A Fast Algorithm for Low Rank + Sparse column-wise Compressive Sensing

Silpa Babu, Namrata Vaswani
ECE dept, Iowa State University, USA

2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton) | 979-8-3503-2814-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/Allerton58177.2023.10313478

***Abstract*—This paper focuses studies the following low rank + sparse (LR+S) column-wise compressive sensing problem. We aim to recover an $n \times q$ matrix, $\boldsymbol{X}^* = [\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_q^*]$ from m independent linear projections of each of its q columns, given by $\boldsymbol{y}_k := \boldsymbol{A}_k \boldsymbol{x}_k^*$, $k \in [q]$. Here, $\boldsymbol{y}_k$ is an m-length vector with $m < n$. We assume that the matrix $\boldsymbol{X}^*$ can be decomposed as $\boldsymbol{X}^* = \boldsymbol{L}^* + \boldsymbol{S}^*$, where $\boldsymbol{L}^*$ is a low rank matrix of rank $r << \min(n, q)$ and $\boldsymbol{S}^*$ is a sparse matrix. Each column of $\boldsymbol{S}$ contains ρ non-zero entries. The matrices $\boldsymbol{A}_k$ are known and mutually independent for different k. To address this recovery problem, we propose a novel fast GD-based solution called AltGDmin-LR+S, which is memory and communication efficient. We numerically evaluate its performance by conducting a detailed simulation-based study.**

I. Introduction

The modeling of an unknown matrix as the sum of a low-rank (LR) matrix and a sparse matrix is widely employed in various dynamic imaging applications - foreground-background separation from videos [1], [2], dynamic MRI reconstruction [3], [4], and low-rank sketching [5] are three important example applications. The design of recommendation systems with outliers is another one [1], [2]. The low-rank component characterizes the static and slowly changing background across the frames, while the sparse component represents the sparse foreground – moving objects in the case of videos, brain activation patterns in case of functional MRI, and motion of the contrast agents across the organs in case of contrast enhanced dynamic MRI.

A. Problem and Notation

We study the problem of recovering an $n \times q$ matrix $\boldsymbol{X}^*$ from undersampled measurements of each of its columns, i.e., from

$$\boldsymbol{y}_k = \boldsymbol{A}_k \boldsymbol{x}_k^*, \quad k \in [q]$$

Here, $\boldsymbol{A}_k$ is a known matrix of dimension $m \times n$, with $m < n$. The different $\boldsymbol{A}_k$s are i.i.d. and each is a dense matrix: either a random Gaussian matrix (each entry i.i.d. standard Gaussian) or a random Fourier matrix (a random subset of rows of the discrete Fourier transform matrix). The vectors $\boldsymbol{y}_k$ and $\boldsymbol{x}_k^*$ denote the k-th column of matrix $\boldsymbol{Y}$ and $\boldsymbol{X}^*$, respectively.

We assume that $\boldsymbol{X}^* = \boldsymbol{L}^* + \boldsymbol{S}^*$, where $\boldsymbol{L}^*$ is a low rank matrix and $\boldsymbol{S}^*$ is a sparse matrix. The rank of matrix $\boldsymbol{L}^*$ is denoted by r, and we assume $r \ll \min(n, q)$ (low-rank). Each column of $\boldsymbol{S}^*$ has ρ non-zero entries. There is no bound on the magnitude of the non-zero entries of matrix $\boldsymbol{S}^*$. Therefore, the undersampled Gaussian measurements can be expressed as

$$\boldsymbol{y}_k = \boldsymbol{A}_k(\boldsymbol{l}_k^* + \boldsymbol{s}_k^*), \quad k \in [q]$$

where $\boldsymbol{l}_k^*$ and $\boldsymbol{s}_k^*$ is the k-th column of matrix $\boldsymbol{L}^*$ and $\boldsymbol{S}^*$ respectively. The above problem occurs in dynamic MRI ($\boldsymbol{A}_k$s are a subset of rows of the 2D-DFT matrix) or in sketching ($\boldsymbol{A}_k$'s are random Gaussian) [5]

The reduced (rank r) Singular Value Decomposition of matrix $\boldsymbol{L}^*$ can be represented as $\boldsymbol{U}^* \boldsymbol{\Sigma}^* \boldsymbol{V}^*$. The matrix $\boldsymbol{L}^*$ can be written as

$$\boldsymbol{L}^* = \boldsymbol{U}^* \boldsymbol{B}^*$$

where $\boldsymbol{U}^*$ is an $n \times r$ matrix with orthonormal columns, and $\boldsymbol{B}^*$ is an $r \times q$ matrix given by $\boldsymbol{B}^* = \boldsymbol{\Sigma}^* \boldsymbol{V}^*$. Consequently, each column of $\boldsymbol{X}^*$ is given by $\boldsymbol{x}_k^* = \boldsymbol{U}^* \boldsymbol{b}_k^* + \boldsymbol{s}_k^*$. Hence the undersampled Gaussian measurements can be written as

$$\boldsymbol{y}_k = \boldsymbol{A}_k(\boldsymbol{U}^* \boldsymbol{b}_k^* + \boldsymbol{s}_k^*), \quad k \in [q]$$

We use $\|.\|$ without a subscript denotes the l_2 norm of a vector or the induced l_2 norm of a matrix, and we use $\|.\|_F$ to denote the matrix Frobenius norm. For a tall matrix $\boldsymbol{M}$, $\boldsymbol{M}^\dagger := (\boldsymbol{M}^\top \boldsymbol{M})^{-1} \boldsymbol{M}^\top$ denotes its Moore-Penrose pseudo-inverse.

We studied the $\boldsymbol{S}^* = 0$ special case of the above problem in our recent work [6], [7]. Recall that $\boldsymbol{L}^* \overset{SVD}{=} \boldsymbol{U}^* \boldsymbol{\Sigma}^* \boldsymbol{V}^* := \boldsymbol{U}^* \boldsymbol{B}^*$ denote its reduced (rank r) SVD, and $\kappa := \sigma_{\max}^* / \sigma_{\min}^*$ the condition number of $\boldsymbol{\Sigma}^*$. Notice that our problem is asymmetric across rows and columns. Also, each measurement $\boldsymbol{y}_{ki}$ is a global functions of column $\boldsymbol{x}_k^*$, but not of the entire matrix. Hence, we need the following assumption, which is a subset of the assumptions used in the LR matrix completion literature.

Assumption 1.1. *We assume that* $\max_k \|\boldsymbol{b}_k^*\| \le \mu \sqrt{r/q} {\sigma_{\max}^*}^2$ *for a constant* $\mu \ge 1$.

B. Related Work

To our best knowledge, the above problem has only been studied in the dynamic MRI setting [8], [9], [10], [11]. All these algorithms are extremely slow and memory-inefficient because they requires storing and processing the entire matrix $\boldsymbol{L}$ (these do not factorize $\boldsymbol{L}$ as $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{B}$). Moreover, all these works only focused on accurately recovering real MRI sequences from the available measurements. To our knowledge, none of these do a careful simulation-based or theoretical study of when the proposed algorithm converges and why. Another related work from the MRI literature is [12]; this explored the LR&S model for MRI datasets.

While the above problem has not been explored in the theoretical or signal processing literature, related LR+S problems have been extensively studied. The robust PCA problem is the above problem with $\boldsymbol{A}_k = \boldsymbol{I}$, i.e. the goal is to separate $\boldsymbol{L}^*$ and $\boldsymbol{S}^*$ from $\boldsymbol{X}^* := \boldsymbol{L}^* + \boldsymbol{S}^*$. The robust LR matrix completion (LRMC) problem, also sometimes referred to as partially observed robust PCA, involves recovering $\boldsymbol{L}^*$ from a subset of the entries of $\boldsymbol{X}^*$. This can be understood as the above problem with $\boldsymbol{A}_k$ being a 1-0 matrix with exactly one 1 in each row. Provably correct convex [1], [13], alternating minimization [14], and GD-based solutions to robust PCA and robust LRMC [15], [16] have been developed in the literature. The work of [15] factors $\boldsymbol{L}$ as $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{B}$ and uses an alternating GD (AltGD) algorithm that alternatively updates $\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{S}$. Since $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{B} = \boldsymbol{U}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{B}$ for any $r \times r$ invertible matrix $\boldsymbol{R}$ (the decomposition is not unique), the alternating GD algorithm requires adding a term to the cost function that ensures the norms of $\boldsymbol{U}$ and $\boldsymbol{B}$ are balanced. Thus, this work develops an AltGD algorithm that minimizes $\sum_k \|\boldsymbol{y}_k - \boldsymbol{A}_k\boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{A}_k\boldsymbol{s}_k\|^2 + \|\boldsymbol{U}^\top\boldsymbol{U} - \boldsymbol{B}\boldsymbol{B}^\top\|_F$.

Another related problem is that of recovering $\boldsymbol{X}^* := \boldsymbol{L}^* + \boldsymbol{S}^*$ from dense linear projections of the entire matrix $\boldsymbol{X}^*$, i.e., from $\boldsymbol{y}_j := \mathcal{A}_j(\boldsymbol{X}^*)$, $j = 1, 2, \ldots, j_{max}$ (this can be referred as low-rank plus sparse compressive sensing) [17]. Here $\mathcal{A}_j(\boldsymbol{X}^*) := \langle \boldsymbol{A}_j, \boldsymbol{X}^* \rangle$ with $\boldsymbol{A}_j$ being random Gaussian. Notice that this problem is different from ours where we have dense linear projections of each column of $\boldsymbol{X}^*$ but not of the entire matrix.

In our recent work [6], [7], [18], we studied the LR column-wise compressive sensing (LRCCS) problem which is the above problem with $\boldsymbol{S}^* = \boldsymbol{0}$. This is also where we introduced the alternating GD and minimization (AltGDmin) algorithm idea. In [6], we provided theoretical sample complexity and iteration complexity guarantees for it and extensive numerical simulations. In [7], [18], we developed a practical modification for fast accelerated dynamic MRI reconstruction. Via extensive experiments on real MRI image sequences (with simulated Cartesian and radial sampling) and on real scanner data, we showed that AltGDmin-MRI is both significantly faster and more accurate than many state-of-the-art approaches from MRI literature.

C. Contributions

We develop a novel GD-based algorithm called Alternating GD and minimization or AltGDmin for solving the above LR+S column-wise compressive sensing problem (LR+S-CCS) and argue why it is both fast and memory and communication-efficient. The communication-efficiency claim assumes a federated setting in which subsets of $\boldsymbol{y}_k, \boldsymbol{A}_k$ are available at different distributed nodes. Using extensive simulation experiments, we also numerically evaluate the sample complexity (value of m needed for a given n, q, r, ρ) to guarantee algorithm convergence.

The AltGDmin algorithmic framework was first introduced for solving the LRCCS problem in [6]. Its idea is to split the unknown variables into two parts, $\boldsymbol{Z}_a, \boldsymbol{Z}_b$ and alternatively update each of them using GD or projected GD for $\boldsymbol{Z}_a$ and minimization for $\boldsymbol{Z}_b$. We let $\boldsymbol{Z}_b$ be the subset of variables for which the minimization can be "decoupled", i.e., subsets of $\boldsymbol{Z}_b$ are functions of only a subset of $\boldsymbol{Y}$. In problems such as LRCCS or the above LR+S-CCS problem, the decoupling is column-wise. Assume a factored representation of $\boldsymbol{L}$, i.e., $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{B}$ where $\boldsymbol{U}$ and $\boldsymbol{B}$ are matrices with r columns and rows respectively. In our current problem, $\boldsymbol{Z}_b \equiv \{\boldsymbol{B}, \boldsymbol{S}\}$ and $\boldsymbol{Z}_a = \boldsymbol{U}$. Thus, our algorithm alternates between a projected GD step for updating $\boldsymbol{U}$ and a minimization step for updating $\boldsymbol{B}$ and $\boldsymbol{S}$.

Because of the column-wise decoupling, the minimization step is about as fast as the GD step. As we argue later, the per-iteration time complexity is $mqnr$, memory complexity is $\max(n, q) \cdot \max(r, \rho)$ and the communication complexity is nr per node. Memory and communication cost wise our algorithm is much better than projected GD. Time-wise it is faster than AltMin which solves a minimization problem also for updating $\boldsymbol{U}$. This problem is coupled across all columns and hence is a much more expensive problem to fully resolve at each algorithm iteration. Replacing this by a single GD update is what makes AltGDmin much faster than AltMin.

To our knowledge, this work is the first to do a detailed simulation-based study of the above LR+S column-wise compressive sensing problem. As is well known from older compressive sensing literature for sparse recovery problems, as well as for LRCCS, algorithms developed originally for the random Gaussian $\boldsymbol{A}_k$ setting also often work with simple modifications for solving the MRI reconstruction problem, e.g., [19], [20] and [21] for CS, and [6] and [22] for LRCCS.

II. AltGDmin for L+S column-wise compressive sensing (L+S-CCS)

We are interested in developing a fast gradient descent (GD) based algorithm to find matrices $\boldsymbol{L}$ and $\boldsymbol{S}$ that minimize the cost function:

$$f(\boldsymbol{L}, \boldsymbol{S}) = \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{A}_k(\boldsymbol{l}_k + \boldsymbol{s}_k)\|^2$$

subject to the constraints (i) matrix $\boldsymbol{L}$ has rank r or less, (ii) each column of matrix $\boldsymbol{S}$ is ρ sparse. This means that the number of non-zero entries in each column $\boldsymbol{s}_k$ is at most ρ. As previously mentioned, we can impose the rank constraint implicitly by expressing the matrix $\boldsymbol{L}$ as $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{B}$ where $\boldsymbol{U}$ and $\boldsymbol{B}$ are $n \times r$ and $r \times q$ matrices. The cost function can then be written as:

$$f(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{S}) = \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{A}_k(\boldsymbol{U}\boldsymbol{b}_k + \boldsymbol{s}_k)\|^2$$

Our objective is to find matrices $\boldsymbol{U}$, $\boldsymbol{B}$, and $\boldsymbol{S}$ that minimize the cost function $f(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{S})$ while satisfying the constraint that each column of $\boldsymbol{S}$ is ρ sparse.

A. altGDmin-L+S iteration

The gradient of the cost function with respect to $\boldsymbol{U}$ is given by $\nabla_U f(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{S})$.

$$\nabla_U f(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{S}) = \sum_{k=1}^{q} \boldsymbol{A}_k^\top(\boldsymbol{A}_k(\boldsymbol{U}\boldsymbol{b}_k + \boldsymbol{s}_k) - \boldsymbol{y}_k)\boldsymbol{b}_k^\top.$$

At each new iteration, the following steps are performed:

- The matrix $\boldsymbol{U}$ is updated using projected gradient descent.
- For a given $\boldsymbol{U}$, the matrices $\boldsymbol{S}$ and $\boldsymbol{B}$ are updated by using minimization, as explained in detail in Section II-B.

These steps iteratively update $\boldsymbol{U}$, $\boldsymbol{B}$, and $\boldsymbol{S}$ to minimize the cost function and obtain better estimates. We summarize our proposed algorithm altGDmin-L+S in Algorithm 1.

B. Minimization: Updating S and B for a given U

For a given $\boldsymbol{U}$, we need to find $\boldsymbol{s}_k$ and $\boldsymbol{b}_k$ that minimize $f(\boldsymbol{U},\boldsymbol{B},\boldsymbol{S}) = \sum_{k=1}^{q} \|\boldsymbol{y}_k - \boldsymbol{A}_k(\boldsymbol{U}\boldsymbol{b}_k + \boldsymbol{s}_k)\|^2$. Observe that $\boldsymbol{s}_k, \boldsymbol{b}_k$ appear only in the k-th term. Thus,

$$\min_{\boldsymbol{B},\boldsymbol{S}} f(\boldsymbol{U},\boldsymbol{B},\boldsymbol{S}) = \sum_{k=1}^{q} \min_{\boldsymbol{s}_k,\boldsymbol{b}_k} \|\boldsymbol{y}_k - \boldsymbol{A}_k(\boldsymbol{U}\boldsymbol{b}_k + \boldsymbol{s}_k)\|^2$$

This means that the minimization problem can be decoupled across each columns, making it much faster. Moreover, notice that, for a given s_k, the recovery of b_k is a standard least squares (LS) problem with a closed-form solution that can be written in terms of $\boldsymbol{s}_k$:

$$\boldsymbol{b}_k = (\boldsymbol{A}_k\boldsymbol{U})^{\dagger}(\boldsymbol{y}_k - \boldsymbol{A}_k\boldsymbol{s}_k) \tag{1}$$

Recall here that $\boldsymbol{M}^{\dagger} = (\boldsymbol{M}^{\top}\boldsymbol{M})^{-1}\boldsymbol{M}^{\top}$. We can substitute this for $\boldsymbol{b}_k$ and hence get a minimization over only the $\boldsymbol{s}_k$s, i.e.,

$$\sum_{k=1}^{q} \min_{\boldsymbol{s}_k,\boldsymbol{b}_k} \|\boldsymbol{y}_k - \boldsymbol{A}_k(\boldsymbol{U}\boldsymbol{b}_k + \boldsymbol{s}_k)\|^2 = \sum_{k=1}^{q} \min_{\boldsymbol{s}_k} \|\boldsymbol{z}_k - \boldsymbol{M}_k\boldsymbol{s}_k\|^2$$

where $\boldsymbol{z}_k := \boldsymbol{y}_k - \boldsymbol{A}_k\boldsymbol{U}(\boldsymbol{A}_k\boldsymbol{U})^{\dagger}\boldsymbol{y}_k$ and $\boldsymbol{M}_k := \boldsymbol{A}_k - \boldsymbol{A}_k\boldsymbol{U}(\boldsymbol{A}_k\boldsymbol{U})^{\dagger}\boldsymbol{A}_k$. In summary,

$$\min_{\boldsymbol{B},\boldsymbol{S}} f(\boldsymbol{U},\boldsymbol{B},\boldsymbol{S}) = \sum_{k=1}^{q} \min_{\boldsymbol{s}_k} \mathcal{L}(\boldsymbol{s}_k),\ \mathcal{L}(\boldsymbol{s}_k) := \|\boldsymbol{z}_k - \boldsymbol{M}_k\boldsymbol{s}_k\|^2$$

with $\boldsymbol{z}_k, \boldsymbol{M}_k$ as given above. Thus, we have simplified the min over $\boldsymbol{B},\boldsymbol{S}$ into a problem of solving q individual problems, each of which is a standard sparse recovery problem, $\min_{\boldsymbol{s}_k} \mathcal{L}(\boldsymbol{s}_k)$. Various algorithms can be used to solve the sparse recovery problem. In this paper, we choose to use the Iterative Hard Thresholding (IHT) algorithm provided in [23] due to simplicity and efficiency. For a given $\boldsymbol{U}$ and $\boldsymbol{S}$, the matrix $\boldsymbol{B}$ can be updated using (1).

C. altGDmin-L+S initialization

Since the cost function $f(\boldsymbol{U},\boldsymbol{B},\boldsymbol{S})$ is non-convex, it needs a careful initialization. As in all previous work on iterative algorithms for the most related L+S problem, robust PCA, we initialize $\boldsymbol{S}$ first while letting $\boldsymbol{B} = \boldsymbol{0}$. The reason this is done is there is no upper bound on entries of the sparse matrix which usually models large magnitude, but infrequently occurring, outlier entries. On the other hand, when we assume that the condition number of $\boldsymbol{L}$ is a numerical constant, we are implicitly assuming an upper bound on the entries of $\boldsymbol{L}$. Also, as noted earlier, the recovery of $\boldsymbol{s}_k$'s is decoupled across columns.

Given an initialize estimate of $\boldsymbol{S}$, we initialize $\boldsymbol{U}$ by a standard spectral initialization approach: compute $\boldsymbol{U}$ as the top r singular vectors of the matrix

$$\boldsymbol{L}_0 := [\boldsymbol{A}_1^{\top}(\boldsymbol{y}_1 - \boldsymbol{A}_1\boldsymbol{s}_1), ..., \boldsymbol{A}_k^{\top}(\boldsymbol{y}_k - \boldsymbol{A}_k\boldsymbol{s}_k), ..., \boldsymbol{A}_q^{\top}(\boldsymbol{y}_q - \boldsymbol{A}_q\boldsymbol{s}_q)]$$

Algorithm 1 *altGDmin-L+S: Low Rank plus Sparse Model. Let $\boldsymbol{M}^{\dagger} := (\boldsymbol{M}^{\top}\boldsymbol{M})^{-1}\boldsymbol{M}^{\top}$.*

```
1: Input: $y_k, A_k, k \in [q]$.
2: Initialization: Set $\tau_{0max} = 10$.
3: $s_k$=IHT($y_k, A_k, \rho, \tau_{0max}, 0$), $k \in [q]$
4:
   $L_0 := [A_1^{\top}(y_1 - A_1 s_1), ..., A_k^{\top}(y_k - A_k s_k), ..., A_q^{\top}(y_q - A_q s_q)]$
5: Set $U_0 \leftarrow$ top $r$ left singular vectors of $L_0$.
6: Let $U \leftarrow U_0$.
7: $b_k \leftarrow (A_k U)^{\dagger}(y_k - A_k s_k)$ , $k \in q$.
8: GDmin iterations: Set $T_{max} = 120$, $\tau_{max} = 3$, set $\eta = \frac{0.14}{\|\nabla_U f(U_0, B_1, S_1)\|}$
9: for $t = 1$ to $T_{max}$ do
10:    Compute: $(z_k)_t = y_k - A_k U (A_k U)^{\dagger} y_k$ and
11: $(M_k)_t = A_k - A_k U (A_k U)^{\dagger} A_k$, $k \in [q]$
12:    $(s_k)_t$=IHT($(z_k)_t, (M_k)_t, \rho, \tau_{max}, (s_k)_{t-1}$), $k \in [q]$
13:    $(b_k)_t \leftarrow (A_k U)^{\dagger}(y_k - A_k (s_k)_t)$ , $k \in q$.
14:    Gradient compute:
       $\nabla_U f(U,B,S) \leftarrow \sum_{k=1}^{q} A_k^{\top}(A_k(U(b_k)_t + (s_k))_t) - y_k)(b_k)_t^{\top}$
15:    Projected GD step : $U^{+} \leftarrow QR(U - \eta \nabla_U f(U,B,S))$.
16:    Set $U \leftarrow U^{+}$.
17: end for
18: Output: $X := [x_1, x_2, \ldots, x_q]$, where $x_k = U(b_k)_T + (s_k)_T$.
```

D. Iterative Hard Thresholding

At each iteration of IHT [23], a vector $\boldsymbol{s}$ is updated as follows:

- Compute gradient of the cost function with respect to $\boldsymbol{s}$.
- Compute the step size using the gradient.
- Update $\boldsymbol{s}$ using gradient descent.
- Perform Hard Thresholding on the updated $\boldsymbol{s}$ to keep only the largest ρ entries of $\boldsymbol{s}$ in magnitude.

We summarize IHT algorithm in Algorithm 2. The computational time for updating a vector $\boldsymbol{s}$ using the IHT algorithm in initialization mainly depends on the operator $\boldsymbol{M}$ and its transpose $\boldsymbol{M}^{\top}$. If these operators are general matrices, the computational time requirement is $O(mn)$ per iteration for each column.

E. Time, Communication, and Memory Complexity

Time Complexity. The time complexity of the IHT algorithm for recovering a vector $\boldsymbol{s}$ in the initialization is $O(mn)$ per iteration, as previously explained. Consequently, the time complexity of the IHT algorithm for updating the entire matrix $\boldsymbol{S}$ is $O(\tau_{0max}mnq)$. The initialization step requires a time complexity of mqn for computing $\boldsymbol{L}_0$. The time needed for computing the r-SVD of $\boldsymbol{L}_0$ is given by $nqr\times$(number of iterations of power method). Summing up the individual time

Algorithm 2 *IHT*

```
1: Input: $\boldsymbol{z}, \boldsymbol{M}, \rho, \tau_{max}, \boldsymbol{s}$.
2: for $t = 1$ to $\tau_{max}$ do
3:     Gradient Compute $\nabla_{\boldsymbol{s}}\mathcal{L}(\boldsymbol{s})$, which is the gradient of $\mathcal{L}(\boldsymbol{s}) = \|\boldsymbol{M}\boldsymbol{s} - \boldsymbol{z}\|^2$ with respect to $\boldsymbol{s}$.
4:     Compute step size $\mu = \frac{\|\nabla_{\boldsymbol{s}}\mathcal{L}(\boldsymbol{s})\|}{\|\boldsymbol{M}\nabla_{\boldsymbol{s}}\mathcal{L}(\boldsymbol{s})\|}$
5:     GD step for $(\boldsymbol{s})_t \leftarrow \boldsymbol{s} - \mu\nabla_{\boldsymbol{s}}\mathcal{L}(\boldsymbol{s})$
6:     $\boldsymbol{s} \leftarrow$ HardThreshold $((\boldsymbol{s})_t, \rho)$, which keep only the largest $\rho$ entries of $\boldsymbol{s}$ in magnitude.
7: end for
8: Output: $\boldsymbol{s}$.
```

complexities mentioned above, the time complexity of the initialization step is given by $O(\tau_{0max}mqn + nqr \times$(number of iterations of power method)$+mqn)$. However, since the dominant term in this expression is $O(\tau_{0max}mqn)$, the overall time complexity of initialization is $O(\tau_{0max}mqn)$.

The computational time required for updating each column of $\boldsymbol{S}$ using IHT is $O(mn)$ per iteration (explained earlier). The time complexity for $\boldsymbol{S}$ update using IHT algorithm is $O(\tau_{max}mqn)$, where τ_{max} is the maximum number of iterations in the IHT algorithm. The update of columns of $\boldsymbol{B}$ by LS also needs time $O(mnqr)$. One gradient computation with respect to $\boldsymbol{U}$ needs time $O(mqnr)$. The QR decomposition requires time nr^2. We need to repeat these steps T_{max} times. Thus, the total time complexity of altGDmin-L+S iterations is $O((mnqr + nr^2 + mnqr + \tau_{max}mqn)T_{max}) = O(mnqrT_{max})$.

Therefore, the total time complexity of the altGDmin-L+S algorithm is $O(mnqrT_{max})$.

Communication Complexity. The communication complexity per node per iteration in a distributed implementation (where subsets of $\boldsymbol{y}_k$s are processed at different distributed nodes) is given as $O(nr)$.

Memory Complexity. If each column of matrix $\boldsymbol{S}$ has only ρ non-zero entries, then storing each column requires 2ρ numbers. Since $\boldsymbol{S}$ has q columns, the total memory required to store $\boldsymbol{S}$ is $2\rho q$. The memory required to store $n \times r$ matrix $\boldsymbol{U}$ and $r \times q$ matrix $\boldsymbol{B}$ is nr and qr. In comparison, storing the entire matrix $\boldsymbol{X}$ would require memory of nq. Therefore, by storing $\boldsymbol{S}$, $\boldsymbol{U}$, and $\boldsymbol{B}$ instead of the full matrix $\boldsymbol{X}$, the algorithm can significantly reduce the memory requirements. The memory complexity of storing $\boldsymbol{U}$, $\boldsymbol{B}$, and $\boldsymbol{S}$ is given by $\max(nr, qr, q\rho)$.

F. Settings parameters automatically

In real-world applications, the rank r of matrix $\boldsymbol{L}^*$ and the exact sparsity of each column in matrix $\boldsymbol{S}$ are unknown. To estimate a suitable value for r, the "$b\%$ energy threshold" method can be employed on the singular values of the initial estimate of matrix $\boldsymbol{L}$ denoted as $\boldsymbol{L}_0$. Additionally, the estimated value of r should be sufficiently less than the minimum of n and q to ensure a low-rank property. To determine r, we perform the "$b\%$ energy threshold" on the first $\min\left(\frac{n}{10}, \frac{q}{10}, \frac{m}{10}\right)$ singular values. In our experiments, a 65% energy threshold was used. In practical scenarios, the exact value of ρ (the sparsity level per column) is often unknown. In many cases the maximum possible nonzero entries (upper bound on ρ), denoted $\rho_{\max}$ is known. We assume this in our code and experiments. The maximum number of iterations of the IHT algorithm in the initialization τ_{0max} is set to 10.

For the altGDmin-L+S Algorithm, the maximum number of iterations T_{max} is set to 200. The step size η for updating the matrix $\boldsymbol{U}$ is set to $\frac{0.14}{\|\nabla_U f(\boldsymbol{U}_0, \boldsymbol{B}_1, \boldsymbol{S}_1)\|}$, where $\boldsymbol{U}_0$ is the initial estimate, $\boldsymbol{B}_1$ and $\boldsymbol{S}_1$ are the estimates in the first AltGDmin-L+S iteration. Assuming that the gradient norm decreases over iterations, this implies that at any iteration t, we have $\eta\nabla_U f(\boldsymbol{U}_t, \boldsymbol{B}_t, \boldsymbol{S}_t) < 1$. The maximum number of iterations for the IHT algorithm, denoted as τ_{max}, used for updating $\boldsymbol{S}$ is set to 3.

III. Experiments

The matrix $\boldsymbol{U}^*$ is generated by orthonormalizing an $n \times r$ matrix with independent and identically distributed (i.i.d) Gaussian entries. The matrix $\boldsymbol{B}^*$ and $\boldsymbol{A}_k$ are matrices of normally distributed random numbers of dimensions $r \times q$ and $m \times n$, respectively. The locations of ρ non zero entries of each column of matrix $\boldsymbol{S}$, is sampled uniformly at random, without replacement, from the integers 1 to n. Each column of $\boldsymbol{S}^*$ contains exactly ρ non-zero entries. In each of the 100 Monte Carlo runs, the measurement matrices $\boldsymbol{A}_k$ consist of i.i.d standard Gaussian entries. We obtained the Gaussian measurements as $\boldsymbol{y}_k = \boldsymbol{A}_k(\boldsymbol{U}^*\boldsymbol{b}_k^* + \boldsymbol{s}_k^*)$, $k \in [q]$. The error is computed as $Error(\boldsymbol{X}^*, \boldsymbol{X}) = \frac{\|\boldsymbol{X}^* - \boldsymbol{X}\|_F}{\|\boldsymbol{X}^*\|_F}$,where $\|\cdot\|_F$ denotes the Frobenius norm. For IHT implementation, we used the modified version of code provided in [24].

For Fig. 1, the parameters were set as follows: $T_{max} = 200$, $\tau_{0max} = 10$, $\tau_{max} = 3$, and $\eta = \frac{0.14}{\|\nabla_U f(\boldsymbol{U}_0, \boldsymbol{B}_1, \boldsymbol{S}_1)\|}$. In this experiment, we fixed the parameter values to $n = 600$, $q = 600$, $m = 80$, $r = 4$. Data was generated for different ρ values, specifically $\rho = 2, 5, 6$ and 7. We considered $\rho_{\max} = 7$. The values of each nonzero element of $\boldsymbol{S}^*$ were drawn uniformly from the interval $[-\alpha, \alpha]$, where α was chosen to be 6. This choice of values for the entries of $\boldsymbol{S}^*$ ensures that certain sparse components are smaller or larger than certain entries of $\boldsymbol{L}$, while some other components fall within the range of $\boldsymbol{L}$. This generation method for matrix $\boldsymbol{S}$ is referred to as S1 in this paper to simplify the explanation. On the y-axis, we display the empirical average of the error of matrix $\boldsymbol{X}$ using a semilog scale, which is averaged over 100 Monte Carlo simulations. Notably, from Fig. 1, it can be observed that error converges to 10^{-15} in most cases. However, when ρ is sufficiently large, our algorithm only converges to 10^{-3}.

In Fig. 2, we used the same set of parameters as in Fig. 1. The data was generated using the following parameter values: $n = 600$, $q = 600$, $m = 80$, $r = 4$, $\rho = 2$. We considered $\rho_{\max} = 5$. In this experiment, the non-zero entries of matrix $\boldsymbol{S}^*$ were generated using two methods: (1) S1, as explained earlier, and (2) we referred second method as S2, where each nonzero element of $\boldsymbol{S}^*$ was randomly chosen from the set $\{-1, -10, -100, 1, 10, 100\}$. In this experiment, our aim was to compare altGDmin-L+S with the well-known L+S-Lin [9] algorithm in the MRI literature, using two different methods (S1 and S2) for generating nonzero entries matrix $\boldsymbol{S}$. To conduct this comparison, we used the code provided by the authors for L+S-Lin [9], which is available on GitHub at https://github.com/JeffFessler/reproduce-l-s-dynamic-mri/tree/main; we used its version developed for dynamic MRI of abdomen. This code fails completely for our current simulated random Gaussian measurements' setting. In general too this algorothm needs application specific parameter tuning. To make it work, we made necessary modifications to the code to accommodate Gaussian measurements instead of Fourier measurements; and we provided it with our estimate of the rank of $\boldsymbol{L}$ (computed as explained earlier) and the value of $\rho_{\max}$. We compare its performance with our proposed algorithm, AltGDmin-L+S, in Fig. 2. With these changes, the recovery error of L+S-Lin goes down to about 0.0001 but the algorithm does not converge. AltGDmin-L+S does converge.

Next we compared our proposed algorithm with both L+S-Lin (its modified version explained above) [9] and with our old algorithm altGDmin-LR [6] designed for LRCCS problem for various values of m. We show the results in Table I. The data was generated using the following parameter values: $n = 400$, $q = 400$, $r = 4$, and $\rho = 2$. We considered $\rho_{\max} = 5$ and used the same parameter settings as in Fig. 2, except for T_{max}, which was changed to 10. The matrix $\boldsymbol{S}$ was generated using method S1, and Fourier measurements were used, indicating that the matrices $\boldsymbol{A}_k$ were random Fourier measurements. We compare the performance of the altGDmin-L+S (proposed) algorithm with other two algorithms, L+S-Lin [9] and altGDmin-LR [6], for different values of m. From table, it is clear that the altGDmin-L+S algorithm performs better than L+S-Lin and altGDmin-LR in Fourier settings.

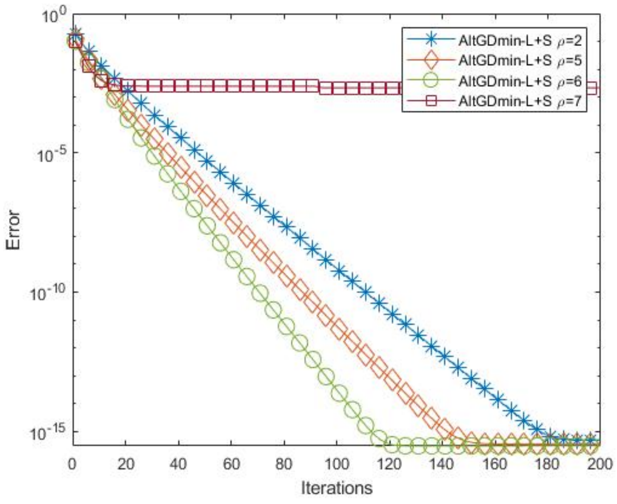

Fig. 1: **Random Gaussian Measurements:** *We compare altGDmin-L+S performance for different values of* ρ. *Parameters used:* $n = 600$, $q = 600$, $m = 80$, $r = 4$, $\rho_{\max} = 7$, $T_{max} = 200$.

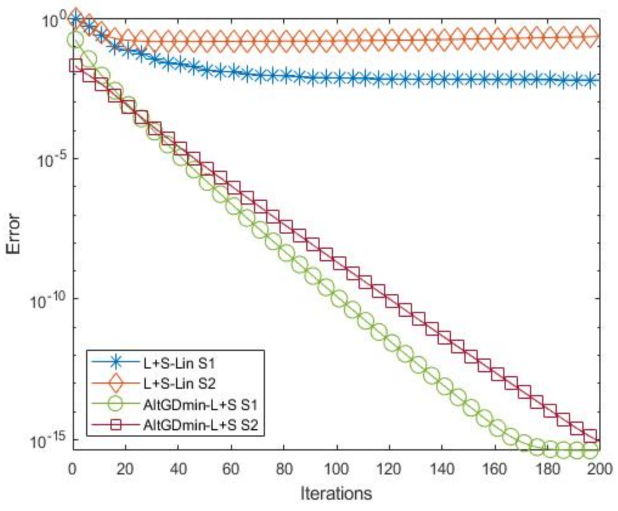

Fig. 2: **Random Gaussian Measurements:** *We compare the performance of altGDmin-L+S and L+S-Lin [9] algorithm, using two different methods (S1 and S2) for generating nonzero entries Sparse matrix. Parameters used:* $n = 600$, $q = 600$, $m = 80$, $r = 4$, $\rho = 2$, $\rho_{\max} = 5$, $T_{max} = 200$.

m	AltGDmin	L+S-Lin	AltGDmin-L+S
40	1	0.6986	0.4299
60	1	0.5054	0.0366
80	1	0.3153	0.0093
100	1	0.1749	0.0037
150	1	0.0685	0.0005
200	0.9957	0.0739	0.0001
250	0.9824	0.0509	0.0000
300	0.9691	0.0281	0.0000

TABLE I: **Random Fourier Measurements:** *We report* **Error** *for algorithms AltGDmin [6], L+S-Lin [9] and AltGDmin-L+S algorithms for different values of m. Parameters used:* $n = 400$, $q = 400$, $r = 4$, $\rho = 2$, $\rho_{\max} = 5$, $T_{max} = 10$.

IV. Conclusions

We developed a fast, memory and communication-efficient gradient descent (GD) based recovery algorithm, called altGDmin-L+S for the LR+S-CCS problem. Through simulations, we have shown that altGDmin-L+S achieves successful recovery of simulated data from their sketches. In future work, we plan to evaluate the performance of altGDmin-L+S in real-time video applications.

References

[1] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, 2011.

[2] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust subspace learning: Robust pca, robust subspace tracking and robust subspace recovery," *IEEE Signal Proc. Magazine*, July 2018.

[3] Z.-P. Liang, "Spatiotemporal imaging with partially separable functions," in *4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2007, pp. 988–991.

[4] S. G. Lingala, Y. Hu, E. DiBella, and M. Jacob, "Accelerated dynamic mri exploiting sparsity and low-rank structure: kt slr," *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1042–1054, 2011.

[5] R. S. Srinivasa, K. Lee, M. Junge, and J. Romberg, "Decentralized sketching of low rank matrices," in *Neur. Info. Proc. Sys. (NeurIPS)*, 2019, pp. 10 101–10 110.

[6] S. Nayer and N. Vaswani, "Fast and sample-efficient federated low rank matrix recovery from column-wise linear and quadratic projections," *IEEE Trans. Info. Th.*, Feb. 2023.

[7] S. Babu, S. Nayer, S. G. Lingala, and N. Vaswani, "Fast low rank compressive sensing for accelerated dynamic mri," in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, 2022.

[8] R. Otazo, E. Candes, and D. K. Sodickson, "Low-rank plus sparse matrix decomposition for accelerated dynamic mri with separation of background and dynamic components," *Magnetic resonance in medicine*, vol. 73, no. 3, pp. 1125–1136, 2015.

[9] C. Y. Lin and J. A. Fessler, "Efficient dynamic parallel mri reconstruction for the low-rank plus sparse model," *IEEE transactions on computational imaging*, vol. 5, no. 1, pp. 17–26, 2018.

[10] B. Trémoulhéac, N. Dikaios, D. Atkinson, and S. R. Arridge, "Dynamic mr image reconstruction–separation from undersampled $(\mathbf{k}, t)$-space via low-rank plus sparse prior," *IEEE Transactions on Medical Imaging*, vol. 33, no. 8, pp. 1689–1701, 2014.

[11] F. Xu, J. Han, Y. Wang, M. Chen, Y. Chen, G. He, and Y. Hu, "Dynamic magnetic resonance imaging via nonconvex low-rank matrix approximation," *IEEE Access*, vol. 5, pp. 1958–1966, 2017.

[12] S. G. Lingala, Y. Hu, E. DiBella, and M. Jacob, "Accelerated dynamic mri exploiting sparsity and low-rank structure: kt slr," *Medical Imaging, IEEE Transactions on*, vol. 30, no. 5, pp. 1042–1054, 2011.

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

[14] P. Netrapalli, U. N. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, "Non-convex robust pca," in *Neur. Info. Proc. Sys. (NeurIPS)*, 2014.

[15] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust pca via gradient descent," in *Neur. Info. Proc. Sys. (NeurIPS)*, 2016.

[16] Y. Cherapanamjeri, K. Gupta, and P. Jain, "Nearly-optimal robust matrix completion," *ICML*, 2016.

[17] J. Tanner and S. Vary, "Compressed sensing of low-rank plus sparse matrices," *Applied and Computational Harmonic Analysis*, vol. 64, pp. 254–293, 2023.

[18] S. Babu, S. G. Lingala, and N. Vaswani, "Fast low rank compressive sensing for accelerated dynamic mri," *IEEE Trans. Comput. Imaging*, revised and resubmitted, 2022.

[19] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Info. Th.*, vol. 52(2), pp. 489–509, February 2006.

[20] D. Donoho, "Compressed sensing," *IEEE Trans. Info. Th.*, vol. 52(4), pp. 1289–1306, April 2006.

[21] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine*, vol. 58(6), pp. 1182–1195, December 2007.

[22] S. Babu, S. G. Lingala, and N. Vaswani, "Fast low rank column-wise compressive sensing for accelerated dynamic mri," *IEEE transactions on computational imaging*, 2023.

[23] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and computational harmonic analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[24] M. Jabbari, "Iterative hard thresholding," https://www.mathworks.com/matlabcentral/fileexchange/124415-iterative_hard_thresholding, 2023, [Online;accessed July 10, 2023].