# Fully Decentralized and Federated Low Rank Compressive Sensing

Shana Moothedath and Namrata Vaswani

***Abstract*— In this work we develop a fully decentralized, federated, and fast solution to the recently studied Low Rank Compressive Sensing (LRCS) problem: recover an $n \times q$ low-rank matrix $\mathbf{X}^\star = [\mathbf{x}_1^\star, \mathbf{x}_2^\star, \dots, \mathbf{x}_q^\star]$ from column-wise linear projections, $\mathbf{y}_k := \mathbf{A}_k \mathbf{x}_k^\star$, $k = 1, 2, \dots, q$, when each $\mathbf{y}_k$ is an $m$-length vector with $m < n$. A simple federated sketching solution is to left multiply the $k$-th vectorized image by a random matrix $\mathbf{A}_k$ and to store only $\mathbf{y}_k$. When $m \ll n$, this requires much lesser storage than storing the full image, and is much faster to implement than traditional image compression. Suppose there are $L$ nodes (say $L$ smartphones), and each stores a set of $(q/L)$ sketches of its images. We develop a decentralized projected gradient descent (GD) based approach to jointly reconstruct the images of all the phones/users from their respective stored sketches. The algorithm is such that the phones/users never share their raw data (their subset of $\mathbf{y}_k$s) but only summaries of this data with the other phones at each algorithm iteration. Also, the reconstructed images of user $g$ are obtained only locally and other users cannot reconstruct them. Only the column span of the matrix $\mathbf{X}^\star$ is reconstructed globally. By "decentralized" we mean that there is no central node to which all nodes are connected and the only way to aggregate the summaries from the various nodes is by use of an iterative consensus algorithm that provides an estimate of the aggregate at each node, for strongly connected network. We validated the effectiveness of our algorithm via extensive simulation experiments.**

## I. Introduction

Due to the growing need for reliable high-speed computing and the increasing focus on security and privacy, it is often preferred to store and process data in a distributed manner, and to recover the whole data in a federated way [1], [2]. *Federated learning* is an approach where devices collaborate to learn a global model from data stored on distributed devices, under the constraint that device-generated data are stored and processed locally, with only intermediate updates being shared between the devices [3], [4]. In the traditional federated setting, so-called as *centralized federated learning*, the devices periodically communicate their local intermediate updates with a central server. The central server then aggregates the information received from all the devices and communicates it with all devices. The key limitation of a centralized federated setting is that (i) the central server orchestrates the whole process and hence is a single point of failure and (ii) the central server may become a bottle neck in certain applications as the number of nodes increases. In applications such as federated sketching of data/images from smart phones or IoT devices, a decentralized setting is more practical [5]. This motivates a fully decentralized and federated framework to learn from distributed data. By "decentralized" we mean that there is no central node to which all nodes are connected and thus the only way to aggregate the summaries from the various nodes is via information exchange between the nodes.

In this paper, we develop a fully decentralized solution to the recently studied Low Rank Compressive Sensing (LRCS) problem [6]: how to reconstruct a low rank (LR) matrix from linear projection measurements of its columns in a decentralized and federated setting. Specifically, an $n \times q$ (low rank) rank-$r$ matrix, $\mathbf{X}^\star = [\mathbf{x}_1^\star, \mathbf{x}_2^\star, \dots, \mathbf{x}_q^\star]$, needs to be recovered from distributed column-wise linear measurements $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q$, where $\mathbf{y}_k := \mathbf{A}_k \mathbf{x}_k^\star$ for $k \in \{1, 2, \dots, q\}$, and each $\mathbf{y}_k$ is an $m$-length vector with $m < n$. The measurement signals, $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q$, are distributed across $p$ nodes and the nodes collaborate to recover $\mathbf{X}^\star$ by periodically sharing their local information with the neighboring nodes via a communication network. Moreover, the information sharing is federated such that the nodes share the parameters of their local model, rather than the raw signal itself. An important application where this problem occurs and a decentralized solution is needed is for the federated sketching: efficiently compressing the vast amounts of distributed images/videos generated by smartphones and various other devices while respecting the users' privacy [2], [7], [8], [9]. Images from different devices, once grouped by category, are pretty similar and hence the matrix formed by the vectorized images of a certain category is well-modeled as being low rank.

*A. Related Work:* The centralized LRCS problem has been studied in three recent works. The first is an Alternating Minimization solution that solves the harder magnitude-only generalization of LRCS (LR Phase Retrieval) [10], [11], [12]. The second (parallel work) studies a convex relaxation called mixed norm min [1]. The third [6] is a gradient descent (GD) based provable solution to LRCS, that we called GDmin. The convex solution is very slow, has very bad experimental performance, and has a worse sample complexity than GDmin for highly accurate recovery settings [6]. The AltMin solution [10], [11], [12] is also much slower than GDmin. Also, since it is designed for a harder problem, its sample complexity guarantee for LRCS is sub-optimal compared to that of GDmin, and consequently it has worse recovery performance with fewer samples [6]. While [6] considered federated, it is the centralized setting where a central node aggregates information from all nodes. The centralized setting is also what was considered in [1] although the paper title contains the word "decentralized".

We should mention here that LRCS is significantly different from the other more commonly studied LR recovery problems: LR matrix sensing (LRMS) [13], LR matrix completion (LRMC) [13], [14], multivariate regression (MVR) [8], or robust PCA [15]. MVR is the LRCS problem with $\mathbf{A}_k = \mathbf{A}_1$, for $k \in \{1, 2, \dots, q\}$, but this simple change makes it

*Department of Electrical and Computer Engineering, Iowa State University, USA. Email: {mshana,namrata}@iastate.edu.

a very different problem: with this change, the measurements of the different columns are no longer mutually independent, conditioned on $\mathbf{X}^\star$. This, in turn, implies that the required sample complexity per column, $m$, can never be less than the signal length $n$. This point is explained in detail in [6].

Distributed iterative algorithms for consensus and averaging problems are well studied in the literature [16], [17], [18]. The general decentralized learning problem, and in particular, decentralized convex optimization, has also been studied extensively. Recently, decentralized GD algorithms with provable guarantees have been developed as well, starting with the seminal works of Nedić *et al.* [19]. There is also some work on using projected GD for decentralized learning [20], [21], [22]. However, all existing approaches with guarantees assume convex cost functions and convex constraints (or no constraints). [20] considers the unconstrained optimization problem, and uses projection onto an appropriately defined subspace to impose the consensus constraint at each algorithm iteration. The works of [21], [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 at each GD iteration. However, these approaches are not suitable for the decentralized LRCS problem as our cost functions are not convex and the constraint set (set of low rank matrices with orthonormal columns) is not a convex set.

*B. Our Contribution and Paper Organization:* In this work, we develop a fast Gradient Descent (GD) algorithm for solving the decentralized federated LRCS problem. The centralized federated setting considered in earlier work [6] meant that the algorithm for dealing with the distributed nodes was pretty straightforward and not too different from the centralized setting. For example, there was no change to the analysis and almost no extra steps needed in the algorithm itself when compared with a fully centralized setting. However, without any central server to aggregate the summary statistics, the algorithm design becomes much more difficult. In this work, we borrow ideas from the scalar consensus literature [16] to develop (i) a decentralized spectral initialization approach; and (ii) develop a decentralized approach to aggregate the gradients. Our proposed algorithm, DeF-GD, integrates a consensus-based approach with projected GD. We present numerical validation of our approach through extensive experiments on simulated data.

The rest of the paper is organized as follows. Section II introduces the problem formulation and discusses the notations used in the paper. Section III presents the proposed algorithm, DeF-GD. Section IV gives the numerical validation results of the proposed algorithm. Section V, presents the concluding remarks and future work.

## II. Problem Formulation and Notations

*A. Problem Formulation: DLRCS Problem:* We first specify the LRCS problem below and then explain the decentralized setting. The goal is to recover a set of $q$ $n-$dimensional vectors/signals, $\mathbf{x}_1^\star, \mathbf{x}_2^\star, \ldots, \mathbf{x}_q^\star$ 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 column-wise linear measurements of the form

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

Here the matrices $\mathbf{A}_k \in \mathbb{R}_{m\times n}$ are known, $\mathbb{R}$ denotes the set of real numbers, and $\mathbf{y}_k$ is an $m$-length vector. We refer to $\mathbf{X}^\star$ as a Low Rank (LR) matrix as $r \ll \min(n,q)$.

In this work, we assume a decentralized federated setting. The $q$ signals $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q$ are not sensed/measured centrally at one node. Instead, there is a set of $L$ distributed nodes/sensors and each node can observe $q/L$ linear projection measurements. For simplicity we assume here that $q/L$ is an integer. Thus, for example, node 1 observes $\mathbf{y}_1, \ldots, \mathbf{y}_{(q/L)}$, node 2 observes $\mathbf{y}_{(q/L)+1}, \ldots, \mathbf{y}_{2q/L}$, and so on.

Moreover, there is *no central node* to aggregate the summaries computed by the individual nodes. The individual nodes exchange information about the parameters of their measurement signals, rather than the raw signal itself, with their neighboring nodes via a communication network. The communication network is specified by a directed graph $\mathcal{G} = (V,E)$, where $V$, $|V| = L$, denotes the set of nodes and $E$ denotes the set of directed edges. The neighbor set of the $g^{\text{th}}$ node (sensor) is given by $\mathcal{N}_g = \{j : (g,j) \in E\}$. We denote the local measurement available to node $g$ by $\mathcal{Y}_g$, where $\mathcal{Y}_g \subset \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q\}$ such that $\cup_{g=1}^{L} \mathcal{Y}_g = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q\}$ and $\mathcal{Y}_g \cap \mathcal{Y}_j = \emptyset$ for $g, j \in \{1,2,\ldots,L\}$ and $g \neq j$. The goal is to recover the matrix $\mathbf{X}^\star$ from the measurements of $p$ sensors in a fully decentralized and federated manner, specifically when $m << n$. We refer to this problem as the *Decentralized Low Rank Compressive Sensing (DLRCS) problem*.

We note that, the measurements are not global, since each measurement, $\mathbf{y}_k$, is a function of a particular column of $\mathbf{X}^\star$, i.e., $\mathbf{x}_k^\star$, rather the full matrix $\mathbf{X}^\star$. The measurements are global for each column but not across the different columns. We thus need the following incoherence assumption to enable correct interpolation across the different columns [11]. This was introduced in [14] for LR Matrix Completion (LRMC) which is another LR problem with non-global measurements, but its model is symmetric across rows and columns.

Let us denote the reduced (rank $r$) Singular Value Decomposition (SVD) of the rank-$r$ matrix $\mathbf{X}^\star$ as $\mathbf{X}^\star \overset{\text{SVD}}{=} \mathbf{U}^\star \, \Sigma^\star \, \mathbf{V}^{\star\top}$. Here $\mathbf{U}^\star \in \mathbb{R}^{n\times r}$ and $\mathbf{V}^\star \in \mathbb{R}^{q\times r}$ are rank-$r$ orthonormal matrices. Let $\sigma_{\max}$, $\sigma_{\min}$ denote the maximum and minimum singular values of $\Sigma^\star$, respectively. Thus $\kappa = \sigma_{\max}/\sigma_{\min}$ is the condition number of $\Sigma^\star$ (since $\mathbf{X}^\star$ is rank deficient, its condition number is infinite). We define $\mathbf{B} := \mathbf{V}^\top$ and $\tilde{\mathbf{B}} := \Sigma \mathbf{V}^\top$. Thus $\mathbf{X}^\star \overset{\text{SVD}}{=} \mathbf{U}^\star \, \Sigma^\star \, \mathbf{B}^\star = \mathbf{U}^\star \, \tilde{\mathbf{B}}^\star$. In our approach, we will recover columns of $\tilde{\mathbf{B}}^\star$, denoted as $\tilde{\mathbf{b}}_k^\star$, individually.

**Assumption 1** (Right singular vectors' incoherence)**.** *We assume that* $\max_k \|\mathbf{b}_k^\star\| \leq \mu\sqrt{r/q}$. *Treating the condition number* $\kappa$ *of* $\Sigma^\star$ *as a constant, up to constants, this is equivalent to requiring that* $\max_k \|\mathbf{x}_k^\star\|^2 \leq \tilde{\mu} \sum_{k=1}^{q} \|\mathbf{x}_k^\star\|^2/q$ *for a constant* $\tilde{\mu}$ *that can depend on* $\kappa$.

*B. Notation:* We denote the Frobenius norm as $\|\cdot\|_F$, the induced $\ell_2$ norm as $\|\cdot\|$, and the (conjugate) transpose of a matrix $\mathbf{Z}$ as $\mathbf{Z}^\top$. We use $\mathbf{e}_k$ to denote the $k^{\text{th}}$ 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. Note that, for two $r$-dimensional subspaces, $SD(\cdot,\cdot)$ is the $\ell_2$ norm of the sines of the $r$ principal angles between span($\mathbf{U}_1$) and span($\mathbf{U}_2$) and is a measure of distance between the two subspaces.

## III. Proposed Algorithm: DeF-GD

In this section, we present the proposed fully decentralized federated algorithm for solving the DLRCS problem. We would like to find a matrix $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_q]$ that minimizes $f(\mathbf{X}) := \sum_{k=1}^{q} ||\mathbf{y}_k - \mathbf{A}_k\mathbf{x}_k||^2$ subject to the constraint that its rank is $r$ or less, in a fully decentralized and federated manner. The pseudocode for the proposed algorithm is given in Algorithm III.3. Algorithm III.3 integrates a projected GD algorithm with a consensus algorithm. The projected GD serves the matrix recovery part and the consensus algorithm serves the decentralized aggregation in a federated manner. We first present the details for projected GD and then present the details of consensus-based projected GD.

### A. Main idea of the centralized projected GD algorithm [6]:

To recover matrix $\mathbf{X}$, we write $\mathbf{X}=\mathbf{U}\mathbf{B}$ where $\mathbf{U}$ is $n\times r$ and $\mathbf{B}$ is $r\times q$ and do alternating projected GD on $\mathbf{U}$ and $\mathbf{B}$. We use projected GD for updating $\mathbf{U}$ (one GD step followed by projecting onto the space of orthonormal matrices); the projection step is needed to ensure the norm of $\mathbf{U}$ does not keep increasing over iterations). For each new estimate of $\mathbf{U}$, we solve for $\mathbf{B}$ by minimizing over it keeping $\mathbf{U}$ fixed. Because of the specific asymmetric nature of our measurement model, the min problem for columns of $\mathbf{B}$ is decoupled. Thus the minimization over $\mathbf{B}$ only involves solving $q$ $r$-dimensional Least Squares (LS) problems, in addition to also first computing the $q$ matrices, $\mathbf{A}_k\mathbf{U}$, for use in the LS step. Thus the time needed is only $O(qmr^2+qmnr)=O(mqnr)$. This is order-wise equal to the time needed to compute gradient with respect to $\mathbf{U}$, and thus, the per-iteration cost of GDmin is only $O(mqnr)$.

Notice that, for $m<n$, our problem is convex but not strongly convex. As a result GD starting from any arbitrary initialization may converge to *a* minimum but the minimum is not unique. Consequently, it is not guaranteed to converge to the true matrix that we want to recover. To address this issue, a class of approaches known as spectral initialization have been used frequently in the literature. The idea is to define a matrix that is close to a matrix whose top $r$ left or right singular vectors span the column span of the true $\mathbf{X}^\star$ which is equivalent to a matrix that is close to the top $r$ eigenvectors of $\mathbf{X}^\star\mathbf{X}^{\star\top}$. In our setting this involves computing the matrix $\mathbf{U}^{(0)}$ given in Algorithm III.2.

### B. Decentralized and Federated Projected GD (DeF-GD):

We note that the measurements of the nodes are distributed as $\mathcal{Y}_1,\ldots,\mathcal{Y}_L$, where $\mathcal{Y}_g \subset \{\mathbf{y}_1,\ldots,\mathbf{y}_q\}$, $\cup_{g=1}^{L}\mathcal{Y}_g = \{\mathbf{y}_1,\ldots,\mathbf{y}_q\}$ and $\mathcal{Y}_g \cap \mathcal{Y}_j = \emptyset$ for $j\neq g$. In a *decentralized federated* setting, the nodes only share the parameters or estimates of the local updates, rather than the raw signal or the local measurements itself, with other nodes. Additionally, each node shares the parameters or estimate of the local update with the neighboring nodes only. As a result, a direct implementation of projected GD is not feasible. To propose a decentralized, federated version of the projected GD, we integrate projected GD with a consensus algorithm.

**Algorithm III.1** Pseudocode for distributed average consensus: AvgConsensus ($\mathbf{D}_1,\mathbf{D}_2,\ldots,\mathbf{D}_L,\mathbf{W},C$)

**Input:** Matrices $\mathbf{D}_1,\ldots,\mathbf{D}_L$, where $\mathbf{D}_g\in\mathbb{R}^{u\times v}$ for $g\in\{1,\ldots,L\}$, Weight matrix $\mathbf{W}\in\mathbb{R}^{L\times L}$, iteration number $C$

Initialize $\mathbf{D}_g^{(0)} \leftarrow \mathbf{D}_g$, for $g\in[L]$
**for** $\tau=0$ to $C$ **do**
  $\mathbf{D}_g^{(\tau+1)} \leftarrow \mathbf{D}_g^{(\tau)} + \sum_{j\in\mathcal{N}_g} W_{gj}\left(\mathbf{D}_j^{(\tau)} - \mathbf{D}_g^{(\tau)}\right)$, for $g\in[L]$
**end for**

In each iteration of the algorithm, the nodes run a local projected GD utilizing its local data. The parameters of the GD is then communicated with the neighboring nodes. Each node aggregates its own local GD parameter with the neighbors' GD parameters and performs a distributed consensus until all nodes converge to the same GD parameter. Once consensus is achieved, all nodes update their local estimate using the converged GD parameter in order to minimize the estimation error. The projected GD iteration continues until all nodes converge to a global estimate with an acceptable error tolerance. The convergence of the consensus algorithm is guaranteed when $\mathcal{G}$ is strongly connected and the weight matrix is doubly stochastic and symmetric [16], [18].

**Proposition 1** ([18])**.** *Let $\mathcal{G}$ be a strongly connected graph and suppose that each node of $\mathcal{G}$ performs a distributed linear protocol $z_g(t+1) = z_g(t) + \sum_{j\in\mathcal{N}_g}\mathbf{W}_{gj}(z_j(t)-z_g(t))$. Then if the graph $\mathcal{G}$ is connected and $\mathbf{W}$ is doubly stochastic and symmetric, then $\lim_{t\to\infty} z_g(t) = \frac{1}{p}\sum_{g=1}^{L} z_g(0)$ (average consensus), where L is the number of nodes.*

We consider a strongly connected[1] and symmetric[2] network $\mathcal{G}$, and a doubly stochastic and symmetric weight matrix $\mathbf{W}$. The consensus algorithm converges to the average value by Proposition 1. Below, we explain the details of our DeF-GD algorithm (Algorithm III.3), including the initialization chosen for the projected GD. We note that the initialization step for Algorithm III.3 also need to be done in a federated setting. As the measurements are distributed across the nodes and since the communication is federated, we will need a federated algorithm for initialization so that all nodes are initialized to a common value. We explain this below and the pseudocode of the initialization algorithm is presented in Algorithm III.2.

*1) Federated Initialization (Algorithm III.2):* For federated initialization, we use two steps, (i) federated computation of the threshold of the indicator function and (ii) federated Power Method (PM). To compute the threshold for the indicator function, each node, $g\in\{1,\ldots,L\}$, computes

[1] A directed graph $\mathcal{G}$ is said to be strongly connected if for each ordered pair of vertices $(v_i,v_j)$ there exists an elementary path from $v_i$ to $v_j$ [23].

[2] $\mathcal{G}$ is said to be symmetric if node $g$ communicates with node $j$, then node $j$ also communicates with node $g$, for any arbitrary pair of nodes $g,j$.

**Algorithm III.2** Pseudocode for federated initialization

**Input:** $\mathcal{Y}_g, \mathbf{A}_k$, where $g = 1,2,\ldots,L$ and $k = 1,2,\ldots,L$

**Output:** $\mathbf{U}^{(0)}$, $\eta$

1: Initialize $\delta_g^{(0)} \leftarrow \sum_{k\in[q]:\mathbf{y}_k\in\mathcal{Y}_g}\sum_{i=1}^{m}\mathbf{y}_{ik}^2$, $(\mathbf{U}_g^{(0)})_0 \leftarrow (\mathbf{U}^{(0)})_0$, for $g \in [L]$

2: **return** $\delta \leftarrow L\times$ AvgConsensus $\left(\delta_1^{(0)},\ldots,\delta_L^{(0)},\mathbf{W},C\right)$

3: **for** $\ell = 0$ to $B$ **do**

4: $(\hat{\mathbf{U}}_g^{(0)})_\ell \leftarrow \sum_{k\in[q]:\mathbf{y}_k\in\mathcal{Y}_g}\frac{1}{m^2}\Big(\sum_{i=1}^{m}\mathbf{a}_{ik}\mathbf{y}_{ik}\mathbb{1}_{\{\mathbf{y}_{ik}^2\leqslant 9\delta/(mq)\}}\Big)\Big(\sum_{i=1}^{m}\mathbf{a}_{ik}\mathbf{y}_{ik}\mathbb{1}_{\{\mathbf{y}_{ik}^2\leqslant 9\delta/(mq)\}}\Big)^\top(\mathbf{U}_g^{(0)})_{\ell-1}$, for $g \in [L]$

5: **return** $\hat{\mathbf{U}}^{(0)} \leftarrow$ AvgConsensus $\left(\{(\hat{\mathbf{U}}_g^{(0)})_0\}_{g=1}^L,\mathbf{W},C\right)$

6: Obtain $\mathbf{U}^{(0)}$ by QR, i.e., compute $\hat{\mathbf{U}}^{(0)} \overset{\text{QR}}{=} \mathbf{U}^{(0)}R^{(0)}$

7: Set $(\mathbf{U}_g^{(0)})_\ell \leftarrow \mathbf{U}^{(0)}$, for all $g = 1,2,\ldots,L$

8: **end for**

9: **return** $\mathbf{U}^{(0)}$ and $\eta \leftarrow 1/\lambda_{\max}(R^{(0)})$, $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue

$\delta_g^{(0)} := \sum_{k\in[q]:\mathbf{y}_k\in\mathcal{Y}_g}\sum_{i=1}^{m}\mathbf{y}_{ik}^2$ using the measurement available to node $g$. Each node then communicates this value with its neighboring nodes and performs a distributed average consensus (step 2 of Algorithm III.2). We present the subroutine code for distributed average consensus in Algorithm III.1.

AvgConsensus takes as input $u \times v$ matrices $\mathbf{D}_1,\mathbf{D}_2,\ldots,\mathbf{D}_L$ corresponding to $L$ nodes, a doubly stochastic and symmetric weight matrix $\mathbf{W} \in \mathbb{R}^{L\times L}$, and the maximum number of iterations $C$. In each iteration, nodes update their values by taking weighted sum of its own and its neighbors' values (step 3 of Algorithm III.1). Convergence of the AvgConsensus algorithm is guaranteed by Proposition 1. In our algorithm we use consensus for scalar values (i.e., $u = v = 1$) and matrices. In the case of scalar, each node has a scalar value associated with it and the nodes communicate with neighbors to reach consensus to the average value (e.g., step 2 in Algorithm III.3). In the matrix case, each node is associated with a matrix and the nodes communicate with neighbors to reach consensus to the element-wise weighted average (e.g., step 5 in Algorithm III.3), which is a direct extension of the scalar case. The number of iterations are chosen such that the values of the nodes are within an acceptable tolerance. We obtain the threshold value of the indicator function using AvgConsensus. Once consensus of the threshold value is achieved, a federated PM algorithm is executed using the converged threshold.

As explained in [6], we plan to initialize $\mathbf{U}_0$ as the top $r$ left singular vectors of $\mathbf{X}_0$, where

$$\mathbf{X}_0 := \frac{1}{m}\Big(\sum_{k=1}^{q}\sum_{i=1}^{m}\mathbf{a}_{ik}\mathbf{y}_{ik}\mathbb{1}_{\{\mathbf{y}_{ik}^2\leqslant 9\delta/(mq)\}}\Big),$$

where $\delta := \sum_{g=1}^{L}\sum_{k\in[q]:\mathbf{y}_k\in\mathcal{Y}_g}\sum_{i=1}^{m}\mathbf{y}_{ik}^2 = L\times$ AvgConsensus $(\delta_1^{(0)},\ldots,\delta_L^{(0)},\mathbf{W},C)$. This is equivalent to initializing $\mathbf{U}_0$ as the top $r$ eigenvectors of $\mathbf{X}_0\mathbf{X}_0^\top$. In order to compute the eigenvalues of $\mathbf{X}_0\mathbf{X}_0^\top$ in a federated and decentralized manner, we perform a federated PM [6] followed by an average consensus. In the federated PM, all nodes first jointly does a random initialization $(\mathbf{U}_g^{(0)})_0 := (\mathbf{U}^{(0)})_0$ for all $g \in \{1,2,\ldots,L\}$, where $(\mathbf{U}^{(0)})_0$ is a random matrix. Then, during each PM iteration, $\ell = 1,2,\ldots,B$, node $g$ computes

$$\sum_{k\in[q]:\mathbf{y}_k\in\mathcal{Y}_g}\frac{1}{m^2}\Big(\sum_{i=1}^{m}\mathbf{a}_{ik}\mathbf{y}_{ik}\mathbb{1}_{\{\mathbf{y}_{ik}^2\leqslant 9\delta/(mq)\}}\Big)\frac{1}{m}\Big(\sum_{i=1}^{m}\mathbf{a}_{ik}\mathbf{y}_{ik}\mathbb{1}_{\{\mathbf{y}_{ik}^2\leqslant 9\delta/(mq)\}}\Big)^\top(\mathbf{U}_g^{(0)})_{\ell-1}$$

and communicates this information with its neighboring nodes. Each node now aggregates its own and the neighbors' information using the AvgConsensus as in step 5 of Algorithm III.2. The nodes perform a distributed consensus until all nodes converge to the same $(\hat{\mathbf{U}}_g^{(0)})_C$. Finally all nodes compute a QR factorization to obtain $\mathbf{U}^{(0)}$ and proceeds to the next iteration of the federated PM. The outputs of Algorithm III.2 are $\mathbf{U}_0$ and $\eta$ which serves as the initialization of the gradient descent and the step size for gradient descent, respectively, for the decentralized, federated LRCS algorithm, DeF-GD, presented in Algorithm III.3.

*2) Decentralized Projected GD (Algorithm III.3):* Using the federated initialization, we propose a decentralized projected gradient descent algorithm to reconstruct the signal matrix $\mathbf{X}^\star$. Each node update its local estimation via a negative-gradient step, by combining the local gradient computed by the node using the data available to the node, and the average of its neighbors' gradient estimates.

Let $\mathbf{X}^\star = \mathbf{U}^\star\tilde{\mathbf{B}}^\star$. We define the notations $f(\mathbf{U},\mathbf{B}) := \sum_{k=1}^{q}\sum_{i=1}^{m}(\mathbf{y}_{ik}-\mathbf{a}_{ik}^\top\mathbf{U}\mathbf{b}_k)^2$ and $f_k(\mathbf{U},\mathbf{B}) := \sum_{i=1}^{m}(\mathbf{y}_{ik}-\mathbf{a}_{ik}^\top\mathbf{U}\mathbf{b}_k)^2$.

We initialize the $\mathbf{U}$ matrix corresponding to the nodes, denoted as $\mathbf{U}_g$, as $\mathbf{U}^{(0)}$ computed in Algorithm III.2. Then in each iteration of the DeF-GD algorithm (Algorithm III.3), we update the gradient of node $g$, $\sum_{k:\mathbf{y}_k\in\mathcal{Y}_g}\nabla_{\mathbf{U}_g}f_k(\cdot,\cdot)$, denoted as $\Psi_g$, by one step of GD on $\mathbf{U}_g$, combined with a weighted average of the neighbors' information, i.e., $\Psi_j = \sum_{k:\mathbf{y}_k\in\mathcal{Y}_j}\nabla_{\mathbf{U}_j}f_k(\cdot,\cdot)$ for $j \in \mathcal{N}_g$ (steps 9 and 11). Once the local gradient updates of all nodes reach consensus to a common $\Psi$ (step:13), the $\mathbf{U}$ matrix is updated as $\hat{\mathbf{U}}^+ = \mathbf{U} - \eta\Psi$, where $\eta$ is the gradient step computed in Algorithm III.2. We then perform QR factorization to get a matrix with orthonormal columns. For each new $\mathbf{U}$, we update $\mathbf{B}$ by minimizing $f(\mathbf{U},\mathbf{B})$ over $\mathbf{B}$. For a fixed $\mathbf{U}$, we note that, the minimization of $f(\mathbf{U},\mathbf{B})$ over $\mathbf{B}$ involves solving $q$ decoupled $r$-dimensional least squares problem.

*C. Discussion:* We note that the decentralized GD approach proposed in [19] (and follow up works) for standard GD is not applicable for DLRCS. In DLRCS, we use GD to update estimates of the column span of the true matrix $\mathbf{X}^\star = \mathbf{U}^\star\mathbf{B}^\star$, i.e., span of columns of $\mathbf{U}^\star$. The $n \times r$ matrix $\mathbf{U}^\star$ is unique only up to right multiplication by an $r \times r$ rotation matrix since $\mathbf{U}^\star\mathbf{B}^\star = \mathbf{U}^\star\mathbf{Q}\mathbf{Q}^{-1}\mathbf{B}^\star$, where $\mathbf{Q}$ is a rotation matrix. Consequently, in Algorithm III.3, we use projected GD to update the subspace estimates $\mathbf{U}$; run one step of GD with respect to the cost function followed by projecting the output onto the set of matrices with orthonormal columns via QR decomposition. The approach of [19] designed for

**Algorithm III.3** Pseudo-code for proposed DeF-GD

**Input:** $\mathcal{Y}_g, \mathbf{A}_k$, where $g = 1,2,\ldots,L$ and $k = 1,2,\ldots,q$
**Parameters:** GD step size $\eta$, number of iterations $T$, error tolerance $\gamma$
**Output:** $\mathbf{U}$

Execute Algorithm III.2 and obtain $\mathbf{U}^{(0)}$
Initialize $t = 1$, $\mathbf{U}_g^{(0)} \leftarrow \mathbf{U}^{(0)}$, for all $g \in \{1,2,\ldots,L\}$
**while** $t \leqslant T$ and Err $> \gamma$ **do**
  $\mathbf{U} \leftarrow \mathbf{U}^{(t-1)}$
  **for** $g = 1$ to $L$ **do**
    Let $\mathbf{U}_g \leftarrow \mathbf{U}_g^{(t-1)}$
    **for** $\mathbf{y}_k \in \mathcal{Y}_g$ **do**
      Set $(\mathbf{b}_k)_t \leftarrow (\mathbf{A}_k\mathbf{U}_g)^{\dagger}\mathbf{y}_k$ and $(\mathbf{x}_k)_t \leftarrow \mathbf{U}_g(\mathbf{b}_k)_t$
      Compute $\nabla_{U_g} f_k(\mathbf{U}_g, (\mathbf{b}_k)_t) = \sum_{i=1}^{m}(\mathbf{y}_{ik} - \mathbf{a}_{ik}^{\top}\mathbf{U}_g(\mathbf{b}_k)_t)\mathbf{a}_{ik}(\mathbf{b}_k)_t^{\top}$
    **end for**
    $\Psi_g \leftarrow \sum_{k\in[q]:\mathbf{y}_k\in\mathcal{Y}_g} \nabla_{U_g} f_k(\mathbf{U}_g, (\mathbf{b}_k)_t)$
  **end for**
  **return** $\Psi \leftarrow$ AvgConsensus $\left(\Psi_1, \Psi_2, \ldots, \Psi_L, \mathbf{W}, C\right)$
  Set $\hat{\mathbf{U}}^{+} \leftarrow \mathbf{U} - \eta\Psi$
  Obtain $\mathbf{U}^{+}$ by QR, i.e., compute $\hat{\mathbf{U}}^{+} = \mathbf{U}^{+}R^{+}$
  Set $\mathbf{U}^{(t)} \leftarrow \mathbf{U}^{+}$ and Err $\leftarrow$ SD$(\mathbf{U}^{(t)}, \mathbf{U}^{(t-1)})$
**end while**

standard GD cannot be used for updating $\mathbf{U}$ because it involves averaging the partial estimates $\mathbf{U}_g$, $g \in \{1,2,\ldots,L\}$, 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" [3].

## IV. Simulations

In this section, we present the numerical validation of the proposed algorithm. We note that all the experiments were done using MATLAB. The communication network $\mathcal{G}$ and the dataset $\mathbf{A}_k$'s and $\mathbf{y}_k$'s were generated randomly.

We simulate the network as an Erdős Rényi graph with $L$ vertices and with probability of an edge between any pair of nodes being $p$. This means that there is an edge between any two nodes (vertices) $i$ and $j$ with probability $p$ independent of all other node pairs. For such a graph, if $p > (1+\zeta)\log L/L$, then, for large values of $L$, with high probability (w.h.p.), the graph is strongly connected) holds. The probability that this holds goes to one as $L \to \infty$. Also, if $L < (1+\zeta)\log L/L$, then, for large values of $L$, w.h.p., the graph is not strongly connected. Since the guarantees are not deterministic, for a particular simulated graph, we used the *conncomp* function in MATLAB to verify that the graph is strongly connected.

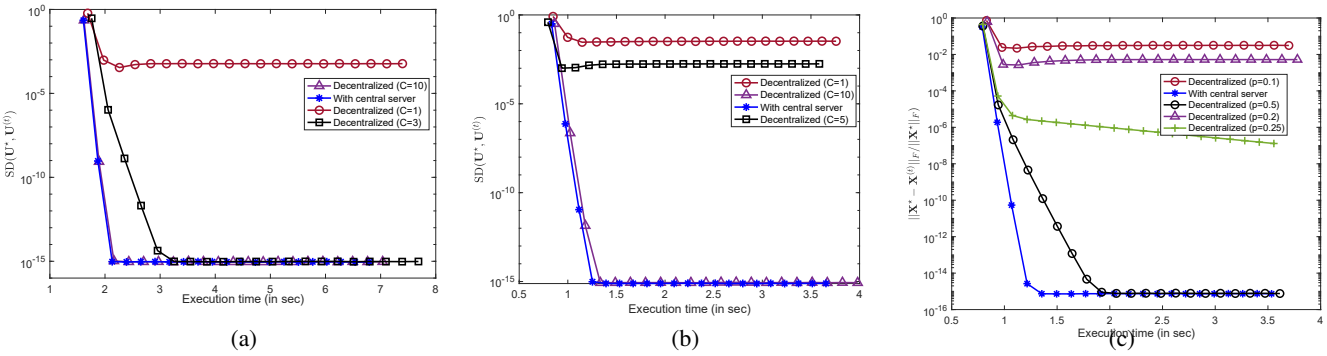
We generated the data for our experiment as follows. We note that, $\mathbf{X}^{\star} = \mathbf{U}^{\star}\tilde{\mathbf{B}}^{\star}$ , where $\mathbf{U}^{\star}$ is an $n \times r$ orthonormal matrix. We generate the entries of $\mathbf{U}^{\star}$ by orthonormalizing an i.i.d standard Gaussian matrix. Similarly, the entries of $\tilde{\mathbf{B}}^{\star} \in \mathbb{R}^{r\times q}$ are generated from a different i.i.d Gaussian distribution. The matrices $\mathbf{A}_k$s were i.i.d. standard Gaussian. We performed two experiments on the generated dataset. (1) Variation of the estimation error, denoted as $SD(\mathbf{U}^{\star}, \mathbf{U}^{(t)})$, where $\mathbf{U}^{\star}$ is the actual matrix and $\mathbf{U}^{(t)}$ is the estimate returned by our algorithm at iteration $t$, with respect to the time taken for execution for different values of $q$. (2) Variation of the estimation error, $\left\|\mathbf{X}^{\star} - \mathbf{X}^{(t)}\right\|_F / \|\mathbf{X}^{\star}\|_F$, where $\mathbf{X}^{\star}$ is the actual data matrix and $\mathbf{X}^{(t)}$ is the estimate of $\mathbf{X}^{\star}$ corresponding to the output of the algorithm at iteration $t$, with respect to the time taken for execution for different values of edge probability in the network $\mathcal{G}$.

**Experiment 1:** For this experiment, we generated the communication network $\mathcal{G}$ such that there exists a link between two nodes with probability 0.25. Thus the communication network is an Erdős Rényi graph with edge probability 0.25. We plot 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 chosen for this experiment are: $n = 100$, $r = 4$, $m = 40$, and $L = 20$. We provide results of the DeF-GD algorithm for three different values of the consensus iteration; (i) $C = 1$, (ii) $C = 3$, and (iii) $C = 10$, when $q = 400$ and $q = 200$. We ran the DeF-GD algorithm for these cases and also implemented the GDmin algorithm in [6] for the centralized case where there is a central server that performs the aggregation of the gradients of all the nodes.

The experimental results are presented in Figure 1. Figures 1a and 1b correspond to $q = 400$ and $q = 200$, respectively. From the experiments, we notice that, for a certain range of $C$, the rate of decay of error increases as $C$ increases (for $C = 1, 3$). However, for $C = 10$ the results are as good as centralized. On the other hand, consensus iterations introduces additional computational overhead resulting in the increase of the execution time. We thus infer that, the structure of the network and the partition of the data across the nodes, play a crucial role in deciding the number of iterations required for achieving consensus and consequently the amount of computations required to recover the data.

**Experiment 2:** For this experiment, we varied the edge probability of the communication network $\mathcal{G}$ and analyze the estimation error. The parameters chosen for this experiment are: $n = 100$, $r = 4$, $q = 400$, $m = 40$, $L = 20$, and $C = 200$. We plot the matrix estimation error (at the end of the iteration) $\left\|\mathbf{X}^{\star} - \mathbf{X}^{(t)}\right\|_F / \|\mathbf{X}^{\star}\|_F$ and the execution time-taken (until the end of that iteration) on the y-axis and x-axis, respectively. We provide results of the DeF-GD algorithm for four different values of the edge probability; (i) 0.1, (ii) 0.2, (iii) 0.25, and (iv) 0.5. We note that, for edge probability 0.1, the network $\mathcal{G}$ was not connected and this explains the reason for no decay in the error. On the other hand, for edge probability values $0.2, 0.25$, and 0.5, the resulting network was connected. We compared the performance of the DeF-GD algorithm with the GDmin algorithm given in [6], where there is a central server. The experimental results are presented in Figure 1c. From the results, as expected, our decentralized algorithm gives lower error and faster convergence when the network is a well connected graph.

[3]To compute the subspace mean of $\mathbf{U}_g$'s w.r.t. the subspace distance $SD(.,.)$, 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.




(a) (b) (c)

Fig. 1: Error versus execution time plot with time in seconds. We compare performance our fully decentralized algorithm (DeF-GD) by varying the number of iterations of AvgConsensus as $C = 1, 3, 10$. We also compare DeF-GD with the GDmin algorithm in [6], which is the memory efficient existing approach with guarantees when there is a central server. In Figure (1a), $n = 100$, $r = 4$, $q = 400$, $m = 40$, and $L = 20$ and in Figure (1b), $n = 100$, $r = 4$, $q = 200$, $m = 40$, and $L = 20$. In Figure 1c, we compare performance our fully decentralized algorithm (DeF-GD) by varying the probability of edge in the communication network $\mathcal{G}$ as $0.1, 0.2, 0.25$, and $0.5$. We also compare DeF-GD with the GDmin algorithm in [6], which is the memory efficient approach with guarantees when there is a central server. The parameters used are: $n = 100$, $r = 4$, $q = 400$, $m = 40$, $L = 20$, and $C = 3$.

From Figure 1c we infer that the decay rate of error increases as the probability of an edge in the network increases. As a result, the convergence rate of the DeF-GD algorithm improves, and the gap between the centralized approach and the decentralized approach decreases as the connectivity of the network increases.

## V. Conclusion

In this paper we studied the Low Rank Compressive Sensing (LRCS) problem in a fully decentralized setting, where the measurement signals are distributed across a set of nodes that are allowed to exchange their information only with a pre-specified set of neighboring nodes. We referred to this problem as the Decentralized Low Rank Phase Retrieval (DLRCS) problem. We considered the federated setting of the DLRCS problem where the nodes only share the parameters of their local estimate rather than the raw signal itself. For solving the DLRCS problem, we proposed a fully decentralized, federated algorithm, referred to as DeF-GD. Our algorithm incorporated a projected gradient descent to serve the matrix recovery part and an average consensus algorithm for achieving the collaboration of nodes. We validated the effectiveness of our algorithm on randomly generated synthetic data and compared with the existing memory efficient approach in [6], which addresses the case when there is a central server in the system. We plan to investigate convergence guarantees of the proposed DeF-GD algorithm as part of our future work.

## References


[1] R. S. Srinivasa, K. Lee, M. Junge, and J. Romberg, "Decentralized sketching of low rank matrices," *Advances in Neural Information Processing Systems*, vol. 32, pp. 10 101–10 110, 2019.

[2] 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.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[5] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," *IEEE Communications Magazine*, vol. 59, no. 2, pp. 16–21, 2021.

[6] S. Nayer and N. Vaswani, "Fast and sample-efficient federated low rank matrix recovery from column-wise linear and quadratic projections," *arXiv:2102.10217*, 2021.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[8] S. Negahban, M. J. Wainwright, *et al.*, "Estimation of (near) low-rank matrices with noise and high-dimensional scaling," *The Annals of Statistics*, vol. 39, no. 2, pp. 1069–1097, 2011.

[9] Y. Chen, Y. Chi, and A. J. Goldsmith, "Exact and stable covariance estimation from quadratic sampling via convex programming," *IEEE Transactions on Information Theory*, vol. 61, pp. 4034–4059, 2015.

[10] N. Vaswani, S. Nayer, and Y. C. Eldar, "Low-rank phase retrieval," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4059–4074, 2017.

[11] S. Nayer, P. Narayanamurthy, and N. Vaswani, "Provable low rank phase retrieval," *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5875–5903, 2020.

[12] S. Nayer and N. Vaswani, "Sample-efficient low rank phase retrieval," *IEEE Transaction on Information Theory*, 2021, to appear.

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

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

[15] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, 2011.

[16] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[17] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[18] 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.

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

[21] 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.

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

[23] R. Diestel, *Graph Theory*. Springer: New York, 2000.