: number of iterations, T_{con}
Initialize $\mathbf{Z}_0^{(\mathrm{g})} \leftarrow \mathbf{Z}_{\mathrm{in}}^{(\mathrm{g})}$, for all $g \in [L]$
Set the mixing matrix $\boldsymbol{W}$ as $\boldsymbol{W}_{gj} = \begin{cases} 1/d_g, & \text{for } j \in \mathcal{N}_g, \text{ where } d_g \text{ is the degree of node g.} \\ 0, & \text{for } g \notin \mathcal{N}_g. \end{cases}$
for $t = 0$ to T_{con} **do**
  $\mathbf{Z}_{t+1}^{(\mathrm{g})} \leftarrow \mathbf{Z}_t^{(\mathrm{g})} + \sum_{j\in\mathcal{N}_g} W_{gj}\left(\mathbf{Z}_t^{(\mathrm{j})} - \mathbf{Z}_t^{(\mathrm{g})}\right)$, for $g \in [L]$
end for
return $\mathbf{Z}_{\mathrm{out}}^{(\mathrm{g})} \leftarrow L \cdot \mathbf{Z}_{T_{\mathrm{con}}}^{(\mathrm{g})}$
Output: $\mathrm{AvgCon}_{(\mathrm{g})}(\{\mathbf{Z}_{\mathrm{in}}^{(\mathrm{g})}\}_{g=1}^L, \mathcal{G}, T_{\mathrm{con}}) := \mathbf{Z}_{\mathrm{out}}^{(\mathrm{g})}$, for all $g \in [L]$

B. Main Result and Sample, Time, & Communication complexities

We can prove the following result for Algorithm 2.

Theorem 2.1. *Consider Algorithm 2. 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* $\varepsilon_{fin} < 1$*, set* $\tilde{C} = 9\kappa^2\mu^2$*,* $\eta = 0.8/{\sigma_{\max}^\star}^2$*,* $T = C\kappa^2\log(1/\varepsilon_{fin})$*,* $T_{PM} = C\kappa^2(\log n + \log\kappa)$*,* $T_{\mathrm{con},\alpha} = CL^3\log(L)$*,*

Algorithm 2 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 iterations: 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 $\mathbf{U}_0^{(g)} = \mathbf{U}_0$ as the same $n \times r$ matrix with i.i.d. standard Gaussian entries for all $g \in [L]$ (use the same random seed at all nodes)

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 $k \in [q]$. For each $k \in [q]$ with $k \in \mathcal{S}_g$, set $\mathbf{b}_k^{(g)} \leftarrow (\mathbf{A}_k \mathbf{U}_{t-1}^{(g)})^\dagger \mathbf{y}_k$ and set $\mathbf{x}_k^{(g)} \leftarrow \mathbf{U}_{t-1}^{(g)}\mathbf{b}_k^{(g)}$

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

$T_{\text{con},PM} = CL^3(\kappa^2\log(1/\varepsilon_{fin}) + \log L + \log n + \log\kappa)$, $T_{\text{con},GD} = CL^3(\kappa^2\log(1/\varepsilon_{fin}) + \log L + \log n + \log\kappa)$.

If m satisfies $mq \ge C\kappa^6\mu^2(n+q)r^2 := m_{\text{init}}q$ for the initialization step, and

$$mq \ge C\kappa^4\mu^2(n+q)r^2\log\kappa := m_{\text{GD}}q \text{ and}$$
$$m \ge C\max(\log L, \log q, \log n, r)$$

for each GD iteration, then, with probability (w.p.) at least $1-n^{-10}$, at all nodes $g \in [L]$,

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

Sample 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 $m_{tot}q \ge C\kappa^6\mu^2(n+q)r^2\log(1/\varepsilon_{fin})\log\kappa$ along with needing $m_{tot} \ge C\max(\log L, \log q, \log n, r)$.

Time complexity. 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 $\mathbf{A}_k\mathbf{U}$ 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 $T_{\text{con},GD} \cdot T$. This simplifies to $O(L^3\kappa^4\log^2(1/\varepsilon_{fin})\log(Ln\kappa))$.

Thus, the time complexity is $O(mqnr \cdot L^3\kappa^4\log^2(1/\varepsilon_{fin})\log(Ln\kappa))$. Treating κ as a numerical constant, this simplifies to $O(mqnr \cdot L^3\log^2(1/\varepsilon_{fin})\log(Ln))$.

Communication complexity. 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 $nr \cdot deg \cdot L$.

C. Proof of Main Result (Theorem 2.1)

In this section, we assume that Assumption 1 holds, and the graph is connected. We first present the two main results that help prove our main result, Theorem 2.1. In the interest of space, we present a proof outline with novelty and followed up by a discussion. For complete proofs we refer the readers to [25].

In the entire proof, to keep notation simple, we often define quantities and state results only for the first node.

The next theorem analyzes the GD iterations for each agent g and shows that the GD error decays at an exponential rate. Recall that the initialization for the GDmin iterations is the final output of the PM algorithm, i.e., that $\mathbf{U}_0^{(g)} = \mathbf{U}_{T_{PM}}^{(g)}$.

Theorem 2.2 (Decentralized GD)**.** *If $\eta = 0.8/{\sigma_{\max}^\star}^2$, $T = C\kappa^2\log(1/\varepsilon_{fin})$, $T_{\text{con},GD} = CL^3(\kappa^2\log(1/\varepsilon_{fin}) + \log(n\kappa L/\varepsilon_{fin}) = CL^3(\kappa^2\log(1/\varepsilon_{fin}) + \log(Ln\kappa))$, if*

$$\text{SD}(\mathbf{U}^\star, \mathbf{U}_0^{(l)}) \le \delta_0 = c/\kappa^2 \text{ and } \max_g \|\mathbf{U}_0^{(l)} - \mathbf{U}_0^{(g)}\|_F \le \tilde{b}_0 = \varepsilon_{fin}/3^T,$$

and if $mq \ge C\kappa^4\mu^2(n+q)r^2\log\kappa\log(1/\varepsilon_{fin})$ and $m > C\max(\log L, \log q, \log n)\log(1/\varepsilon_{fin})$; then w.p. at least $1-n^{-10}$,

$$\text{SD}(\mathbf{U}^\star, \mathbf{U}_T^{(l)}) \le \varepsilon_{fin}.$$

The next theorem provides a bound on the initialization error when the agents perform decentralized initialization.

Theorem 2.3 (Decentralized PM for initialization)**.** *Pick a $\delta_0 < 0.1$. If $mq \ge C\max(\kappa^2\mu^2(n+q)r^2/\delta_0^2, \log L) = C\kappa^2\mu^2(n+q)r^2/\delta_0^2$ (since $L \le q$), if $T_{PM} = C\kappa^2\log(n/\delta_0)$, if $T_{con} > CL^3(\log(Ln\kappa/\delta_0) + \log(1/\tilde{b}_0))$, then w.p at least $1-(n^{-C_2}+e^{-C_3 n})-n^{-10}$,*

$$\text{SD}(\mathbf{U}_{T_{PM}}^{(l)}, \mathbf{U}^\star) \le \delta_0 \text{ and } \max_g \|\mathbf{U}_{T_{PM}}^{(g)} - \mathbf{U}_{T_{PM}}^{(l)}\|_F \le \tilde{b}_0.$$

Theorem 2.1 is an easy consequence of the above two results. We need to set δ_0 and $\tilde{b}_0$ in the initialization result in order to satisfy the requirements of the GD result.

Proof of Theorem 2.1. The subspace distance bound is a direct consequence of the above three theorems along with setting $\delta_0 =$

c/κ^2 and $\tilde{b}_0 = \varepsilon_{fin}/3^T$ with $T = C\kappa^2 \log(1/\varepsilon_{fin})$ in the initialization theorem.

Thus the initialization step needs $T_{\text{con}} = CL^3(\log(Ln\kappa/\delta_0) + T + \log(1/\varepsilon_{fin})) = CL^3(\log(Ln\kappa) + \kappa^2 \log(1/\varepsilon_{fin})$. The GD step needs the same expression for T_{con}.

In addition, the initialization step also needs $T_{PM} = C\kappa^2 \log(n\kappa)$.

The subspace distance bounds of Theorem 2.1 follow using the above two results. The bounds on the errors in $\mathbf{x}_k^{\star(g)}$ and $\mathbf{X}^\star$ follow using the results from [2] (Lemma 4.8 in [25]) and the lemma below. Let

$$\mathbf{B}^{(g)} = [\mathbf{b}_k^{(g)},\ k \in \mathcal{S}_g], \quad \mathbf{X}^{(g)} = [\mathbf{U}^{(g)}\mathbf{b}_k^{(g)},\ k \in \mathcal{S}_g] \text{ and}$$
$$\mathbf{B} = [\mathbf{B}^{(g)},\ g \in [L]], \quad \mathbf{X} = [\mathbf{X}^{(g)},\ g \in [L]],$$

Also let $\mathbf{X}^{\star(g)} = [\mathbf{x}_k^{\star(g)},\ k \in \mathcal{S}_g]$ be a sub-matrix of $\mathbf{X}^\star$, and similarly define $\mathbf{B}^{\star(g)}$.

Lemma 2.4. *Recall that* $\mathbf{B} := [\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)}]$ *and similarly for* $\mathbf{X}$. *Define* $\boldsymbol{D} := [\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \ldots, \boldsymbol{D}^{(L)}]$. *Assume that*

$$\text{SD}(\mathbf{U}^\star, \mathbf{U}_t^{(g)}) \leqslant \delta_t, \text{ and } \max_g \|\mathbf{U}_t^{(l)} - \mathbf{U}_t^{(g)}\|_F \leqslant \tilde{b}_t$$

with $\delta_t < \delta_0 = c/\kappa^2$ *and* $\tilde{b}_t \leq 0.5\delta_t/\sqrt{r}$. *Then, for any agent* $g \in [L]$, *w.p.* $\geqslant 1 - \exp(\log q + r - cm)$,

1) $\|\mathbf{B} - \boldsymbol{D}\|_F = \sqrt{\sum_{g\in[L]} \|\boldsymbol{D}^{(g)} - \mathbf{B}^{(g)}\|_F^2} \leqslant 0.6\delta_t \sigma_{\max}^\star$,
2) $\|\mathbf{X} = \mathbf{X}^\star\|_F = \sqrt{\sum_{g\in[L]} \|\mathbf{X}^{\star(g)} - \mathbf{X}^{(g)}\|_F^2} \leqslant 1.6\delta_t \sigma_{\max}^\star$,
3) $\sigma_{\min}(\mathbf{B}) \geqslant 0.7\sigma_{\min}^\star$ *and* $\sigma_{\max}(\mathbf{B}) \leqslant 1.1\sigma_{\max}^\star$.

III. Proof novelty, outline, and discussion

A. Proof novelty

Consider the GDmin iterations and consider node $g = 1$. Assume for this discussion that κ, μ are numerical constants. The overall proof is similar to that for the centralized setting, however the details are different. In the centralized setting, we first obtain a deterministic bound on $\text{SD}(\mathbf{U}^\star, \mathbf{U}_{t+1})$ in terms of $\text{SD}(\mathbf{U}^\star, \mathbf{U}_t)$ and three other terms. This proof uses the fundamental theorem of calculus [26]. Next, we use the analysis of the minimization step and the sub-exponential Bernstein inequality to bound these terms in order to show exponential subspace recovery error decay. In the current decentralized setting, we still need the same two steps.

The first step is similar, except we get an extra term $\|\text{ConsErr}^{(1)}\|_F$, with $\text{ConsErr} = \text{GradU}^{(1)} - \text{GradU}$. Most of the new work is in the second step. This is harder for two reasons. First, $\text{GradU}^{(1)} \neq \text{GradU} = \sum_g \nabla f_g(\mathbf{U}^{(g)}, \mathbf{B}^{(g)})$, but is a consensus based approximation of it. This is easy to handle using a standard consensus guarantee. The second, and more difficult, issue is that we can only compute a consensus approximation of $\sum_g \nabla f_g(\mathbf{U}^{(g)}, \mathbf{B}^{(g)})$: the $\mathbf{U}^{(g)}$ is different at different nodes g while, in the centralized setting, we compute $\sum_g \nabla f_g(\mathbf{U}, \mathbf{B})$. This difference implies that $\boldsymbol{b}_k^{(g)} = (\mathbf{A}_k \mathbf{U}^{(g)})^\dagger \mathbf{y}_k$ is an estimate of $\mathbf{U}^{(g)\top}\mathbf{x}_k^{\star(g)}$ with $\mathbf{U}^{(g)}$ being different for different nodes g. Consequently it is not straightforward to convert a bound on $\|\boldsymbol{b}_k^{(g)} - \mathbf{U}^{(g)\top}\mathbf{x}_k^{\star(g)}\|$ into a bound on $\|\mathbf{X} - \mathbf{X}^\star\|_F$. In order to do this, we now need to use the consensus result, and a guarantee for perturbed QR decomposition to show first consensus for $\mathbf{U}^{(g)}$s, see Lemma 4.6 in [25]. Next, we need to use this and develop a more involved argument to get the desired tight bound on $\|\mathbf{X} - \mathbf{X}^\star\|_F$, see Lemma 2.4. We explain the main ideas below.

B. Proof outline

We use the analysis of the minimization step, fundamental theorem of calculus [26], sub-exponential Bernstein inequality [27], scalar consensus guarantee (see Proposition 4.2 in [25]), and analysis of perturbed QR decomposition (see Proposition 4.2 in [25]) to show the following. If $\text{SD}(\mathbf{U}^\star, \mathbf{U}_t^{(1)}) \leq \delta_t$ for a $\delta_t < \delta_0 = c < 1$, and $\|\mathbf{U}_t^{(g)} - \mathbf{U}_t^{(1)}\|_F \leq \tilde{b}_t \leq \delta_t/\sqrt{r}$, then w.h.p.,

$$\text{SD}(\mathbf{U}^\star, \mathbf{U}_{t+1}^{(1)}) \leq (1 - c/\kappa^2)^t \delta_t := \delta_{t+1}, \text{ and}$$
$$\|\mathbf{U}_{t+1}^{(g)} - \mathbf{U}_{t+1}^{(1)}\|_F \leq \tilde{b}_{t+1} \leq \delta_{t+1}/\sqrt{r} \tag{2}$$

This, along with a simple induction argument, and the analysis of the initialization step to get the desired initial bounds, completes our proof.

To do the above, we use the fundamental theorem of calculus to obtain a deterministic bound on $\text{SD}(\mathbf{U}^\star, \mathbf{U}_{t+1}^{(1)})$ in terms of $\text{SD}(\mathbf{U}^\star, \mathbf{U}_t^{(1)})$. See Lemma 4.4 in [25]. In order to prove the result, we need a careful analysis of the minimization step, which is done in Lemma 2.4 and Lemma 4.8 in [25]. Let $\mathbf{U}^{(g)} \equiv \mathbf{U}_t^{(g)}$. Lemma 4.8 in [25] bounds $\|\boldsymbol{b}_k^{(g)} - \mathbf{U}^{(g)\top}\mathbf{x}_k^{\star(g)}\|$, $\|\mathbf{x}_k^{(g)} - \mathbf{x}_k^{\star(g)}\|$, $\|\boldsymbol{b}_k^{(g)}\|$ and a fourth similar term follows exactly as in [2]. However Lemma 2.4, which uses these bounds to get bounds on $\|\mathbf{X} - \mathbf{X}^\star\|_F$, and on singular values of $\mathbf{B}$, needs a new proof (explained below). This uses the assumption $\|\mathbf{U}_t^{(g)} - \mathbf{U}_t^{(1)}\|_F \leq \delta_t/\sqrt{r}$.

The last step to is to show the second row of (2). This is done in Lemma 4.7 in [25]. This proof uses the assumption $\|\mathbf{U}_t^{(g)} - \mathbf{U}_t^{(1)}\|_F \leq \tilde{b}_t$ with $\tilde{b}_t \leq \delta_t/\sqrt{r}$, and the bound on ConsErr from Lemma 4.6 in [25] to bound $\|\tilde{\mathbf{U}}_{t+1}^{(g)} - \tilde{\mathbf{U}}_{t+1}^{(1)}\|_F$. Next, it uses this bound and the perturbed QR result to show that $\|\mathbf{U}_{t+1}^{(g)} - \mathbf{U}_{t+1}^{(1)}\|_F \leq \tilde{b}_{t+1} = 3(\tilde{b}_t + 0.2\sqrt{nr}\varepsilon_{\text{con}})$, where ε_{con} is the consensus error bound for one scalar entry of the gradient matrix. In the proof of our main result (Theorem 2.2), using these lemmas, we set ε_{con} in order to guarantee that $\tilde{b}_t \leq \delta_t/\sqrt{r}$ for all t.

Main idea for Lemma 2.4. We provide the main ideas here.

Lemma 4.8 in [25] shows that $\|\boldsymbol{b}_k^{(g)} - \mathbf{U}^{(g)\top}\mathbf{x}_k^{\star(g)}\| \leqslant 0.4\|\left(\mathbf{I} - \mathbf{U}^{(g)}\mathbf{U}^{(g)\top}\right)\mathbf{U}^\star \mathbf{b}_k^{\star(g)}\|$ for all $k \in \mathcal{S}_g$ and for all g. Lemma 2.4 uses this to bound $\sum_g \|\mathbf{B}^{(g)} - \mathbf{U}^{(g)\top}\mathbf{X}^{\star(g)}\|_F^2$. This bound is then used to bound $\|\mathbf{X} - \mathbf{X}^\star\|_F^2$ and the maximum and minimum singular values of $\mathbf{B}$,

1) Observe that $\sum_g \|\mathbf{B}^{(g)} - \mathbf{U}^{(g)\top}\mathbf{X}^{\star(g)}\|_F^2 \leq 0.4^2 \sum_g \|\left(\mathbf{I} - \mathbf{U}^{(g)}\mathbf{U}^{(g)\top}\right)\mathbf{U}^\star \mathbf{B}^{\star(g)}\|_F^2$.
2) Bounding $\sum_g \|\left(\mathbf{I} - \mathbf{U}^{(g)}\mathbf{U}^{(g)\top}\right)\mathbf{U}^\star \mathbf{B}^{\star(g)}\|_F^2$ by $2(\delta_t \sigma_{\max}^\star)^2$:
   To bound this, we first add and subtract $\mathbf{U}^{(1)}\mathbf{U}^{(1)\top}$ from the parentheses. For the first term, we use $\|\mathbf{U}^{(g)}\mathbf{U}^{(g)\top} - \mathbf{U}^{(1)}\mathbf{U}^{(1)\top}\|_F \leq 2\|\mathbf{U}_t^{(g)} - \mathbf{U}_t^{(1)}\|_F$ and our assumption to show that
   $$\sum_{g\in[L]} \|(\mathbf{U}^{(1)}\mathbf{U}^{(1)\top} - \mathbf{U}^{(g)}\mathbf{U}^{(g)\top})\mathbf{U}^\star \mathbf{B}^{\star(g)}\|_F^2$$
   $$\leq (\delta_t/\sqrt{r})^2 \|\mathbf{U}^\star \mathbf{B}^\star\|_F^2 = (\delta_t/\sqrt{r})^2 (\sqrt{r}\sigma_{\max}^\star)^2 = (\delta_t \sigma_{\max}^\star)^2$$
   For the second term, we can use an approach similar to [2] since $(\mathbf{I} - \mathbf{U}^{(1)}\mathbf{U}^{(1)\top})$ is the same for all g.
   $$\sum_{g\in[L]} \|(\mathbf{I} - \mathbf{U}^{(1)}\mathbf{U}^{(1)\top})\mathbf{U}^\star \mathbf{B}^{\star(g)}\|_F^2$$
   $$= \|(\mathbf{I} - \mathbf{U}^{(1)}\mathbf{U}^{(1)\top})\mathbf{U}^\star \mathbf{B}^\star\|_F^2 \leq (\delta_t \|\mathbf{B}^\star\|)^2 = (\delta_t \sigma_{\max}^\star)^2$$
3) This then implies that $\sum_g \|\mathbf{B}^{(g)} - \mathbf{U}^{(g)\top}\mathbf{X}^{\star(g)}\|_F^2 \leq 0.4^2 \cdot 2(\delta_t \sigma_{\max}^\star)^2$.
4) The second item is also used to bound $\|\mathbf{X} - \mathbf{X}^\star\|_F^2 \leqslant (0.32 + 1)\sum_{g\in[L]} \|(\mathbf{I} - \mathbf{U}^{(g)}\mathbf{U}^{(g)\top})\mathbf{U}^\star \mathbf{B}^{\star(g)}\|_F^2$
5) The third item is used to lower bound $\sigma_{\min}(\mathbf{B})$ and upper bound $\sigma_{\max}(\mathbf{B})$ as follows. Consider $\sigma_{\min}(\mathbf{B})$.
   Define $\boldsymbol{D} = [\mathbf{U}^{(g)\top}\mathbf{X}^{\star(g)},\ g \in [L]]$ and $\boldsymbol{D}^{tmp} = [\mathbf{U}^{(1)\top}\mathbf{X}^{\star(g)},\ g \in [L]] = \mathbf{U}^{(1)\top}\mathbf{X}^\star$.
   We have $\sigma_{\min}(\mathbf{B}) \geq \sigma_{\min}(\boldsymbol{D}^{tmp}) - \|\mathbf{B} - \boldsymbol{D}\| - \|\boldsymbol{D} - \boldsymbol{D}^{tmp}\| = \sigma_{\min}^\star - \|\mathbf{B} - \boldsymbol{D}\| - \|\boldsymbol{D} - \boldsymbol{D}^{tmp}\|$.

$\|\mathbf{B}-\boldsymbol{D}\|_F$ is upper bounded in the third item above. $\|\boldsymbol{D}-\boldsymbol{D}^{tmp}\|_F^2 = \sum_g \|(\mathbf{U}^{(1)}-\mathbf{U}^{(g)})\mathbf{X}^{\star(g)}\|_F^2 \le \sum_g \|\mathbf{U}^{(1)}-\mathbf{U}^{(g)}\|_F^2\|\mathbf{X}^{\star(g)}\|_F^2 \le \max_g \|\mathbf{U}^{(1)}-\mathbf{U}^{(g)}\|_F^2\|\mathbf{X}\|_F^2 \le (\delta_t/\sqrt{r})^2(\sqrt{r}\sigma^\star_{\max})^2 = (\delta_t\sigma^\star_{\max})^2$ by our assumption. Thus, $\sigma_{\min}(\mathbf{B}) \ge \sigma^\star_{\min} - 3\delta_t\sigma^\star_{\max} \ge 0.9\sigma^\star_{\min}$ by using the assumption. $\sigma_{\max}(\mathbf{B})$ is bounded similarly.

Initialization. Consider our initialization guarantee (Theorem 2.3). Our proof strategy consists of interpreting the decentralized PM as an instance of the "noisy" PM meta-algorithm studied in [23]. We then carefully use the results for the centralized setting [2], and results for scalar consensus [24], to show that the "noise" in each PM iteration is small enough as needed by the result of [23]. We combine this with that used for centralized LRcCS in [2] to get our result.

C. Discussion

Consider our GDmin iterations' guarantee. It requires T_{con} in each iteration, t, to be proportional to $\log(1/\varepsilon_{fin})$. It may be possible to improve our algorithm and results by borrowing the gradient tracking ideas from [28] to modify our GD step.

In all the above works, the required T_{con} depends on the mixing time, which is a function of the second largest eigenvalue, of the mixing matrix. For a given *connected* network, if we use the equal neighbor weight matrix, the mixing time is $O(L^3)$ [24], [29]. We assume the network is given and cannot be designed. If the network topology can be designed, it may be possible to improve the required value of T_{con} [29].

The algorithm in [30] studies non-convex unconstrained problems. The guarantee assumes that the cost function satisfies smoothness and the Polyak Łojasiewicz (PL) condition, which is a generalization of strong convexity to non-convex functions. These assumptions are not satisfied in our problem.

IV. Conclusion

This work developed a fully-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 columnwise linear projections (LR column-wise Compressive Sensing). We proved the correctness of the proposed algorithm under simple assumptions and showed that Dec-AltProjGDmin is both faster and more communication-efficient than various other potential solution approaches, in addition to also having one of the best sample complexity guarantees. To our best knowledge, this work is the first attempt that developed a provably correct fully-decentralized algorithm for any problem involving the use of an alternating projected GD algorithm and one in which the constraint set to be projected to is a non-convex set.

References

[1] R. S. Srinivasa, K. Lee, M. Junge, and J. Romberg, "Decentralized sketching of low rank matrices," in *Neural Information Processing Systems (NeurIPS)*, 2019, pp. 10 101–10 110.

[2] S. Nayer and N. Vaswani, "Fast and sample-efficient federated low rank matrix recovery from column-wise linear and quadratic projections (old title: Fast low rank column-wise compressive sensing)," *arXiv:2102.10217*, 2021.

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

[4] F. P. Anaraki and S. Hughes, "Memory and computation efficient pca via very sparse random projections," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1341–1349.

[5] S. Nayer, P. Narayanamurthy, and N. Vaswani, "Provable low rank phase retrieval," *IEEE Transactions on Information Theory*, March 2020.

[6] E. J. Candes and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, no. 9, pp. 717–772, 2008.

[7] Y. Cherapanamjeri, K. Gupta, and P. Jain, "Nearly-optimal robust matrix completion," *International Conference on Machine Learning*, 2016.

[8] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust pca via gradient descent," in *Neural Information Processing Systems (NeurIPS)*, 2016.

[9] Q. Zheng and J. Lafferty, "Convergence analysis for rectangular matrix completion using burer-monteiro factorization and gradient descent," *arXiv preprint arXiv:1605.07051*, 2016.

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

[11] A. 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.

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

[13] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[14] P. Netrapalli, P. Jain, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Annual ACM Symposium on Theory of Computing (STOC)*, 2013.

[15] Q. Ling, Y. Xu, W. Yin, and Z. Wen, "Decentralized low-rank matrix completion," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 2925–2928.

[16] A.-Y. Lin and Q. Ling, "Decentralized and privacy-preserving low-rank matrix completion," *Journal of the Operations Research Society of China*, vol. 3, no. 2, pp. 189–205, 2015.

[17] M. Mardani, G. Mateos, and G. Giannakis, "Decentralized sparsity-regularized rank minimization: Algorithms and applications," *IEEE Transactions on Signal Processing*, 2013.

[18] P. Jain and P. Netrapalli, "Fast exact matrix completion with finite samples," in *Conference on Learning Theory*, 2015, pp. 1007–1034.

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

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

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

[22] F. Shahriari-Mehr, D. Bosch, and A. Panahi, "Decentralized constrained optimization: Double averaging and gradient projection," *arXiv preprint arXiv:2106.11408*, 2021.

[23] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," *Advances in neural information processing systems (NeurIPS)*, 2014.

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

[25] S. Moothedath and N. Vaswani, "Fast, communication-efficient, and provable decentralized low rank matrix recovery," *arXiv preprint, arxiv: 2204.08117*, 2022.

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

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

[28] 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, 2015.

[29] L. Lovász, "Random walks on graphs," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1-46, p. 4, 1993.

[30] R. Xin, U. A. Khan, and S. Kar, "A fast randomized incremental gradient method for decentralized non-convex optimization," *IEEE Transactions on Automatic Control*, 2021.