Fast Federated Low Rank Matrix Completion

Ahmed Ali Abbasi, Shana Moothedath, and Namrata Vaswani

Department of Electrical and Computer Engineering, Iowa State University. Email: aabbasi1@iastate.edu

2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton) | 979-8-3503-2814-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/Allerton58177.2023.10313472

***Abstract*—For the low rank matrix completion problem, we develop a fast, communication-efficient and private algorithm, called Alternating Gradient Descent and Minimization (AltGDMin). The communication-efficiency and privacy claims assume a vertically federated setting in which each node observes some entries of a different subset of columns of the unknown LR matrix $\boldsymbol{X}^*$. By extensive simulation experiments, we demonstrate the superior communication efficiency of AltGDMin. We also numerically evaluate the sample complexity and show competitive performance compared to benchmark methods.**

I. Introduction

We develop a fast, communication-efficient and private algorithm, called Alternating gradient descent (GD) and minimization (AltGDMin), for solving the Low Rank Matrix Completion (LRMC) problem. The communication-efficiency and privacy claims assume a vertically federated setting in which each node observes some entries of a different subset of columns of the unknown LR matrix $\boldsymbol{X}^*$. AltGDMin is a novel algorithmic framework introduced in [1] for solving the LR column-wise compressive sensing (LRCCS) problem. [1]

1) Problem setup: We consider the problem of recovering rank-r matrix $\boldsymbol{X}^* \in \mathbb{R}^{n\times q}$, where $r \ll \min(n,q)$, from partial observations. Specifically, entry i of column j, denoted $\boldsymbol{X}^*_{ij}$, is observed, independently of all other observations, with probability p. The observed matrix $\boldsymbol{Y} \in \mathbb{R}^{n\times q}$ is

$$\boldsymbol{Y}_{ij} = \begin{cases} \boldsymbol{X}^*_{ij} & \text{with probability } p, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

As in all other works on LRMC, we assume μ-incoherence for the left and right singular vectors of $\boldsymbol{X}^*$. Let $\boldsymbol{X}^* \overset{\text{SVD}}{=} \boldsymbol{U}^*\boldsymbol{\Sigma}^*\mathbf{V}^*$. Then, we are assuming the following bounds on each row-norm of $\boldsymbol{U}^*$, $\|\mathbf{u}_i^{*\intercal}\|_2 \le \mu\sqrt{r/n}$, and column-norm of $\boldsymbol{V}^*$, $\|\boldsymbol{v}_k^*\|_2 \le \mu\sqrt{r/q}$, $\forall i \in [n]$, $k \in [q]$.

2) Federation: Assume that there are a total of γ nodes with $\gamma \le q$. Each node observes a subset of columns of $\boldsymbol{Y}$ defined in (1). We use $\mathcal{S}_\ell$ to denote the subset of columns of $\boldsymbol{Y}$ observed by node ℓ. The sets $\mathcal{S}_\ell$ are mutually disjoint and $\cup_{\ell=1}^{\gamma}\mathcal{S}_\ell = [q]$. All nodes can communicate with a central node or "center". The algorithm needs to maintain privacy. This means that the center should not be able reconstruct the entire matrix $\boldsymbol{X}^*$. Each node should only be able reconstruct its own sub-matrix of $\boldsymbol{X}^*$. This implicitly also means that the nodes' raw data cannot be shared with the center.

[1]This work was supported in part by NSF grant: 2213069.

3) Applications: An important application of the LRMC problem is product recommendation systems. For example, in the Netflix problem, the k-th column of $\boldsymbol{X}^*$, denoted $\boldsymbol{x}^*_k$ corresponds to ratings of all movies by user k. Of course, no user will rate all the movies, in fact each user rates a very small subset of all the movies. Thus, only a few entries of $\boldsymbol{X}^*$ are actually observed. The LR assumption is justified by the fact that ratings of different users depend only a few factors such as their age, ethinicity, location etc. When we assume that $\boldsymbol{X}^*$ is rank r, we are claiming that the number of factors governing the user ratings is at most r. A federated setting may involve different nodes denoting users in different households.

4) Notation: Ω denotes the set of indices (i,j) of the observed entries of $\boldsymbol{X}^* \in \mathbb{R}^{n\times q}$. $\boldsymbol{Y} = \boldsymbol{X}^*_\Omega$ is the matrix defined in (1), with entries in Ω equal to the corresponding entries of $\boldsymbol{X}^*$ and zero elsewhere. $\boldsymbol{x}^*_k \in \mathbb{R}^n$ denotes the k-th column of $\boldsymbol{X}^*$. $\boldsymbol{x}_i^{*\intercal} \in \mathbb{R}^{1\times q}$ denotes the i-th row of $\boldsymbol{X}^*$. $\|\boldsymbol{X}^*\|_{op}$ denotes the largest singular value of $\boldsymbol{X}^*$. $\Omega_{:,k} = \{(i,k) \mid (i,k) \in \Omega\}$ is the set of observed entries of $\boldsymbol{x}^*_k$, column k of $\boldsymbol{X}^*$. Similarly, $\Omega_{i,:} = \{(i,j) \mid (i,j) \in \Omega\}$ denotes the set observed entries of row i. $\|\cdot\|_2$ denotes the vector ℓ_2 norm. $\|\cdot\|_F$ denotes the matrix Frobenius norm. $\boldsymbol{M}^\dagger \triangleq (\boldsymbol{M}^\intercal\boldsymbol{M})^{-1}\boldsymbol{M}^\intercal$ denotes the Moore-penrose pseudo-inverse. $(\cdot)^\intercal$ denotes the transpose. For matrices $\boldsymbol{U}_1, \boldsymbol{U}_2$ with orthonormal columns, we define the subspace distance between their column spans as

$$\text{SD}(\boldsymbol{U}_1, \boldsymbol{U}_2) := \|(\mathbf{I} - \boldsymbol{U}_1\boldsymbol{U}_1^\intercal)\boldsymbol{U}_2\|_F. \tag{2}$$

A. Related work

Starting with the seminal work of [5, 6] which introduced a nuclear norm based convex relaxation, the LRMC problem has been extensively studied in the last decade and a half [2, 5, 6, 7, 8, 9, 10, 11, 12]. Two types of algorithms feature prominently in this literature. Convex relaxations [5, 6, 8] have optimal sample complexity, but these methods are slow. For ϵ-accuracy, the required number of iterations typically grows as $1/\sqrt{\epsilon}$. The first provably accurate Alternating Minimization (AltMin) algorithm with a spectral initialization was proposed in [2]. AltMin converges geometrically and requires a sample complexity of $O(nr^{4.5}\log(1/\epsilon))$, with subsequent work [13] improving the sample complexity to $O(nr^{2.5}\log(1/\epsilon))$. Later works proposed gradient descent (GD) based algorithms such as Projected GD (ProjGD) [4] and Alternating GD (AltGD) [3]. These methods need sample complexity of $O(nr^2\log^2 n\log^2 1/\epsilon)$ samples and also converge geometrically. Generally, per iteration, AltMin is slower than GD. However, GD approaches need a step size of $O(1/r)$ or smaller. Consequently, their iteration complexity is higher. The proposed AltGDMin algorithm is most closely related to

Algorithm	Time (node) per iter.	Time (center) per iter.	Comm. (node) per iter.	Comm. (center) per iter.	Iteration (T) Complexity	Private
AltMin (Exact) [2]	$\lvert\Omega_{:,k}\rvert r^2$	$\lvert\Omega\rvert r^2$	r	nr	$\log\left(\frac{1}{\epsilon}\right)$	no
AltMin (Private) [2]	$\max(\lvert\Omega_{:,k}\rvert r\kappa\log(\frac{1}{\epsilon}), \lvert\Omega_{:,k}\rvert r^2)$	0	$\lvert\Omega_{:,k}\rvert r\log(\frac{1}{\epsilon})$	nr	$\log\left(\frac{1}{\epsilon}\right)$	yes
AltGD [3]	$\lvert\Omega_{:,k}\rvert r$	$nr^2 + \lvert\Omega\rvert r$	$\lvert\Omega_{:,k}\rvert$	$\max(n,q)r$	$\mu r\kappa\log(\frac{1}{\epsilon})$	no
ProjGD [4]	0	$\lvert\Omega\rvert r^2$	$\lvert\Omega_{:,k}\rvert$	$\lvert\Omega\rvert$	$\log\left(\frac{n+q}{\epsilon}\right)$	no
AltGDMin (Proposed)	$\lvert\Omega_{:,k}\rvert r^2$	nr^2	$\lvert\Omega_{:,k}\rvert r$	nr	$\log(\frac{1}{\epsilon})$ (**conjecture**)	yes

TABLE I: κ, μ denote the condition number and incoherence parameter of $\boldsymbol{X}^* \in \mathbb{R}^{n\times q}$, and $|\Omega_{:,k}|$ denotes the number of measurements observed in column k. Iteration complexity is the number of iterations needed for an ϵ-accurate solution $\|\boldsymbol{X} - \boldsymbol{X}^*\|_F \le \epsilon\|\boldsymbol{X}^*\|_F$. The time costs are calculated assuming zero cost for addition/subtraction operations. Since $|\Omega_{:,k}| \le n$ and $|\Omega| \ge nr$, AltGDMin has the lowest $\max$(Time (center), Time (node)), while also being private.

the algorithmic framework introduced in [1] for solving the LRCCS problem. We discuss this further in Sec. II.

B. Contributions

We develop a novel GD-based algorithm called Alternating GD and Minimization (AltGDMin) for solving the LRMC problem. The maximum time complexity of AltGDMin per iteration is $\max(|\Omega_{:,k}|r^2, nr^2) = nr^2$ and the maximum communication complexity per iteration is $\max(|\Omega_{:,k}|r, nr) = nr$, where the maximum is of either the node or center complexity. As can be seen from Table I, both time-wise and communication-wise, AltGDMin is the most efficient. In addition, it is private, while the only other algorithm that is also private is AltMin implemented using GD to solve the LS problem for updating $\boldsymbol{U}$; we denote this by AltMin (Prvt). As we show in Table I and explain in Sec III, AltMin(Prvt) is slower than AltGDMin in a federated setting because of its higher communication cost. Using extensive simulation experiments, we also numerically compare the sample/iteration complexity of AltGDMin. Our experiments indicate that AltGDMin has lower sample/iteration complexity than ProjGD and AltGD, and slightly higher complexity than that of AltMin.

II. Proposed AltGDMin algorithm

As in [1], we split the unknown variables into two parts, $\boldsymbol{Z}_a, \boldsymbol{Z}_b$ and alternatively update them using (projected) GD for $\boldsymbol{Z}_a$ and minimization for $\boldsymbol{Z}_b$. We let $\boldsymbol{Z}_b$ be the subset of variables for which the minimization can be "decoupled", i.e., subsets of $\boldsymbol{Z}_b$ are functions of only a subset of $\boldsymbol{Y}$. Moreover, if these subsets of $\boldsymbol{Y}$ are observed at different distributed nodes (federated setting), this decoupling also implies that privacy is guaranteed. For the LRMC problem, we can pick either of $\boldsymbol{U}$ or $\boldsymbol{B}$ to be $\boldsymbol{Z}_b$ (and let the other be $\boldsymbol{Z}_a$) since the k-th column of $\boldsymbol{B}$ only depends on the k-th column of $\boldsymbol{Y}$, while the i-th row of $\boldsymbol{U}$ only depends on the i-th row of $\boldsymbol{Y}$. However, since we are assuming vertical federation (different columns of $\boldsymbol{Y}$ available at different nodes), we set $\boldsymbol{Z}_b \equiv \boldsymbol{B}$ and $\boldsymbol{Z}_a \equiv \boldsymbol{U}$.

Algorithm 1 AltGDMin

Require: partial observations $\boldsymbol{Y}$, rank r, step size η, and number of iterations T

1: Initialize $\boldsymbol{U}_0$ by top r singular vectors of $\boldsymbol{Y}$
2: **for** $i \in 1 \cdots T$ **do**
3: $\quad \boldsymbol{b}_k^{(t+1)} \leftarrow \boldsymbol{U}_{\Omega_{:,k}}^{(t)^\dagger} \boldsymbol{y}_{\Omega_{:,k}} \ \ \forall\, k \in [q]$
4: $\quad \boldsymbol{U} \leftarrow \boldsymbol{U}^{(t)} - \eta \nabla_{\boldsymbol{U}} f(\boldsymbol{U}^{(t)}, \boldsymbol{B}^{(t+1)})$
5: $\quad \boldsymbol{U}^{(t+1)} \leftarrow \mathrm{QR}(\boldsymbol{U})$
6: **Return** $\boldsymbol{U}, \boldsymbol{B}$

A. AltGDMin algorithm

Factorizing $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{B}$, where $\boldsymbol{U} \in \mathbb{R}^{n\times r}, \boldsymbol{B} \in \mathbb{R}^{r\times q}$, we propose to recover $\boldsymbol{X}^*$ from its observed entries $\boldsymbol{Y}$ by minimizing the following objective function

$$\min_{\boldsymbol{B},\, \boldsymbol{U}:\, \boldsymbol{U}^\intercal\boldsymbol{U}=\mathbf{I}} \|(\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{B})_\Omega\|_F^2. \tag{3}$$

The objective function in (3) was also considered in [2], but without the constraint $\boldsymbol{U}^\intercal\boldsymbol{U} = \mathbf{I}$. Different from that work which proposes alternating exact minimization of $\boldsymbol{U}$ and $\boldsymbol{B}$, we optimize (3) by alternating exact minimization of $\boldsymbol{B}$ and a single projected gradient step for $\boldsymbol{U}$. This change makes the resulting algorithm communication efficient and private, as described in section II-B. For $k \in [q]$,

$$\boldsymbol{b}_k^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{b}} \|\boldsymbol{y}_{\Omega_{:,k}} - \boldsymbol{U}_{\Omega_{:,k}}^{(t)}\boldsymbol{b}\|_2^2 = \boldsymbol{U}_{\Omega_{:,k}}^{(t)\dagger}\boldsymbol{y}_{\Omega_{:,k}}, \tag{4}$$

$$\boldsymbol{U}^{(t+1)} = \mathrm{QR}(\boldsymbol{U}^{(t)} - \eta\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}^{(t)}), \tag{5}$$

where $f(\boldsymbol{U}, \boldsymbol{B}) \triangleq \|(\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{B})_\Omega\|_F^2$ is the objective function from (3), $\boldsymbol{y}_{\Omega:,k} \in \mathbb{R}^{|\Omega:,k|}$ denotes the sub-vector of entries $\Omega_{:,k}$ of column $\boldsymbol{y}_k$, and $\mathrm{QR}(\boldsymbol{A})$ maps matrix $\boldsymbol{A} \in \mathbb{R}^{n\times r}$ to $\boldsymbol{Q} \in \mathbb{R}^{n\times r}$ such that $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$ is the QR decomposition of $\boldsymbol{A}$. The gradient is

$$\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}^{(t)}) = 2(\boldsymbol{U}\boldsymbol{B}^{(t)} - \boldsymbol{Y})_\Omega \boldsymbol{B}^{(t)\intercal}, \tag{6}$$

and step size η is set as discussed in Sec. IV. Since the optimization problem in (3) is non-convex, a 'good' initialization to (4), (5) is needed. Like [2, 3], we initialize $\boldsymbol{U}^{(0)}$ to the left-singular vectors of $\boldsymbol{Y}$. An upper bound on the initialization error $\mathrm{SD}(\boldsymbol{U}^{(0)}, \boldsymbol{U}^*)$ is given in Lemma 5.2 of [2].

B. Time/Comm. Complexity and Privacy of AltGDMin

1) Communication Complexity: Each node performs two operations: i) update of $\boldsymbol{b}_k$ by the least-squares solution, equation (4), and ii) computation of the partial gradient $[\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}^{(t)})]_k = 2(\boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{y}_k)_{\Omega_{:,k}}\boldsymbol{b}_k^\intercal \in \mathbb{R}^{|\Omega_{:,k}|\times r}$. The maximum upstream (node to center) communication complexity is therefore $\max_{k\in[q]}|\Omega_{:,k}|r$. Total node communication complexity, for all q nodes, is $\sum_{k=1}^{k=q}|\Omega_{:,k}|r = |\Omega|r$. The center communicates the updated $\boldsymbol{U}^{(t+1)}$ (line 5, Algorithm 1) to the nodes with a downstream (center to node) communication complexity of nr.

2) Computational Complexity: For Algorithm 1, line 3: the q least-squares problems can be solved with complexity $O(\sum_{k=1}^{k=q}|\Omega_{:,k}|r^2) = O(|\Omega|r^2)$. The complexity of least-squares solution with design matrix of size $a \times b$, where $a > b$, is $O(ab^2)$, see equation 11.14 in [14]. Line 4: the per-node gradient computation $(\boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{y}_k)_{\Omega_{:,k}}\boldsymbol{b}_k^\intercal$ costs $|\Omega_k|r$, with $|\Omega_{:,k}|r$ being the complexity of computing $\boldsymbol{z} \triangleq \boldsymbol{U}_{\Omega_{:,k}}\boldsymbol{b}_k$ and the same for $\boldsymbol{z}\boldsymbol{b}_k^\intercal$. The total node complexity for gradient computation is $|\Omega|r$. Line 5: The QR decomposition of $\boldsymbol{U} \in \mathbb{R}^{n\times r}$ is performed at the center and costs $O(2nr^2)$, see equation 10.9 in [14]. It follows that the the per-iteration complexity of Algorithm 1 is $O(|\Omega|r^2)$, which follows from noting that the dominating cost is that of solving the q least-squares problems in line 3.

3) Privacy: The algorithm is private because, given only the partial gradient terms $[\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}^{(t)})]_k = 2(\boldsymbol{U}\boldsymbol{b}_k - \boldsymbol{y}_k)_{\Omega_{:,k}}\boldsymbol{b}_k^\intercal$ and $\boldsymbol{U}$, the center cannot recover the unknown $\boldsymbol{b}_k, \boldsymbol{y}_k$.

III. Time/Communication Complexity of Benchmark Methods

1) AltMin [2]: AltMin can be federated by solving the $\boldsymbol{U}$-update least squares (LS) problem using gradient descent. While private, the federated implementation of AltMin has a much higher communication complexity than AltGDMin, as we discuss next. Note that the $\boldsymbol{U}$-update decouples along the rows of the matrix $\boldsymbol{Y}$ to form n least-squares problems

$$\boldsymbol{U}_i^{(t+1)} = \underset{\boldsymbol{U}_i\in\mathbb{R}^r}{\operatorname{argmin}} \sum_{i=1}^{i=n} \|\boldsymbol{B}_{\Omega_{i,:}}^{(t)\intercal}\boldsymbol{U}_i - \boldsymbol{y}_{\Omega_{i,:}}\|_2^2, \tag{7}$$

where $\boldsymbol{B}_{\Omega_{i,:}}^{(t)} \in \mathbb{R}^{r\times|\Omega_{i,:}|}$ is the subset of matrix $\boldsymbol{B}^{(t)} \in \mathbb{R}^{r\times q}$, comprised of columns with indices in $|\Omega_{i,:}|$, $\boldsymbol{U}_i \in \mathbb{R}^r$ represents the i-th row of the matrix $\boldsymbol{U}$ (represented as a column vector), and $\boldsymbol{y}_{\Omega_{i,:}} \in \mathbb{R}^{|\Omega_{i,:}}$ are the observed entries of row i of $\boldsymbol{Y}$ (represented as a column vector). The gradient of the objective function in (7) is

$$\nabla_{\boldsymbol{U}_i} = \underbrace{\boldsymbol{B}_{\Omega_{i,:}}}_{\triangleq\boldsymbol{A}} \underbrace{\left(\boldsymbol{B}_{\Omega_{i,:}}^\intercal\boldsymbol{U}_i - \boldsymbol{y}_{\Omega_{i,:}}\right)}_{\triangleq\boldsymbol{z}\in\mathbb{R}^{|\Omega_i|}}. \tag{8}$$

The gradient, with notation defined in (8), is

$$\nabla_{\boldsymbol{U}_i} = \sum_{k\in\Omega_{i,:}} \boldsymbol{A}_k\boldsymbol{z}_k, \tag{9}$$

where $\boldsymbol{A}_k$ is the k-th column of $\boldsymbol{A}$, and $\boldsymbol{z}_k$ is the k-th entry of $\boldsymbol{z}$. Each of the r-dimensional vectors in the linear combination in (9) can be calculated locally at node k. To compute the gradient once, node k communicates r-dimensional vectors $\boldsymbol{A}(:,k)\boldsymbol{z}_k \in \mathbb{R}^r$ (the elements of the linear combination in (9)) a total of $|\Omega_{:,k}|$ times. For an ϵ-accurate solution to the LS problem, the gradient is transmitted $\log(1/\epsilon)$ times, with a per-node communication complexity of $|\Omega_{:,k}|r\log(1/\epsilon)$. The total node communication complexity is $\sum_{k=1}^{k=q}|\Omega_{:,k}|r\log(1/\epsilon) = |\Omega|r\log(1/\epsilon)$. The center sums the partial gradient terms and communicates the updated $\boldsymbol{U}^{(t+1)}$ to the nodes with a downstream (center to node) communication complexity of $O(nr)$.

The per-iteration node computational cost of AltMin is at least as high as that of AltGDMin, and will be higher if $|\Omega_{:,k}|r\kappa_{\text{LS}}\log(\frac{1}{\epsilon}) > |\Omega_{:,k}|r^2$, where $|\Omega_{:,k}|r\kappa_{\text{LS}}\log(\frac{1}{\epsilon})$ is the cost of solving LS using gradient descent to ϵ accuracy. The condition number κ_{LS} is $\kappa_{\text{LS}} = O(\kappa)$, with high probability under the assumed sample complexity (see Section VI-B). Neglecting the cost of adding up partial gradient terms, AltMin has zero computational cost at the center, compared to $O(nr^2)$ cost of the QR decomposition for AltGDMin, but AltGDMin has lower communication cost.

2) ProjGD [15]: Each iteration of Projected GD (ProjGD) involves one step of GD w.r.t. the cost function to be minimized, followed by projecting onto the constraint set which is the set of rank r matrices [4, 10].

$$\boldsymbol{X}^{(t+1)} = \text{Proj}_r(\boldsymbol{X}^{(t)} - \eta\nabla_{\boldsymbol{X}}(\|(\boldsymbol{X}^{(t)} - \boldsymbol{X}^*)_\Omega\|_F^2)) \tag{10}$$

The nodes transmit the partial gradient $(\boldsymbol{y}_{:,\Omega_k} - \boldsymbol{x}_{:,\Omega_k}^{(t)}) \in \mathbb{R}^{|\Omega_{:,k}|}$ to the center. The center transmits $\boldsymbol{x}_{:,\Omega_k}^{(t+1)}$ to node k, with total downstream communication complexity $|\Omega|$. The center also computes the SVD of $\boldsymbol{X}^{(t)} - \eta\nabla_X$, see equation (10). Using the fact that $\boldsymbol{X}^{(t)}$ is rank-r, the SVD can be computed in $O(|\Omega|r^2)$ time. The method is not private because the center knows estimate $\boldsymbol{X}^{(t)}$.

3) Federated AltGD [3]: Alternating GD (AltGD) factorizes $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{B}$, and alternatively updates $\boldsymbol{U}$ and $\boldsymbol{B}$ using one GD step for $\boldsymbol{U}$ keeping $\boldsymbol{B}$ fixed and vice versa, followed by projecting each of them onto the set of matrices with incoherent rows and columns respectively. The GD steps are for the cost function $f(\boldsymbol{U}, \boldsymbol{B}) + \|\boldsymbol{U}^\intercal\boldsymbol{U} - \boldsymbol{B}\boldsymbol{B}^\intercal\|_F^2$. The second term is a norm balancing term that ensures the norm of $\boldsymbol{U}$ does not keep increasing with iterations while that of $\boldsymbol{B}$ decreases (or vice versa).

The gradients for $\boldsymbol{U}$ and $\boldsymbol{B}$ are $\nabla_{\boldsymbol{U}} = \nabla_{\boldsymbol{M}}\boldsymbol{B}^\intercal$ and $\nabla_{\boldsymbol{B}} = \boldsymbol{U}^\intercal\nabla_{\boldsymbol{M}}$, where $\nabla_{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{B} - \boldsymbol{Y}$. Node k computes and communicates the observed entries of the k-th column of $\nabla\boldsymbol{M} \in \mathbb{R}^{n\times q}$, with a per node computation and communication cost of $|\Omega_{:,k}|r$ and $|\Omega_{:,k}|$, respectively. The gradient operations $\nabla_{\boldsymbol{U}}$, $\nabla_{\boldsymbol{B}}$ and norm-balancing $\|\boldsymbol{U}^\intercal\boldsymbol{U} - \boldsymbol{B}\boldsymbol{B}^\intercal\|_F^2$, with cost $O(|\Omega_{:,k}|r), O(|\Omega_{:,k}|r)$, and $O(nr^2)$, respectively, are carried out at the center. AltGD is not private because the center has access to both $\boldsymbol{U}$ and $\boldsymbol{V}$.

4) SVD Init: AltMin [2], AltGD [3] and AltGDMin (Proposed) are initialized to the left-singular vectors of $\boldsymbol{Y}$, which are computed by the simultaneous power iteration method for $\boldsymbol{Y}\boldsymbol{Y}^\intercal$. Computing $\boldsymbol{M} \triangleq \boldsymbol{Y}(\boldsymbol{Y}^\intercal\boldsymbol{Z})$, where $\boldsymbol{Z} \in \mathbb{R}^{n\times r}$, costs $O(|\Omega|r)$. The QR decomposition of $\boldsymbol{M} \in \mathbb{R}^{n\times r}$, performed at

the center, costs $O(nr^2)$, see equation 10.9 in [14]. Combined, total iteration cost is $O(|\Omega|r^2)$, since $|\Omega| \geq n$. The upstream per-node communication cost $O(|\Omega_{:,k}|r)$ is that of transmitting $\boldsymbol{y}_{\Omega:,k}\boldsymbol{y}_{\Omega:,k}^{\mathsf{T}}\boldsymbol{Z}$. The downstream communication cost is of transmitting $\boldsymbol{Z}$ with $O(nr)$. The method converges linearly and for ϵ_{svd} accuracy, $\log(1/\epsilon_{svd})$ iterations are required.

IV. Simulation Results

MATLAB code to reproduce the results in this paper is at the first author's github repository https://github.com/aabbas02. For AltGD [3], we used the code provided by the authors of that work.

Synthetic data generation. The entries of matrix $\tilde{\boldsymbol{X}} \in \mathbb{R}^{n\times q}$ are sampled i.i.d. from the normal distribution. Rank-r matrix $\boldsymbol{X}^*$ is generated by truncating the r singular vectors and singular values of $\tilde{\boldsymbol{X}}$. The resulting matrix has incoherence $\mu(\boldsymbol{X}^*) \simeq 1$.

Parameter selection. The step size for both AltGDMin and AltMin (Private) was set equal to $\eta = p/\|\boldsymbol{Y}\|_{op}^2$. In the appendix (section VI), we show that our choice of η is based on an estimated upper bound of the initial operator norm $\|\mathbb{E}_\Omega[\nabla_{\boldsymbol{U}} f(\boldsymbol{U}, \boldsymbol{B}^{(0)})]\|_{op}$.

For AltMin (Prvt.), η is an upper bound on the Lipschitz constant of $\mathbb{E}_{\Omega_{i,:}}[\nabla_{\boldsymbol{U}_i} f(\boldsymbol{U}_i, \boldsymbol{B}^{(0)})]$, as shown in section VI-B. We also note that setting the step size as described showed consistently fast convergence for both algorithms. For AltGD, the step size was set $\eta_{\text{AltGD}} = pc/\|\boldsymbol{Y}\|_{op}$, as also done in the authors' own implementation of their algorithm. Setting $c = 0.75$ showed the fastest convergence for our simulations. For ProjGD, the step size was set $\eta_{\text{ProjGD}} = 1/p$, according to [15].

Figures 1a, 1c. Figure plots the subspace distance $\text{SD}(\boldsymbol{U}^{(t)}, \boldsymbol{U}^*)$ against iteration number. We note that, while all algorithms converge to $\boldsymbol{U}^*$, AltMin requires the fewest number of iterations for convergence, whereas ProjGD requires the highest number of iterations. For AltMin (Prvt.), to ensure that the the center does not have access to the nodes' raw data, the $\boldsymbol{U}$-update least-squares problems are solved by gradient descent, as discussed also in section III-1. The number of gradient descent iterations was set to 10, which is sufficient as the plot shows that the SD for AltMin (Prvt.) is identical to that for AltMin (Exact). The least squares problems are solved exactly by the closed form solution for AltMin (Exact). Figure 1c repeats the experiment in Figure 1a with rank $r = 10$.

Figures 1b, 1d. For the same simulation as in Fig. 1a, the subspace distance is now plotted against run-time. We observe that out of the two 'private' algorithms, AltMin (Prvt.) and AltGDMin, the proposed AltGDMin is significantly faster. For example, AltGDMin takes approximately 7 seconds to converge to $\boldsymbol{U}^*$, compared to nearly 15 seconds taken by AltMin (Prvt.). AltGDMin also has a lower runtime than AltGD and ProjGD, two methods which are centralized and not private. We now describe the details of our distributed simulation setup. For both AltMin and AltGDMin, we used the 'parfor' loop in MATLAB to distribute the $\boldsymbol{U}$ and $\boldsymbol{V}$ updates across 4 workers. The results show that, despite a small number of GD iterations (10 iterations), the communication overhead of transmitting the individual gradient terms in equation (9) greatly increases the runtime of AltMin(Prvt.). Finally, we note that the run-time for AltGDMin includes the cost of the QR decomposition of $\boldsymbol{U}^{(t)}$ performed at the center whereas, for AltMin (Prvt.) , the cost of adding up the individual gradient terms from the nodes was neglected. Figure 1d repeats the experiment in Figure 1b with rank $r = 10$.

Figure 2. AltGDMin is compared with benchmark methods based on sample complexity, that is, the number of observed entries required for successful matrix completion. The results show that AltGDMin has lower sample complexity than ProjGD and AltGD, and slightly higher sample complexity than that of AltMin. We emphasize, however, that AltGDMin is faster in a distributed setting that AltMin.

V. Conclusion and future work

We proposed the novel AltGDMin algorithm for low rank matrix completion. The proposed algorithm is fast and private in a federated setting. Numerical experiments confirm the efficiency of the proposed algorithm. In future work, we will bound the sample and iteration complexity of AltGDMin.

References

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

[2] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," 2012.

[3] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust pca via gradient descent," *Advances in neural information processing systems*, vol. 29, 2016.

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

[5] M. Fazel, "Matrix rank minimization with applications," *PhD thesis, Stanford Univ*, 2002.

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

[7] P. Netrapalli, P. Jain, and S. Sanghavi, "Phase retrieval using alternating minimization," in *Neur. Info. Proc. Sys. (NeurIPS)*, 2013, pp. 2796–2804.

[8] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Trans. Info. Th.*, vol. 56, no. 6, pp. 2980–2998, 2010.

[9] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via non-convex factorization," *IEEE Trans. Info. Th.*, vol. 62, no. 11, pp. 6535–6579, 2016.

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

[11] C. Ma, K. Wang, Y. Chi, and Y. Chen, "Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution," in *Intl. Conf. Machine Learning (ICML)*, 2018.

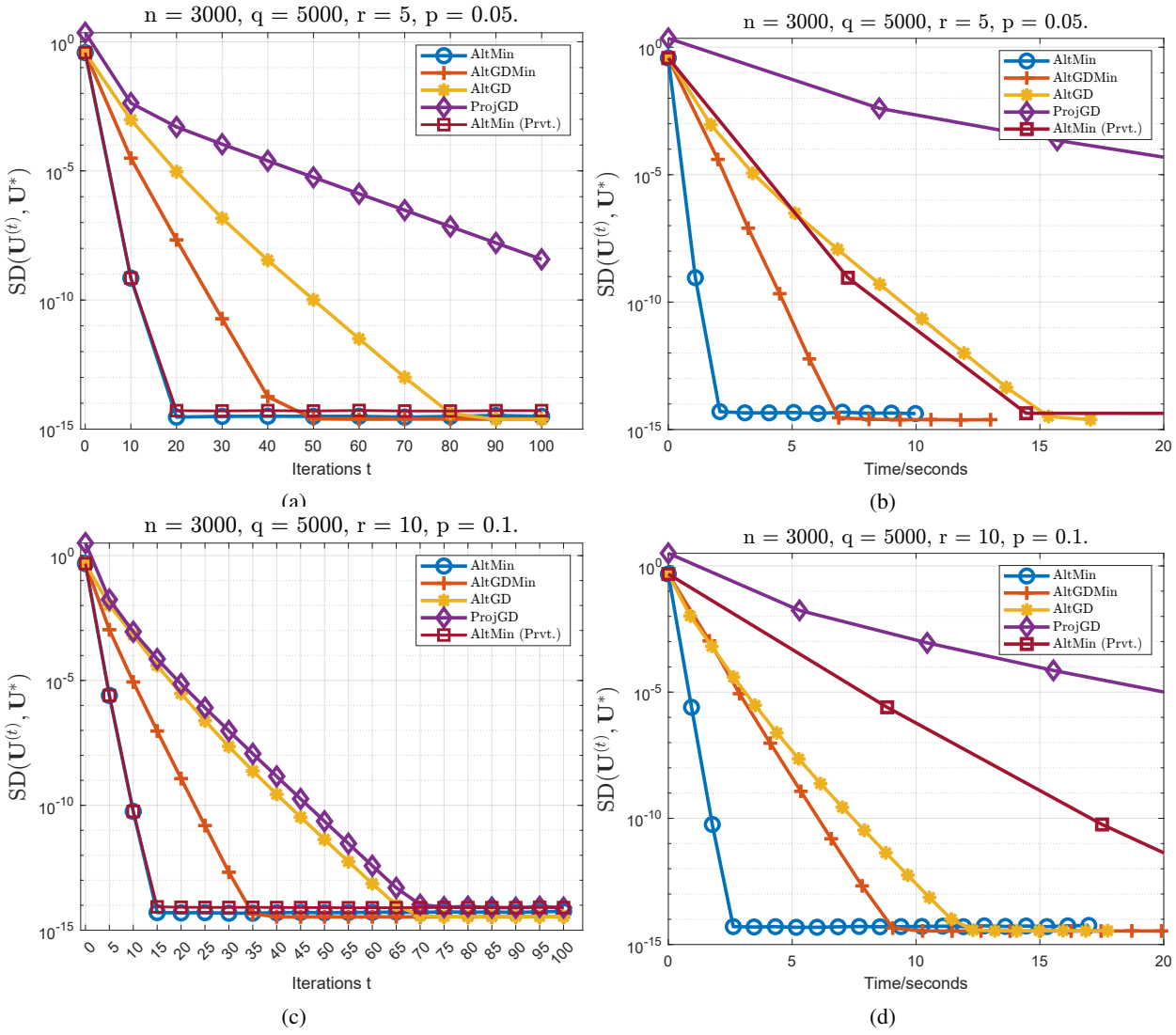


Fig. 1: Comparison of benchmark methods with AltGDMin based on the subspace distance $\mathrm{SD}(\boldsymbol{U}^{(t)}, \boldsymbol{U}^*)$, where $\boldsymbol{U}^{(t)}$ is an estimate of $\boldsymbol{U}^*$, the left-singular vectors of the rank-r matrix $\boldsymbol{X}^* \in \mathbb{R}^{n\times q}$. Of the two private algorithms, AltMin (Prvt.) and AltGDMin, observe, from figures 1b and 1d, that AltGDMin has a much lower runtime than AltMin (Prvt.). See section IV for an explanation of the results. For a description of the benchmark methods, see section III.

[12] M. Hardt and M. Wootters, "Fast matrix completion without the condition number," in *Conf. on Learning Theory*, 2014.

[13] M. Hardt, "Understanding alternating minimization for matrix completion," in *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 2014, pp. 651–660.

[14] L. N. Trefethen and D. Bau, *Numerical linear algebra*. Siam, 2022, vol. 181.

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

VI. Appendix

A. Step size selection for AltMin and AltGDMin

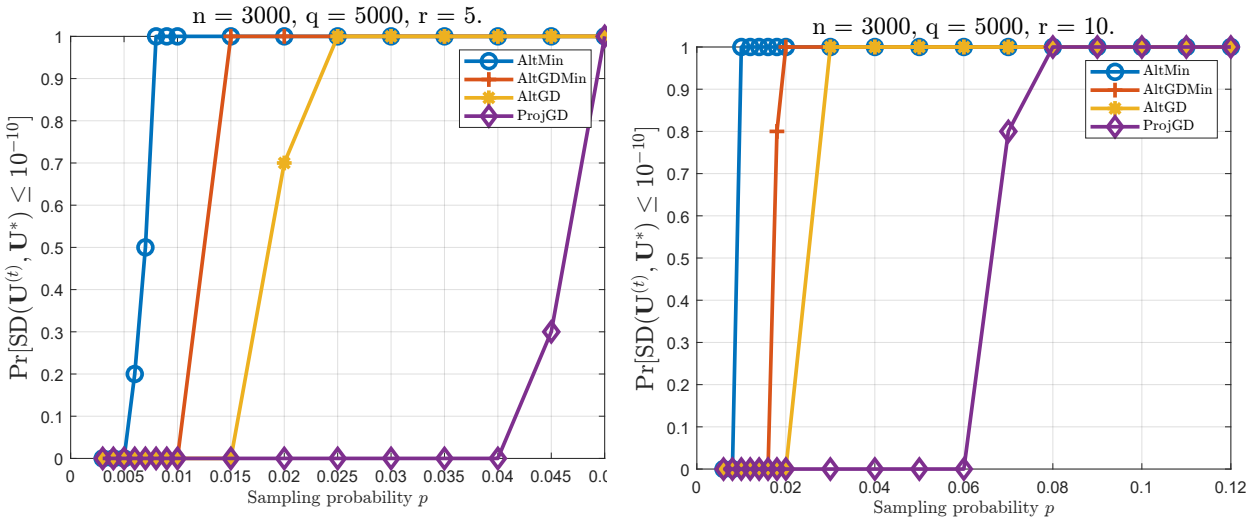
Recall notation that partial observations $\boldsymbol{Y} = \boldsymbol{X}^*_\Omega$ and that the indices Ω are sampled i.i.d. with probability p. For AltGDMin, the expectation of the gradient is

$$\mathbb{E}_\Omega[\nabla_{\boldsymbol{U}} f(\boldsymbol{U}^{(0)}, \boldsymbol{B}^{(1)})] = 2p(\boldsymbol{X}^{(0)} - \boldsymbol{X}^*)\boldsymbol{B}^{(1)\mathsf{T}}. \tag{11}$$

where $f(\boldsymbol{U}, \boldsymbol{B}) \triangleq \|(\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{B})_\Omega\|_F^2$ is the objective function from (3). Substituting $\boldsymbol{X}^{(0)} \triangleq \boldsymbol{U}^{(0)}\boldsymbol{B}^{(1)}$ and $\boldsymbol{X}^* = \boldsymbol{U}^*\boldsymbol{B}^*$ and using the triangle inequality, the operator norm is upper bounded as

$$\begin{aligned}\|2p(\boldsymbol{X}^{(0)} - \boldsymbol{X}^*)\boldsymbol{B}^{(1)\mathsf{T}}\|_{op} &\le 2p(\|\boldsymbol{B}^{(1)}\boldsymbol{B}^{(1)\mathsf{T}}\|_{op} + \|\boldsymbol{B}^{(1)}\boldsymbol{B}^{*\mathsf{T}}\|_{op}) \\ &\lesssim 2p\|\boldsymbol{X}^*\|_{op}^2, \end{aligned} \tag{12}$$

where we have used $\boldsymbol{U}^{(0)\mathsf{T}}\boldsymbol{U}^{(0)} = \boldsymbol{U}^{*\mathsf{T}}\boldsymbol{U}^* = \mathbf{I}$, that is $\sigma_{\max}(\boldsymbol{U}^{(0)}) = \sigma_{\max}(\boldsymbol{U}^*) = 1$, and substituted (22) for $t = 0$. We note that (22), derived for iterates of the alternating minimization (AltMin) algorithm in [2], also holds at $t = 0$ for AltGDMin. This is because the iterates $\boldsymbol{U}^{(0)}, \boldsymbol{B}^{(1)}$ for AltGDMin are the same as the corresponding AltMin iterates. Specifically, the iterates begin to differ with $\boldsymbol{U}^{(1)}$ being



(a)

Fig. 2: Sample complexity comparison of AltGDMin with benchmark methods.

updated by gradient descent for AltGDMin and least-squares for AltMin. We set the step-size η_{AltGDMin} proportional to the inverse of the operator norm, that is

$$\eta_{\text{AltGDMin}} = \frac{p}{\|\boldsymbol{Y}\|_{op}^2}, \tag{13}$$

where we have estimated $\|\boldsymbol{X}^*\|_{op} \simeq \|\boldsymbol{Y}\|_{op}/p$ because $\mathbb{E}_\Omega[\boldsymbol{Y}] = p\boldsymbol{X}^*$.

For AltMin, the expectation of the gradient with respect to row $\boldsymbol{U}_i \in \mathbb{R}^{1\times r}$ is $2p(\boldsymbol{X}_i^{(0)} - \boldsymbol{X}_i^*)\boldsymbol{B}^{(1)\intercal}$. Since $2p(\boldsymbol{X}_i^{(0)} - \boldsymbol{X}_i^*)\boldsymbol{B}^{(1)\intercal} = 2p(\boldsymbol{U}_i^{(0)}\boldsymbol{B}^{(1)}\boldsymbol{B}^{(1)\intercal} - \boldsymbol{X}_i^*\boldsymbol{B}^{(1)\intercal})$, note that $L = 2p\|(\boldsymbol{B}^{(1)})\|_{op}^2$ is the Lipschitz constant of the expectation of the gradient. By (22), $\|\boldsymbol{B}^{(t)}\|_{op} \le \|\boldsymbol{X}^*\|_{op}(1 + \frac{1}{20r})$. Therefore, $2\|\boldsymbol{Y}\|_{op}^2/p$ is an estimated upper bound on L.

B. Derivation of κ upper bound for AltMin(GD)

All steps till (20) are a reproduction of the original work in [2]. The rest is also a straightforward adaption of the proof of Lemma 5.7 therein. The notation for this subsection is the following: $\|\boldsymbol{M}\|_2$ and $\sigma_{\max}(\boldsymbol{M})$ both denote the largest singular value of $\boldsymbol{M}$ and $\sigma_{\min}$ denotes the minimum non-zero singular value. σ_1^*, $\sigma_{\min}^*$ denote singular values of $\boldsymbol{X}^*$. Following equation (18) in [2], let $\widehat{\boldsymbol{B}}^{(t+1)} = \boldsymbol{B}^{(t+1)}\boldsymbol{R}^{(t+1)}$ be the QR decomposition of the least squares solution $\widehat{\boldsymbol{B}}^{(t+1)}$. Then,

$$\begin{aligned}
\sigma_{\min}(\widehat{\boldsymbol{B}}^{(t+1)}) &= \sigma_{\min}(\boldsymbol{B}^{(t+1)}\boldsymbol{R}^{(t+1)}) \\
&= \min_{\boldsymbol{z},\|\boldsymbol{z}\|_2=1} \|\boldsymbol{B}^{(t+1)}\boldsymbol{R}^{(t+1)}\boldsymbol{z}\|_2 \\
&= \|\boldsymbol{V}^*\Sigma^*\boldsymbol{U}^{*\intercal}\boldsymbol{U}^{(t)}z - \boldsymbol{F}z\|_2 \qquad (14)\\
&\ge \sigma_{\min}^*\sigma_{\min}(\boldsymbol{U}^{*\intercal}\boldsymbol{U}^{(t)}) - \|\boldsymbol{F}\|_{op} \qquad (15)\\
&\ge \sigma_{\min}^*\sqrt{1-\text{SD}_2^2(\boldsymbol{U}^{(t)},\boldsymbol{U}^*)} - \|\boldsymbol{F}(\Sigma)^{-1}\Sigma^*\|_{op} \qquad (16)\\
&\ge \sigma_{\min}^*\sqrt{1-\text{SD}_2^2(\boldsymbol{U}^{(t)},\boldsymbol{U}^*)} - \sigma_1^*\|\boldsymbol{F}(\Sigma)^{-1}\|_{op}. \qquad (17)
\end{aligned}$$

where (14) follows from equation 19 in [2], and (16) follows from expanding $1 = \sigma_{\min}(\boldsymbol{U}^{(t)}) = \sigma_{\min}(\boldsymbol{U}^{*\dagger}\boldsymbol{U}^{(t)} + \boldsymbol{U}_\perp^{*\dagger}\boldsymbol{U}^{(t)}) \le \sigma_{\min}^2(\boldsymbol{U}^{*\dagger}\boldsymbol{U}^{(t)}) + \text{SD}_2^2(\boldsymbol{U}^{(t)},\boldsymbol{U}^*)$. Here, $\boldsymbol{U}^{*\dagger}$ denotes the projection matrix onto the space spanned by the columns of $\boldsymbol{U}^*$. For (16), express $\boldsymbol{U}^{(t)} = \boldsymbol{U}^{*\dagger}\boldsymbol{U}^{(t)} + \boldsymbol{U}_\perp^{*\dagger}\boldsymbol{U}^{(t)}$. Then, expand the quadratic form

$$\sigma_{\min}^2(\boldsymbol{U}^{(t)}) = \min_{\|z\|_2=1} \boldsymbol{z}(\cdot)^\intercal(\boldsymbol{U}^{*\dagger}\boldsymbol{U}^{(t)} + \boldsymbol{U}_\perp^{*\dagger}\boldsymbol{U}^{(t)})\boldsymbol{z} \le \tag{18}$$
$$\sigma_{\min}^2(\boldsymbol{U}^{*\dagger}\boldsymbol{U}^{(t)}) + \sigma_{\max}^2(\boldsymbol{U}_\perp^{*\dagger}\boldsymbol{U}^{(t)}) = 1 + \text{SD}_2^2(\boldsymbol{U}^*,\boldsymbol{U}^{(t)}).$$

Repeating the same arguments as above and observing for (15) that $\sigma_{\max}(\boldsymbol{U}^{*\intercal}\boldsymbol{U}^{(t)}) \le \sigma_{\max}(\boldsymbol{U}^{*\intercal})\sigma_{\max}(\boldsymbol{U}^{(t)}) = 1$, we bound $\sigma_{\max}$ as

$$\sigma_{\max}(\widehat{\boldsymbol{B}}^{(t+1)}) \le \sigma_{\max}^* + \sigma_{\max}^*\|\boldsymbol{F}(\Sigma)^{-1}\|_{op}. \tag{19}$$

Lemma 5.6 upper bounds the term $\sigma_1^*\|\boldsymbol{F}(\Sigma)^{-1}\|_{op}$

$$\|\boldsymbol{F}(\Sigma)^{-1}\|_{op} \le \frac{\delta_{2k}}{1-\delta_{2k}}\text{SD}_2(\boldsymbol{U}^{(t)},\boldsymbol{U}^*). \tag{20}$$

Substituting $\text{SD}_2(\boldsymbol{U}^{(t)},\boldsymbol{U}^*) \le \text{SD}_2(\boldsymbol{U}^{(0)},\boldsymbol{U}^*) \le \frac{1}{2}$ (Lemma 5.2 in [2]) and $\delta_{2k}/(1-\delta_{2k}) \le 1/(12r\kappa - 1) \le 1/(10r\kappa)$, where $\delta_{2k} \le 1/(12r\kappa)$ (page 17 of [2]) in (17) and (19),

$$\sigma_{\min}(\widehat{\boldsymbol{B}}^{(t+1)}) \ge \frac{\sqrt{3}}{2}\sigma_{\min}^* - \frac{1}{20r\kappa}\sigma_{\max}^* = \sigma_{\min}^*(\frac{\sqrt{3}}{2} - \frac{1}{20r}), \tag{21}$$
$$\sigma_{\max}(\widehat{\boldsymbol{B}}^{(t+1)}) \le \sigma_{\max}^*(1+\frac{1}{20r\kappa}) = \sigma_{\max}^* + \frac{\sigma_{\min}^*}{20r} \le \sigma_{\max}^*(1+\frac{1}{20r}). \tag{22}$$

From (21), (22), for large r,

$$\kappa(\widehat{\boldsymbol{B}}) = \frac{\sigma_{\max}(\widehat{\boldsymbol{B}}^{(t)})}{\sigma_{\min}(\widehat{\boldsymbol{B}}^{(t)})} = O(\kappa). \tag{23}$$