

Foundations and Trends® in Optimization

AltGDmin: Alternating GD and Minimization for Partly-decoupled (Federated) Optimization

Suggested Citation: Namrata Vaswani (2025), "AltGDmin: Alternating GD and Minimization for Partly-decoupled (Federated) Optimization", Foundations and Trends® in Optimization: Vol. 8, No. 4, pp 333–414. DOI: 10.1561/2400000051.

Namrata Vaswani
Iowa State University
namrata@iastate.edu

This article may be used only for the purpose of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval.

now
the essence of knowledge
Boston — Delft

Contents

| | | |
|----------|--|------------|
| 1 | Introduction | 335 |
| 1.1 | Partly Decoupled Optimization Examples | 337 |
| I | Existing Optimization Solutions and AltGDmin | 342 |
| 2 | Commonly Used Optimization Algorithms | 343 |
| 2.1 | The Optimization Problem and its Federated Version | 343 |
| 2.2 | Gradient Descent (GD) | 344 |
| 2.3 | Block Coordinate Descent and Alternating Minimization (AltMin) | 345 |
| 2.4 | Non-linear Least Squares (NLLS) | 347 |
| 2.5 | Algorithm Initialization | 347 |
| 3 | AltGDmin for Partly Decoupled Optimization Problems | 349 |
| 3.1 | Partly-decoupled Optimization: Precise Definition | 349 |
| 3.2 | Alternating GD and Minimization (AltGDmin) | 350 |

| | |
|---|------------|
| II AltGDmin for Partly-decoupled Low Rank (LR) Recovery Problems: Algorithms and Guarantees | 354 |
| 4 AltGDmin for Three LR Matrix Recovery Problems | 355 |
| 4.1 Notation | 355 |
| 4.2 LRCS, LRPR, and LRMC Problems | 355 |
| 4.3 Federation | 358 |
| 4.4 AltGDmin for LRCS: Algorithm and Guarantees | 358 |
| 4.5 AltGDmin for LRPR: Algorithm and Guarantees | 363 |
| 4.6 AltGDmin for LRMC: Algorithm and Guarantees | 365 |
| III AltGDmin Analysis – Overall Proof Technique and Details for LR Problems | 369 |
| 5 General Proof Approach for any Problem | 370 |
| 6 AltGDmin for LR Problems: Overall Proof Ideas | 373 |
| 6.1 AltGDmin for any LR Matrix Recovery Problem | 374 |
| 6.2 Proof Approach: Clean and Noise-free Case | 374 |
| 6.3 Proof Approach: Noisy Gradient Approach to Deal with Nonlinear or Noisy or Attack-prone Cases | 378 |
| 7 AltGDmin for LR Problems: Proof Details | 380 |
| 7.1 Key Results Used | 380 |
| 7.2 Analyzing the Initialization Step | 381 |
| 7.3 Clean Noise-free Case | 383 |
| 7.4 Nonlinear or Noisy or Attack-prone or Outlier Corrupted Settings | 386 |
| 8 Linear Algebra and Random Matrix Theory Preliminaries | 389 |
| 8.1 Linear Algebra: Maximum and Minimum Singular Value and the Induced 2-norm | 389 |
| 8.2 Linear Algebra: Wedin and Davis-Kahan $\sin \Theta$ Theorems . | 390 |
| 8.3 Probability Results: Markov's Inequality and its Use to Prove Concentration Bounds | 391 |
| 8.4 Probability Results: Chernoff Bounding Idea | 392 |

| | | |
|-------------------|--|------------|
| 8.5 | Probability Results: Bounds on Sums of Independent Scalar r.v.s (Scalar Concentration Bounds) | 393 |
| 8.6 | Probability Results: Epsilon Netting Argument Used for Extending Union Bound to Uncountable but Compact Sets | 394 |
| 8.7 | Probability Results: Bounding Sums of Independent Matrix r.v.s (Matrix Concentration Bounds) | 395 |
| IV | Open Questions: AltGDmin and Generalized-AltGDmin for Other Partly-decoupled Problems | 398 |
| 9 | Open Questions | 399 |
| 9.1 | Guarantees for a General Optimization Problem | 399 |
| 9.2 | Generalized AltGDmin | 399 |
| 9.3 | Robust PCA and Extensions: A Partly Decoupled Example Problem for Generalized AltGDmin | 400 |
| 9.4 | Partly Decoupled Tensor LR: Tensor LR Slicewise Sensing . | 401 |
| 9.5 | Partly Decoupled Not-differentiable Problems | 403 |
| Appendix | | 406 |
| References | | 409 |

AltGDmin: Alternating GD and Minimization for Partly-decoupled (Federated) Optimization

Namrata Vaswani

Iowa State University, USA; namrata@iastate.edu

ABSTRACT

This monograph describes a novel optimization solution framework, called alternating gradient descent (GD) and minimization (AltGDmin), that is useful for many problems for which alternating minimization (AltMin) is a popular solution. AltMin is a special case of the block coordinate descent algorithm that is useful for problems in which minimization w.r.t one subset of variables keeping the other fixed is closed form or otherwise reliably solved. Denote the two blocks/subsets of the optimization variables \mathbf{Z} by $\mathbf{Z}_{slow}, \mathbf{Z}_{fast}$, i.e., $\mathbf{Z} = \{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}$. AltGDmin is often a faster solution than AltMin for any problem for which (i) the minimization over one set of variables, \mathbf{Z}_{fast} , is much quicker than that over the other set, \mathbf{Z}_{slow} ; and (ii) the cost function is differentiable w.r.t. \mathbf{Z}_{slow} . Often, the reason for one minimization to be quicker is that the problem is “decoupled” for \mathbf{Z}_{fast} and each of the decoupled problems is quick to solve. This decoupling is also what makes AltGDmin communication-efficient for federated settings.

Important examples where this assumption holds include (a) low rank column-wise compressive sensing (LRCS), low

rank matrix completion (LRMC), (b) their outlier-corrupted extensions such as robust PCA, robust LRCS and robust LRMC; (c) phase retrieval and its sparse and low-rank model based extensions; (d) tensor extensions of many of these problems such as tensor LRCS and tensor completion; and (e) many partly discrete problems where GD does not apply – such as clustering, unlabeled sensing, and mixed linear regression. LRCS finds important applications in multi-task representation learning and few shot learning, federated sketching, and accelerated dynamic MRI. LRMC and robust PCA find important applications in recommender systems, computer vision and video analytics.

1

Introduction

This monograph describes a novel algorithmic framework, called Alternating Gradient Descent (GD) and Minimization or AltGDmin for short, that is useful for optimization problems that are “partly decoupled” [37]. Consider the optimization problem $\min_{\mathbf{Z}} f(\mathbf{Z})$. This is partly-decoupled if we can split the set of optimization variables \mathbf{Z} into two blocks, $\mathbf{Z} = \{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}$, so that the minimization over \mathbf{Z}_{fast} , keeping \mathbf{Z}_{slow} fixed, is decoupled. This means that it can be solved by solving many smaller-dimensional, and hence much faster, minimization problems over disjoint subsets of \mathbf{Z}_{fast} . That over \mathbf{Z}_{slow} , keeping \mathbf{Z}_{fast} fixed, may or may not be decoupled. We provide examples below and define this mathematically in Section 3.1.

For problems for which one of the two minimizations is decoupled, and hence fast, while the other is not, AltGDmin often provides a much faster solution than the well-known Alternating Minimization (AltMin) [7, 19] approach. Even if both problems are decoupled, AltGDmin still often has a communication-efficiency advantage over AltMin when used in distributed or federated settings. This is the case when the data is distributed across the nodes in such a way that the decoupled minimization over a subset of \mathbf{Z}_{fast} also depends on the subset of data available at a node; so this can be solved locally.

Federated learning is a setting in which multiple distributed nodes or entities or clients collaborate to solve a machine learning (ML) problem and where different subsets of the data are acquired at the different nodes. Each node can only communicate with a central server or service provider that we refer to as “center” in this monograph. Communication-efficiency is a key concern with all distributed algorithms, including federated ones. Privacy is another key concern in federated learning. Both concerns dictate that the data observed or measured at each node/client be stored locally and not be shared with the center. Summaries of it can be shared with the center. The center typically aggregates the received summaries and broadcasts the aggregate to all the nodes [29]. In this monograph, “privacy” only means the following: the nodes’ raw data cannot be shared with the center and the algorithm should be such that the center cannot reconstruct the entire unknown true signal (vector/matrix/tensor).

One of the challenges in federated learning is developing algorithms that are resilient to adversarial attacks on the nodes; resilience to Byzantine attacks is especially critical. An important challenge in distributed computing settings (data is available centrally, but is distributed to nodes, e.g., over the cloud, to parallelize and hence speed up the computing) is to have algorithms that are resilient to stragglers (some worker nodes occasionally slowing down or failing) [45, 49]. As will become clear in this monograph, the design of both attack resilient and straggler resilient modifications of AltGDmin is also efficient. One example of Byzantine attack resilient AltGDmin is studied in [46].

Monograph organization. This monograph begins by giving some examples of partly decoupled optimization problems and their applications below. In Section 2, we provide a short overview of some of the popular optimization algorithms - gradient descent (GD), block coordinate descent and AltMin, and nonlinear least squares – and when these work well. All these are iterative algorithms that need an initialization. We describe common initialization approaches as well. Then, in Section 3, we precisely define a partly decoupled problem and develop and discuss the AltGDmin algorithmic framework. In the second part of this monograph, in Section 4, we provide the AltGDmin algorithm details, including initialization, for three important LR matrix recovery

problems - LR column-wise sensing, LR phase retrieval and LR matrix completion. We also state and discuss the theoretical sample and iteration complexity guarantees that we can prove for these problems. The iteration complexity helps provide total computational and communication complexity bounds. The third part of this monograph discusses proof techniques. We first provide the general proof approach that can be used to analyze the AltGDmin in Section 5 and then describe the key ideas for LR problems in Section 6. Details are in Section 7. Preliminaries used in these proofs are provided and explained in Section 8. This section provides a short overview of the most useful linear algebra and random matrix theory topics from [53] and [16]. In the last part of this monograph, Section 9 describes open questions including other problems where AltGDmin or its generalization may be useful.

1.1 Partly Decoupled Optimization Examples

We provide a few examples of partly decoupled problems.

Low rank column-wise compressive sensing (LRCS). This problem involves recovering an $n \times q$ rank- r matrix \mathbf{X}^* , with $r \ll \min(n, q)$, from column-wise undersampled (compressive) measurements, $\mathbf{y}_k := \mathbf{A}_k \mathbf{x}_k^*$, $k \in [q]$. The matrices \mathbf{A}_k are dense (non-sparse) matrices that are known. Each \mathbf{y}_k is an m -length vector with $m < n$. Let $\mathbf{Y} := [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q]$ denote the observed data matrix. We can solve this problem by considering the squared loss function. It then becomes a problem of finding a matrix \mathbf{X} of rank at most r that minimizes $\sum_{k=1}^q \|\mathbf{y}_k - \mathbf{A}_k \mathbf{x}_k\|_2^2$. Suppose that r or an upper bound on it is known. This problem can be converted into an unconstrained, and smaller dimensional, one by factorizing \mathbf{X} as $\mathbf{X} = \mathbf{U} \mathbf{B}$, where \mathbf{U} and \mathbf{B} are matrices with r columns and rows respectively. Thus, the goal is to solve

$$\arg \min_{\mathbf{U}, \mathbf{B}} f(\mathbf{U}, \mathbf{B}) := \arg \min_{\mathbf{U}, \mathbf{B}} \sum_{k=1}^q \|\mathbf{y}_k - \mathbf{A}_k \mathbf{U} \mathbf{b}_k\|_2^2. \quad (1.1)$$

Notice that \mathbf{b}_k appears only in the k -th term of the above summation. Thus, if we needed to minimize over \mathbf{B} , while keeping \mathbf{U} fixed, the

problem decouples column-wise. The opposite is not true. We refer to such a problem as a partly decoupled problem.

In solving the above problem iteratively, there can be numerical issues because $\mathbf{U}\mathbf{B} = \mathbf{U}\mathbf{R}\mathbf{R}^{-1}\mathbf{B}$ for any $r \times r$ invertible matrix \mathbf{R} . The norm of \mathbf{U} could keep increasing over iterations while that of \mathbf{B} decreases or vice versa. To prevent this, either the cost function is modified to include a norm balancing term, e.g., as in [56], or one orthonormalizes the estimate of \mathbf{U} after each update.

Three important practical applications where the LRCS problem occurs include (i) federated sketching [3, 17, 22, 23, 44, 48, 55], (ii) accelerated (undersampled) dynamic MRI with the low rank (LR) model on the image sequence, and (iii) multi-task linear representation learning to enable few shot learning [18, 20, 46, 50]. In fact, some works refer to the LRCS problem as multi-task representation learning. (iv) The LRCS problem also occurs in for parameter estimation in multi-task linear bandits [33].

Low rank phase retrieval (LRPR). This is the phaseless extension of LRCS [37, 39, 40] but it was studied in detail before LRCS was studied. This involves solving

$$\arg \min_{\mathbf{U}, \mathbf{B}} f(\mathbf{U}, \mathbf{B}) := \arg \min_{\mathbf{U}, \mathbf{B}} \sum_{k=1}^q \|\mathbf{y}_k - |\mathbf{A}_k \mathbf{U} \mathbf{b}_k| \|^2_2 \quad (1.2)$$

where $|\cdot|$ computes the absolute value of each vector entry. LRPR finds applications in dynamic Fourier ptychography [26, 27].

LR matrix completion (LRMC). In this case, the cost function is partly decoupled w.r.t. both \mathbf{U} and \mathbf{B} (keeping the other fixed). This involves recovering a LR matrix from a subset of its observed entries. Letting Ω denote the set of observed matrix entries, and letting \mathcal{P}_Ω denote the linear projection operator that returns a matrix of size $n \times q$ with the unobserved entries set to zero, this can be expressed as a problem of learning \mathbf{X}^* from $\mathbf{Y} := \mathcal{P}_\Omega(\mathbf{X}^*)$. Letting the unknown \mathbf{X} as $\mathbf{X} = \mathbf{U}\mathbf{B}$ as above, the optimization problem to solve now becomes:

$$\begin{aligned}
\arg \min_{\mathbf{U}, \mathbf{B}} f(\mathbf{U}, \mathbf{B}) &:= \|\mathbf{Y} - \mathcal{P}_{\Omega}(\mathbf{X}^*)\|_F^2 \\
&= \sum_{k=1}^q \|\mathbf{y}_k - \mathcal{P}_{\Omega_k}(\mathbf{U} \mathbf{b}_k)\|_2^2 \\
&= \sum_{j=1}^n \|\mathbf{y}^j - \mathcal{P}_{\Omega^j}(\mathbf{u}^{j\top} \mathbf{B})\|_2^2
\end{aligned} \tag{1.3}$$

with $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_k, \dots, \mathbf{b}_q]$, $\mathbf{U}^\top = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_j, \dots, \mathbf{u}_n]$, $\Omega_k := \{j : (j, k) \in \Omega\}$ and $\Omega^j := \{k : (j, k) \in \Omega\}$. Notice that the above problem is decoupled over \mathbf{B} for a given \mathbf{U} , and vice-versa. LRMC finds important applications in recommender systems' design, survey data analysis, and video inpainting [11]. LRMC also finds applications in parameter estimation for reinforcement learning, in particular for filling in the missing entries of its state transition probability matrix.

Other partly-decoupled examples. Other examples of partly decoupled problems include non-negative matrix factorization, sparse PCA, robust PCA and extensions (robust LRCS and robust LRMC), tensor LR slice-wise sensing and its robust extension, and LR tensor completion; and certain partly discrete problems – clustering, shuffled or unlabeled sensing, and mixed linear regression. We describe these in Section 9.

1.1.1 Detailed Description of Some Applications

Why the LR model? Medical image sequences change slowly over time and hence these are well modeled as forming a low-rank matrix with each column of the matrix being one vectorized image [5, 34]. The same is often also true for similar sets of natural images and videos [12, 36]. The matrix of user ratings of different products, e.g., movies, is modeled as a LR matrix under the commonly used hypothesis that the ratings are explained by much fewer factors than the number of users, q , or products, n [11]. In fact, many large matrices are well modeled as being LR [51]; these model any image sequence or product ratings or survey dataset, in which most of the differences between the different images or ratings or survey data, q , are explained by only a small number r of factors.

MRI. In MRI, which is used in medicine for cross-sectional imaging of human organs, after some pre-processing, the acquired data can be modeled as the 2D discrete Fourier transform (FT) of the cross-section being imaged. This is acquired one FT coefficient (or one row or line of coefficients) at a time [9, 35]. The choice of the sampled coefficients can be random or it may be specified by carefully designed trajectories. The goal is to reconstruct the image of the cross-section from this acquired data. If we can reconstruct accurately from fewer samples, it means that the acquisition can be speeded up. This is especially useful for dynamic MRI because it can improve the temporal resolution for imaging the changes over time, e.g. the beating heart. Accelerated dynamic MRI involves doing this to recover a sequence of q images, \mathbf{x}_k^* , $k \in [q]$, say, of the beating heart or of brain function as brain neurons respond to a stimuli, or of the vocal tract (larynx) as a person speaks, from undersampled DFT measurements \mathbf{y}_k , $k \in [q]$. Here \mathbf{x}_k^* is a vectorized image. The matrices \mathbf{A}_k are the partial Fourier matrices represented by the 2D DFT (or sometimes the FT in case of radial sampling) computed at the specified frequencies.

Multi-task learning. Multi-task representation learning refers to the problem of jointly estimating the model parameters for a set of related tasks. This is typically done by learning a common lower-dimensional “representation” for all of their feature vectors. This learned representation can then be used for solving the meta-learning or learning-to-learn problem: learning model parameters in a data-scarce environment. This strategy is referred to as “few-shot” learning. In recent work [20], a very interesting low-dimensional linear representation was introduced and the corresponding low rank matrix learning optimization problem was defined. This linear case will be solved if we can solve (1.1). Simply said, this can be understood as a problem of jointly learning the coefficients’ for q related linear regression problems, each with their own dataset \mathbf{A}_k , and with the regression vectors \mathbf{x}_k^* being correlated (so that low rank is a good model on the matrix formed by these vectors, \mathbf{X}^*). Once the “common representation” (the column span subspace matrix \mathbf{U}) can be estimated, we can solve a new linear regression problem that is related

(correlated) with these hold ones by only learning a new r -dimensional vector \mathbf{b}_k for it.

Federated sketching. For the vast amounts of data acquired on smartphones/other devices, there is a need to compress/sketch it before it can be stored or transmitted. The term “sketch” refers to a compression approach, where the compression end is very inexpensive [3, 17, 22, 23, 44, 48, 55]. A common approach to sketching, that is especially efficient in distributed settings, is to multiply each vectorized image by a different independent $m \times n$ random matrix (typically random Gaussian or Rademacher matrix) with $m < n$, and to store or transmit this sketch.

Part I

**Existing Optimization
Solutions and AltGDmin**

2

Commonly Used Optimization Algorithms

2.1 The Optimization Problem and its Federated Version

Our goal is to solve

$$\arg \min_{\mathbf{Z}} f(\mathbf{Z}; \mathcal{D})$$

Here \mathcal{D} is the available data.

Federation. We assume that there are a total of γ distributed nodes, and that disjoint subsets of data are observed / sensed / measured at the different nodes. We use \mathcal{D}_ℓ to denote the subset of the data available at node ℓ . Thus $\cup_{\ell=1}^{\gamma} \mathcal{D}_\ell = \mathcal{D}$ is all the available data and \mathcal{D}_ℓ s are disjoint. We assume that $f(\mathbf{Z}; \mathcal{D})$ is a sum of γ functions, each of which depends on a subset of the data \mathcal{D}_ℓ , i.e., that

$$f(\mathbf{Z}; \mathcal{D}) = \sum_{\ell=1}^{\gamma} f_\ell(\mathbf{Z}; \mathcal{D}_\ell)$$

In the partly decoupled problems that we study, f_ℓ is a function of only a subset of \mathbf{Z} , e.g., in LRCS, f_ℓ depends only on \mathbf{U}, \mathbf{b}_k . We clarify this later.

2.2 Gradient Descent (GD)

The simplest, and most popular, class of solutions for solving an unconstrained (convex or non-convex) optimization problem is gradient descent (GD) [6, 13] and its modifications, the most common being various versions of stochastic GD. GD starts with an initial guess and attempts to move little steps in the direction opposite to that of the gradient of the cost function at the previous iterate. Pseudo-code for simple GD is as follows.

- Initialize \mathbf{Z} to $\hat{\mathbf{Z}}$.
- Run the following steps T times, or until a stopping criterion is reached.
 1. Update \mathbf{Z} by one GD step: $\hat{\mathbf{Z}} \leftarrow \hat{\mathbf{Z}} - \eta \nabla_{\mathbf{Z}} f(\hat{\mathbf{Z}}; \mathcal{D})$. Here η is the GD step size, also often referred to as the learning rate.

The stopping criterion usually involves checking if $\|\nabla_{\mathbf{Z}} f(\hat{\mathbf{Z}})\| / \|\hat{\mathbf{Z}}\|$ is small enough.

- Federated setting: At each algorithm iteration, each node ℓ computes the partial gradient $\nabla_{\mathbf{Z}} f_{\ell}(\hat{\mathbf{Z}}; \mathcal{D}_{\ell})$ and sends it to center, which computes $\nabla f = \sum_{\ell} \nabla f_{\ell}$ and the GD update.

For strongly convex cost functions, GD provably converges to the unique global minimizer of $f(\mathbf{Z})$ starting from any initial guess. For convex functions, one can prove that it converges to a global minimizer (there are multiple global minimizers in this case). For non-convex functions, convergence to a global minimizer cannot be guaranteed. One can only show that GD will converge to a stationary point of the cost function under mild assumptions. All the above results require that the GD step size is small enough [6].

In signal processing and machine learning, the goal is to learn the “true solution” which is one of many local minimizers of the specified cost function. Henceforth, we refer to this as the “desired minimizer”. For certain classes of non-convex cost functions that are “nice”, such as those that arise in phase retrieval [10, 15] or in various LR matrix recovery

problems, one can prove results of the following flavor. If the available number of data samples is large enough, if the step size is small enough, and if the initialization is within a certain sized window of the desired minimizer, then, the GD estimate will converge to the desired minimizer, with high probability (w.h.p.), e.g., see [10, 15, 56]. Such results are also non-asymptotic and provide an order-wise bound on the iteration complexity (number of iterations needed by the algorithm to get within ϵ normalized distance of the desired minimizer). However, in some cases, such results require a very small step size, e.g., for phase retrieval in [10]. This, in turn, means that GD has high iteration complexity. In other cases, such as for LRCS, it is not possible to show that a GD algorithm converges at all [37].

2.3 Block Coordinate Descent and Alternating Minimization (Alt-Min)

Coordinate descent involves minimizing over one scalar optimization variable at a time, keeping others fixed, and repeating this sequentially for all variables. Block Coordinate Descent or BCD involves doing this for blocks (subsets) of variables, instead of one variable at a time. AltMin is a popular special case of BCD that splits the variables into two blocks $\mathbf{Z} = \{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}$. This is extensively used and studied theoretically because of its simplicity [7, 19]. It solves $\arg \min_{\mathbf{Z}} f(\mathbf{Z}) = \arg \min_{\{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}} f(\mathbf{Z}_{slow}, \mathbf{Z}_{fast})$ using an iterative algorithm that starts with initializing \mathbf{Z}_{slow} , and then alternatively updates \mathbf{Z}_{fast} and \mathbf{Z}_{slow} using minimization over one of them keeping the other fixed. The following is pseudo-code for AltMin.

- Initialize \mathbf{Z}_{slow} to $\hat{\mathbf{Z}}_{slow}$.
- Alternate between the following two steps T times (or until a stopping criterion is reached).
 1. Update \mathbf{Z}_{fast} keeping \mathbf{Z}_{slow} fixed: $\hat{\mathbf{Z}}_{fast} \leftarrow \min_{\mathbf{Z}_{fast}} f(\hat{\mathbf{Z}}_{slow}, \mathbf{Z}_{fast})$
 2. Update \mathbf{Z}_{slow} keeping \mathbf{Z}_{fast} fixed: $\hat{\mathbf{Z}}_{slow} \leftarrow \min_{\mathbf{Z}_{slow}} f(\mathbf{Z}_{slow}, \hat{\mathbf{Z}}_{fast})$
- Federated setting: This is not easy and has to be considered on a problem-specific basis. For most problems it is not efficient.

BCD is a generalization of this algorithm to the case when \mathbf{Z} needs to be split into more than two subsets for the individual optimizations to be closed form or otherwise reliably solvable.

AltMin works well, and often can be shown to provably converge, for problems in which the two minimizations can either be solved in closed form, or involve use of a provably convergent algorithm. Bilinear problems, such as the LRCS and LRMC problems described earlier, are classic examples of settings in which each of the minimization problems is a Least Squares (LS) problem, and hence, has a closed form solution [40, 41]. LRPR [40] is an example problem in which one minimization is LS while the other is a standard phase retrieval problem, with many provably correct iterative algorithms, e.g., [15]. A second class of problems where AltMin is used, while GD is not even applicable, is those for which the cost function is not differentiable w.r.t. some of the variables. Clustering is one example of such a problem; the k-means clustering algorithm is an AltMin solution.

As explained above in Section 2.2, for some of the problems for which both AltMin and GD are applicable, either GD cannot be shown to converge or it requires a very small step size to provably converge, making its iteration complexity too high. For certain classes of “nice” problems such as the LR problems described earlier, when initialized carefully, AltMin makes more progress towards the minimizer in each iteration, and hence converges faster: it can be shown to have an iteration complexity that depends logarithmically on the final error level. However, typically, AltMin is much slower per iteration than GD. An exception is problems in which both the minimization problems of AltMin are decoupled and hence very fast.

Moreover, a federated or distributed modification of AltMin is almost never efficient. This is the case even when both minimizations are decoupled like for LRMC. The reason is one of the minimization steps will require using data from multiple nodes. This will either require use of multiple GD iterations to solve the minimization problem (slow and communication-inefficient) or it will require all nodes to send their raw data to the center which distributes it (communication-inefficient and not private either).

2.4 Non-linear Least Squares (NLLS)

The Non-linear Least Squares (NLLS) approach was originally developed within the telecommunications literature for frequency estimation and related problems (all low-dimensional problems) [30]. This also splits \mathbf{Z} into two blocks, $\mathbf{Z} = \{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}$ and solves problems in which, for any value of \mathbf{Z}_{slow} , the minimization over \mathbf{Z}_{fast} is an over-determined least squares (LS) problem. For such problems, the NLLS approach substitutes a closed form expression for $\hat{\mathbf{Z}}_{fast}$ in terms of \mathbf{Z}_{slow} into the original cost function. It then minimizes the new cost function over \mathbf{Z}_{slow} using GD or one of its modifications such as the Newton method. To be precise, it uses GD (or Newton's method or any solver) to solve

$$\min_{\mathbf{Z}_{slow}} f(\mathbf{Z}_{slow}, \hat{\mathbf{Z}}_{fast}(\mathbf{Z}_{slow}))$$

where $\hat{\mathbf{Z}}_{fast}(\mathbf{Z}_{slow}) = \arg \min_{\mathbf{Z}_{fast}} f(\mathbf{Z}_{slow}, \mathbf{Z}_{fast})$ is the closed form for the LS solution for \mathbf{Z}_{fast} in terms of \mathbf{Z}_{slow} .

NLLS is easy to use for problems for which it is easy to compute the gradient of $\tilde{f}(\mathbf{Z}_{slow}) := f(\mathbf{Z}_{slow}, \hat{\mathbf{Z}}_{fast}(\mathbf{Z}_{slow}))$ w.r.t. \mathbf{Z}_{slow} . However, for problems like LRCS, the new cost function is $\tilde{f}(\mathbf{U}) := \sum_k \|\mathbf{y}_k - \mathbf{A}_k \mathbf{U} (\mathbf{A}_k \mathbf{U})^\dagger \mathbf{y}_k\|^2$, with $\mathbf{M}^\dagger := (\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top$. The gradient of this new cost function does not have a simple expression. Moreover, it is expensive to compute, and it makes the algorithm too complicated to analyze theoretically. To our knowledge, guarantees do not exist for NLLS.

2.5 Algorithm Initialization

All iterative algorithms require an initialization. If the cost function is strongly convex, it has a unique minimizer. In this case, any initialization will work. For all other cases, the solution that an iterative algorithm converges to depends on the initialization.

There are a few possible ways that an algorithm can be initialized. The most common approach is random initialization. In this case, one runs the entire algorithm with multiple random initializations and stores the final cost function value for each. The output corresponding to the initialization that results in the smallest final cost is then chosen.

In some other settings, some prior knowledge about \mathbf{Z}_{slow} is available and that is used as the initialization. For example, in multi-modal imaging, an approximate image estimate may be available from one source and that can serve as an initialization.

For a large number of structured signal (vector, matrix, or tensor) recovery problems and for phase retrieval problems, one can come up with a carefully designed spectral initialization: one computes the top, or top few, singular vectors of an appropriately defined matrix, that is such that the top singular vector(s) of its expected value are equal to, or close to, the unknown quantity of interest, or to a part of it.

3

AltGDmin for Partly Decoupled Optimization Problems

We first precisely define a partly decoupled problem and then develop the AltGDmin framework.

3.1 Partly-decoupled Optimization: Precise Definition

Consider an optimization problem $\arg \min_{\mathbf{Z}} f(\mathbf{Z})$. Suppose, as before, that \mathbf{Z} can be split into two blocks $\mathbf{Z} = \{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}$, such that optimization over one keeping the other fixed can be correctly solved (has a closed form or provably correct iterative solution). We say this problem is partly decoupled (is decoupled for \mathbf{Z}_{fast}) if

$$f(\mathbf{Z}; \mathcal{D}) := f(\mathbf{Z}_{slow}, \mathbf{Z}_{fast}; \mathcal{D}) = \sum_{\ell=1}^{\gamma} f_{\ell}(\mathbf{Z}_{slow}, (\mathbf{Z}_{fast})_{\ell}; \mathcal{D}_{\ell})$$

with $(\mathbf{Z}_{fast})_{\ell}$ being disjoint subsets of the variable set \mathbf{Z}_{fast} , e.g., in case of the LR problems described earlier, these are different columns of the matrix \mathbf{B} .

Thus, partial decoupling implies that the minimization over \mathbf{Z}_{fast} (keeping \mathbf{Z}_{slow} fixed at its previous value, denoted $\hat{\mathbf{Z}}_{slow}$) can be solved by solving γ smaller dimensional optimization problems, i.e.,

$$\min_{\mathbf{Z}_{fast}} f(\hat{\mathbf{Z}}_{slow}, \mathbf{Z}_{fast}; \mathcal{D}) = \sum_{\ell=1}^{\gamma} \min_{(\mathbf{Z}_{fast})_{\ell}} f_{\ell}(\hat{\mathbf{Z}}_{slow}, (\mathbf{Z}_{fast})_{\ell}; \mathcal{D}_{\ell})$$

The computation cost of most optimization problems is more than linear, and hence, the γ smaller dimensional problems are quicker to solve, than one problem that jointly optimizes over all of \mathbf{Z}_{fast} . Moreover, notice that the minimization over $(\mathbf{Z}_{fast})_{\ell}$ only depends on the data subset \mathcal{D}_{ℓ} . Thus, if all of \mathcal{D}_{ℓ} is available at a node, then the minimization over $(\mathbf{Z}_{fast})_{\ell}$ can be solved locally at the node itself.

Remark 3.1. It is possible that there are partly decoupled optimization problems for which the cost function is not just a sum of simpler cost functions, but is some other composite function. We attempt to define this most general case in Appendix A.

3.2 Alternating GD and Minimization (AltGDmin)

For partly decoupled problems, in recent work [37], we introduced the following *Alternating GD and Minimization (AltGDmin)* algorithmic framework.

- Initialize \mathbf{Z}_{slow} to $\hat{\mathbf{Z}}_{slow}$. Approaches discussed in Section 2.5 can be used.
- Alternate between the following two steps T times, or until a stopping criterion is reached.
 1. Update \mathbf{Z}_{fast} by minimization: $\hat{\mathbf{Z}}_{fast} \leftarrow \min_{\mathbf{Z}_{fast}} f(\hat{\mathbf{Z}}_{slow}, \mathbf{Z}_{fast})$. Because of the decoupling, this simplifies to

$$(\hat{\mathbf{Z}}_{fast})_{\ell} \leftarrow \arg \min_{(\mathbf{Z}_{fast})_{\ell}} f_{\ell}(\hat{\mathbf{Z}}_{slow}, (\mathbf{Z}_{fast})_{\ell}) \text{ for all } \ell \in [\gamma]$$

- (a) Federated setting: Each node ℓ solves the above problem locally. No data exchange needed.

2. Update \mathbf{Z}_{slow} by GD:

$$\hat{\mathbf{Z}}_{slow} \leftarrow \hat{\mathbf{Z}}_a - \eta \sum_{\ell=1}^{\gamma} \nabla_{\mathbf{Z}_{slow}} f_{\ell}(\mathbf{Z}_{slow}, (\mathbf{Z}_{fast})_{\ell}; \mathcal{D}_{\ell})$$

Here η is the GD step size.

- (a) Federated setting: Each node ℓ computes the partial gradient $\nabla_{\mathbf{Z}_{slow}} f_\ell(\hat{\mathbf{Z}_{slow}}, (\hat{\mathbf{Z}_{fast}})_\ell; \mathcal{D}_\ell)$ and sends it to the center, which computes $\nabla f = \sum_\ell \nabla f_\ell$ and the GD update.

Time complexity per iteration: centralized. The time cost of gradient computation w.r.t. \mathbf{Z}_{slow} is much lower than that of solving a full minimization w.r.t. it. In addition, if the time cost of solving $\min_{\mathbf{Z}_{fast}} f(\hat{\mathbf{Z}_{slow}}, \mathbf{Z}_{fast})$ is comparable to that of computing the gradient w.r.t. \mathbf{Z}_{slow} , then, per iteration, AltGDmin is as fast as GD, and much faster than AltMin. This is the case for the LRCS problem, for example.

Communication complexity. If the data is federated as assumed earlier (data subset \mathcal{D}_ℓ is at node ℓ), then one can develop an efficient distributed federated implementation that is also more communication-efficient per iteration than AltMin, and comparable to GD. Node ℓ updates $(\mathbf{Z}_{fast})_\ell$ locally and computes its partial gradient which it shares with the center. The center needs to only sum these, implement GD (just a subtraction), and (if needed) process the final output, e.g., orthonormalize the columns of $\mathbf{Z}_{slow} = \mathbf{U}$ in case of LR recovery problems. This step is quick, of order nr^2 since \mathbf{U} is an $n \times r$ matrix.

Iteration complexity and sample complexity. It is often possible to also prove that the AltGDmin iteration complexity is only slightly worse than that of AltMin and much better than that of GD; and this is true under sample complexity lower bounds that comparable to what AltMin or GD need. The reason is the minimization over \mathbf{Z}_{fast} in each iteration helps ensure sufficient error decay with iteration, even with using a constant GD step size. For example, we have proved this for LRCS, LRPR, and LRMC; see Section 4 and Table 4.1. This claim treats the matrix condition number as a numerical constant. We should mention here that, the GD algorithm for \mathbf{U}, \mathbf{B} , referred to as Factorized GD (FactGD), does not provably converge for LRCS or LRPR; the reasons are explained in Section 4. Hence we do not have a bound on its iteration complexity. FactGD does converge for LRMC but its iteration complexity is r times worse than that of AltGDmin or AltMin.

Overall computation and communication complexity and sample complexity. As explained above, for many partly decoupled problems, one can prove that AltGDmin is as fast per iteration as GD, while having iteration complexity that is almost as good as that of AltMin with a sample complexity bound that is comparable to that of AltMin. This makes it one of the fastest algorithms in terms of total time complexity. In terms of communication cost, its per iteration cost is usually comparable to that of GD, while its iteration complexity is better, making it the most communication efficient.

As an example, for LRCS, AltGDmin is much faster and much more communication-efficient than AltMin. This is true both in terms of order-wise complexity and practically in numerical experiments. For LRMC, which is partly decoupled for both Z_{fast} and for Z_{slow} , all of AltGDmin, AltMin and GD have similar order-wise time complexity. However communication-complexity of AltGDmin is the best. Consequently, in numerical simulations on federated AWS nodes, AltGDmin is overall the fastest algorithm for large problems; see [2].

Non-differentiable cost functions. Another useful feature of AltGDmin is that it even applies for settings for which the cost function is not differentiable w.r.t. the decoupled set of variables. Some examples include clustering and unlabeled/shuffled sensing. We describe these in Section 9.5.

Byzantine-resilient or straggler-resilient modifications. Federated algorithms are often vulnerable to attacks by adversaries. One of the most difficult set of attacks to deal with is Byzantine attacks. Because AltGDmin involves exactly one round of partial gradients exchange per iteration, designing Byzantine resilient modifications for AltGDmin is easy to do and the resulting algorithm retains its efficiency properties. In recent work [46], we developed a Byzantine-resilient AltGDmin solution for federated LRCS. We postpone the discussion of these modifications to a later review.

In distributed computing, resilience to straggling nodes is an important practical requirement. Straggler resilient GD using the “gradient coding” approach has been extensively studied [45, 49]. These approaches are directly applicable also for AltGDmin.

Part II

AltGDmin for
Partly-decoupled Low Rank
(LR) Recovery Problems:
Algorithms and Guarantees

4

AltGDmin for Three LR Matrix Recovery Problems

4.1 Notation

We use $\|\cdot\|_F$ to denote the Frobenius norm, $\|\cdot\|$ without a subscript to denote the (induced) l_2 norm, $^\top$ to denote matrix or vector transpose, and $\mathbf{M}^\dagger := (\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top$. For a tall matrix \mathbf{M} , $QR(\mathbf{M})$ orthonormalizes \mathbf{M} . We use $diag(\mathbf{v})$ to create a diagonal matrix with entries given by entries of vector \mathbf{v} . For two $n \times r$ matrices $\mathbf{U}_1, \mathbf{U}_2$ that have orthonormal columns, we use

$$\begin{aligned} \text{SubsDist}_2(\mathbf{U}_1, \mathbf{U}_2) &:= \|(\mathbf{I} - \mathbf{U}_1 \mathbf{U}_1^\top) \mathbf{U}_2\|, \\ \text{SubsDist}_F(\mathbf{U}_1, \mathbf{U}_2) &:= \|(\mathbf{I} - \mathbf{U}_1 \mathbf{U}_1^\top) \mathbf{U}_2\|_F \end{aligned}$$

as two measures of Subspace Distance (SD). Clearly, $\text{SubsDist}_F \leq \sqrt{r} \text{SubsDist}_2$.

We reuse the letters c, C to denote different numerical constants in each use with the convention that $c < 1$ and $C \geq 1$.

4.2 LRCS, LRPR, and LRMC Problems

In all three problems, the goal is to recover an $n \times q$ rank- r matrix $\mathbf{X}^* = [\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_q^*]$, with $r \ll \min(n, q)$, from different types of

under-sampled linear or element-wise nonlinear functions of it. Let $\mathbf{X}^* \stackrel{\text{SVD}}{=} \mathbf{U}^* \boldsymbol{\Sigma}^* \mathbf{V}^* := \mathbf{U}^* \mathbf{B}^*$ denote its reduced (rank r) SVD, and $\kappa := \sigma_{\max}^* / \sigma_{\min}^*$ the condition number of $\boldsymbol{\Sigma}^*$. We let $\mathbf{B}^* := \boldsymbol{\Sigma}^* \mathbf{V}^*$.

4.2.1 LRCS Problem

The goal is to recover an $n \times q$, rank- r , matrix $\mathbf{X}^* = [\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_q^*]$ from independent linear projections of it, i.e., from

$$\mathbf{y}_k := \mathbf{A}_k \mathbf{x}_k^*, \quad k \in [q] \quad (4.1)$$

where each \mathbf{y}_k is an m -length vector, with $m < n$, $[q] := \{1, 2, \dots, q\}$, and the measurement/sketching matrices \mathbf{A}_k are mutually independent and known. For obtaining theoretical guarantees, each \mathbf{A}_k is assumed to be random-Gaussian: each entry of it is independent and identically distributed (i.i.d.) standard Gaussian.

Since no measurement \mathbf{y}_{ki} is a global function of the entire matrix, \mathbf{X}^* , we need the following assumption to make our problem well-posed (allow for correct interpolation across columns). This assumption is a subset of the “incoherence assumption” introduced for correctly solving the LR matrix completion problem [11, 31, 41].

Assumption 4.1 (μ -incoherence of right singular vectors). Assume that $\|\mathbf{b}_k^*\|^2 \leq \mu^2 r \sigma_{\max}^* / q$ for a numerical constant μ .

4.2.2 LRPR Problem

In LRPR, which is a generalization of LRCS, the goal is to recover \mathbf{X}^* from undersampled phaseless linear projections of its columns, i.e., from $\mathbf{z}_k := |\mathbf{y}_k|$, $k \in [q]$. Here $|\cdot|$ of a vector takes the magnitude of each element of the vector. If the vector is real-valued, then this just means that the sign is not measured. In case of Fourier ptychography, \mathbf{y}_k are complex-valued and in that case, one takes the absolute value of each complex number entry. LRPR also needs Assumption 4.1 for the same reason.

4.2.3 LRMC Problem

LRMC involves recovering an $n \times q$ rank- r matrix $\mathbf{X}^* = [\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_q^*]$ from a subset of its entries. Entry j of column k , denoted \mathbf{X}_{jk}^* , is observed, independently of all other observations, with probability p .

Let $\xi_{jk} \stackrel{\text{iid}}{\sim} \text{Bernoulli}(p)$ for $j \in [n], k \in [q]$. Then, the set of observed entries, denoted by Ω , is

$$\Omega := \{(j, k) : \xi_{jk} = 1\}$$

By setting the unobserved entries to zero, the observed data matrix $\mathbf{Y} \in \mathbb{R}^{n \times q}$ can be defined as

$$\mathbf{Y}_{jk} := \begin{cases} \mathbf{X}_{jk}^* & \text{if } (j, k) \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \quad \text{or, equivalently, } \mathbf{Y} := \mathbf{X}_{\Omega}^* \quad (4.2)$$

Here and below, \mathbf{M}_{Ω} refers to the matrix \mathbf{M} with all entries whose indices are not in the set Ω are zeroed out; while the rest of the entries remain unchanged.

We use $\Omega_k := \{j \in [n] \mid \xi_{jk} = 1\}$ to denote the set of indices of the observed entries in column k . To easily explain the AltGDmin algorithm idea, we define a diagonal 1-0 matrix $\mathbf{S}_k \in \mathbb{R}^{n \times n}$ as

$$\mathbf{S}_k := \text{diag}([\xi_{jk}, j \in [n]])$$

With this, our goal is to learn \mathbf{X}^* from

$$\mathbf{y}_k := \mathbf{S}_k \mathbf{x}_k^*, \quad k \in [q]$$

Remark 4.1. In the above, we let the matrix \mathbf{S}_k be an $n \times n$ diagonal matrix with entries being 1 or 0, only for ease of notation. It contains a lot of zero entries. The expected number of nonzero rows (diagonal entries only) in \mathbf{S}_k is pn .

We need the following assumption on the singular vectors of \mathbf{X}^* ; this is a way to guarantee that the rows and columns of \mathbf{X}^* are dense (non-sparse) [11, 31, 41]. This helps ensure that one can correctly interpolate (fill in) the missing entries even with observing only a few entries of each row or column.

Assumption 4.2 (μ -incoherence of singular vectors of \mathbf{X}^*). Assume row norm bounds on \mathbf{U}^* : $\max_{j \in [n]} \|\mathbf{u}^{*j}\| \leq \mu \sqrt{r/n}$, and column norm bounds on \mathbf{V}^* : $\max_{k \in [q]} \|\mathbf{v}_k^*\| \leq \mu \sqrt{r/q}$. Since $\mathbf{B}^* = \Sigma^* \mathbf{V}^*$, this implies that $\|\mathbf{b}_k^*\| \leq \mu \sqrt{r/q} \sigma_{\max}^*$.

4.3 Federation

We assume that there are a total of γ nodes, with $\gamma \leq q$ and each node has access to a different subset of the columns of the observed data matrix \mathbf{Y} and the corresponding matrices \mathbf{A}_k (or enough information to define them). This type of federation where the columns are distributed is often referred to as “vertical federation”. In case of LRCS, \mathbf{Y} is $m \times q$. In case of LRMC, \mathbf{Y} is $n \times q$ with a lot of zero entries. All nodes can only communicate with a central node or “center”.

We use \mathcal{S}_ℓ to denote the subset of columns of \mathbf{Y} available at node ℓ . The sets \mathcal{S}_ℓ form a partition of $[q]$. i.e., they are mutually disjoint and $\cup_{\ell=1}^{\gamma} \mathcal{S}_\ell = [q]$. Thus, the data at node ℓ ,

$$\mathcal{D}_\ell = \{\mathbf{y}_k, \mathbf{A}_k, k \in \mathcal{S}_\ell\}$$

To keep notation simple, we assume q is a multiple of γ and that $|\mathcal{S}_\ell| = q/\gamma$. Our discussion of complexities assumes $\gamma \ll q$ and treats γ as a numerical constant. Thus order $|\Omega|/\gamma$ is equal to order $|\Omega|$ with $|\Omega| \geq (n+q)r$ (the number of samples needs to be larger than the number of unknowns in rank r matrix).

For LRCS with \mathbf{A}_k being random Gaussian, the storage (or communication in case of distributed computing) required is significant, it is mnq/γ per node. For LRMC, \mathbf{S}_k is fully specified by just the set observed indices $\cup_{k \in \mathcal{S}_\ell} \Omega_k$. This is much cheaper to store or transmit with a cost of only $|\Omega|/\gamma$. The same is true for the LRCS problem for the MRI application where \mathbf{A}_k is a partial Fourier matrix; in this case only the observed frequency locations need to be stored or transmitted.

4.4 AltGDmin for LRCS: Algorithm and Guarantees

AltGDmin for LRCS was introduced and studied in parallel works [18, 37, 50] and follow-up work [52]. In [37, 52], we referred to the problem as LR column-wise compressive sensing (LRCS), while [18, 50] referred to the same problem as multi-task linear representation learning. The initialization introduced in [37, 52] is the best one (needs fewer samples for a certain accuracy level). The best sample complexity guarantee for AltGDmin is the one proved in our recent work [52].

4.4.1 AltGDmin-LRCS Algorithm

We first summarize the algorithm and then explain each step. Our development follows [37, 52]. AltGDmin for the LRCS problem involves minimizing $f(\mathbf{U}, \mathbf{B}) := \sum_{k=1}^q \|\mathbf{y}_k - \mathbf{U}\mathbf{b}_k\|^2$ over \mathbf{U}, \mathbf{B} . Clearly, this is decoupled for columns of \mathbf{B} (with holding \mathbf{U} fixed). Thus, we use $\mathbf{Z}_{\text{slow}} \equiv \mathbf{U}$, $\mathbf{Z}_{\text{fast}} \equiv \mathbf{B}$. It proceeds as follows.

1. *Spectral initialization:* We initialize \mathbf{U} by computing the top r singular vectors of the following matrix

$$\mathbf{X}_0 := \sum_k \mathbf{A}_k^\top \mathbf{y}_{k,\text{trnc}} \mathbf{e}_k^\top, \quad \mathbf{y}_{k,\text{trnc}} := \text{trunc}(\mathbf{y}_k, \alpha)$$

Here $\alpha := \tilde{C} \sum_k \|\mathbf{y}_k\|^2 / mq$ with $\tilde{C} := 9\kappa^2\mu^2$, and the function trunc truncates (zeroes out) all entries of the vector \mathbf{y}_k with magnitude greater than $\sqrt{\alpha}$, i.e., for all $j \in [n]$, $\text{trunc}(\mathbf{y}, \alpha)_j = (\mathbf{y})_j \mathbb{1}_{|\mathbf{y}_j| \leq \sqrt{\alpha}}$, with $\mathbb{1}$ being the indicator function.

2. At each iteration, update \mathbf{B} and \mathbf{U} as follows:

- (a) *Minimization for \mathbf{B} :* keeping \mathbf{U} fixed, update \mathbf{B} by solving $\min_{\mathbf{B}} f(\mathbf{U}, \mathbf{B})$. Due to the form of the LRCS model, this minimization decouples across columns, making it a cheap least squares problem of recovering q different r length vectors. It is solved as $\mathbf{b}_k = (\mathbf{A}_k \mathbf{U})^\dagger \mathbf{y}_k$ for each $k \in [q]$.
- (b) *GD for \mathbf{U} :* keeping \mathbf{B} fixed, update \mathbf{U} by a GD step, followed by orthonormalizing its columns: $\mathbf{U}^+ = QR(\mathbf{U} - \eta \nabla_{\mathbf{U}} f(\mathbf{U}, \mathbf{B}))$. Here $QR(\cdot)$ orthonormalizes the columns of its input.

Computation cost. The use of minimization to update \mathbf{B} at each iteration is what helps ensure that we can show exponential error decay with a constant step size. At the same time, due to the column-wise decoupled nature of LRCS, the time complexity for this step is only as much as that of computing one gradient w.r.t. \mathbf{U} . Both steps need

time¹ of order $mqnr$. This is only r times more than “linear time” (time needed to read the algorithm inputs, here $\mathbf{y}_k, \mathbf{A}_k$ ’s). To our knowledge, r -times linear-time is the best known time complexity for any algorithm for any LR matrix recovery problem. Moreover, due to the use of the $\mathbf{X} = \mathbf{U}\mathbf{B}$ factorization, AltGDmin is also communication-efficient. Each node needs to only send nr scalars (gradients w.r.t \mathbf{U}) at each iteration.

Understanding the initialization step. To understand the initialization step, note the following. It can be shown that $\mathbb{E}[\mathbf{X}_0] = \mathbf{X}^* \mathbf{D}(\alpha)$ where $\mathbf{D}(\alpha)$ is a diagonal $q \times q$ matrix with $\sigma_{\min}(\mathbf{D}) \geq 0.9$ with high probability (w.h.p.) [37, 52]. Thus, $\mathbb{E}[\mathbf{X}_0]$ is a rank r matrix with column-span equal to that of \mathbf{U}^* (or \mathbf{X}^*). Furthermore, it is easy to see that

$$\mathbf{X}_0 = \sum_{k=1}^q \sum_{i=1}^m \mathbf{a}_{ki} (\mathbf{a}_{ki}^\top \mathbf{x}_k^*) \mathbb{1}_{(\mathbf{a}_{ki}^\top \mathbf{x}_k^*)^2 \leq \alpha}$$

Using concentration bounds and linear algebra results,² it can be shown that, w.h.p., \mathbf{X}_0 is a good approximation of its expected value and hence, in terms of subspace distance, \mathbf{U}_0 is a good approximation of \mathbf{U}^* (column span of \mathbf{X}^*).

Sample-splitting is assumed, i.e., each new update of \mathbf{U} and \mathbf{B} uses a new independent set of measurements and measurement matrices, $\mathbf{y}_k, \mathbf{A}_k$.

4.4.2 Federated Implementation

Consider the GDmin steps. Update of \mathbf{b}_k s and \mathbf{x}_k s is done locally at the node that stores the corresponding \mathbf{y}_k . For gradient w.r.t. \mathbf{U} computation, the partial sums over $k \in \mathcal{S}_\ell$ are computed at node ℓ and

¹The LS step time is $\max(q \cdot mnr, q \cdot mr^2) = mqnr$ (maximum of the time needed for computing $\mathbf{A}_k \mathbf{U}$ for all k , and that for obtaining \mathbf{b}_k for all k) while the GD step time is $\max(q \cdot mnr, nr^2) = mqnr$ (maximum of the time needed for computing the gradient w.r.t. \mathbf{U} , and time for the QR step).

²Using sub-exponential Bernstein inequality to lower and upper bound α ; and using the sub-Gaussian Hoeffding inequality and an easy epsilon-net argument [53] to bound $\|\mathbf{X}_0 - \mathbb{E}[\mathbf{X}_0]\|$, one can argue that, w.h.p., \mathbf{X}_0 is close to its expected value if mq is large enough. This, along with using the Wedin sin θ theorem [16], and lower bounding the smallest entry of $\mathbf{D}(\alpha)$, helps bound subspace distance (SD) between \mathbf{U}_0 and \mathbf{U}^* .

transmitted to the center which adds all the partial sums to obtain $\nabla_{\mathbf{U}} f(\mathbf{U}, \mathbf{B})$. GD step and QR are done at the center. The updated \mathbf{U} is then broadcast to all the nodes for use in the next iteration. The per node time complexity is thus $mnrq_\ell$ at each iteration. The center only performs additions and a QR decomposition, which is an order nr^2 operation, in each iteration. The communication cost is order nr per node per iteration.

The initialization step can be federated by using the Power Method [24, 25] to compute the top r eigenvectors of $\mathbf{X}_0 \mathbf{X}_0^\top$. Power method starts with a random initialization and runs the iteration $\hat{\mathbf{U}}_0 \leftarrow QR(\mathbf{X}_0 \mathbf{X}_0^\top \hat{\mathbf{U}}_0)$. Any power method guarantee, e.g. [25], can be used to guarantee that its output is within a subspace distance δ_0 to the span of the top r singular vectors of \mathbf{X}_0 within order $\log(1/\delta_0)$ iterations. The communication complexity is thus just nr per node per iteration. The number of iterations needed is only order $\log r$ because \mathbf{U}_0 only needs to be order $1/r$ accurate.

Communication cost. The total communication cost is order $\max(nr \log r, nr \cdot T)$ where T is the total number AltGDmin iterations needed to achieve ϵ accuracy. We show below that $T = C\kappa^2 \log(1/\epsilon)$ suffices. For accurate solutions $\epsilon < \exp(-r)$ and hence the total communication cost is order $nrT = \kappa^2 nr \log(1/\epsilon)$.

Privacy. Observe from above that the information shared with the center is not sufficient to recover \mathbf{X}^* centrally. It is only sufficient to estimate $\text{span}(\mathbf{U}^*)$. The recovery of the columns of \mathbf{B} , \mathbf{b}_k^* , is done locally at the node where the corresponding \mathbf{y}_k is stored, thus ensuring privacy.

4.4.3 Theoretical Guarantees

We provide below the best known guarantee for LRCS; this is taken from [52]. We state the noise-free case result here for simplicity. Let m_0 denote the total number of samples per column needed for initialization and let m_1 denote this number for each GDmin iteration. Then, the total sample complexity per column is $m = m_0 + m_1 T$. Our guarantee given next provides the required minimum value of m .

Theorem 4.1 (AltGDmin-LRCS [52]). Assume that Assumption 4.1 holds. Set $\eta = 0.4/m\sigma_{\max}^2$ and $T = C\kappa^2 \log(1/\epsilon)$. If

$$mq \geq C\kappa^4\mu^2(n+q)r(\kappa^4r + \log(1/\epsilon))$$

and $m \geq C \max(\log n, \log q, r) \log(1/\epsilon)$, then, with probability (w.p.) at least $1 - n^{-10}$,

$$\text{SubsDist}_2(\mathbf{U}, \mathbf{U}^*) \leq \epsilon \text{ and } \|\mathbf{x}_k - \mathbf{x}_k^*\| \leq \epsilon \|\mathbf{x}_k^*\| \text{ for all } k \in [q].$$

The time complexity is $mqnr \cdot T = mqnr \cdot \kappa^2 \log(1/\epsilon)$. The communication complexity is $nr \cdot T = nr \cdot \kappa^2 \log(1/\epsilon)$ per node.

Remark 4.2. More generally, for any $\eta = c_\eta/(m\sigma_{\max}^2)$ with $c_\eta \leq 0.8$, one can show that $\text{SubsDist}_2(\mathbf{U}^+, \mathbf{U}^*) \leq (1 - \frac{cc_\eta}{\kappa^2})\text{SubsDist}_2(\mathbf{U}, \mathbf{U}^*)$. In short, the above result applies, with only changes to numerical constants.

4.4.4 Discussion

Existing approaches for LRCS include the AltMin solution studied in our work on LR phase retrieval (LRCS is a special case of LRPR) [38, 39, 40] and the convex relaxation studied in [48]. For reasons explained in detail in [37], for LRCS, there does not seem to be a way to guarantee convergence of either of the GD algorithms that have been studied for LRMC and robust PCA – Factorized GD (FactGD) and Projected GD (PGD) [28, 56]. Factorized GD is GD for \mathbf{U}, \mathbf{B} for the cost function $f(\mathbf{U}, \mathbf{B}) + \lambda \|\mathbf{U}^\top \mathbf{U} - \mathbf{B} \mathbf{B}^\top\|_F$ (the second term is a norm balancing term). PGD is GD for \mathbf{X} , with each GD step followed by projection onto the set of rank r matrices (by SVD). The reason is: to show convergence, we need to bound the norm of the gradient w.r.t. \mathbf{U} or \mathbf{X} of $f(\mathbf{U}, \mathbf{B})$ or $f(\mathbf{X})$, and show that it decays with iterations, under the desired roughly nr^2 sample complexity. To obtain this bound, one needs a tight bound on the column-wise recovery error $\max_k \|\mathbf{x}_k - \mathbf{x}_k^*\|$. This is not possible to get for either FactGD or PGD because, for both, the estimates of \mathbf{x}_k are coupled (PGD) or coupled given \mathbf{U} (FactGD).³

³Consider FactGD. The gradient w.r.t \mathbf{U} of $f(\mathbf{U}, \mathbf{B})$ is $\sum_{k=1}^q \mathbf{A}_k^\top \mathbf{A}_k (\mathbf{x}^* - \mathbf{x}_k) \mathbf{b}_k^\top$. To bound the norm of its deviation from its expected value, we need a small enough bound on the sub-exponential norm of each summand [53, Chap 2]; this requires a

AltGDmin is the fastest and most communication-efficient compared to both of AltMin and convex relaxation. Convex relaxation (mixed norm minimization) is known to be much slower. Its time complexity is not discussed in the monograph, however, it is well known that solvers for convex programs are much slower when compared to direct iterative algorithms: they either require number of iterations proportional to $1/\sqrt{\epsilon}$ or the per-iteration cost has cubic dependence on the problem size, here $(nr)^3$. AltMin is also slower than AltGDmin, both in terms of theoretical complexity and experimentally, because, for updating both \mathbf{U} and \mathbf{B} , it requires solving a minimization problem keeping the other variable fixed. The minimization step for \mathbf{U} is the slow one. The same is true for its communication cost. The minimization step for updating \mathbf{U} needs to use multiple GD iterations instead of just one in case of AltGDmin, or it needs to share matrices of size $nr \times nr$ (even more inefficient). This is why both the time and communication cost of AltMin depend on $\log^2(1/\epsilon)$ instead of just $\log(1/\epsilon)$ in case of AltGDmin.

4.5 AltGDmin for LRPR: Algorithm and Guarantees

To explain the ideas simply here, we consider the real-valued case. This means we do not need to worry about complex conjugation. The phaseless measurements z_k can be rewritten as

$$\mathbf{z}_k = \text{diag}(\mathbf{c}_k^*) \mathbf{y}_k = \text{diag}(\mathbf{c}_k^*) \mathbf{A}_k \mathbf{U}^* \mathbf{b}_k^*$$

where \mathbf{c}_k^* is a vector of signs/phases of \mathbf{y}_k and diag converts this into a diagonal matrix. Thus, the cost function to minimize now becomes

$$f(\mathbf{U}, \mathbf{B}, \{\mathbf{c}_k, k \in [q]\}) := \sum_{k=1}^q \|\mathbf{z}_k - \text{diag}(\mathbf{c}_k) \mathbf{A}_k \mathbf{U} \mathbf{b}_k\|^2$$

small enough bound on the column-wise error $\max_k \|\mathbf{x}_k - \mathbf{x}_k^*\|$, here $\mathbf{x}_k = \mathbf{U} \mathbf{b}_k$. It is not possible to get a tight bound on this quantity for FactGD because its estimates of the different \mathbf{b}_k s are coupled, due to the gradient term coming from the second norm balancing term. Consider PGD. The gradient w.r.t. \mathbf{X} is $\sum_{k=1}^q \mathbf{A}_k^\top \mathbf{A}_k (\mathbf{x}^* - \mathbf{x}_k)$; bounding it again requires a bound on $\max_k \|\mathbf{x}_k - \mathbf{x}_k^*\|$. The estimates \mathbf{x}_k are coupled for different k because of the rank r projection step.

Clearly this problem is again decoupled with $\mathbf{Z}_{slow} = \mathbf{U}$ and $\mathbf{Z}_{fast} = \{\mathbf{c}_k, \mathbf{b}_k, k \in [q]\}$. Notice also that when \mathbf{U} is fixed, solving for $\{\mathbf{b}_k, \mathbf{c}_k\}$ is a standard r -dimensional PR problem with many fast and provably correct solutions, e.g., [10, 15]. The cost of r -dimensional PR is order $mr \log(1/\epsilon)$ and the cost of computing $\mathbf{A}_k \mathbf{U}$ is mnr , per column. Thus, the total cost of standard PR for all columns is just $q \max(mnr, mr \log(1/\epsilon))$. Typically the first term dominates. Gradient computation cost is still $mnrq$. Thus the total cost of AltGDmin iterations is $mqnr \cdot T$ with T bounded in the result below.

The initialization in this case also needs to be different. We initialize \mathbf{U} by computing the top r singular vectors of [40]

$$\mathbf{M} = \sum_k \mathbf{A}_k^\top \mathbf{z}_{k,trunc} \mathbf{z}_{k,trunc}^\top \mathbf{A}_k$$

with the truncation done exactly as explained above for LRCS (truncation only uses magnitudes of observations). We can prove the following [37].

Theorem 4.2 (AltGDmin-LRPR [37]). Assume that Assumption 4.1 holds. Set $\eta = 0.4/m\sigma_{\max}^{*2}$ and $T = C\kappa^2 \log(1/\epsilon)$. If

$$mq \geq C\kappa^6 \mu^2 (n + q)r^2 (\kappa^4 r + \log(1/\epsilon))$$

and $m \geq C \max(\log n, \log q, r) \log(1/\epsilon)$, then, the conclusions of Theorem 4.1 hold with SubsDist₂ replaced by SubsDist_F. The time complexity is $mqnr \cdot T = \max(mqnr, mqr \log(1/\epsilon)) \cdot \kappa^2 \log(1/\epsilon)$. The communication complexity is $nr \cdot T = nr \cdot \kappa^2 \log(1/\epsilon)$ per node.

Remark 4.3. We can use any $\eta = c_\eta / (m\sigma_{\max}^{*2})$ with $c_\eta \leq 0.8$, see Remark 4.2.

Notice that the only change in the above result compared to LRCS is an extra factor of r in the sample complexity. This trend is well-known from other work on structured phase retrieval [8, 39, 40]. The rest of the discussion is the same as in case of LRCS. AltGDmin is much faster than AltMin. FactGD or PGD do not provably converge for the same reasons. The proof strategy for this case involves interpreting the gradient w.r.t. \mathbf{U} as a noisy version of the LRCS case. The overall idea for handling this case is provided in Section 6.3.

4.6 AltGDmin for LRMC: Algorithm and Guarantees

AltGDmin for LRMC was studied in [2]. There are two differences between LRMC and LRCS. The first is that LRMC measurements are row-wise and column-wise local while those for LRCS are global functions of each column. This is why LRMC needs incoherence of left and right singular vectors of \mathbf{X}^* , and needs to prove this for each estimate $\mathbf{X} = \mathbf{U}\mathbf{B}$ at each iteration. LRCS needs this only for right singular vectors. The second is that the measurements are bounded and this is why the initialization does not need a truncation step.

The goal is to minimize

$$\min_{\check{\mathbf{B}}, \check{\mathbf{U}} : \check{\mathbf{U}}^\top \check{\mathbf{U}} = \mathbf{I}} f(\check{\mathbf{U}}, \check{\mathbf{B}}), \quad f(\check{\mathbf{U}}, \check{\mathbf{B}}) := \|(\mathbf{Y} - \check{\mathbf{U}}\check{\mathbf{B}})_\Omega\|_F^2 \quad (4.3)$$

As before, we impose the orthonormal columns constraint on $\check{\mathbf{U}}$ as one way to ensure that the norm of \mathbf{U} does not keep increasing or decreasing continuously with algorithm iterations, while that of \mathbf{B} decreases or increases.

As explained earlier in (1.3), this cost function is partly decoupled for \mathbf{B} as well as for \mathbf{U} . This means that we could pick either of the two to serve as \mathbf{Z}_{fast} ; the choice depends on how the data is federated. In fact, since the LRMC problem is symmetric w.r.t. rows and columns, one can always assume vertical federation as stated earlier and, if needed, transpose the matrices to satisfy the assumption.

Conceptually, the only difference for the AltGDmin algorithm in this case is in the initialization step. However, its efficient implementation requires some careful work. The analysis to derive the theoretical guarantees needs significant extra work as well. Most importantly, it requires showing incoherence of \mathbf{U} at each iteration including the initialization. For the iterations, this can be proved; we explain the main ideas in Section 6 and 7. For the initialization of \mathbf{U} , we need to ensure this by construction. We do this by adapting the idea of [56]. We first compute the top r singular vectors of \mathbf{Y} ; denote the matrix formed by these singular vectors by \mathbf{U}_{00} . We then project \mathbf{U}_{00} onto the space of row incoherent matrices, $\mathcal{U} := \{\check{\mathbf{U}} : \|\check{\mathbf{u}}^j\| \leq \mu\sqrt{r/n}\}$ to obtain $\Pi_{\mathcal{U}}(\mathbf{U}_{00})$. We finally obtain \mathbf{U}_0 by orthonormalizing it by QR. Here,

$$[\Pi_{\mathcal{U}}(\mathbf{M})]^j = \mathbf{m}^j \cdot \min \left(1, \frac{\mu \sqrt{r/n}}{\|\mathbf{m}^j\|} \right), \quad \text{for all } j \in [n] \quad (4.4)$$

In words, if a row of \mathbf{M} has ℓ_2 norm that is more than the threshold $\mu\sqrt{r/n}$, then one renormalizes the row so that its norm equals the threshold. If the norm is less than this threshold, then we do not change it. Clearly this is an order nr time operation. In summary, $\mathbf{U}_0 := QR(\Pi_{\mathcal{U}}(\mathbf{U}_{00}))$ with \mathbf{U}_{00} being the top r left singular vectors of \mathbf{Y} .

The rest of the AltGDmin algorithm is conceptually similar to that for LRCS, we use $\mathbf{Z}_{slow} = \mathbf{U}$ and $\mathbf{Z}_{fast} = \mathbf{B}$. However, its efficient implementation is very different and hence, so is its time and communication complexity. Briefly, the reason is that \mathbf{S}_k is just a row selection matrix. Thus, for example, $\mathbf{S}_k \mathbf{U}$ is actually implemented by sub-selecting the rows of \mathbf{U} and not by matrix multiplication. A lot of other steps use similar ideas for efficient implementation. We summarize the complexities in Table 4.1. We can prove the following for AltGDmin-LRMC

Theorem 4.3 (AltGDmin-LRMC [2]). Pick an $\epsilon < 1$. Assume that Assumption 4.2 holds, and that, entries of \mathbf{X}^* are observed independently of other entries with probability p . Set $\eta = 0.5/(p\sigma_{\max}^{*2})$ and $T = C\kappa^2 \log(1/\epsilon)$. If $nqp > C\kappa^6 \mu^2 \max(n, q)r^2 \log \max(n, q) \log(1/\epsilon)$, then, with probability (w.p.) at least $1 - 4T/\min(n, q)^3$,

$$\text{SubsDist}_F(\mathbf{U}^{(T)}, \mathbf{U}^*) \leq \epsilon \text{ and } \|\mathbf{X}^{(T)} - \mathbf{X}^*\|_F \leq \epsilon \|\mathbf{X}^*\|. \quad (4.5)$$

The total per-node computation complexity of federated AltGDmin is $C\kappa^2 \log(1/\epsilon) \cdot \max(n, |\Omega|)r^2 \cdot \frac{1}{\gamma}$ and its total per-node communication complexity is $C\kappa^2 \log(1/\epsilon) \cdot nr$.

Observe that $nq \cdot p = \mathbb{E}[|\Omega|]$, i.e., it is the expected value of the sample complexity. We often just use the phrase “sample complexity” when referring to it in our writing.

Remark 4.4. More generally, we can use any $\eta = c_\eta/(p\sigma_{\max}^{*2})$ with $c_\eta \leq 0.8$; see Remark 4.2.

Table 4.1: Comparing AltGDmin with AltMin and GD (FactGD) for recovering an $n \times q$ rank r matrix from a subset of m linear projections of its columns (LRCS), m phaseless linear projections (LRPR), or from a subset of its entries, when each entry is observed with probability p independent of all others (LRMC). γ is the total number of federated nodes. Ω is the set of observed entries for LRMC. The table assumes $n \approx q$, γ is a numerical constant, κ, μ are numerical constants, $\max(\log(1/\epsilon), r) = \log(1/\epsilon)$, and $|\Omega| \geq nr$ (necessary). Here $\text{Communic. Comp} = T \cdot \max(\text{Communic.}(node), \text{Communic.}(center))$. Similarly for the computation cost.

| LRCS | Computation Complexity | Communic. Complexity | Sample Complexity | Resilient Modific |
|--|---|---|---|-------------------|
| AltGDmin [52] | $m \frac{2}{\gamma} nr \cdot \log(1/\epsilon)$ | $nr \log(1/\epsilon)$ | $nr \max(r, \log(1/\epsilon))$ | Efficient |
| GD (FactGD) | $m \frac{2}{\gamma} nr \cdot T$ (cannot bound T) | $nr \cdot T$ | (cannot bound) | Efficient |
| AltMin [40] | $m \frac{2}{\gamma} nr \cdot \log^2(1/\epsilon)$ | $nr \log^2(1/\epsilon)$ | $nr^2 \log(1/\epsilon)$ | Not Efficient |
| Convex [48] (mixed norm min) | $m q n r \cdot \min\left(\frac{1}{\sqrt{\epsilon}}, n^3 r^3\right)$ | | $\frac{nr}{\epsilon^4}$ | Not Efficient |
| LRPR | Computation Complexity | Communic. Complexity | Sample Complexity | Resilient Modific |
| AltGDmin [37] | $\max(m \frac{2}{\gamma} nr, m \frac{2}{\gamma} r \log(1/\epsilon)) \cdot \log(1/\epsilon)$ | $nr \log(1/\epsilon)$ | $nr^2 \max(r, \log(1/\epsilon))$ | Efficient |
| GD (FactGD) | $m \frac{2}{\gamma} nr \cdot T$ (cannot bound T) | $nr \cdot T$ | (cannot bound) | Efficient |
| AltMin [40] | $m \frac{2}{\gamma} nr \cdot \log^2(1/\epsilon)$ | $nr \log^2(1/\epsilon)$ | $nr^2 \log(1/\epsilon)$ | Not |
| LRMC | Computation Complexity | Communic. Complexity | Sample Complexity | Resilient Modific |
| AltGDmin [2] | $\frac{ \Omega }{\gamma} r^2 \log\left(\frac{1}{\epsilon}\right)$ | $nr \log\left(\frac{1}{\epsilon}\right)$ | $nr^2 \log n \log\left(\frac{1}{\epsilon}\right)$ | Efficient |
| GD (FactGD) [56, 57] | $\frac{ \Omega }{\gamma} r^2 \log\left(\frac{1}{\epsilon}\right)$ | $nr^2 \log\left(\frac{1}{\epsilon}\right)$ | $nr^2 \log n$ | Efficient |
| AltMin [41] (use GD for updating \mathbf{U}) | $\frac{ \Omega }{\gamma} r \log^2\left(\frac{1}{\epsilon}\right)$ | $nr \log^2\left(\frac{1}{\epsilon}\right)$ | $nr^{4.5} \log n \log\left(\frac{1}{\epsilon}\right)$ | Not |
| AltMin [2, 41] (use closed form for updating \mathbf{U}) | $\frac{ \Omega }{\gamma} r^2 \log\left(\frac{1}{\epsilon}\right)$ | $\frac{ \Omega }{\gamma} \log\left(\frac{1}{\epsilon}\right)$ | $nr^2 \log n \log\left(\frac{1}{\epsilon}\right)$ | Not |
| Convex [11] (nuclear norm min) | $ \Omega r \cdot \min\left(\frac{1}{\sqrt{\epsilon}}, n^3 r^3\right)$ | | $n^{1.2} r \log^2 n$ | Not |

4.6.1 Discussion

Recall that for LRCS, AltGDmin is much faster than AltMin because the minimization step w.r.t. \mathbf{U} is coupled and hence expensive. However, in case of LRMC, the recovery problem is decoupled for both \mathbf{U} and for \mathbf{B} . Consequently, AltMin is the fastest centralized solution and order-wise (ignoring dependence on κ, μ), all of AltMin, AltGDmin and FactGD are equally fast. PGD is much slower. In a federated

setting, when considering communication cost, AltGDmin is the most communication efficient compared with both AltMin and GD (FactGD). Compared with FactGD, AltGDmin iteration complexity is better by a factor of r . Compared with AltMin, its per-iteration cost is lower. It requires sharing just the gradient w.r.t. \mathbf{U} , which is at most nr entries, in each iteration. AltMin, on the other hand, requires sharing all the observed entries and this has a communication cost of order $|\Omega| > nr$. The required $|\Omega|$ is order nr^2 at least (see sample complexity). We provide a summary of comparisons of guarantees for both LRCS and LRMC in Table 4.1.

Part III

AltGDmin Analysis – Overall Proof Technique and Details for LR Problems

5

General Proof Approach for any Problem

The following approach generalizes the ideas used for LRCS [52] and the other LR problems. Let NormDist_a be the relevant measure of normalized distance for \mathbf{Z}_{slow} and NormDist_b for \mathbf{Z}_{fast} . The distance metric used can be different for \mathbf{Z}_{slow} and \mathbf{Z}_{fast} , e.g., for the LR matrix recovery problems discussed above, we used the subspace distance for \mathbf{Z}_{slow} , and normalized Euclidean norm distance for \mathbf{Z}_{fast} . The following is the overall approach that can be considered to analyze AltGDmin for solving a problem. This generalizes the ideas used for the above guarantees for LRCS and LRMC.

- Analyze the initialization step to try to show that $\text{NormDist}_a(\hat{\mathbf{Z}}_{\text{slow}}, \mathbf{Z}_{\text{slow}}^*) \leq \delta_0$ with a certain probability.
 - Typically, the initialization is a spectral initialization for which existing approaches (if any) can be used.
 - In many cases, δ_0 being a small numerical constant suffices. For certain problems, it may even be possible to prove results with random initialization; in this case, δ_0 is very close to 1.
- At iteration t , suppose that we are given an estimate $\hat{\mathbf{Z}}_{\text{slow}}$ satisfying $\text{NormDist}_a(\hat{\mathbf{Z}}_{\text{slow}}, \mathbf{Z}_{\text{slow}}^*) \leq \delta_{t-1}$ with δ_{t-1} “small enough”.

- Analyze the minimization step to show that $\text{NormDist}_b(\hat{\mathbf{Z}}_{fast}, \mathbf{Z}_{fast}^*) \lesssim \delta_{t-1}$ with a certain probability.
 - * This analysis will typically be the easier one, because this step is often a well studied problem, e.g., in case of LRCS or LRMC, it is the standard least squares (LS) problem. For LRPR, it is a standard phase retrieval problem.
- Analyze the GD step to try to show that, for the updated $\hat{\mathbf{Z}}_{slow}$ estimate, $\text{NormDist}_a(\hat{\mathbf{Z}}_{slow}^+, \mathbf{Z}_{slow}^*) \leq \delta_t := c_1(\eta, \delta_0, \epsilon_1) \delta_{t-1}$ with a certain probability. This bound would hold under an upper bound on the step size η and the initialization error δ_0 .

We set δ_0 , η and ϵ_1 to ensure that $c_1(\eta, \delta_0, \epsilon_1) \leq c$ for a $c < 1$ (exponential error decay). If we can set η to be a constant (w.r.t. n, q, r), it will help guarantee that the iteration complexity grows logarithmically with $1/\epsilon$ (fast convergence). All the above steps should hold with a certain probability that depends on the problem dimensions (n, q, r in case of LR problems), sample complexity (m or np in case of LRCS and LRMC), and the values of δ_0 and ϵ_1 . We use our values of δ_0 and ϵ_1 to find a lower bound on the sample complexity in terms of n, q, r and κ, μ , in order to guarantee that all the above steps hold with a high enough probability.

Sample-splitting is assumed across iterations in order to make the analysis easier (a common technique for analyzing iterative algorithms that we learned about in [41] and follow-up works). This helps guarantee that the estimates $\hat{\mathbf{Z}}_{slow}, \hat{\mathbf{Z}}_{fast}$ used in a given step are independent of the data used in that step. Using this assumption, (i) the expected value of terms can be computed more easily; and (ii) the summands in a given term are independent conditioned on past data making it possible to bound the deviation from the expected value using concentration bounds for sums of independent random variables/vectors/matrices [53].

In the AltGDmin analysis, analyzing the GD step is the most challenging part. The reason is AltGDmin is not a GD or projected GD algorithm (both of which are well studied) for any variable(s). This

means that the gradient at $\mathbf{Z}_{slow} = \mathbf{Z}_{slow}^*$, $\nabla_{\mathbf{Z}_{slow}} f(\mathbf{Z}_{slow}^*, \mathbf{Z}_{fast})$ is not zero.

We emphasise here that because one of the steps in AltGDmin is a minimization step, there does not seem to be a way to prove guarantees without sample-splitting. All guarantees for AltMin require sample splitting [40, 41, 42]. On the other hand, for GD / factorized GD, it is possible to prove guarantees without sample splitting as done in [15, 56].

6

AltGDmin for LR Problems: Overall Proof Ideas

In this section, we explain the main ideas that can be used for analyzing AltGDmin for solving an LR recovery problem. We begin below by specifying AltGDmin for any LR problem. Next in Section 6.2, we provide the main ideas for analyzing the noise-free attack-free linear measurements problems - LRCS and LRMC. In Section 6.3, we explain how to analyze the noisy, attack-resilient or nonlinear measurement settings. More details for both sections are provided in Section 7. The mathematical tools used in this analysis (linear algebra, probability and random matrix theory ideas) are summarized in Section 8. Many of these are from [16] and [53] and include: (i) singular value bounds, (ii) results such as the Davis-Kahan or the Wedin $\sin \Theta$ theorem, that can be used to obtain a deterministic bound the subspace distance between the estimate of the column-span of the unknown LR matrix \mathbf{X}^* and the true one [16]; and (iii) matrix concentration bounds (or scalar ones combined with an appropriate epsilon-net argument) such as sub-Gaussian Hoeffding, sub-exponential Bernstein or the matrix Bernstein inequality [53]. In addition some basic linear algebra tricks are needed as well.

6.1 AltGDmin for any LR Matrix Recovery Problem

Consider the problem of recovering $\mathbf{X} = \mathbf{U}\mathbf{B}$ from $\mathbf{Y} := \mathcal{A}(\mathbf{U}\mathbf{B})$ where \mathcal{A} is a linear operator. We consider the squared loss function

$$f(\mathbf{U}, \mathbf{B}) := \nu \|\mathbf{Y} - \mathcal{A}(\mathbf{U}\mathbf{B})\|_F^2$$

where ν is a quantity that does not depend on \mathbf{U}, \mathbf{B} and that is used to normalize the loss function so that

$$\mathbb{E}[\nabla_{\mathbf{U}} f(\mathbf{U}, \mathbf{B})] = (\mathbf{X} - \mathbf{X}^*)\mathbf{B}^\top, \quad \mathbf{X} := \mathbf{U}\mathbf{B}$$

when \mathbf{U}, \mathbf{B} are independent of the data $\{\mathbf{Y}, \mathcal{A}\}$. For example, for LRCS, with \mathbf{A}_k containing i.i.d. standard Gaussian entries, $\nu = 1/m$; while for LRMC, $\nu = 1/p$.

The AltGDmin algorithm proceeds as follows.

- Initialize \mathbf{U} : use a carefully designed spectral initialization approach to get \mathbf{U}_0
- Repeat the following for all $t = 1$ to T :
 1. Update \mathbf{B} by minimization: obtain

$$\mathbf{B}_t := \arg \min_{\mathbf{B}} f(\mathbf{U}_{t-1}, \mathbf{B})$$

(this statement assumes that the minimizer is unique and this fact is proved in the algorithm analysis). This step is efficient if it decouples column-wise, as in the case of LRCS, LRPR and LRMC.

2. Update \mathbf{U} by GD followed by orthonormalization:

$$\mathbf{U}_t := QR(\mathbf{U}_{t-1} - \eta \nabla_{\mathbf{U}} f(\mathbf{U}, \mathbf{B}_t))$$

- Sample splitting is assumed as noted earlier.

6.2 Proof Approach: Clean and Noise-free Case

To explain the main ideas of our proof approach, we use the simplest setting: noise-free and attack-free LRCS and LRMC. Depending on the

problem, we use SubsDist₂ or SubsDist_F. For any problem, SubsDist_F can be used. For attack-free LRCS, use of SubsDist₂ gives in a better result (sample complexity lower by a factor of r). When SubsDist₂ is used, all the norms below are $\|\cdot\|$. When SubsDist_F is used, all the numerator term norms are $\|\cdot\|_F$.

Let \mathbf{U} be the estimate at the t -th iteration. Define

$$\begin{aligned}\mathbf{g}_k &:= \mathbf{U}^\top \mathbf{x}_k^*, k \in [q], \text{ and } \mathbf{G} := \mathbf{U}^\top \mathbf{X}^*, \\ \mathcal{P}_{\mathbf{U}^*, \perp} &:= \mathbf{I} - \mathbf{U}^* \mathbf{U}^{*\top}, \\ \text{gradU} &:= \nabla_{\mathbf{U}} f(\mathbf{U}, \mathbf{B}) \\ \delta_t &:= \text{SubsDist}(\mathbf{U}, \mathbf{U}^*) = \|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}\|\end{aligned}\tag{6.1}$$

Under the sample-splitting assumption, it can be shown that

$$\mathbb{E}[\text{gradU}] = \mathbb{E}[\nabla_{\mathbf{U}} f(\mathbf{U}, \mathbf{B})] = (\mathbf{X} - \mathbf{X}^*) \mathbf{B}^\top\tag{6.2}$$

Recall the Projected GD step for \mathbf{U} :

$$\tilde{\mathbf{U}}^+ = \mathbf{U} - \eta \text{gradU} \text{ and } \tilde{\mathbf{U}}^+ \stackrel{\text{QR}}{=} \mathbf{U}^+ \mathbf{R}^+\tag{6.3}$$

Since $\mathbf{U}^+ = \tilde{\mathbf{U}}^+ (\mathbf{R}^+)^{-1}$ and since $\|(\mathbf{R}^+)^{-1}\| = 1/\sigma_{\min}(\mathbf{R}^+) = 1/\sigma_{\min}(\tilde{\mathbf{U}}^+)$, thus, $\text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) = \|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}^+\|$ can be bounded as

$$\begin{aligned}\delta_{t+1} &:= \text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \tilde{\mathbf{U}}^+\|}{\sigma_{\min}(\tilde{\mathbf{U}}^+)} \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \tilde{\mathbf{U}}^+\|}{\sigma_{\min}(\mathbf{U}) - \eta \|\text{gradU}\|} \\ &= \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \tilde{\mathbf{U}}^+\|}{1 - \eta \|\text{gradU}\|}\end{aligned}$$

This follows by Weyl's inequality and $\sigma_{\min}(\mathbf{U}) = 1$. Consider the numerator. Using (6.3), adding/subtracting $\eta \mathbb{E}[\text{gradU}]$, and using (6.2) which implies that $\mathcal{P}_{\mathbf{U}^*, \perp} \mathbb{E}[\text{gradU}] = \mathcal{P}_{\mathbf{U}^*, \perp} (\mathbf{X} - \mathbf{X}^*) \mathbf{B}^\top = \mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{X} \mathbf{B}^\top = \mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U} \mathbf{B} \mathbf{B}^\top$, we get

$$\begin{aligned}\mathcal{P}_{\mathbf{U}^*, \perp} \tilde{\mathbf{U}}^+ &= \mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U} - \eta \mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U} \mathbf{B} \mathbf{B}^\top + \eta \mathcal{P}_{\mathbf{U}^*, \perp} ((\mathbb{E}[\text{gradU}] - \text{gradU})) \\ &= \mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U} (\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top) + \eta \mathcal{P}_{\mathbf{U}^*, \perp} (\mathbb{E}[\text{gradU}] - \text{gradU})\end{aligned}$$

Thus, using (6.1),

$$\begin{aligned}\text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) &= \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}(\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top) + \eta \mathcal{P}_{\mathbf{U}^*, \perp} (\mathbb{E}[\text{gradU}] - \text{gradU})\|}{1 - \eta \|\mathbb{E}[\text{gradU}] + \text{gradU} - \mathbb{E}[\text{gradU}]\|} \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}\| \cdot \|\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top\| + \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{1 - \eta \|\mathbb{E}[\text{gradU}]\| - \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|}\end{aligned}$$

Notice that

$$\lambda_{\min}(\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top) = 1 - \eta \|\mathbf{B}\|^2.$$

Thus, if $\eta < 0.9/\|\mathbf{B}\|^2$, then $1 - \eta \|\mathbf{B}\|^2 > 0.1 > 0$, i.e., the matrix $(\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top)$ is positive semi-definite (p.s.d.). This means that

$$\|\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top\| = \lambda_{\max}(\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top) = 1 - \eta \sigma_r(\mathbf{B})^2$$

Thus, if $\eta \leq 0.9/\|\mathbf{B}\|^2$, then

$$\begin{aligned}\delta_{t+1} &:= \text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \tilde{\mathbf{U}}^+\|}{1 - \eta \|\mathbb{E}[\text{gradU}]\|} \\ &= \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}(\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top) + \eta \mathcal{P}_{\mathbf{U}^*, \perp} (\mathbb{E}[\text{gradU}] - \text{gradU})\|}{1 - \eta \|\mathbb{E}[\text{gradU}] + \text{gradU} - \mathbb{E}[\text{gradU}]\|} \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}\| \cdot \|\mathbf{I} - \eta \mathbf{B} \mathbf{B}^\top\| + \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{1 - \eta \|\mathbb{E}[\text{gradU}]\| - \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|} \\ &\leq \frac{\delta_t (1 - \eta \sigma_r(\mathbf{B})^2) + \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{1 - \eta \|\mathbb{E}[\text{gradU}]\| - \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|} \\ &= \frac{\delta_t \left(1 - \eta \left(\sigma_r(\mathbf{B})^2 - \frac{\|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{\delta_t}\right)\right)}{1 - \eta \|\mathbb{E}[\text{gradU}]\| - \eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|} \\ &= \delta_t \left(1 - \eta \left(\sigma_r(\mathbf{B})^2 - \frac{\|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{\delta_t}\right)\right) \\ &\quad (1 + 2\eta \|\mathbb{E}[\text{gradU}]\| + 2\eta \|\mathbb{E}[\text{gradU}] - \text{gradU}\|) \\ &\leq \delta_t \left(1 - \eta \left(\sigma_r(\mathbf{B})^2 - \frac{\|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{\delta_t}\right)\right) \\ &\quad - 2\|\mathbb{E}[\text{gradU}]\| - 2\|\mathbb{E}[\text{gradU}] - \text{gradU}\| \\ &\leq \delta_t \left(1 - \eta \left(\sigma_r(\mathbf{B})^2 - 3 \frac{\|\mathbb{E}[\text{gradU}] - \text{gradU}\|}{\delta_t} - 2\|\mathbb{E}[\text{gradU}]\|\right)\right)\end{aligned}\tag{6.4}$$

using $(1-x)^{-1} \leq (1+2x)$ for $x < 0.5$; $(1-z)(1+2x) = 1-z+2x-2xz < 1-z+2x = 1-(z-2x)$; and $\delta_t < 0.5$ (this allows us to replace $\frac{\|\mathbb{E}[\text{gradU}]-\text{gradU}\|}{\delta_t} + 2\|\mathbb{E}[\text{gradU}]-\text{gradU}\|$ by 3 times the first term).

The next step is to upper bound the expected gradient norm $\|\mathbb{E}[\text{gradU}]\|$ and the gradient deviation norm $\|\mathbb{E}[\text{gradU}]-\text{gradU}\|$ and to lower bound $\sigma_r(\mathbf{B})$. We need tight enough bounds in order to be able to show that for η small enough,

$$\delta_{t+1} \leq (1 - c_1/\kappa^2)\delta_t$$

To bound the terms, we use matrix concentration bounds from Section 8 and the incoherence assumptions on \mathbf{B}^* (LRCS) or on \mathbf{B}^* and \mathbf{U}^* (LRMC). The overall approach is as follows. We provide more details in Section 7. At each iteration,

- The first step is to analyze the minimization step to bound $\|\mathbf{B} - \mathbf{G}\|$: for LRCS, we can bound $\max_k \|\mathbf{b}_k - \mathbf{g}_k\|$ and use it to bound $\|\mathbf{B} - \mathbf{G}\|$; for LRMC, we can only bound the matrix error.
 - This is used to bound $\|\mathbf{X} - \mathbf{X}^*\|$ using triangle inequality
 - The above is used to upper bound $\sigma_{\max}(\mathbf{B})$ and lower bound $\sigma_r(\mathbf{B})$ using tricks from Section 8.
- We use the minimization step bounds to bound $\|\mathbb{E}[\text{gradU}]\| \leq \|\mathbf{X} - \mathbf{X}^*\| \sigma_{\max}(\mathbf{B})$.
- We bound $\|\mathbb{E}[\text{gradU}] - \text{gradU}\|$ using matrix concentration inequalities from Section 8 and the minimization step bounds.
- Most above results also use incoherence of \mathbf{B} (for LRCS) and of \mathbf{B} and of \mathbf{U} (for LRMC), which needs to be proved.
 - Incoherence of \mathbf{B} is easy to show.
 - Incoherence of \mathbf{U} for LRMC requires an inductive argument and a concentration bound on $\max_j \|\mathbf{e}_j^\top (\text{gradU} - \mathbb{E}[\text{gradU}])\|$. We need a bound on it that contains a factor of $\sqrt{r/n}$; but it need not contain a factor of δ_t .

We provide details of the above steps in Section 7.

6.3 Proof Approach: Noisy Gradient Approach to Deal with Non-linear or Noisy or Attack-prone Cases

Above we explained the proof strategy for the simple noise-free case. Here, we explain how to modify it to deal with various modifications of the basic LRCS or LRMC problems. One simple example where this occurs is the LR phase retrieval (LRPR) problem which is the phaseless measurements' generalization of LRCS. A second example is noise-corrupted LRCS or LRMC. A third setting is dealing with attacks, such as the Byzantine attack, by malicious nodes. The algorithm itself may remain the same (noisy case) or may change (LRPR or attack setting).

In all the above cases, the gradient expression will be different in the GD step to update \mathbf{U} . To bound $\text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*)$, we need to define and bound an extra term that we refer to as Err for "Error term". Let

$$\text{Err} := \text{gradU}_{cln} - \text{gradU}$$

where gradU_{cln} is the gradient from the noise-free case section that satisfies

$$\mathbb{E}[\text{gradU}_{cln}] = (\mathbf{X} - \mathbf{X}^*)\mathbf{B}^\top$$

We proceed exactly as in the noise-free case, with the following modification: we add/subtract $\mathbb{E}[\text{gradU}_{cln}] = (\mathbf{X} - \mathbf{X}^*)\mathbf{B}^\top$ and we add/subtract gradU_{cln} . This gives the following: if $\eta \leq 0.9/\|\mathbf{B}\|^2$, then

$$\begin{aligned} \delta_{t+1} &:= \text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \tilde{\mathbf{U}}^+\|}{1 - \eta \|\text{gradU}\|} \\ &= \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}(\mathbf{I} - \eta \mathbf{B}\mathbf{B}^\top) + \eta \mathcal{P}_{\mathbf{U}^*, \perp} (\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}) + \eta \mathcal{P}_{\mathbf{U}^*, \perp} \text{Err}\|}{1 - \eta \|\text{gradU} - \text{gradU}_{cln} + \text{gradU}_{cln} - \mathbb{E}[\text{gradU}_{cln}] - \mathbb{E}[\text{gradU}_{cln}]\|} \\ &\leq \frac{\|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}\| \cdot \|\mathbf{I} - \eta \mathbf{B}\mathbf{B}^\top\| + \eta \|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| + \eta \|\text{Err}\|}{1 - \eta \|\mathbb{E}[\text{gradU}_{cln}]\| - \eta \|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| - \eta \|\text{Err}\|} \\ &\leq \frac{\delta_t (1 - \eta \sigma_r(\mathbf{B})^2) + \eta \|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| + \eta \|\text{Err}\|}{1 - \eta \|\mathbb{E}[\text{gradU}_{cln}]\| - \eta \|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| - \eta \|\text{Err}\|} \end{aligned} \tag{6.5}$$

Using $(1 - x)^{-1} \leq (1 + 2x)$ for $x < 0.5$; $(1 - z)(1 + 2x) < 1 - (z - 2x)$; $\delta_t < 0.5$; and using $\|\mathbb{E}[\text{gradU}_{cln}]\| < 1$, $\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| < 1$, and $\|\text{Err}\| < 1$,

$$\begin{aligned}
\delta_{t+1} &:= \text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) \\
&\leq \frac{\delta_t(1 - \eta\sigma_r(\mathbf{B})^2) + \eta\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| + \eta\|\text{Err}\|}{1 - \eta\|\mathbb{E}[\text{gradU}_{cln}]\| - \eta\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| - \eta\|\text{Err}\|} \\
&\leq \frac{\delta_t(1 - \eta(\sigma_r(\mathbf{B})^2 - \eta\frac{\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\|}{\delta_t})) + \eta\|\text{Err}\|}{1 - \eta\|\mathbb{E}[\text{gradU}_{cln}]\| - \eta\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| - \eta\|\text{Err}\|} \\
&\leq \delta_t \left(1 - \eta(\sigma_r(\mathbf{B})^2 - \frac{\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\|}{\delta_t}) \right) (1 + 2\eta\|\mathbb{E}[\text{gradU}_{cln}]\| \\
&\quad + 2\eta\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| + 2\eta\|\text{Err}\|) \\
&\quad + \eta\|\text{Err}\|(1 + 2\eta\|\mathbb{E}[\text{gradU}_{cln}]\| + 2\eta\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| + 2\eta\|\text{Err}\|) \\
&\leq \delta_t \left(1 - \eta(\sigma_r(\mathbf{B})^2 - \frac{\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\|}{\delta_t} - 2\|\mathbb{E}[\text{gradU}_{cln}]\| - 2\|\text{Err}\|) \right) \\
&\quad + \eta\|\text{Err}\|(1 + 2\eta\|\mathbb{E}[\text{gradU}_{cln}]\| + 2\eta\|\mathbb{E}[\text{gradU}_{cln}] - \text{gradU}_{cln}\| + 2\eta\|\text{Err}\|) \tag{6.6}
\end{aligned}$$

The next steps are similar to those in the noise-free case. What result we can finally prove depends on how small Err is.

1. If $\|\text{Err}\|$ is of the same order as (or smaller than) the gradient deviation term, then the noise-free case analysis extends without much change. This requires $\|\text{Err}\|$ to decay as $c\delta_t$ for a $c < 1$, which can be shown for the LR phase retrieval problem.
2. If $\|\text{Err}\|$ is not as small, but is of order δ_0 or smaller, then the final bound will contain two terms: the first decays with t , and the second is a constant term that is of the order of $\max_t \|\text{Err}_t\|$.

We provide details for both steps in Section 7.

7

AltGDmin for LR Problems: Proof Details

7.1 Key Results Used

By combining Theorem 8.8 given in Section 8 with the scalar sub-exponential Bernstein or sub-Gaussian Hoeffding inequalities, we obtain the following two results which have been widely used in the LR recovery and phase retrieval literature. These study sums of rank-one matrices which are outer products of specific types of random vectors (r.vec). The last result below is the matrix Bernstein inequality.

Corollary 7.1 (Sum of rank-one matrices that are outer products of two sub-Gaussian r.vecs.). Consider a sum of m zero-mean independent rank-one $n \times r$ random matrices $\mathbf{x}_i \mathbf{z}_i^\top$ with $\mathbf{x}_i, \mathbf{z}_i$ being sub-Gaussian random vectors with sub-Gaussian norms $K_{x,i}, K_{z,i}$ respectively. For a $t \geq 0$,

$$\left\| \sum_{i=1}^m \mathbf{x}_i \mathbf{z}_i^\top \right\| \leq 1.4t$$

with probability at least

$$1 - \exp \left((\log 17)(n + r) - c \min \left(\frac{t^2}{\sum_i (K_{x,i}, K_{z,i})^2}, \frac{t}{\max_i (K_{x,i}, K_{z,i})} \right) \right)$$

By combining Theorem 8.8 with the scalar sub-Gaussian Hoeffding inequality, we conclude the following.

Corollary 7.2 (Sum of rank-one matrices that are outer products of a sub-Gaussian r. vec. and a bounded r.vec.). Consider a sum of m zero-mean independent rank-one $n \times r$ random matrices $\mathbf{x}_i \mathbf{z}_i^\top$ with \mathbf{x}_i being sub-Gaussian random vector with sub-Gaussian norms $K_{x,i}$ and \mathbf{z}_i being a bounded random vector with $\|\mathbf{z}_i\| \leq L_i$. Then, clearly, for any \mathbf{w}, \mathbf{w}' , $\mathbf{w}^\top \mathbf{x}_i \mathbf{z}_i^\top \mathbf{w}'$ is a sub-Gaussian r.v. with sub-Gaussian norm $K_{x,i} L_i$. Thus, for a $t \geq 0$,

$$\left\| \sum_{i=1}^m \mathbf{x}_i \mathbf{z}_i^\top \right\| \leq 1.4t$$

with probability at least

$$1 - \exp \left((\log 17)(n + r) - c \frac{t^2}{\sum_i (K_{x,i} L_i)^2} \right)$$

For bounded matrices, the following matrix Bernstein result gives a much tighter bound than what would be obtained by combining scalar bounded Bernstein and Theorem 8.8. See Section 8 for details.

Theorem 7.3 (Matrix Bernstein). Let $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m$ be independent, zero-mean, $n \times r$ matrices with $\|\mathbf{X}_i\| \leq L$ for all $i = 1, 2, \dots, m$. Define the “variance parameter” of the sum

$$v := \max \left(\left\| \sum_i \mathbb{E}[\mathbf{X}_i \mathbf{X}_i^\top] \right\|, \left\| \sum_i \mathbb{E}[\mathbf{X}_i^\top \mathbf{X}_i] \right\| \right).$$

Then,

$$\left\| \sum_{i=1}^m \mathbf{X}_i \right\| \leq t$$

with probability at least

$$1 - 2 \exp \left(\log \max(n, r) - c \min \left(\frac{t^2}{v}, \frac{t}{L} \right) \right)$$

7.2 Analyzing the Initialization Step

The following is the overall approach to analyze spectral initialization.

LRCS initialization. Recall the LRCS initialization from Section 4.4. It is not hard to show that

$$\mathbb{E}[\mathbf{X}_0] = \mathbf{X}^* \mathbf{D}(\alpha)$$

where $\mathbf{D}(\alpha)$ is a diagonal $q \times q$ matrix with all non-zero entries and that $\sigma_{\min}(\mathbf{D}) = \min_k (\mathbf{D})_{k,k} \geq 0.9$ with high probability (w.h.p.) [37, 52]. Thus, $\mathbb{E}[\mathbf{X}_0]$ is a rank r matrix with column-span equal to that of \mathbf{U}^* (or of \mathbf{X}^*). Using Wedin $\sin \theta$ theorem,

$$\text{SubsDist}_F(\mathbf{U}_0, \mathbf{U}^*) \leq \frac{\sqrt{r} \|\mathbf{X}_0 - \mathbb{E}[\mathbf{X}_0]\|}{\sigma_r(\mathbb{E}[\mathbf{X}_0]) - 0 - \|\mathbf{X}_0 - \mathbb{E}[\mathbf{X}_0]\|}$$

Using the bounds from Section 8.1, it is not hard to see that

$$\begin{aligned} \sigma_r(\mathbb{E}[\mathbf{X}_0]) &= \sigma_r(\mathbf{X}^* \mathbf{D}(\alpha)) \\ &\geq \sigma_r(\mathbf{X}^*) \sigma_{\min}(\mathbf{D}(\alpha)^\top) \\ &= \sigma_{\min}^* \sigma_{\min}(\mathbf{D}(\alpha)) \\ &\geq 0.9 \sigma_{\min}^* \end{aligned}$$

To upper bound $\|\mathbf{X}_0 - \mathbb{E}[\mathbf{X}_0]\|$ observe that \mathbf{X}_0 can be rewritten as

$$\mathbf{X}_0 = \sum_{k=1}^q \sum_{i=1}^m \mathbf{a}_{ki} (\mathbf{a}_{ki}^\top \mathbf{x}_k^*) \mathbb{1}_{(\mathbf{a}_{ki}^\top \mathbf{x}_k^*)^2 \leq \alpha}$$

where \mathbf{a}_{ki} is a standard Gaussian vector. We first use the sub-exponential Bernstein inequality to lower and upper bound α by a constant times $\|\mathbf{X}^*\|_F^2/q$ w.h.p. Next, we use Corollary 7.2 (sub-Gaussian Hoeffding inequality + epsilon-netting) to bound $\|\mathbf{X}_0 - \mathbb{E}[\mathbf{X}_0]\|$ under a lower bound on the sample complexity mq . This, along with using the Wedin $\sin \theta$ theorem (Theorem 8.2) [16], and lower bounding the smallest entry of $\mathbf{D}(\alpha)$, helps bound the subspace distance between \mathbf{U}_0 and \mathbf{U}^* .

LRMC initialization. For LRMC, $\mathbf{X}_0 = \mathbf{Y}$ and it is easy to see that $\mathbb{E}[\mathbf{Y}] = \mathbf{X}^*$. Thus, we can again use Wedin and a different matrix concentration bound (matrix Bernstein) to show that \mathbf{U}_{00} is a good subspace estimate of \mathbf{U}^* . Analyzing the second step (projection onto

row incoherent matrices) uses the fact that the set of row incoherent matrices is convex, this argument is borrowed from [56]). Finally we analyze the orthonormalization

7.3 Clean Noise-free Case

Recall equation (6.4) from the previous section. We make the following assumption temporarily just to show the flow of our proof. Later, we explain how to show that this assumption holds w.h.p. under just a sample complexity lower bound and incoherence Assumption 4.1 or 4.2. The final guarantee only needs to make assumptions on the true \mathbf{X}^* or on \mathbf{Y} (data).

Temporary Assumption 7.1. Suppose that

1. $\sigma_r(\mathbf{B}) \geq 0.9\sigma_r(\mathbf{B}^*) = 0.9\sigma_{\min}^*$
2. $\|\mathbb{E}[\text{gradU}] - \text{gradU}\| \leq c_1\delta_t\sigma_{\min}^{*2}$ for a $c_1 < 0.2$,
3. $\|\mathbb{E}[\text{gradU}]\| \leq C_2\kappa^2\delta_t\sigma_{\min}^{*2}$,
4. with δ_t such that $c_1 + (2C_2\kappa^2 + 2c_1)\delta_t < 0.1$ for all $t \geq 0$, and
5. $\eta = c_\eta/\sigma_{\max}^{*2}$ with $c_\eta \leq 0.9/(1.1\sigma_{\max}^{*2})$, and
6. equation (6.2) holds.

We note here that the big $C_?$ numbered constant C_2 can depend on r : if SubsDist₂ is used as the subspace distance measure then, $C_2 = C\sqrt{r}$ while if SubsDist_F is used then $C_2 = C$.

Using (6.4) and the bounds from the above Temporary Assumption 7.1,

$$\begin{aligned}
\delta_{t+1} &:= \text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) \\
&\leq \delta_t \left(1 - \eta \left(0.9\sigma_{\min}^{*2} - c_1\sigma_{\min}^{*2} - 2C_2\delta_t\sigma_{\min}^{*2} - 2c_1\delta_t\sigma_{\min}^{*2} \right) \right) \\
&\leq \delta_t \left(1 - \eta\sigma_{\min}^{*2} (0.9 - c_1 - (2C_2\kappa^2 + 2c_1)\delta_t) \right) \\
&\leq \delta_t (1 - \eta(0.9 - c_1 - 0.1)\sigma_{\min}^{*2}) \\
&= \delta_t \left(1 - (0.8 - c_1) \frac{c_\eta}{\kappa^2} \right)
\end{aligned}$$

Proving the bounds of temporary assumption 7.1. We use the matrix concentration bounds from Section 8 to bound the following with high probability.

1. $\max_k \|\mathbf{b}_k - \mathbf{g}_k\|$ and use it to bound $\|\mathbf{B} - \mathbf{G}\|_F$ (for LRCS) or directly bound $\|\mathbf{B} - \mathbf{G}\|_F$ (for LRMC)
2. $\|\mathbb{E}[\text{gradU}] - \text{gradU}\|$ (in case of LRCS) or $\|\mathbb{E}[\text{gradU}] - \text{gradU}\|_F$ (for LRMC)
3. $(\mathbf{S}_k \mathbf{U})^\top (\mathbf{S}_k \mathbf{U})$ and $(\mathbf{S}_k \mathbf{U}^*)^\top (\mathbf{S}_k \mathbf{U})$ (for LRMC)
4. and $\|\mathbf{e}_j^\top (\mathbb{E}[\text{gradU}] - \text{gradU})\|$ (for LRMC).

The LRCS proofs use Corollary 7.1 (sub-exponential Bernstein inequality followed by the epsilon-net argument). The LRMC proofs use the Theorem 7.3 (matrix Bernstein inequality).

The $\mathbf{B} - \mathbf{G}$ bound assumes incoherence of \mathbf{U} (in case of LRMC). The gradient deviation $\|\mathbb{E}[\text{gradU}] - \text{gradU}\|$ bound assumes the $\mathbf{B} - \mathbf{G}$ bound and incoherence of \mathbf{B} and of \mathbf{U} (in case of LRMC). The bound on $\|\mathbf{e}_j^\top (\mathbb{E}[\text{gradU}] - \text{gradU})\|$ (used to show incoherence of the updated \mathbf{U}^+ in case of LRMC) uses incoherence of \mathbf{B} .

- **Bounding** $\|\mathbb{E}[\text{gradU}]\| = C\|(\mathbf{X} - \mathbf{X}^*)\mathbf{B}^\top\|$ **and** $\sigma_r(\mathbf{B})$. The bound on $\|\mathbf{B} - \mathbf{G}\|$ is used to upper bound $\|\mathbf{X} - \mathbf{X}^*\|$ and $\|\mathbf{B}\| = \sigma_{\max}(\mathbf{B})$ and to lower bound $\sigma_r(\mathbf{B})$. The first two bounds are used to bound $\|\mathbb{E}[\text{gradU}]\| \leq \|\mathbf{X} - \mathbf{X}^*\| \|\mathbf{B}\|$. The σ_{\max} bound is straightforward. The σ_{\min} bound follows using the bounds given next; these follow using the preliminaries from Section 8.1.

$$\begin{aligned}
\sigma_r(\mathbf{B}) &\geq \sigma_r(\mathbf{G}) - \|\mathbf{B} - \mathbf{G}\|, \\
\sigma_r(\mathbf{G}) &= \sigma_{\min}(\mathbf{G}^\top) = \sigma_{\min}(\mathbf{B}^{*\top} \mathbf{U}^{*\top} \mathbf{U}) \geq \sigma_{\min}^* \sigma_{\min}(\mathbf{U}^{*\top} \mathbf{U}) \\
\sigma_{\min}^2(\mathbf{U}^{*\top} \mathbf{U}) &= \lambda_{\min}(\mathbf{U}^\top \mathbf{U}^* \mathbf{U}^{*\top} \mathbf{U}) \\
&= \lambda_{\min}(\mathbf{U}^\top (\mathbf{I} - \mathcal{P}_{\mathbf{U}^*, \perp}) \mathbf{U}) \\
&= \lambda_{\min}(\mathbf{I} - \mathbf{U}^\top \mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}) \\
&= \lambda_{\min}(\mathbf{I} - \mathbf{U}^\top \mathcal{P}_{\mathbf{U}^*, \perp}^2 \mathbf{U}) \\
&= 1 - \|\mathcal{P}_{\mathbf{U}^*, \perp} \mathbf{U}\|^2 = 1 - \text{SubsDist}(\mathbf{U}^*, \mathbf{U})^2
\end{aligned}$$

- **Showing Incoherence of updated \mathbf{B} .** LRCS only needs incoherence of \mathbf{B} . LRMC needs that of both.
 - Incoherence of \mathbf{b}_k (columns of \mathbf{B}) for LRCS: Since we can bound $\|\mathbf{b}_k - \mathbf{g}_k\|$ for each k , the bound on $\|\mathbf{b}_k\|$ follows directly from this bound and the fact that $\|\mathbf{g}_k\| \leq \|\mathbf{b}_k^*\|$.
 - Incoherence of \mathbf{b}_k (columns of \mathbf{B}) for LRMC: We use the exact expression for updating \mathbf{b}_k and concentration bounds on $(\mathbf{S}_k \mathbf{U})^\top (\mathbf{S}_k \mathbf{U})$ and $(\mathbf{S}_k \mathbf{U}^*)^\top (\mathbf{S}_k \mathbf{U})$ to get a bound that is a constant times $\|\mathbf{b}_k^*\|$.
- **Showing Incoherence of updated \mathbf{U} : only for LRMC.** This is shown in two steps:
 - First we bound the deviation of the j -th row of gradU from its expected value, $\max_j \|\mathbf{e}_j^\top (\mathbb{E}[\text{gradU}] - \text{gradU})\|$. We need a bound on it that contains a factor of $\sqrt{r/n}$; but it need not contain a factor of δ_t . To get such a bound we use $|x - z| \leq 2 \max(|x|, |z|)$ and obtain a bound of the form $c_4 \max(\|\mathbf{e}_j^\top \mathbf{U}^*\|, \|\mathbf{e}_j^\top \mathbf{U}\|) \sigma_{\max}^2$.
 - Next, we use (6.3) and proceed in a fashion similar to (6.4) to bound $\|\mathbf{e}_j^\top \tilde{\mathbf{U}}^+\|$. We show that $\|\mathbf{e}_j^\top \tilde{\mathbf{U}}^+\| \leq (1 - (0.9 - 0.1)\eta) \|\mathbf{e}_j^\top \mathbf{U}\| + \|\mathbf{e}_j^\top \mathbf{U}^*\| + c_4 \max(\|\mathbf{e}_j^\top \mathbf{U}^*\|, \|\mathbf{e}_j^\top \mathbf{U}\|)$ followed by simplifying this expression using the denominator expression. We can simplify this bound by using $\max(x, z) \leq x + z$. We get $\|\mathbf{e}_j^\top \mathbf{U}^+\| \leq (1 - c/\kappa^2) \|\mathbf{e}_j^\top \mathbf{U}\| + 2 \|\mathbf{e}_j^\top \mathbf{U}^*\|$.

Finally, recursively applying the above expression we get a bound on $\|\mathbf{e}_j^\top \mathbf{U}_t\|$ in terms of $\|\mathbf{e}_j^\top \mathbf{U}_0\|$ and $\|\mathbf{e}_j^\top \mathbf{U}^*\|$ that does not grow with t . We show that

$$\begin{aligned} \|\mathbf{e}_j^\top \mathbf{U}_t\| &\leq (1 - c/\kappa^2)^t \|\mathbf{e}_j^\top \mathbf{U}_0\| + \sum_{\tau=0}^{t-1} (1 - c/\kappa^2)^\tau 2 \|\mathbf{e}_j^\top \mathbf{U}^*\| \\ &\leq \|\mathbf{e}_j^\top \mathbf{U}_0\| + C\kappa^2 \|\mathbf{e}_j^\top \mathbf{U}^*\| \end{aligned}$$

- **Ensuring $(2C_2\kappa^2 + 2c_1)\delta_t < 0.1$.** Note from above that we are showing exponential decay for δ_t with t . A corollary of this is that

$\delta_t < \delta_0$. Thus, our bound on δ_t holds if δ_0 satisfies the same bound, i.e., if the initialization is good enough i.e. if $\delta_0 < 0.1/(2C_2\kappa^2+2c_1)$. The proof approach given above shows exponential error decay and hence $\delta_{t+1} < \delta_t < \delta_0$ for each t . (We note here the above numbered constants c_1, C_2 etc can depend on n, q, r, κ^2 . For example, in case of LRMC, $C_2 = \sqrt{r}$.)

7.4 Nonlinear or Noisy or Attack-prone or Outlier Corrupted Settings

Using equation (6.5) from the previous section and Temporary Assumption 7.1,

$$\begin{aligned} \delta_{t+1} &:= \text{SubsDist}(\mathbf{U}^+, \mathbf{U}^*) \\ &\leq \frac{\delta_t(1 - \eta 0.9 \sigma_{\min}^{*2}) + \eta c_1 \delta_t \sigma_{\min}^{*2} + \eta \|\text{Err}\|}{1 - \eta C_2 \kappa^2 \delta_t \sigma_{\min}^{*2} - \eta c_1 \delta_t \sigma_{\min}^{*2} - \eta \|\text{Err}\|} \\ &\leq \frac{\delta_t \left(1 - \frac{c_1}{\kappa^2} (0.9 - c_1)\right) + \frac{c_1}{\kappa^2} \|\text{Err}\|}{1 - \frac{c_1}{\kappa^2} (C_2 \kappa^2 + c_1) \delta_t - \frac{c_1}{\kappa^2} \|\text{Err}\|} \end{aligned} \quad (7.1)$$

1. First, if we can show that the Err term is of the same order as the gradient deviation term, i.e., both decay at the same rate as δ_t w.h.p., under the desired sample complexity bound, i.e., if

$$\|\text{Err}\| \leq c_4 \delta_t \sigma_{\min}^{*2}, \quad \text{with a } c_4 < (0.9 - c_1),$$

then, the old analysis applies without change. One can still show exponential decay of δ_t with t . This is the case for example for LRPR for which the Err term bound is taken from [40] since this term also occurs when studying AltMin for LRPR.

This would also be the case for noisy LRCS or LRMC if one assumes a small enough bound on the noise (noise to signal ratio smaller than c times the final desired error ϵ).

2. If we want to obtain error bounds without making any assumptions on the noise, then the Err term does not satisfy the above bound. This is also the case for Byzantine-resilient AltGDmin for the

vertically federated LRCS and (any) federated LRMC setting. Both cases are instances of heterogeneous gradients. In these cases, suppose a much looser bound on $\|\text{Err}_t\|$ holds: suppose that it is of the order of the initial error δ_0 , i.e., suppose that

$$\max_t \|\text{Err}_t\| \leq 0.1\delta_0\sigma_{\min}^*{}^2$$

Substituting this bound only into the denominator term of (7.1) (in the numerator we leave $\|\text{Err}_t\|$ as is, this allows for a tighter bound in cases where the error is much smaller than its upper bound)

$$\begin{aligned} \delta_{t+1} &\leq \delta_t \left(1 - \frac{c_\eta}{\kappa^2} \left(0.9 - c_1 - 0.1 - \frac{2\|\text{Err}\|}{\sigma_{\min}^*{}^2} \right) \right) \\ &\quad + \left(1 + \frac{c_\eta}{\kappa^2} \left(0.1 + \frac{2\|\text{Err}\|}{\sigma_{\min}^*{}^2} \right) \right) \frac{c_\eta}{\sigma_{\max}^*{}^2} \|\text{Err}\| \\ &\leq \delta_t \left(1 - \frac{c_\eta}{\kappa^2} \left(0.8 - c_1 - \frac{2\|\text{Err}\|}{\sigma_{\min}^*{}^2} \right) \right) \\ &\quad + \left(1 + \frac{c_\eta}{\kappa^2} \left(0.1 + \frac{2\|\text{Err}\|}{\sigma_{\min}^*{}^2} \right) \right) \frac{c_\eta}{\sigma_{\max}^*{}^2} \|\text{Err}\| \\ &\leq \delta_t \left(1 - \frac{c_\eta}{\kappa^2} (0.8 - c_1 - 0.2\delta_0) \right) \\ &\quad + \left(1 + \frac{c_\eta}{\kappa^2} (0.1 + 0.2\delta_0) \right) \frac{c_\eta}{\sigma_{\max}^*{}^2} \|\text{Err}\| \end{aligned}$$

Using the upper bound on δ_0 from Assumption 7.1 (it assumes a bound on δ_t for any $t \geq 0$), we get

$$\delta_{t+1} \leq \delta_t \left(1 - \frac{c_\eta}{\kappa^2} (0.8 - c_1 - 0.02) \right) + 1.12c_\eta \frac{\|\text{Err}_t\|}{\sigma_{\max}^*{}^2} \quad (7.2)$$

Using this inequality, and the bound on $\|\text{Err}_t\|$, we can argue that $\delta_t \leq \delta_0$ for all t . To see this, suppose $\delta_t \leq \delta_0$. Using above, and $c_\eta < 1$, $\delta_{t+1} \leq \delta_0(1 - 0.78 + c_1) + 0.112\delta_0 = (0.22 + c_1 + 0.112)\delta_0 < \delta_0$ since $c_1 < 0.2$. The recursion in (7.2) can be simplified to get

$$\begin{aligned}
\delta_t &\leq \left(1 - \frac{c_\eta}{\kappa^2}(0.78 - c_1)\right)^t \delta_0 \\
&\quad + \sum_{\tau=1}^t \left(1 - \frac{c_\eta}{\kappa^2}(0.78 - c_1)\right)^{t-\tau} 1.12 c_\eta \kappa^2 \frac{\|\text{Err}_\tau\|}{\sigma_{\min}^{*2}} \\
&\leq \left(1 - \frac{c_\eta}{\kappa^2}(0.78 - c_1)\right)^t \delta_0 + \frac{1.12}{c_\eta(0.78 - c_1)} \kappa^4 \max_{\tau \in [t]} \frac{\|\text{Err}_\tau\|}{\sigma_{\min}^{*2}}
\end{aligned} \tag{7.3}$$

In summary, if all the bounds from the noise-free case holds and if $\|\text{Err}_t\| \leq 0.1\delta_0\sigma_{\min}^{*2}$, then (7.3) holds.

8

Linear Algebra and Random Matrix Theory Preliminaries

Most of the below review is taken from [53].

8.1 Linear Algebra: Maximum and Minimum Singular Value and the Induced 2-norm

We denote the hyper-sphere in \mathbb{R}^n by \mathcal{S}_{n-1} ; thus $\mathcal{S}_n := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| = 1\}$.

For any matrix \mathbf{M} of size $n_1 \times n_2$,

$$\begin{aligned}\|\mathbf{M}\| &= \sigma_{\max}(\mathbf{M}) \\ &= \sigma_1(\mathbf{M}) = \max_{\mathbf{x} \in \mathcal{S}_{n_2-1}} \|\mathbf{M}\mathbf{x}\| \\ &= \max_{\mathbf{x} \in \mathcal{S}_{n_2-1}, \mathbf{y} \in \mathcal{S}_{n_1-1}} \mathbf{y}^\top \mathbf{M}\mathbf{x} \\ &= \sqrt{\lambda_{\max}(\mathbf{M}^\top \mathbf{M})}\end{aligned}$$

If \mathbf{M} is a symmetric matrix, then

$$\|\mathbf{M}\| = \max_{\mathbf{x} \in \mathcal{S}_{n-1}} |\mathbf{x}^\top \mathbf{M}\mathbf{x}|$$

For an $n_1 \times n_2$ matrix \mathbf{M} ,

$$\sigma_{\min}(\mathbf{M}) := \sigma_{n_2}(\mathbf{M}) := \min_{\mathbf{x} \in \mathcal{S}_{n_2-1}} \|\mathbf{M}\mathbf{x}\| = \sqrt{\lambda_{\min}(\mathbf{M}^\top \mathbf{M})}$$

i.e., it is the n_2 -the singular value. Thus, for a rectangular matrix, $\sigma_{\min}(\mathbf{M}) \neq \sigma_{\min}(\mathbf{M}^\top)$ but $\sigma_{\max}(\mathbf{M}) = \sigma_{\max}(\mathbf{M}^\top)$ and, more generally,

$$\sigma_i(\mathbf{M}^\top) = \sigma_i(\mathbf{M})$$

Weyl's inequality for singular values implies that

$$\sigma_i(\mathbf{M}) - \|\mathbf{A}\| \leq \sigma_i(\mathbf{M} + \mathbf{A}) \leq \sigma_i(\mathbf{M}) + \|\mathbf{A}\|$$

For an $n_1 \times n_2$ matrix \mathbf{M} and an $n_2 \times n_3$ matrix \mathbf{A} , if \mathbf{A} has rank n_3 , then,

$$\begin{aligned} \sigma_{\min}(\mathbf{M}\mathbf{A}) &= \min_{\mathbf{x} \in \mathcal{S}_{n_3-1}} \|\mathbf{M}\mathbf{A}\mathbf{x}\| \cdot \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{A}\mathbf{x}\|} \\ &\geq \sigma_{\min}(\mathbf{M}) \min_{\mathbf{x} \in \mathcal{S}_{n_3-1}} \|\mathbf{A}\mathbf{x}\| \\ &= \sigma_{\min}(\mathbf{M})\sigma_{\min}(\mathbf{A}) \end{aligned}$$

8.2 Linear Algebra: Wedin and Davis-Kahan $\sin \Theta$ Theorems

The following results are used to bound the subspace distance between the top r singular or eigen vectors of a matrix and its estimate.

Theorem 8.1 (Davis-Kahan for eigenvectors of symmetric matrices). For symmetric matrices $\mathbf{S}, \hat{\mathbf{S}}$, let $\mathbf{U}, \hat{\mathbf{U}}$ denote the matrices of their top r eigenvectors respectively. Then,

$$\text{SubsDist}(\mathbf{U}, \hat{\mathbf{U}}) \leq \frac{\|\mathbf{S} - \hat{\mathbf{S}}\|}{\lambda_r(\mathbf{S}) - \lambda_{r+1}(\hat{\mathbf{S}})} \leq \frac{\|\mathbf{S} - \hat{\mathbf{S}}\|}{\lambda_r(\mathbf{S}) - \lambda_{r+1}(\mathbf{S}) - \|\mathbf{S} - \hat{\mathbf{S}}\|}$$

Also, let \mathbf{u}_i denote the i -th eigenvector. Then, we have the following bound.

$$\sin \theta(\mathbf{u}_i, \hat{\mathbf{u}}_i) \leq \frac{\|\mathbf{S} - \hat{\mathbf{S}}\|}{\min_{j \neq i} |\lambda_j(\mathbf{S}) - \lambda_i(\mathbf{S})|}$$

Here $\sin \theta(\mathbf{u}_i, \hat{\mathbf{u}}_i) = \sqrt{1 - (\mathbf{u}_i^T \hat{\mathbf{u}}_i)^2}$

Theorem 8.2 (Wedin $\sin \Theta$ theorem for Frobenius norm subspace distance [16, 54][Theorem 2.3.1]). For two $n_1 \times n_2$ matrices \mathbf{M}^*, \mathbf{M} , let \mathbf{U}^*, \mathbf{U} denote the matrices containing their top r left singular vectors and let $\mathbf{V}^{*\top}, \mathbf{V}^\top$ be the matrices of their top r right singular vectors (recall

from problem definition that we defined SVD with the right matrix transposed). Let $\sigma_r^*, \sigma_{r+1}^*$ denote the r -th and $(r+1)$ -th singular values of \mathbf{M}^* . If $\|\mathbf{M} - \mathbf{M}^*\| \leq \sigma_r^* - \sigma_{r+1}^*$, then

$$\begin{aligned} \text{SubsDist}_F(\mathbf{U}, \mathbf{U}^*) \\ \leq \frac{\sqrt{2} \max(\|(\mathbf{M} - \mathbf{M}^*)^\top \mathbf{U}^*\|_F, \|(\mathbf{M} - \mathbf{M}^*)^\top \mathbf{V}^{*\top}\|_F)}{\sigma_r^* - \sigma_{r+1}^* - \|\mathbf{M} - \mathbf{M}^*\|} \\ \\ \text{SubsDist}_2(\mathbf{U}, \mathbf{U}^*) \\ \leq \frac{\sqrt{2} \max(\|(\mathbf{M} - \mathbf{M}^*)^\top \mathbf{U}^*\|, \|(\mathbf{M} - \mathbf{M}^*)^\top \mathbf{V}^{*\top}\|)}{\sigma_r^* - \sigma_{r+1}^* - \|\mathbf{M} - \mathbf{M}^*\|} \end{aligned}$$

8.3 Probability Results: Markov's Inequality and its Use to Prove Concentration Bounds

All the concentration bounds stated below use the Markov inequality, which itself is an easy application of the integral identity

Theorem 8.3 (Markov's inequality). For a non-negative random variable (r.v.) Z ,

$$\Pr(Z > s) \leq \frac{\mathbb{E}[Z]}{s}$$

This result forms the basis of the entire set of results on non-asymptotic random scalar, vector, and matrix theory. We obtain the Chebyshev inequality by applying Markov's inequality to $Z = |X - \mu|$ with $\mu = \mathbb{E}[X]$. For all the other inequalities we use the Chernoff bounding technique explained below. This requires using an upper bound on the moment generating function (MGF) of the r.v.'s. This, in turn, requires assuming that the r.v.'s belong to a certain class of "nice enough" probability distributions (bounded, sub-Gaussian, or sub-exponential). With making one of these assumptions, the probability bound obtained is much tighter (decays exponentially) than what Chebyshev provides. However, the Chebyshev bound is the most general since it does not assume any distribution on the r.v.s.

8.4 Probability Results: Chernoff Bounding Idea

The MGF of a random variable (r.v.) X is defined as

$$M_X(\lambda) := \mathbb{E}[\exp(\lambda X)]$$

Chernoff bounding involves applying the Markov inequality to $Z = e^{tX}$ for any $t \geq 0$. Notice e^{tX} is always non-negative.

$$\Pr(X > s) = \Pr(e^{tX} > e^{ts}) \leq e^{-ts} \mathbb{E}[e^{tX}] = e^{-ts} M_X(t)$$

Since this bound holds for all $t \geq 0$, we can take a $\min_{t \geq 0}$ of the RHS or we can substitute in any convenient value of t .

If $S = \sum_{i=1}^m X_i$ with X_i 's independent, then $M_X(\lambda) = \prod_i M_{X_i}(\lambda)$. The next step involves either using an exact expression for MGF or a bound on the MGF for a class of distributions, e.g., Hoeffding's lemma. This is often followed by a scalar inequality such as $1 + x \leq e^x$ or using $\cosh(x) \leq e^{x^2/2}$ (or other bounds) to simplify the expressions to try to get a summation over i in the exponent.

$$\begin{aligned} \Pr(S > s) &= \Pr\left(\sum_i X_i > s\right) \\ &= \Pr\left(e^{t \sum_i X_i} > e^{ts}\right) \\ &\leq e^{-ts} \mathbb{E}[e^{t \sum_i X_i}] \\ &\leq \min_{\lambda \geq 0} e^{-\lambda s} \prod_i M_{X_i}(\lambda) \end{aligned}$$

The final step is to minimize over $\lambda \geq 0$ by differentiating the expression and setting it to zero, or picking a convenient value of $\lambda \geq 0$ to substitute.

A similar approach is then used to bound $\Pr(\sum_i X_i < -s)$. The only difference is we use $Z = e^{-t \sum_i X_i}$ for $t \geq 0$ and so,

$$\begin{aligned} \Pr(S < -s) &= \Pr\left(\sum_i X_i < -s\right) \\ &= \Pr\left(e^{-t \sum_i X_i} > e^{-t \cdot (-s)}\right) \\ &\leq e^{-ts} \mathbb{E}[e^{-t \sum_i X_i}] \\ &\leq \min_{\lambda \geq 0} e^{-\lambda s} \prod_i M_{X_i}(-\lambda) \end{aligned}$$

Combine both of the above bounds to bound

$$\Pr\left(\left|\sum_i X_i\right| > s\right) = \Pr\left(\sum_i X_i > s\right) + \Pr\left(\sum_i X_i < -s\right)$$

8.5 Probability Results: Bounds on Sums of Independent Scalar r.v.s (Scalar Concentration Bounds)

The results summarized below are taken from [53, Chap 2]. We state these results for sums of zero mean r.v.s. However, usually these are applied to show concentration of sums of nonzero mean r.v.s around their means. Given a set of nonzero mean r.v.s Z_i ,

$$X_i = Z_i - \mathbb{E}[Z_i]$$

is zero mean.

The first result below, Chebyshev inequality, requires no assumptions on the r.v.s except that they have a finite second moment. But its probability bound is also the weakest. The three results below it are for sums of bounded, sub-Gaussian, and sub-exponential r.v.s.

Theorem 8.4 (Chebyshev's inequality). Let $X_i, i = 1, 2, \dots, n$ be independent r.v.s with $\mathbb{E}[X_i^2] < \infty$. Then,

$$\Pr\left(\left|\sum_i X_i\right| > t\right) \leq \frac{1}{t^2} \sum_i \mathbb{E}[X_i^2]$$

Theorem 8.5 (Bounded Bernstein inequality). Let $X_i, i = 1, 2, \dots, n$ be independent zero-mean bounded r.v.s with $\Pr(-M_i \leq X_i \leq M_i) = 1$. Let $\sigma_i^2 := \max(\mathbb{E}[X_i^2])$. Then

$$\Pr\left(\left|\sum_i X_i\right| \geq t\right) \leq 2 \exp\left(-\frac{0.5t^2}{\sum_i \sigma_i^2 + 0.33(\max_i M_i)t}\right)$$

Definition 8.1 (Sub-Gaussian and Sub-exponential r.v.). We say a r.v. X is sub-Gaussian with sub-Gaussian norm K if $\Pr(|X| > t) \leq 2 \exp(-t^2/K^2)$. Equivalently, $K = C \sup_{p \geq 1} \frac{1}{\sqrt{p}} \mathbb{E}[|X|^p]^{1/p}$.

We say a r.v. X is sub-exponential with sub-exponential norm K if $\Pr(|X| > t) \leq 2 \exp(-t/K)$. Equivalently, $K = C \sup_{p \geq 1} \frac{1}{p} \mathbb{E}[|X|^p]^{1/p}$.

Fact 8.1 (Product of sub-Gaussians is sub-exponential). For two sub-Gaussian r.v.s X, Y with sub-Gaussian norms K_X, K_Y , the r.v. $Z := XY$ is sub-exponential with sub-exponential norm $K_X K_Y$.

Theorem 8.6 (Sub-Gaussian Hoeffding inequality). Let X_1, X_2, \dots, X_n be independent zero-mean sub-Gaussian r.v.s with sub-Gaussian norm K_i . Then, for every $t \geq 0$,

$$\Pr\left(\left|\sum_i X_i\right| \geq t\right) \leq 2 \exp\left(-c \frac{t^2}{\sum_i K_i^2}\right)$$

Theorem 8.7 (Sub-exponential Bernstein inequality). Let X_1, X_2, \dots, X_n be independent zero-mean sub-exponential r.v.s with sub-exponential norm K_i . Then, for every $t \geq 0$,

$$\Pr\left(\left|\sum_i X_i\right| \geq t\right) \leq 2 \exp\left(-c \min\left(\frac{t^2}{\sum_i K_i^2}, \frac{t}{\max_i K_i}\right)\right)$$

8.6 Probability Results: Epsilon Netting Argument Used for Extending Union Bound to Uncountable but Compact Sets

The following discussion is taken from [53, Chap 4]. An “epsilon net” is a finite set of points that is used to “cover” a compact set by balls of radius ϵ . More precisely, it is a finite set of points that are such that any point on the compact set is within an ϵ distance of some point in the epsilon-net. We use the bounds on the size of the smallest epsilon-net that covers a hyper-sphere to convert a scalar concentration bound into a bound on the minimum and maximum singular values of a large random matrix.

Definition 8.2 (Epsilon net on a sphere). We say \mathcal{N}_ϵ is an ϵ -net covering \mathcal{S}_{n-1} in Euclidean distance if $\mathcal{N}_\epsilon \subset \mathcal{S}^{n-1}$ and if, for any $\mathbf{x} \in \mathcal{S}_{n-1}$, there exists a $\bar{\mathbf{x}} \in \mathcal{N}_\epsilon$ s.t. $\|\mathbf{x} - \bar{\mathbf{x}}\| \leq \epsilon$.

It can be shown, using volume arguments [53], that there exists an epsilon-net, \mathcal{N}_ϵ , covering \mathcal{S}_{n-1} whose cardinality can be bounded as

$$|\mathcal{N}_\epsilon| \leq (1 + 2/\epsilon)^n$$

Using this bound, the following result can be proved for obtaining a high probability bound on the ℓ_2 -norm of a matrix \mathbf{M} with random entries.

Theorem 8.8 (Bounding $\|\mathbf{M}\|$). For an $n \times r$ matrix \mathbf{M} and fixed vectors \mathbf{w}, \mathbf{z} with, $\mathbf{w} \in \mathcal{S}_n$ and $\mathbf{z} \in \mathcal{S}_r$, suppose that

$$|\mathbf{w}^\top \mathbf{M} \mathbf{z}| \leq b_0 \text{ w.p. at least } 1 - p_0.$$

where b_0 does not depend on \mathbf{w}, \mathbf{z} . Then,

$$\|\mathbf{M}\| \leq 1.4b_0 \text{ w.p. at least } 1 - \exp((\log 17)(n+r)) \cdot p_0$$

Proof. Denote ϵ_0 -nets covering \mathcal{S}_{n-1} and \mathcal{S}_{r-1} by $\bar{\mathcal{S}}_{n-1}$ and $\bar{\mathcal{S}}_{r-1}$. Using union bound, w.p. at least $1 - (1 + 2/\epsilon_0)^{n+r} p_0$,

- $\max_{\mathbf{w} \in \bar{\mathcal{S}}_{n-1}, \mathbf{z} \in \bar{\mathcal{S}}_{r-1}} |\mathbf{w}^\top \mathbf{M} \mathbf{z}| \leq b_0$ and
- $\|\mathbf{M}\| := \max_{\mathbf{w} \in \mathcal{S}_{n-1}, \mathbf{z} \in \mathcal{S}_{r-1}} \mathbf{w}^\top \mathbf{M} \mathbf{z} \leq \max_{\mathbf{w} \in \bar{\mathcal{S}}_{n-1}, \mathbf{z} \in \bar{\mathcal{S}}_{r-1}} |\mathbf{w}^\top \mathbf{M} \mathbf{z}| \leq \frac{1}{1-2\epsilon_0-\epsilon_0^2} b_0.$

The proof of the second item above follows that of Lemma 4.4.1 of [53].

Using $\epsilon_0 = 1/8$ gives the final conclusion. \square

8.7 Probability Results: Bounding Sums of Independent Matrix r.v.s (Matrix Concentration Bounds)

By combining Theorem 8.8 given above with the scalar sub-exponential Bernstein or sub-Gaussian Hoeffding inequalities, we obtain the following two results which have been widely used in the LR recovery and phase retrieval literature. These study sums of rank-one matrices which are outer products of specific types of random vectors (r.vec). The last result below is the matrix Bernstein inequality.

Corollary 8.9 (Sum of rank-one matrices that are outer products of two sub-Gaussian r.vecs. (repeated from Section 7.1)). Consider a sum of m zero-mean independent rank-one $n \times r$ random matrices $\mathbf{x}_i \mathbf{z}_i^\top$ with

$\mathbf{x}_i, \mathbf{z}_i$ being sub-Gaussian random vectors with sub-Gaussian norms $K_{x,i}, K_{z,i}$ respectively. For a $t \geq 0$,

$$\left\| \sum_{i=1}^m \mathbf{x}_i \mathbf{z}_i^\top \right\| \leq 1.4t$$

with probability at least

$$1 - \exp \left((\log 17)(n+r) - c \min \left(\frac{t^2}{\sum_i (K_{x,i}, K_{z,i})^2}, \frac{t}{\max_i (K_{x,i}, K_{z,i})} \right) \right)$$

By combining Theorem 8.8 with the scalar sub-Gaussian Hoeffding inequality, we conclude the following.

Corollary 8.10 (Sum of rank-one matrices that are outer products of a sub-Gaussian r. vec. and a bounded r.vec. (repeated from Section 7.1)). Consider a sum of m zero-mean independent rank-one $n \times r$ random matrices $\mathbf{x}_i \mathbf{z}_i^\top$ with \mathbf{x}_i being sub-Gaussian random vector with sub-Gaussian norms $K_{x,i}$ and \mathbf{z}_i being a bounded random vector with $\|\mathbf{z}_i\| \leq L_i$. Then, clearly, for any $\mathbf{w}, \mathbf{w}', \mathbf{w}^\top \mathbf{x}_i \mathbf{z}_i^\top \mathbf{w}'$ is a sub-Gaussian r.v. with sub-Gaussian norm $K_{x,i} L_i$. Thus, for a $t \geq 0$,

$$\left\| \sum_{i=1}^m \mathbf{x}_i \mathbf{z}_i^\top \right\| \leq 1.4t$$

with probability at least

$$1 - \exp \left((\log 17)(n+r) - c \frac{t^2}{\sum_i (K_{x,i} L_i)^2} \right)$$

For bounded matrices, the following matrix Bernstein result gives a much tighter bound than what would be obtained by combining scalar bounded Bernstein and Theorem 8.8. See Section 8 for details.

Theorem 8.11 (Matrix Bernstein (repeated from Section 7.1)). Let $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m$ be independent, zero-mean, $n \times r$ matrices with $\|\mathbf{X}_i\| \leq L$ for all $i = 1, 2, \dots, m$. Define the “variance parameter” of the sum

$$v := \max \left(\left\| \sum_i \mathbb{E}[\mathbf{X}_i \mathbf{X}_i^\top] \right\|, \left\| \sum_i \mathbb{E}[\mathbf{X}_i^\top \mathbf{X}_i] \right\| \right).$$

Then,

$$\left\| \sum_{i=1}^m \mathbf{X}_i \right\| \leq t$$

with probability at least

$$1 - 2 \exp \left(\log \max(n, r) - c \min \left(\frac{t^2}{v}, \frac{t}{L} \right) \right)$$

We can also obtain a corollary for the bounded Bernstein inequality by combining it with Theorem 8.8, but as we explain below, that is not useful. It is not as tight as directly using the matrix Bernstein inequality. We state this inequality as a remark next just to explain why it is not useful and why the matrix Bernstein inequality should be used instead for sums of bounded matrices.

Remark 8.1 (Sum of outer products of bounded random vectors (Not Useful)). Consider a sum of m zero-mean independent bounded $n \times r$ random matrices \mathbf{X}_i with $\|\mathbf{X}_i\| \leq M_i$. Let $\sigma_i^2 := \mathbb{E}[\|\mathbf{X}_i\|^2]$. For a $t \geq 0$,

$$\left\| \sum_{i=1}^m \mathbf{X}_i \right\| \leq 1.4t$$

with probability at least

$$1 - \exp \left((\log 17)(n + r) - c \min \left(\frac{t^2}{\sum_i \mathbb{E}[\|\mathbf{X}_i\|^2]}, \frac{t}{\max_i \|\mathbf{X}_i\|} \right) \right)$$

The matrix Bernstein bound given in Theorem 8.11 is always better than the result of the above Remark 8.1. The reason is that the positive term in the exponent is $\max(n, r)$ in the above case and $\log \max(n, r)$ in matrix Bernstein. Consider the negative term in the exponent. L is the same in both cases, but variance parameter v of matrix Bernstein is upper bounded by the one used in the above remark.

Part IV

**Open Questions: AltGDmin
and Generalized-AltGDmin
for Other Partly-decoupled
Problems**

9

Open Questions

We describe open questions next.

9.1 Guarantees for a General Optimization Problem

An open question is: what assumptions do we need on an optimization problem to show that AltGDmin for it will converge. And can we bound its iteration complexity. Moreover, can we prove results similar to those for GD for AltGDmin and under what assumptions. Finally, when can these be extended to analyze Stochastic-AltGDmin which replaces the GD step of AltGDmin by Stochastic GD.

9.2 Generalized AltGDmin

AltGDmin has been established to be a useful (faster, more communication-efficient, or both) modification of AltMin for certain partly decoupled problems. Its overall idea is to replace the slower of the two minimization steps of AltMin by a single GD step. For certain problems, the natural split up of the unknown variable set consists of three or more subsets, an example is robust PCA and extensions – robust matrix completion and robust LRCS – described below. For these problems,

one can generalize the AltGDmin idea as follows. Consider the block coordinate descent (BCD) algorithm which is a generalization of AltMin for multiple blocks. Replace minimization for the slowest variable set in BCD with GD. We explain the idea in detail below using robust PCA as an example.

9.3 Robust PCA and Extensions: A Partly Decoupled Example Problem for Generalized AltGDmin

The modern definition of Robust PCA [12, 14, 28, 43, 56] is the following: recover a LR matrix $\mathbf{X}^* = \mathbf{U}^* \mathbf{B}^*$ and a sparse matrix \mathbf{S}^* from observed data which is their sum, i.e., from

$$\mathbf{Y} := \mathbf{U}^* \mathbf{B}^* + \mathbf{S}^*$$

This problem occurs in foreground background separation for videos as well as in making sense of survey data with some outlier entries. Thus, the optimization problem to be solved is

$$\min_{\mathbf{U}, \mathbf{B}, \mathbf{S}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}} f(\mathbf{U}, \mathbf{B}, \mathbf{S}) := \sum_k \|\mathbf{y}_k - \mathbf{s}_k - \mathbf{U}\mathbf{b}_k\|_2^2$$

In the above, clearly, $\mathbf{Z}_{\text{slow}} = \mathbf{U}$ and $\mathbf{Z}_{\text{fast}} = \{\mathbf{B}, \mathbf{S}\}$. The first approach to try would be to use AltGDmin with this split-up. However, this is a bad idea for two reasons. First there is no good approach to jointly solve for \mathbf{B}, \mathbf{S} . Second, and more importantly, this approach is ignoring the important fact that the sparse component is the one with no bound on its entries' magnitudes, hence it is the one that needs to be initialized and updated first. The entries of \mathbf{U} are bounded because of the unit norm column constraint, while those of \mathbf{B} are bounded because of the incoherence assumption and bounded condition number. Our recommended approach in this case is to consider the following generalization of AltGDmin (Gen-AltGDmin).

- Initialize $\hat{\mathbf{S}}$ and then $\hat{\mathbf{U}}$: Initialize $\hat{\mathbf{S}}$ using thresholding with threshold proportional to $\alpha = \sigma_{\max}^* r/n$. Obtain $\hat{\mathbf{U}}$ as the top r singular vectors of $\mathbf{X}_0 := \mathbf{Y} - \hat{\mathbf{S}}$

- Run T iterations that alternate between the following three steps:
 - Obtain $\hat{\mathbf{b}}_k = \mathbf{U}^\top(\mathbf{y}_k - \mathbf{A}_k \hat{\mathbf{s}}_k)$ for all $k \in [q]$.
 - Obtain $\hat{\mathbf{s}}_k$ by thresholding $(\mathbf{y}_k - \mathbf{A}_k \hat{\mathbf{U}} \hat{\mathbf{b}}_k)$ using a threshold proportional to 0.5α , for all $k \in [q]$ (α decreases with iteration).
 - Obtain $\hat{\mathbf{U}}$ by a GD step followed by orthonormalization.

An open question is when does the above algorithm work? Can we bound its iteration complexity under a reasonable bound on the fraction of outliers (sparse entries) in each row and column?

For Robust LRCS, the goal is to recover $\mathbf{U}^* \mathbf{B}^*$ from $\mathbf{Y} := \mathbf{A}_k \mathbf{U}^* \mathbf{B}^* + \mathbf{A}_k \mathbf{S}^*$. The overall idea above will generalize. Thresholding will get replaced by solving a compressive sensing problem. Update of \mathbf{b}_k will solve an LS problem. Robust LRMC would be handled similarly with some changes because we can only hope to estimate the sparse component of the observed entries.

9.4 Partly Decoupled Tensor LR: Tensor LR Slicewise Sensing

The goal is to learn / recover / estimate a $(J + 1)$ -th order tensor from provided data. The class of tensor problems that we focus on treats the first J dimensions of the tensor differently than the last one [4]. This is used to model tensor time or user sequences, e.g., multidimensional image sequences, or multiple product ratings by users. For the dynamic MRI and federated sketching problems, this models the fact that we have time sequence of 2D or 3D (or higher-dimensional) images. If we consider videos or single-slice dynamic MRI, then $J = 2$. If we consider dynamic multi-slice MRI, then $J = 3$ and so on. For the recommender systems one, this models the fact that we are trying to design a system for J products and the last dimension is the different users. Thus for a Netflix movie and shows recommendation system, we would use $J = 2$, while if the products are movies, shows, and documentaries, we would use $J = 3$ and so on. in case MRI data for different types of imaging are combined.

For a $(J + 1)$ -th order tensor, \mathcal{Z} , we use \mathcal{Z}_k to denote the k -th frontal slice, e.g., if $J = 2$, then $\mathcal{Z}_k = \mathcal{Z}(:, :, k)$ is a matrix for each k . There are many ways to define rank and the notion of LR for tensors; we assume a Tucker LR model [32] on \mathcal{L} , since it is the most relevant model for our asymmetric setting. We define this next.

We need to learn a $(J + 1)$ -th order tensors \mathcal{L} of size $n_1 \times n_2 \times \cdots \times n_J \times q$ from data \mathcal{Y} of size $m_1 \times m_2 \times \cdots \times m_J \times q$ that satisfy

$$\mathcal{Y}_k := \mathcal{A}_k(\mathcal{L}_k + \mathcal{E}_k) \quad \text{for all } k \in [q] \quad (9.1)$$

$$\mathcal{L}_k = \mathcal{B}_k \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_J \mathbf{U}^{(J)} \quad \text{for all } k \in [q] \quad (9.2)$$

when $m_j \ll \min(n_j, q)$, \mathcal{E} is the modeling error or noise, \mathcal{L} is an unknown LR tensor with Tucker ranks r_1, r_2, \dots, r_J, q with \mathcal{B} being a $r_1 \times r_2 \cdots \times r_J \times q$ core tensor, $\mathbf{U}^{(j)}$ being $n_j \times r_j$ matrices denoting the subspace bases along the various dimensions. Here \times_j denotes the j -th mode product [32]. The function $\mathcal{A}_k(\cdot)$ is a *known* dense linear function or is element-wise nonlinear, e.g., in case of phase retrieval. To solve the above problem, we need to minimize

$$\begin{aligned} f(\mathbf{U}^{(j)}, j \in [J], \mathcal{B}, \mathcal{S}) := & \sum_k \|\mathcal{Y}_k - \mathcal{A}_k(\mathcal{S}_k) - \mathcal{A}_k(\mathcal{B}_k \times_1 \mathbf{U}^{(1)} \\ & \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_J \mathbf{U}^{(J)})\|_F^2 \end{aligned} \quad (9.3)$$

Notice that this problem is partly decoupled with $\mathcal{Z}_{slow} = \{\mathbf{U}^{(j)}, j = 1, 2, \dots, J\}$ being the coupled variables and $\mathcal{Z}_{fast} = \mathcal{B}$ being the decoupled one.

Special Case: 3D Tensor LRCS (LR Tensor Slice-wise Sensing). We define below the simplest tensor LR extension of LRCS for a third order tensor, thus $J = 2$. We studied this numerically in [4]. Our goal is to learn a LR tensor \mathcal{L} of size $n_1 \times n_2 \times q$ from available third-order tensor measurements \mathcal{Y} of size $m_1 \times m_2 \times q$ with $m_1 \ll n_1, m_2 \ll n_2$. In this case some of the tensor models simplify as given next.

$$\begin{aligned} \mathcal{Y}_k &= \mathcal{L}_k \times_1 \Phi_k \times_2 \Psi_k = \Phi_k \mathbf{L}_k \Psi_k^\top, \quad \text{for all } k \in [q], \\ \mathcal{L}_k &= \mathcal{B}_k \times_1 \mathbf{U} \times_2 \mathbf{V} = \mathbf{U} \mathbf{G}_k \mathbf{V}^\top \quad \text{for all } k \in [q] \end{aligned}$$

The first equation above models a linear tensor function. In this $J = 2$ special case, it simplifies as a product of the three matrices. The same is

true for the LR model in this case. Here \mathbf{U} is $n_1 \times r_1$, and \mathbf{V} is $n_2 \times r_2$. We need to solve $\min_{\mathbf{U}, \mathbf{V}, \mathbf{B}: \mathbf{U}^\top \mathbf{U} = \mathbf{I}, \mathbf{V}^\top \mathbf{V} = \mathbf{I}} f(\mathbf{U}, \mathbf{V}, \mathbf{B}) := \sum_{k=1}^q \|\mathbf{Y}_k - \Phi_k \mathbf{U} \mathbf{G}_k \mathbf{V}^\top \Psi_k^\top\|_F^2$. This is clearly a partly decoupled problem with $\mathbf{Z}_{\text{slow}} = \{\mathbf{U}, \mathbf{V}\}$ and $\mathbf{Z}_{\text{fast}} = \mathbf{B}$.

AltGDmin in this case proceeds as follows [4]. (1) We initialize using a modification of our matrix case idea. Define the initial tensor $\hat{\mathcal{L}}_0$ as follows: $(\hat{\mathcal{L}}_0)_k = \Phi_k^\top \mathbf{Y}_k \Psi_k$, $k \in [q]$. We initialize \mathbf{U} as the top r_1 singular vectors of the unfolded matrix $(\hat{\mathcal{L}}_0)_{(1)}$: this means unfold $\hat{\mathcal{L}}_0$ along the first dimension to get a matrix of size $n_1 \times n_2 q$; and \mathbf{V} as the top r_2 singular vectors of $(\hat{\mathcal{L}}_0)_{(2)}$. (2) We alternatively update \mathbf{B} and $\{\mathbf{U}, \mathbf{V}\}$ as follows. (a) Given $\{\mathbf{U}, \mathbf{V}\}$, update \mathbf{B} by minimizing the above cost function over it. Since this decouples, this step consists of q inexpensive least squares (LS) problems. For this $J = 2$ special case, these are obtained as $\mathbf{G}_k = (\Phi_k \mathbf{U})^\dagger \mathbf{Y}_k ((\Psi_k \mathbf{V})^\dagger)^\top$ for each $k \in [q]$. Here $\mathbf{M}^\dagger := (\mathbf{M}^\top \mathbf{M})^{-1} \mathbf{M}^\top$. (b) Given \mathbf{B} , we update $\{\mathbf{U}, \mathbf{V}\}$ using GD followed by projecting the output onto the space of matrices with orthonormal columns. This is done using the QR decomposition. The QR decomposition is very cheap, it is of order $n_1 r_1^2$ and $n_2 r_2^2$ respectively. It can be easily derived that the time complexity of the above algorithm is much lower than that of a matrix LRCS algorithm that vectorizes the first $J = 2$ dimensions. The same is true for the communication cost for a distributed implementation.

While the above algorithm converges numerically as shown in [4], its theoretical analysis is an open question.

9.5 Partly Decoupled Not-differentiable Problems

Clustering. Data clustering is another set of problems where an AltGDmin type approach can be very beneficial. We have data points $\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_q^*$ that belong to one of ρ classes with ρ being a small number. Each \mathbf{x}_k^* is an n -length vector (or matrix/tensor) and n is typically large. We do not know any of the class labels which we denote by $c_1^*, c_2^*, \dots, c_q^*$ and would like to find them. We assume that each of the ρ classes is represented by a “low-dimensional model”, \mathcal{M}_j , $j \in [\rho]$. For example, we could assume a Gaussian mixture model (GMM) where

data points from class j occur with probability p_j and have mean and a covariance matrix $\boldsymbol{\mu}_j$ and \mathbf{C}_j respectively. Without any assumptions on \mathbf{C}_j , there are too many ($n^2\rho$) unknowns. Often a diagonal assumption is made but this may not be valid if different entries of \mathbf{x}_k^* are correlated. A milder assumption is to assume that each \mathbf{C}_j is LR with rank $r \ll n$, i.e., $\mathbf{C}_j \stackrel{\text{SVD}}{=} \mathbf{U}_j \boldsymbol{\Sigma}_j^* \mathbf{U}_j^\top$ with \mathbf{U}_j being $n \times r$.

Then AltGDmin with $\mathbf{Z}_{\text{slow}} \equiv \{p_j, \boldsymbol{\mu}_j, \mathbf{U}_j, \boldsymbol{\Sigma}_j^*, j \in [\rho]\}$ (model parameters for all the classes) and $\mathbf{Z}_{\text{fast}} \equiv \{\mathbf{c}_k^*, k \in [q]\}$ (class labels) will be faster than the traditional AltMin based solution (k-means clustering).¹

Maximum or mixed linear regression. This [21, 58] is another problem that can be solved more efficiently using AltGDmin instead of the AltMin algorithm [21]. This assumes that $y = \max_{j \in [\rho]} \beta_j^{*\top} \mathbf{x} + w$. The goal is to learn the ρ regression vectors $\beta_j^*, j \in [\rho]$ using training data pairs $\{\mathbf{x}_k^*, y_k\}, k \in [q]$ with $q > \rho n$. Let $\mathbf{c}_k^*, k \in [q]$ denote the index of the best model for the k -the training data pair. AltMin [21] alternates between estimating \mathbf{c}_k^* 's given β_j 's (q decoupled scalar maximizations) and vice versa: use the now labeled data to learn the β_j 's by solving an LS problem. AltGDmin would replace this LS by a single GD step.

Unlabeled or shuffled sensing. This involves recovering \mathbf{x} from $\mathbf{y} := \mathbf{\Pi} \mathbf{A} \mathbf{x}$ when only \mathbf{y}, \mathbf{A} are given [1, 47]. Here \mathbf{x} is the unknown signal, and $\mathbf{\Pi}$ is an unknown $m \times m$ permutation matrix. This problem occurs in simultaneous location and mapping for robots, multi-target tracking, and record linkage. Polynomial time solutions for it are possible with some assumptions on $\mathbf{\Pi}$, e.g., $\mathbf{\Pi}$ is often assumed to be s -sparse: i.e., only s or less of its rows are permuted. In this case, one common solution is to initialize $\mathbf{\Pi}$ to \mathbf{I} and use AltMin to update \mathbf{x} and $\mathbf{\Pi}$ alternatively. The update of \mathbf{x} is a robust regression problem (LS with sparse outliers), while that of updating $\mathbf{\Pi}$ is a well-studied discrete optimization problem called linear assignment problem (LAP). AltMin can be speeded up if we replace it by AltGDmin: instead of updating

¹AltMin requires a full model parameters' estimate at each iteration; this is expensive when \mathbf{C}_j 's are not assumed to be diagonal.

\boldsymbol{x} by fully solving the robust regression problem in each iteration, we replace it by one GD step. Another assumption on $\boldsymbol{\Pi}$ is the s -local assumption [1]. Under this assumption, the LAP would be decoupled making it faster.

Appendix

A

Partly Decoupled Optimization Problem: Most General Definition

Consider an optimization problem $\arg \min_{\mathbf{Z}} g(\mathbf{Z})$. We say the problem is decoupled if it can be solved by solving smaller dimensional problems over disjoint subsets of \mathbf{Z} . To define this precisely, observe that any function $g(\mathbf{Z})$ can be expressed as a composition of γ functions, for a $\gamma \geq 1$,

$$g(\mathbf{Z}) = h(f^1(\mathbf{Z}), f^2(\mathbf{Z}), \dots, f^\gamma(\mathbf{Z})),$$

Here $h(\dots)$ is a function of γ inputs. This is true always since we can trivially let $\gamma = 1$, $h(\mathbf{Z}) = \mathbf{Z}$ and $f^1(\mathbf{Z}) = g(\mathbf{Z})$.

We say that the optimization problem is decoupled if, for a $\gamma > 1$, \mathbf{Z} can be split into γ disjoint subsets

$$\mathbf{Z} = [\mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_\gamma]$$

so that

$$\begin{aligned} \arg \min_{\mathbf{Z}} g(\mathbf{Z}) = & [\arg \min_{\mathbf{Z}_1} f^1(\mathbf{Z}_1), \arg \min_{\mathbf{Z}_2} f^2(\mathbf{Z}_2), \dots, \arg \min_{\mathbf{Z}_\ell} f^\ell(\mathbf{Z}_\ell), \\ & \dots, \arg \min_{\mathbf{Z}_\gamma} f^\gamma(\mathbf{Z}_\gamma)] \end{aligned}$$

Observe that, in general, $\arg \min$ is a set and the notation $[\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_\gamma]$ is short for their Cartesian product $\mathcal{S}_1 \times \mathcal{S}_2 \times \dots \times \mathcal{S}_\gamma$. In words, the

set $\arg \min_{\mathbf{Z}} f(\mathbf{Z}) = \{[\hat{\mathbf{Z}}_1, \hat{\mathbf{Z}}_2, \dots, \hat{\mathbf{Z}}_\gamma] : \hat{\mathbf{Z}}_1 \in \arg \min_{\mathbf{Z}_1} f^1(\mathbf{Z}_1), \hat{\mathbf{Z}}_2 \in \arg \min_{\mathbf{Z}_2} f^2(\mathbf{Z}_2), \dots, \hat{\mathbf{Z}}_\gamma \in \arg \min_{\mathbf{Z}_\gamma} f^\gamma(\mathbf{Z}_\gamma)\}$.

If $g(\mathbf{Z})$ is strongly convex, then the $\arg \min$ is one unique minimizer $\hat{\mathbf{Z}}$. In this case, the decoupled functions have a unique minimizer too and $\arg \min_{\mathbf{Z}_1} f^1(\mathbf{Z}_1)$ returns $\hat{\mathbf{Z}}_1$ and so on, and $\hat{\mathbf{Z}} = [\hat{\mathbf{Z}}_1, \hat{\mathbf{Z}}_2, \dots, \hat{\mathbf{Z}}_\gamma]$. Data-decoupled means that the above holds and that $ef^\ell(\mathbf{Z}_\ll)$ depends only on a disjoint subset \mathcal{D}_ℓ of the data \mathcal{D} . Let $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_\gamma]$. We use a subscript to denote the data. Data-decoupled means that

$$\begin{aligned} \arg \min_{\mathbf{Z}} f(\mathbf{Z}) = & [\arg \min_{\mathbf{Z}_1} f_{\mathcal{D}_1}^1(\mathbf{Z}_1), \arg \min_{\mathbf{Z}_2} f_{\mathcal{D}_2}^2(\mathbf{Z}_2), \dots, \arg \min_{\mathbf{Z}_\ell} f_{\mathcal{D}_\ell}^\ell(\mathbf{Z}_\ell), \\ & \dots \arg \min_{\mathbf{Z}_\gamma} f_{\mathcal{D}_\gamma}^\gamma(\mathbf{Z}_\gamma)] \end{aligned}$$

Most practical problems that are decoupled are often also data-decoupled. *Henceforth we use the term “decoupled” to also mean data-decoupled.*

Partly-decoupled is a term used for optimization problems for which the unknown variable \mathbf{Z} can be split into two parts, $\mathbf{Z} = \{\mathbf{Z}_{slow}, \mathbf{Z}_{fast}\}$, so that the optimization over one keeping the other fixed is “easy” (closed form, provably correct algorithm exists, or fast). Decoupled and data-decoupled w.r.t. \mathbf{Z}_{fast} means that decoupling holds only for minimization over \mathbf{Z}_{fast} . To be precise, let

$$\mathbf{Z}_{fast} = [(\mathbf{Z}_{fast})_1, (\mathbf{Z}_{fast})_2, \dots, (\mathbf{Z}_{fast})_\gamma] \text{ and } \mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_\gamma]$$

Then,

$$\begin{aligned} \arg \min_{\mathbf{Z}_{fast}} f(\mathbf{Z}_{slow}, \mathbf{Z}_{fast}) = & [\arg \min_{(\mathbf{Z}_{fast})_1} f_{\mathcal{D}_1}^1(\mathbf{Z}_{slow}, (\mathbf{Z}_{fast})_1), \dots, \\ & \arg \min_{(\mathbf{Z}_{fast})_\ell} f_{\mathcal{D}_\ell}^\ell(\mathbf{Z}_{slow}, (\mathbf{Z}_{fast})_\ell), \\ & \dots \arg \min_{\mathbf{Z}_\gamma} f_{\mathcal{D}_\gamma}^\gamma(\mathbf{Z}_{slow}, (\mathbf{Z}_{fast})_\gamma)] \end{aligned}$$

All the examples of partly decoupled optimization problems that we discuss in this work are those for which $g(\mathbf{Z}) = h(f^1, f^2, \dots, f^\gamma) = \sum_{\ell=1}^\gamma f^\ell$ is a sum of the γ functions f^ℓ . In this case, partly decoupled problems means that

$$\min_{\mathbf{Z}_{fast}} f(\mathbf{Z}_{slow}, \mathbf{Z}_{fast}) = \sum_{\ell} \min_{(\mathbf{Z}_{fast})_\ell} f_{\mathcal{D}_\ell}^\ell(\mathbf{Z}_{slow}, \mathbf{Z}_{fast_\ell})$$

References

- [1] A. A. Abbasi, A. Tasissa, and S. Aeron, “R-local unlabeled sensing: Improved algorithm and applications,” in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, pp. 5593–5597, 2022.
- [2] A. A. Abbasi and N. Vaswani, “Efficient federated low rank matrix completion,” *IEEE Trans. Info. Th.*, 2025. URL: <https://ieeexplore.ieee.org/document/10975055>.
- [3] F. P. Anaraki and S. Hughes, “Memory and computation efficient pca via very sparse random projections,” pp. 1341–1349, 2014.
- [4] S. Babu, S. Aviyente, and N. Vaswani, “Tensor low rank column-wise compressive sensing for dynamic imaging,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 1–5, 2023.
- [5] S. Babu, S. G. Lingala, and N. Vaswani, “Fast low rank compressive sensing for accelerated dynamic MRI,” *IEEE Trans. Comput. Imag.*, 2023.
- [6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [7] C. L. Byrne, “Alternating minimization and alternating projection algorithms: A tutorial,” *Journal of Optimization Theory and Applications*, vol. 156, no. 3, 2013, pp. 554–566.

- [8] T. Cai, X. Li, and Z. Ma, “Optimal rates of convergence for noisy sparse phase retrieval via thresholded wirtinger flow,” *The Annals of Statistics*, vol. 44, no. 5, 2016, pp. 2221–2251.
- [9] E. Candes and T. Tao, “Near optimal signal recovery from random projections: Universal encoding strategies?” *IEEE Trans. on Information Theory*, vol. 52(12), 2006, pp. 5406–5425.
- [10] E. J. Candes, X. Li, and M. Soltanolkotabi, “Phase retrieval via wirtinger flow: Theory and algorithms,” *IEEE Trans. Info. Th.*, vol. 61, no. 4, 2015, pp. 1985–2007.
- [11] E. J. Candes and B. Recht, “Exact matrix completion via convex optimization,” *Found. of Comput. Math.*, no. 9, 2008, pp. 717–772.
- [12] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?” *J. ACM*, vol. 58, no. 3, 2011.
- [13] A.-L. Cauchy, “Méthode générale pour la résolution des systèmes d’équations simultanées,” French, *Comptes Rendus de l’Académie des Sciences*, vol. 25, 1847, pp. 536–538.
- [14] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, “Rank-sparsity incoherence for matrix decomposition,” *SIAM Journal on Optimization*, vol. 21, 2 2011.
- [15] Y. Chen and E. Candes, “Solving random quadratic systems of equations is nearly as easy as solving linear systems,” pp. 739–747, 2015.
- [16] Y. Chen, Y. Chi, J. Fan, C. Ma, *et al.*, “Spectral methods for data science: A statistical perspective,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 5, 2021, pp. 566–806.
- [17] 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, no. 7, 2015, pp. 4034–4059.
- [18] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, “Exploiting shared representations for personalized federated learning,” in *International conference on machine learning*, PMLR, pp. 2089–2099, 2021.
- [19] I. Csiszár, “Information geometry and alternating minimization procedures,” *Statistics and Decisions, Dedewicz*, vol. 1, 1984, pp. 205–237.

- [20] S. S. Du, W. Hu, S. M. Kakade, J. D. Lee, and Q. Lei, “Few-shot learning via learning the representation, provably,” 2021.
- [21] A. Ghosh, A. Pananjady, A. Guntuboyina, and K. Ramchandran, “Max-affine regression: Parameter estimation for gaussian designs,” *IEEE Transactions on Information Theory*, vol. 68, no. 3, 2021, pp. 1851–1885.
- [22] A. C. Gilbert, J. Y. Park, and M. B. Wakin, “Sketched svd: Recovering spectral features from compressive measurements,” *arXiv preprint arXiv:1211.0361*, 2012.
- [23] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, “One sketch for all: Fast algorithms for compressed sensing,” in *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pp. 237–246, 2007.
- [24] G. H. Golub and C. F. Van Loan, “Matrix computations,” *The Johns Hopkins University Press, Baltimore, USA*, 1989.
- [25] M. Hardt and E. Price, “The noisy power method: A meta algorithm with applications,” *Advances in neural information processing systems*, vol. 27, 2014.
- [26] J. Holloway, M. S. Asif, M. K. Sharma, N. Matsuda, R. Horstmeyer, O. Cossairt, and A. Veeraraghavan, “Toward long-distance sub-diffraction imaging using coherent camera arrays,” *IEEE Trans Comput Imaging*, vol. 2, no. 3, 2016, pp. 251–265.
- [27] G. Jagatap, Z. Chen, S. Nayer, C. Hegde, and N. Vaswani, “Sample efficient fourier ptychography for structured data,” *IEEE Transactions on Computational Imaging*, vol. 6, 2019, pp. 344–357.
- [28] P. Jain and P. Netrapalli, “Fast exact matrix completion with finite samples,” in *Conference on Learning Theory*, PMLR, pp. 1007–1034, 2015.
- [29] 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,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, 2021, 1–210.
- [30] S. M. Kay, *Fundamentals of statistical processing: Estimation theory*. 1993.

- [31] R. H. Keshavan, A. Montanari, and S. Oh, “Matrix completion from a few entries,” *IEEE transactions on information theory*, vol. 56, no. 6, 2010, pp. 2980–2998.
- [32] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Review*, vol. 51, no. 3, 2009, pp. 455–500. DOI: 10.1137/07070111X.
- [33] J. Lin, S. Moothedath, and N. Vaswani, “Fast and sample efficient multi-task representation learning in stochastic contextual bandits,” in *International Conference on Machine Learning*, PMLR, pp. 30227–30251, 2024.
- [34] 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, 2011, pp. 1042–1054.
- [35] M. Lustig, J. M. Santos, D. L. Donoho, and J. M. Pauly, “K-t sparse: High frame rate dynamic MRI exploiting spatio-temporal sparsity,” in *Conf. of International Society for Magnetic Resonance in Medicine (ISMRM), Seattle, Washington*, May 2006.
- [36] P. Narayananamurthy and N. Vaswani, “Provable dynamic robust pca or robust subspace tracking,” *IEEE Transactions on Information Theory*, vol. 65, no. 3, 2018, pp. 1547–1577.
- [37] 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.*, 2023.
- [38] S. Nayer, P. Narayananamurthy, and N. Vaswani, “Phaseless PCA: Low-rank matrix recovery from column-wise phaseless measurements,” in *International Conference on Machine Learning*, PMLR, pp. 4762–4770, 2019.
- [39] S. Nayer, P. Narayananamurthy, and N. Vaswani, “Provable low rank phase retrieval,” *IEEE Transactions on Information Theory*, vol. 66, no. 9, 2020, pp. 5875–5903.
- [40] S. Nayer and N. Vaswani, “Sample-efficient low rank phase retrieval,” *IEEE Transactions on Information Theory*, vol. 67, no. 12, 2021, pp. 8190–8206.
- [41] P. Netrapalli, P. Jain, and S. Sanghavi, “Low-rank matrix completion using alternating minimization,” 2013.

- [42] P. Netrapalli, P. Jain, and S. Sanghavi, “Phase retrieval using alternating minimization,” in *Neur. Info. Proc. Sys. (NeurIPS)*, pp. 2796–2804, 2013.
- [43] P. Netrapalli, U. N. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, “Non-convex robust pca,” in *Neur. Info. Proc. Sys. (NeurIPS)*, 2014.
- [44] H. Qi and S. M. Hughes, “Invariance of principal components under low-dimensional random projection of the data,” in *19th IEEE International Conference on Image Processing*, pp. 937–940, 2012.
- [45] A. Ramamoorthy, R. Meng, and V. Girimaji, “Leveraging partial stragglers within gradient coding,” *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 60 382–60 402.
- [46] A. P. Singh and N. Vaswani, “Byzantine resilient and fast federated few-shot learning,” in *Forty-first International Conference on Machine Learning*, 2024.
- [47] M. Slawski and E. Ben-David, “Linear regression with sparsely permuted data,” *Electronic Journal of Statistics*, vol. 13, 2019, pp. 1–36.
- [48] R. S. Srinivasa, K. Lee, M. Junge, and J. Romberg, “Decentralized sketching of low rank matrices,” pp. 10 101–10 110, 2019.
- [49] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, “Gradient coding: Avoiding stragglers in distributed learning,” in *International Conference on Machine Learning*, PMLR, pp. 3368–3376, 2017.
- [50] K. K. Thekumparampil, P. Jain, P. Netrapalli, and S. Oh, “Statistically and computationally efficient linear meta-representation learning,” *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 18 487–18 500.
- [51] M. Udell and A. Townsend, “Why are big data matrices approximately low rank?” *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, 2019, pp. 144–160.
- [52] N. Vaswani, “Efficient federated low rank matrix recovery via alternating gd and minimization: A simple proof,” *IEEE Trans. Info. Th.*, 2024.

- [53] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47. Cambridge university press, 2018.
- [54] P.-Å. Wedin, “Perturbation bounds in connection with singular value decomposition,” *BIT Numerical Mathematics*, vol. 12, no. 1, 1972, pp. 99–111.
- [55] D. P. Woodruff *et al.*, “Sketching as a tool for numerical linear algebra,” *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, 2014, pp. 1–157.
- [56] X. Yi, D. Park, Y. Chen, and C. Caramanis, “Fast algorithms for robust pca via gradient descent,” in *Neur. Info. Proc. Sys. (NeurIPS)*, 2016.
- [57] Q. Zheng and J. Lafferty, “Convergence analysis for rectangular matrix completion using burer-monteiro factorization and gradient descent,” *arXiv preprint arXiv:1605.07051*, 2016.
- [58] K. Zhong, P. Jain, and I. S. Dhillon, “Mixed linear regression with multiple components,” in *Neur. Info. Proc. Sys. (NeurIPS)*, vol. 29, 2016.