Accelerated Factored Gradient Descent for Low-Rank Matrix Factorization

Dongruo Zhou
Department of Computer Science
University of California,
Los Angeles

Yuan Cao
Department of Computer Science
University of California,
Los Angeles

Quanquan Gu
Department of Computer Science
University of California,
Los Angeles

Abstract

We study the low-rank matrix estimation problem, where the objective function $\mathcal{L}(\mathbf{M})$ is defined over the space of positive semidefinite matrices with rank less than or equal to r. A fast approach to solve this problem is matrix factorization, which reparameterizes $\mathbf{M}$ as the product of two smaller matrix such that $\mathbf{M} = \mathbf{U}\mathbf{U}^\top$ and then performs gradient descent on $\mathbf{U}$ directly, a.k.a., factored gradient descent. Since the resulting problem is nonconvex, whether Nesterov's acceleration scheme can be adapted to it remains a long-standing question. In this paper, we answer this question affirmatively by proposing a novel and practical accelerated factored gradient descent method motivated by Nesterov's accelerated gradient descent. The proposed method enjoys better iteration complexity and computational complexity than the state-of-the-art algorithms in a wide regime. The key idea of our algorithm is to restrict all its iterates onto a special convex set, which enables the acceleration. Experimental results demonstrate the faster convergence of our algorithm and corroborate our theory.

1 Introduction

We consider the following low-rank matrix estimation problem:

$$\min_{\mathbf{M}\in\mathbb{R}^{d\times d}} \mathcal{L}(\mathbf{M}), \text{ subject to } \mathbf{M} \succeq 0, \ \text{rank}(\mathbf{M}) \leq r, \tag{1.1}$$

where we denote by $\mathbf{M} \succeq 0$ if matrix $\mathbf{M}$ is positive semidefinite (PSD), and $\mathcal{L} : \mathbb{R}^{d\times d} \to \mathbb{R}$ is a L-smooth and μ-strongly convex function over positive semidefinite matrices with rank less than or equal to r for some $r > 0$. For the sake of simplicity, we consider a symmetric matrix and impose the positive semidefinite constraint on $\mathbf{M}$ [1]. Let $\mathbf{M}^*$ be the global minimizer to the problem (1.1). We are mostly interested in the case where the rank of the matrix is much smaller than the dimension ($r \ll d$) and the condition number (L/μ) of objective function $\mathcal{L}(\cdot)$ is large ($L/\mu \gg 1$). Such a low-rank matrix estimation problem in (1.1) is very generic, which includes matrix completion (Srebro et al., 2004; Candès and Tao, 2010) and matrix sensing (Recht et al., 2010) as special cases, and has wide applications such as collaborative filtering (Srebro et al., 2004) and multi-label learning (Cabral et al., 2011). Significant progress has been made to solve this problem using nuclear norm based convex relaxation (Srebro et al., 2004; Candès and Tao, 2010; Recht et al., 2010; Negahban and Wainwright, 2011, 2012; Gui et al., 2016), which involves a singular value decomposition (SVD) at each iteration and suffers from high computational complexity (i.e., $O(d^3)$) per iteration. Such a computational cost is prohibitively expensive when the dimension d is large, which hinders the application of these methods for large scale problems.

In order to overcome the aforementioned computational burden, a line of research (Bhojanapalli et al., 2015; Burer and Monteiro, 2003; Chen and Wainwright, 2015) uses Burer and Monteiro factorization/matrix factorization to reparameterize the matrix space and use gradient descent (GD) to solve the transformed problem. More specifically, by rewriting $\mathbf{M}$ as $\mathbf{U}\mathbf{U}^\top$ where $\mathbf{U}$ is a $d \times r$ matrix, we obtain the following optimization problem

$$\min_{\mathbf{U}\in\mathbb{R}^{d\times r}} \mathcal{G}(\mathbf{U}) := \mathcal{L}(\mathbf{U}\mathbf{U}^\top). \tag{1.2}$$

Note that the positive semidefinite constraint and the low-rank constraint on $\mathbf{M}$ are automatically satisfied

Proceedings of the 23rdInternational Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

[1] Our algorithm and theory can be easily extended to optimization over rectangle matrices with low rank constraint.

due to the reparameterization. The optimization problem in (1.2) can be solved by alternating minimizatoin (Jain et al., 2013; Hardt, 2014; Hardt and Price, 2014) or gradient descent (Zhao et al., 2015; Chen and Wainwright, 2015; Sun and Luo, 2015; Zheng and Lafferty, 2015, 2016; Tu et al., 2016; Bhojanapalli et al., 2015; Park et al., 2016; Wang et al., 2017b; Zhang et al., 2018) over $\mathbf{U}$ as follows

$$\mathbf{U}_{t+1} = \mathbf{U}_t - \eta \nabla \mathcal{G}(\mathbf{U}_t), \tag{1.3}$$

where $\eta > 0$ is the step size. The gradient descent update in (1.3) is also called factored gradient descent (FGD) (Bhojanapalli et al., 2015) or Procrustes flow (Tu et al., 2016). In contrast to (1.1), (1.2) is a nonconvex optimization problem although $\mathcal{L}(\cdot)$ is convex, which poses a big challenge to analyze the convergence of (1.3). Fortunately, a recent series of work (Zhao et al., 2015; Chen and Wainwright, 2015; Sun and Luo, 2015; Zheng and Lafferty, 2015, 2016; Tu et al., 2016; Bhojanapalli et al., 2015; Park et al., 2016; Wang et al., 2017a) has shown that with good initialization $\mathbf{U}_0$ that is close enough to the optimal solution $\mathbf{U}^*$ where $\mathbf{M}^* = \mathbf{U}^*(\mathbf{U}^*)^\top$, (1.3) is able to converge to the ϵ-optimal solution to (1.2) [2] with $O(\|\mathbf{U}^*\|_2^2/\sigma_r^2(\mathbf{U}^*)\widehat{L}/\mu \log(1/\epsilon))$ iteration complexity[3], where $\sigma_r(\cdot)$ denotes the r-th singular value, and $\widehat{L} = L + \|\nabla\mathcal{L}(\mathbf{U}^*(\mathbf{U}^*)^\top)\|_2/\|\mathbf{U}^*\|_2^2$. Such locally linear rate of convergence suggests that despite the nonconvex nature in (1.2), gradient descent methods still locally behaves like in the strongly convex setting.

It is well-known that for any L-smooth and μ-strongly convex function, Nesterov's accelerated gradient descent (AGD) Nesterov (1983, 1988, 2004) can achieve an accelerated rate and is the optimal first-order optimization algorithm. In specific, by introducing an "estimate sequence" technique, Nesterov (2004) showed that AGD can achieve an ϵ-optimal solution within $O(\sqrt{L/\mu}\log(1/\epsilon))$ iterations, which outperforms the iteration complexity of GD by a factor of $\sqrt{L/\mu}$. This result matches the lower bound of the iteration complexity for first-order methods (Nesterov, 2004), which suggests that AGD is optimal in the first-order optimization setting. Given the success of AGD for smooth and strongly convex functions, a natural question is whether we can adapt AGD to the nonconvex low-rank matrix factorization problem (1.2) and achieve faster convergence rate than that of FGD. Such a research question is highly challenging due to the following two main reasons:

- It is well known that strong convexity is required for AGD to achieve an accelerated linear rate of convergence (Nesterov, 1988). However, although $\mathcal{L}(\mathbf{M})$ is strongly convex in $\mathbf{M}$, strong convexity does not hold for the transformed problem $\mathcal{G}(\mathbf{U})$ with respect to $\mathbf{U}$.
- The linear convergence can be proved for GD in (1.3) since GD ensures *monotonic decreasing*[4] on $\mathcal{G}(\mathbf{U})$, which guarantees that all iterates generated by (1.3) locate in a small neighborhood of optimal solution $\mathbf{U}^*$. For AGD, such monotonic decreasing property does not even hold for strongly convex and smooth function, thus direct application of AGD to (1.2) cannot guarantee that all iterates stay inside a small neighborhood of optimal solution.

In this paper, we overcome the above challenges, and propose a new algorithm namely accelerated factored gradient descent (AFGD). The key idea of our algorithm is to restrict all its iterates on a special convex set to achieve accelerated convergence. We prove that AFGD enjoys a locally accelerated linear rate of convergence to optimal solution to (1.2). Our main contributions can be summarized as follows:

1. We propose a simple and practical algorithm AFGD, which can be seen as the adaptation of Nesterov's AGD (Nesterov, 2004) to low-rank matrix factorization problem (1.2). Our algorithm only requires basic gradient computations, which is computationally very efficient.
2. We analyze the local convergence of AFGD. Under the condition that $\mathcal{L}$ is L-restricted smooth and μ-restricted strongly convex (See Definitions 2.1 and 2.2), we prove that AFGD achieves an ϵ-optimal solution after $O(\|\mathbf{U}^*\|_2^3/\sigma_r^3(\mathbf{U}^*)\sqrt{\widehat{L}/\mu}\log(1/\epsilon))$ iterations, which outperforms that of FGD (Bhojanapalli et al., 2015) for a wide regime of the problem parameters.

1.1 Comparison with Existing Results

In literature, the most related work to ours is Li and Lin (2017), which also proposes an accelerated gradient descent with alternating constraint (AGD-AC) for solving (1.2). However, they assume that two index sets S_1, S_2 with no intersection are known such that the projection of the optimal solution to (1.2) onto S_1 and S_2 should be positive definite. However, as is noted by the authors, finding such index sets is a difficult problem in the theoretical computer science community, and there does not exist a polynomial-time algorithm to do

[2] We say a point $\mathbf{U}$ is an ϵ-optimal solution to $\mathcal{G}(\mathbf{U})$, if $\mathcal{G}(\mathbf{U}) - \min_{\mathbf{U}} \mathcal{G}(\mathbf{U}) \leq \epsilon$.

[3] We define iteration complexity as the total number of iterations an algorithm needs to execute to achieve an ϵ-optimal solution to $\min_{\mathbf{U}} \mathcal{G}(\mathbf{U})$.

[4] We say a sequence $\{\mathbf{U}_k\}$ satisfies monotonic decreasing if $\mathcal{G}(\mathbf{U}_k) \geq \mathcal{G}(\mathbf{U}_{k+1})$ for any k.

Table 1: Comparisons of different algorithm to find an ϵ-optimal solution to $\mathcal{G}$. Here $\mathbf{U}^*$ is the global minimum of $\mathcal{G}$, $\widehat{L} = L + \|\nabla\mathcal{L}(\mathbf{U}^*(\mathbf{U}^*)^\top)\|_2/\|\mathbf{U}^*\|_2^2$ where L is the restricted smoothness constant of $\mathcal{L}$, μ is the restricted convexity constant of $\mathcal{L}$. $\mathcal{T}_g$ is the computational complexity to calculate $\nabla\mathcal{G}(\mathbf{U})$, which is $O(dr^2)$ for matrix completion (Zheng and Lafferty, 2016) and matrix sensing (Zheng and Lafferty, 2015; Tu et al., 2016), and can be much worse for generic problems.

Algorithm	Iteration Complexity	Per-iteration Complexity
FGD (Bhojanapalli et al., 2015)	$O\Big(\Big[\frac{\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)}\Big]^2 \frac{\widehat{L}}{\mu}\log\frac{1}{\epsilon}\Big)$	$\mathcal{T}_g + O(dr)$
AGD-AC (Li and Lin, 2017)	$O\Big(\Big[\frac{\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)}\Big]^2 \sqrt{dr\frac{\widehat{L}}{\mu}}\log\frac{1}{\epsilon}\Big)$	$\mathcal{T}_g + O(dr + r^3)$
AFGD (This work)	$O\Big(\Big[\frac{\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)}\Big]^3 \sqrt{\frac{\widehat{L}}{\mu}}\log\frac{1}{\epsilon}\Big)$	$\mathcal{T}_g + \widetilde{O}\Big(dr^2 + \frac{\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)}r^3\Big)$

this with provable guarantee. Therefore, AGD-AC is not practical, or even intractable. Our algorithm, in contrast, is practical and computationally efficient.

We summarize the iteration complexity and per-iteration complexity of FGD (Bhojanapalli et al., 2015), AGD-AC (Li and Lin, 2017) and our algorithm AFGD in Table 1 for better comparison. We can see that in terms of the iteration complexity, AFGD is better than FGD when $\widehat{L}/\mu > \|\mathbf{U}^*\|_2^2/\sigma_r^2(\mathbf{U}^*)$, and is better than AGD-AC when $dr > \|\mathbf{U}^*\|_2^2/\sigma_r^2(\mathbf{U}^*)$. Both conditions are easily satisfied when the condition number $\widehat{L}/\mu$ and dimension d are large enough. Regarding the per-iteration complexity, since $\mathcal{T}_g$ is at least $O(dr^2)$ and is always the dominating term in the complexity, the per-iteration complexity of all these algorithms are comparable. The total computational complexity of our algorithm can be much better than the that of FGD and AGD-AC.

1.2 Additional Related Work

Since the seminal work of Nesterov's acceleration gradient descent (AGD) (Nesterov, 1983, 1988, 2004), several lines of work have been developed to study AGD for various nonconvex problems. For general nonconvex optimzation, Ghadimi and Lan (2016) proved that AGD can converge to a stationary point with the same rate as GD in the nonconvex setting. Paquette et al. (2018) extended the acceleration technique called Catylast (Lin et al., 2015) from convex setting to nonconvex setting, and proved that accelerated algorithms based on Catylast converge to a stationary point with the same rate as GD. So for general nonconvex optimization, accelerated rate cannot be proved for AGD or its variants. For finding the second-order stationary point (i.e., local minimum), Carmon et al. (2018); Jin et al. (2018) showed that variants of AGD can escape saddle points and find a second-order stationary point faster than their counterparts based on GD. For finding the global minimum of functions that are nonconvex in the Euclidean space, but geodesically strongly convex on Riemannian manifolds, Liu et al. (2017); Zhang and Sra (2018) proposed variants of AGD, which can achieve accelerated linear rate of convergence. Moreover, Agarwal et al. (2017) proposed a second-order method based on cubic regularization for non-convex optimization problems. Ge et al. (2015) proved that stochastic gradient descent can escape from saddle points and converge to local minima.

2 Notation and Preliminaries

We use $[d]$ to denote the set $\{1, 2, \ldots, d\}$. For any d-dimensional vector $\mathbf{x} = [x_1, ..., x_d]^\top$, its ℓ_2 norm is defined as $\|\mathbf{x}\|_2 = (\sum_{i=1}^d |x_i|^2)^{1/2}$. For any matrix $\mathbf{A} = [A_{ij}]$, denote the spectral norm and Frobenius norm of $\mathbf{A}$ by $\|\mathbf{A}\|_2$ and $\|\mathbf{A}\|_F$, respectively. Let $\sigma_r(\mathbf{A})$ be the r-th largest singular value of $\mathbf{A}$ for $r \in [d]$. We denote by $\mathbf{A} \succeq 0$ if $\mathbf{A}$ is positive semidefinite (PSD). Given any two sequences $\{a_n\}$ and $\{b_n\}$, we write $a_n = O(b_n)$ if there exists a constant $0 < C < +\infty$ such that $a_n \leq C\, b_n$, and we use $\widetilde{O}(\cdot)$ to hide the logarithmic factors.

We begin with a few definitions that are used for later theoretical analysis of our algorithms. The first two definitions are regarding restricted strongly convex and smooth functions (Negahban et al., 2009; Negahban and Wainwright, 2011, 2012; Jain et al., 2017).

Definition 2.1. We say $\mathcal{L}$ is μ-restricted strongly convex if for any $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d\times d}$ such that $\mathbf{X}, \mathbf{Y} \succeq 0$ and $\mathrm{rank}(\mathbf{X}) \leq r$, $\mathrm{rank}(\mathbf{Y}) \leq r$, we have

$$\mathcal{L}(\mathbf{X}) \geq \mathcal{L}(\mathbf{Y}) + \langle\nabla\mathcal{L}(\mathbf{Y}), \mathbf{X} - \mathbf{Y}\rangle + \frac{\mu}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2.$$

Definition 2.2. We say $\mathcal{L}$ is L-restricted smooth if for any $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{d\times d}$ such that $\mathbf{X}, \mathbf{Y} \succeq 0$ and $\mathrm{rank}(\mathbf{X}) \leq r$,

$\text{rank}(\mathbf{Y}) \leq r$, we have

$$\mathcal{L}(\mathbf{X}) \leq \mathcal{L}(\mathbf{Y}) + \langle \nabla \mathcal{L}(\mathbf{Y}), \mathbf{X} - \mathbf{Y} \rangle + \frac{L}{2} \|\mathbf{X} - \mathbf{Y}\|_F^2.$$

Note that in our paper, we only require $\mathcal{L}(\mathbf{M})$ to be restricted strongly convex and smooth. The induced function $\mathcal{G}(\mathbf{U})$ from $\mathcal{L}(\mathbf{M})$ by matrix factorization $\mathbf{M} = \mathbf{U}\mathbf{U}^\top$ is not restricted strongly convex or smooth.

Following Tu et al. (2016), we define the Procrustes distance to measure the distance between two matrices up to rotation, and function $\Pi_{\mathbf{V}}(\mathbf{U})$ representing the matrix which is closest to $\mathbf{V}$ in the equivalence class of $\mathbf{U}$ up to rotation.

Definition 2.3 (Procrustes distance)**.** For any $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d\times r}$, define

$$D(\mathbf{U}, \mathbf{V}) := \|\mathbf{U}\mathbf{R} - \mathbf{V}\|_F, \ \Pi_{\mathbf{V}}(\mathbf{U}) := \mathbf{U}\mathbf{R},$$

where $\mathbf{R} = \text{argmin}_{\mathbf{P}\in\mathbb{R}^{r\times r}, \mathbf{P}^\top\mathbf{P}=\mathbf{I}} \|\mathbf{U}\mathbf{P} - \mathbf{V}\|_F$.

In Definition 2.3, the optimal rotation matrix $\mathbf{R} = \mathbf{P}\mathbf{Q}^\top$, where $[\mathbf{P}, \mathbf{D}, \mathbf{Q}]$ is the singular value decomposition (SVD) of $\mathbf{U}^\top\mathbf{V}$ (Tu et al., 2016). It is easy to check that $D(\cdot,\cdot)$ is rotation invariant, i.e., $D(\mathbf{U}\mathbf{P}, \mathbf{V}) = D(\mathbf{U}, \mathbf{V}\mathbf{P}) = D(\mathbf{U}, \mathbf{V})$ for any $\mathbf{P} \in \mathbb{R}^{r\times r}$ with $\mathbf{P}^\top\mathbf{P} = \mathbf{I}$. $D(\cdot,\cdot)$ also satisfies triangle inequality such that for any $\mathbf{U}, \mathbf{V}, \mathbf{W} \in \mathbb{R}^{d\times r}$, we have $D(\mathbf{U}, \mathbf{V}) \leq D(\mathbf{U}, \mathbf{W}) + D(\mathbf{W}, \mathbf{V})$. The intuition behind Definition 2.3 is that, by the definition of $\mathcal{G}$ in (1.2), clearly $\mathcal{G}(\mathbf{U}) = \mathcal{G}(\mathbf{U}\mathbf{P})$ for any orthornormal matrix $\mathbf{P}$. Therefore for two matrices $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d\times r}$, it is natural to consider their corresponding "equivalent classes" $\{\mathbf{U}\mathbf{P} : \mathbf{P}^\top\mathbf{P} = \mathbf{I}\}$, $\{\mathbf{V}\mathbf{P} : \mathbf{P}^\top\mathbf{P} = \mathbf{I}\}$, and $D(\mathbf{U}, \mathbf{V})$ essentially is the minimum Frobenius distance between matrices in these two equivalent classes. Moreover, $\Pi_{\mathbf{V}}(\mathbf{U})$ defines a mapping that maps all matrices in $\{\mathbf{U}\mathbf{P} : \mathbf{P}^\top\mathbf{P} = \mathbf{I}\}$ to a specific one according to some reference matrix $\mathbf{V}$. In our algorithm, we utilize $\Pi_{\mathbf{U}_0}(\cdot)$, where $\mathbf{U}_0$ is the initialization, to make sure that all iterates are well-aligned representatives in their equivalent classes.

In terms of computational complexity, we note that computing the matrix $\mathbf{R}$ in Definition 2.3 consists of the calculation of $\mathbf{U}^\top\mathbf{V}$ and SVD of $\mathbf{U}^\top\mathbf{V}$, which are of $O(dr^2)$ and $O(r^3)$ complexities respectively. Thus the computational complexity to compute $\Pi_{\mathbf{V}}(\mathbf{U})$ is $O(dr^2)$.

3 The Proposed Algorithm

Before we present our algorithm, we first review AGD update scheme proposed in Nesterov (2004). For an objective function $\mathcal{G}$, starting from initial points $\mathbf{X}_0 = \mathbf{V}_0$, AGD maintains three sequences of iterates $\{\mathbf{X}_k\}$, $\{\mathbf{Y}_k\}$ and $\{\mathbf{V}_k\}$ with the following update rule:

$$\begin{aligned} \mathbf{Y}_k &= \frac{\alpha}{\alpha+1}\mathbf{V}_k + \frac{1}{\alpha+1}\mathbf{X}_k, \\ \mathbf{V}_{k+1} &= (1-\alpha)\mathbf{V}_k + \alpha\mathbf{Y}_k - \frac{\alpha}{\gamma}\nabla\mathcal{G}(\mathbf{Y}_k), \\ \mathbf{X}_{k+1} &= \mathbf{Y}_k - \eta\nabla\mathcal{G}(\mathbf{Y}_k), \end{aligned} \tag{3.1}$$

where $\eta > 0$ is the step size and $\alpha > 0, \gamma > 0$ are momentum parameters satisfying $\alpha^2 = \eta\gamma$. It has been shown in Nesterov (2004) that if $\mathcal{G}$ is $\widetilde{\mu}$-strongly convex and $\widetilde{L}$-smooth, with the choice $\gamma = \widetilde{\mu}$ and $\eta = 1/\widetilde{L}$, the AGD update rule in (3.1) achieves the accelerated linear rate of convergence, i.e., $\mathcal{G}(\mathbf{X}_k) - \mathcal{G}(\mathbf{X}^*) \leq O((1 - \sqrt{\widetilde{\mu}/\widetilde{L}})^k)$, where $\mathbf{X}^*$ is the minimizer of $\mathcal{G}(\cdot)$. However, the strong convexity does not hold for our problem (1.2), and such update rule cannot ensure acceleration.

To overcome this technical barrier, we consider the following set defined in terms of the initial point $\mathbf{U}_0$. For any $\mathbf{U}_0 \in \mathbb{R}^{d\times r}$, we define $\Omega(\mathbf{U}_0)$ as

$$\Omega(\mathbf{U}_0) := \{\mathbf{U} \in \mathbb{R}^{d\times r} : \mathbf{U}^\top\mathbf{U}_0 \succeq 0\}. \tag{3.2}$$

In other word, $\Omega(\mathbf{U}_0)$ is the set of all such matrices $\mathbf{U}$ that $\mathbf{U}^\top\mathbf{U}_0$ is positive semidefinite. $\Omega(\mathbf{U}_0)$ is highly related to $D(\cdot,\cdot)$ and $\Pi_{\mathbf{U}_0}(\cdot)$ in Definition 2.3. In fact, it holds that for any $\mathbf{V} \in \mathbb{R}^{d\times r}$, $\Pi_{\mathbf{U}_0}(\mathbf{V}) \in \Omega(\mathbf{U}_0)$ (See Lemma B.3 in the appendix). Our key observation is that restricted strong convexity holds for $\mathcal{G}(\mathbf{U})$ on the set $\Omega(\mathbf{U}_0)$, as is shown in the following proposition:

Proposition 3.1 (Informal)**.** Let $\mathbf{U}^*$ be the optimal solution to (1.2). Then there exist constants C_{convex} and $\bar{\mu} > 0$ that only depend on $\mathcal{G}$ and $\mathbf{U}^*$, such that for any $\mathbf{U}_0$ which satisfies that $D(\mathbf{U}_0, \mathbf{U}^*) \leq C_{\text{convex}}$, and any $\mathbf{U}, \mathbf{V} \in \Omega(\mathbf{U}_0)$ satisfying $D(\mathbf{U}, \mathbf{U}^*), D(\mathbf{V}, \mathbf{U}^*) \leq C_{\text{convex}}$, it holds that

$$\mathcal{G}(\mathbf{U}) \geq \mathcal{G}(\mathbf{V}) + \langle \nabla\mathcal{G}(\mathbf{V}), \mathbf{U} - \mathbf{V} \rangle + \frac{\bar{\mu}}{2}\|\mathbf{U} - \mathbf{V}\|_F^2.$$

Proposition 3.1 suggests that if all iterates $\{\mathbf{X}_k\}$, $\{\mathbf{V}_k\}$, $\{\mathbf{Y}_k\}$ generated by (3.1) belong to the set $\Omega(\mathbf{U}_0)$, the restricted strong convexity holds for $\mathcal{G}(\mathbf{U})$ along the trajectory of the iterates, which is pivotal to prove the accelerated convergence rate. Nevertheless, such property does not hold naturally from the AGD update rules in (3.1), since the gradient descent update in (3.1) can make the new iterate run out of $\Omega(\mathbf{U}_0)$. We address this issue by proposing a new algorithm namely accelerated factored gradient descent (AFGD), as displayed in Algorithm 1.

Like AGD in (3.1), AFGD maintains three sequences of iterates $\{\mathbf{X}_k\}, \{\mathbf{Y}_k\}, \{\mathbf{V}_k\}$. The key feature of AFGD is to keep all three sequences in the set $\Omega(\mathbf{U}_0)$ defined in (3.2). To achieve this, the update scheme of AFGD

Algorithm 1 Accelerated Factored Gradient Descent (AFGD)

Require: Function $\mathcal{G}$, step size η, moment parameter γ, accuracy ϵ_S, initial point $\mathbf{U}_0$, iteration number K, subsolver iteration number T.

1: **Initialize:** $\mathbf{X}_0 = \mathbf{V}_0 = \mathbf{U}_0$
2: Compute $[\mathbf{A}_0, \mathbf{D}_0, \mathbf{B}_0]$ as the r-SVD of $\mathbf{U}_0$.
3: Compute α satisfying that $\alpha^2 = \eta\gamma$.
4: **for** $k = 0, 1, \ldots, K-1$ **do**
5: Compute $\mathbf{Y}_k, \mathbf{V}_{k+1}$ and $\mathbf{X}_{k+1}$ as follows:

$$\mathbf{Y}_k = \frac{\alpha}{\alpha+1}\mathbf{V}_k + \frac{1}{\alpha+1}\mathbf{X}_k, \tag{3.3}$$

$$\mathbf{V}'_{k+1} = (1-\alpha)\mathbf{V}_k + \alpha\mathbf{Y}_k - \frac{\alpha}{\gamma}\nabla\mathcal{G}(\mathbf{Y}_k) \tag{3.4}$$

$$\mathbf{V}_{k+1} = \text{ACCPROJ}(\mathbf{V}'_{k+1}, \mathbf{A}_0, \mathbf{D}_0, \mathbf{B}_0, T) \tag{3.5}$$

$$\mathbf{X}_{k+1} = \Pi_{\mathbf{U}_0}\big(\mathbf{Y}_k - \eta\nabla\mathcal{G}(\mathbf{Y}_k)\big) \tag{3.6}$$

6: **end for**

Ensure: $\mathbf{X}_K$

is very similar to (3.1) except for the update of $\mathbf{V}_{k+1}$ and $\mathbf{X}_{k+1}$. More specifically, AFGD starts from initial point $\mathbf{U}_0$. At the k-th iteration, AFGD first updates $\mathbf{Y}_k$ according to (3.3) in Algorithm 1, which is the same as AGD in (3.1), and can be regarded as a convex combination of $\mathbf{V}_k$ and $\mathbf{X}_k$. It can be easily verified that if both $\mathbf{V}_k$ and $\mathbf{X}_k$ belong to $\Omega(\mathbf{U}_0)$, then $\mathbf{Y}_k$ also belongs to $\Omega(\mathbf{U}_0)$ due to its convexity. Next, AFGD computes $\mathbf{V}'_{k+1}$ according to (3.4), which is the same update rule of $\mathbf{V}_{k+1}$ for AGD in (3.1). Unlike $\mathbf{Y}_k$, $\mathbf{V}'_{k+1}$ can not be guaranteed to belong to $\Omega(\mathbf{U}_0)$ although $\mathbf{V}_k, \mathbf{Y}_k \in \Omega(\mathbf{U}_0)$. In order to address this problem, we aim to set $\mathbf{V}_{k+1}$ to be the projection of $\mathbf{V}'_{k+1}$ onto set $\Omega(\mathbf{U}_0)$, which is defined as follows

$$\mathbf{V}_{k+1} = \operatorname*{argmin}_{\mathbf{V}\in\Omega(\mathbf{U}_0)} \|\mathbf{V} - \mathbf{V}'_{k+1}\|_F. \tag{3.7}$$

However, (3.7) itself is a non-trivial optimization problem, which does not has a closed-form solution. Therefore we use accelerated projection (ACCPROJ) in Algorithm 2 as a subproblem solver to approximately calculate $\mathbf{V}_{k+1}$. We defer the details of ACCPROJ to the next paragraph. Finally, AFGD performs one-step gradient descent from $\mathbf{Y}_k$, followed by setting $\mathbf{X}_{k+1}$ to be the rotation of this update that is closest to $\mathbf{U}_0$, as shown in (3.6). This guarantees that $\mathbf{X}_{k+1} \in \Omega(\mathbf{U}_0)$, as is discussed in Section 2.

We now discuss the subproblem solver ACCPROJ as displayed in Algorithm 2. Generally speaking, ACCPROJ aims to solve the projection problem (3.7) up to certain precision. ACCPROJ solves this problem by making the following two key observations.

Algorithm 2 ACCPROJ($\mathbf{V}'_{k+1}, \mathbf{A}_0, \mathbf{D}_0, \mathbf{B}_0, T$)

Require: $\mathbf{V}'_{k+1}, \mathbf{A}_0, \mathbf{D}_0, \mathbf{B}_0$, iteration number T.

1: Let $\mathbf{T} = \mathbf{A}_0^\top\mathbf{V}'_{k+1}\mathbf{B}_0$, $\mathbf{\Sigma}_0^{(1)} = \mathbf{\Sigma}_0^{(2)} = \mathbf{0}$, $\eta_S = \sigma_r^2(\mathbf{D}_0)$, $\beta_S = (\|\mathbf{D}_0\|_2 - \sigma_r(\mathbf{D}_0))/(\|\mathbf{D}_0\|_2 + \sigma_r(\mathbf{D}_0))$.
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Compute $\mathbf{\Sigma}_{t+1}^{(1)}, \mathbf{\Sigma}_{t+1}^{(2)}$ as follows:

$$\mathbf{\Sigma}'_{t+1} = \mathbf{\Sigma}_t^{(2)} - \eta_S\mathbf{D}_0^{-1}[\mathbf{D}_0^{-1}\mathbf{\Sigma}_t^{(2)} - \mathbf{T}], \tag{3.8}$$

Compute $[\mathbf{D}, \mathbf{A}]$ as the eigendecomposition of $(\mathbf{\Sigma}'_{t+1} + \mathbf{\Sigma}'^\top_{t+1})/2$,

$$\mathbf{\Sigma}_{t+1}^{(1)} = \mathbf{A}\max\{\mathbf{D}, 0\}\mathbf{A}^\top, \tag{3.9}$$

$$\mathbf{\Sigma}_{t+1}^{(2)} = \mathbf{\Sigma}_{t+1}^{(1)} + \beta_S(\mathbf{\Sigma}_{t+1}^{(1)} - \mathbf{\Sigma}_t^{(1)}). \tag{3.10}$$

4: **end for**

Ensure: $\mathbf{V}_{k+1} = (\mathbf{I} - \mathbf{A}_0\mathbf{A}_0^\top)\mathbf{V}'_{k+1} + \mathbf{A}_0\mathbf{D}_0^{-1}\mathbf{\Sigma}_T^{(1)}\mathbf{B}_0^\top$

First, suppose $\bar{\mathbf{V}} = \operatorname{argmin}_{\mathbf{V}\in\Omega(\mathbf{U}_0)} \|\mathbf{V} - \mathbf{V}'_{k+1}\|_F$, it can be shown that (see Appendix A.4) $\bar{\mathbf{V}}$ has the analytic formula,

$$\bar{\mathbf{V}} = (\mathbf{I} - \mathbf{A}_0\mathbf{A}_0^\top)\mathbf{V}'_{k+1} + \mathbf{A}_0\mathbf{D}_0^{-1}\mathbf{\Sigma}^*\mathbf{B}_0^\top, \tag{3.11}$$

$$\mathbf{\Sigma}^* = \operatorname*{argmin}_{\mathbf{\Sigma}\in\mathbb{R}^{r\times r}, \mathbf{\Sigma}\succeq 0} 1/2\big\|\mathbf{D}_0^{-1}\mathbf{\Sigma} - \mathbf{T}\big\|_F^2, \tag{3.12}$$

where $[\mathbf{A}_0, \mathbf{D}_0, \mathbf{B}_0]$ is the r-SVD of $\mathbf{U}_0$[5], $\mathbf{A}_0 \in \mathbb{R}^{d\times r}, \mathbf{D}_0 \in \mathbb{R}^{r\times r}, \mathbf{B}_0 \in \mathbb{R}^{r\times r}$ and $\mathbf{T} = (\mathbf{A}_0)^\top\mathbf{V}'_{k+1}\mathbf{B}_0$. Thus, to find $\bar{\mathbf{V}}$, it is equivalent to find $\mathbf{\Sigma}^*$. Note that the dimension of $\mathbf{\Sigma}$ is only $r \times r$, which suggests that it is more efficient to solve (3.12) than solving (3.7).

Second, since the objective function in (3.12) is strongly convex in $\mathbf{\Sigma}$, we can use standard proximal AGD (Nesterov, 2004) to find an ϵ-optimal solution within T steps. In detail, ACCPROJ maintains two sequences of iterates $\{\mathbf{\Sigma}_t^{(1)}\}$ and $\{\mathbf{\Sigma}_t^{(2)}\}$, both of which start from $\mathbf{0}$. At each iteration t, ACCPROJ performs one gradient descent step from $\mathbf{\Sigma}_t^{(2)}$ and denote the update as $\mathbf{\Sigma}'_{t+1}$, as shown in (3.8). ACCPROJ then updates $\mathbf{\Sigma}_{t+1}^{(1)}$ as the projection of $\mathbf{\Sigma}'_{t+1}$ onto the set of positive semidefinite matrices. The projection can be computed in a closed-form according to (3.9). Finally, ACCPROJ updates $\mathbf{\Sigma}_{t+1}^{(2)}$ as the linear combination of $\mathbf{\Sigma}_{t+1}^{(1)}$ and $\mathbf{\Sigma}_t^{(1)}$ as shown in (3.10).

4 Main Theory

In this section, we present the main theoretical results.

4.1 Iteration complexity of AFGD

In order to calculate the iteration complexity of Algorithm 1, we begin with the following inexact condition

[5] $\mathbf{D}_0$ consists of top-r singular values of $\mathbf{U}_0$ and $\mathbf{A}_0, \mathbf{B}_0$ are corresponding unitary matrices.

for subproblem solver ACCPROJ in Algorithm 2.

Definition 4.1. Suppose $\bar{\mathbf{V}} = \mathrm{argmin}_{\mathbf{V}\in\Omega(\mathbf{U}_0)} \|\mathbf{V} - \mathbf{V}'_{k+1}\|_F$, we say the subproblem solver ACCPROJ outputs an ϵ_S-approximate projection of $\mathbf{V}'_{k+1}$ onto $\Omega(\mathbf{U}_0)$, if the output of ACCPROJ $\mathbf{V}_{k+1}$ satisfies $\mathbf{V}_{k+1} \in \Omega(\mathbf{U}_0)$ and $\|\mathbf{V}_{k+1} - \bar{\mathbf{V}}\|_F \leq \epsilon_S$ for any k.

With Definition 4.1, we are ready to show our main theorem.

Theorem 4.2. Suppose that $\mathcal{L}$ is μ-restricted strongly convex and L-restricted smooth. let $\mathbf{U}^* \in \mathrm{argmin}_{\mathbf{U}\in\mathbb{R}^{d\times r}} \mathcal{G}(\mathbf{U})$ be an optimal solution and $c > 0$ be a sufficiently small constant. Set the parameters in Algorithm 1 as $\gamma = c\mu\sigma_r^6(\mathbf{U}_0)/\|\mathbf{U}_0\|_2^4$, $\eta = c/(L\|\mathbf{U}_0\|_2^2 + \|\nabla\mathcal{L}(\mathbf{U}_0\mathbf{U}_0^\top)\|_2)$. Suppose $D(\mathbf{U}_0, \mathbf{U}^*) \leq C_{\mathrm{ini}} = c^2\sigma_r^{10}(\mathbf{U}^*)\mu^{3/2}/(\|\mathbf{U}^*\|_2^9\widehat{L}^{3/2})$, then for any $\epsilon \leq \widehat{L}\|\mathbf{U}^*\|_2^2 C_{\mathrm{ini}}^2$ and any k, if ACCPROJ outputs an ϵ_S-approximate projection with $\epsilon_S = \min\{\alpha\epsilon/[4c^{1/2}\gamma\sigma_r(\mathbf{U}^*)], c^{1/2}\sigma_r(\mathbf{U}^*)/2\}$, the output of AFGD in Algorithm 1 satisfies

$$\mathcal{G}(\mathbf{X}_k) - \mathcal{G}(\mathbf{U}^*) \leq (1-\sqrt{\eta\gamma})^k\Big[\mathcal{G}(\mathbf{U}_0) - \mathcal{G}(\mathbf{U}^*) + \frac{\gamma}{2}D^2(\mathbf{U}_0, \mathbf{U}^*)\Big] + \frac{\epsilon}{2}.$$

Theorem 4.2 suggests the accelerated linear rate of convergence for AFGD up to precision ϵ. The following corollary further spells out the iteration complexity of AFGD to get an ϵ-optimal solution to (1.2).

Corollary 4.3. Under the same conditions as in Theorem 4.2, AFGD in Algorithm 1 outputs a ϵ-optimal solution $\mathbf{X}_K$ to (1.2) after

$$K = O\bigg(\Big[\frac{\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)}\Big]^3 \cdot \sqrt{\frac{\widehat{L}}{\mu}} \cdot \log\frac{\Delta_{\mathcal{G}}}{\epsilon}\bigg)$$

iterations, where $\Delta_{\mathcal{G}} = \mathcal{G}(\mathbf{U}_0) - \mathcal{G}(\mathbf{U}^*) + \gamma D^2(\mathbf{U}_0, \mathbf{U}^*)/2$.

Remark 4.4. In AFGD, the momentum parameter γ is set to $\gamma = \alpha^2/\eta$. This is essentially the same as that in AGD (3.1), where the momentum parameters γ and α are set to $\gamma = \widetilde{\mu}$, $\alpha = \sqrt{\widetilde{\mu}/\widetilde{L}}$, and the step size η is set to $\eta = 1/\widetilde{L}$. Here $\widetilde{\mu}$ and $\widetilde{L}$ are the strongly convex and smooth parameters of the objective function (Nesterov, 2004).

Remark 4.5. Note that $\Delta_{\mathcal{G}}$ only depends on $\mathbf{U}_0$ and $\mathbf{U}^*$, and can be seen as a constant. We can compare the iteration complexity of AFGD with other baseline algorithms in Table 1. AFGD outperforms FGD Bhojanapalli et al. (2015) when $\widehat{L}/\mu > [\|\mathbf{U}^*\|_2/\sigma_r(\mathbf{U}^*)]^2$, which is satisfied when the condition number of original problem (1.1) L/μ is large. AFGD also outperforms AGD-AC Li and Lin (2017) in the case $dr > \|\mathbf{U}^*\|_2/\sigma_r(\mathbf{U}^*)$, which always holds when d is large. Also note that AGD-AC is not a practical algorithm, because it relies on an assumption that it can sample two subsets with certain properties, which is hard to be satisfied.

Theorem 4.2 requires the initial point $\mathbf{U}_0$ to be within a small neighborhood of the optimal solution $\mathbf{U}^*$ with radius C_{ini}. To find such initial point, we can use the initialization method proposed in Bhojanapalli et al. (2015); Wang et al. (2017a). For special low-rank matrix factorization problems such as matrix completion and matrix sensing, even simpler initialization methods (Jain et al., 2013; Tu et al., 2016; Zheng and Lafferty, 2016; Zhao et al., 2015) exist and can be adopted to generate the desired initial point very efficiently.

4.2 Per-iteration complexity of AFGD

We now calculate the per-iteration complexity of Algorithm 1. At each iteration of Algorithm 1, we need to compute gradient $\nabla\mathcal{G}(\mathbf{Y}_k)$ in (3.4) and (3.6), whose computational complexity is denoted by $\mathcal{T}_g$. We also need to compute $\Pi_{\mathbf{U}_0}(\cdot)$ at (3.6), whose computational complexity is $O(dr^2)$. The remaining thing is to characterize the computational complexity of (3.5), which boils down to the total computational complexity of ACCPROJ in Algorithm 2.

The following theorem characterizes the iteration complexity of Algorithm 2 to output an ϵ_S-approximate projection, where ϵ_S is defined in Theorem 4.2.

Theorem 4.6. Under the same conditions as in Theorem 4.2, ACCPROJ in Algorithm 2 outputs an ϵ_S-approximate projection after

$$T = O\bigg(\frac{\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)} \cdot \log\frac{\|\mathbf{U}^*\|_F\sigma_r^4(\mathbf{U}^*)\sqrt{\mu\widehat{L}}}{\epsilon\|\mathbf{U}^*\|_2}\bigg)$$

iterations.

We now compute the per-iteration complexity for ACCPROJ. ACCPROJ needs to compute $\mathbf{T}$ at Line 1 of Algorithm 2, whose computational complexity is $O(dr^2)$. At each iteration, ACCPROJ needs to compute (3.8), whose computational complexity is $O(r^3)$. ACCPROJ also needs to do SVD of a $r \times r$ matrix in (3.9), whose computational complexity is $O(r^3)$. Finally, at the end of Algorithm 2, ACCPROJ needs to compute $\mathbf{V}_{k+1}$ once, whose computational complexity is $O(dr^2)$. Thus, the total computational complexity of ACCPROJ is

$$O(dr^2 + Tr^3) = O\bigg(dr^2 + \frac{r^3\|\mathbf{U}^*\|_2}{\sigma_r(\mathbf{U}^*)} \cdot \log\frac{\|\mathbf{U}^*\|_F\sigma_r^4(\mathbf{U}^*)\sqrt{\mu\widehat{L}}}{\epsilon\|\mathbf{U}^*\|_2}\bigg).$$

Thus by putting together the above computational complexity results, we conclude that the per-iteration

computational complexity of Algorithm 1 is $\widetilde{O}(dr^2 + r^3\|\mathbf{U}^*\|_2/\sigma_r(\mathbf{U}^*))$.

Remark 4.7. We compare the per-iteration complexity of AFGD with that of FGD (Bhojanapalli et al., 2015) and AGD-AC (Li and Lin, 2017) in Table 1. Note that the computational complexity of calculating the gradient of (1.2) (denoted by $\mathcal{T}_g$) is at least $O(dr^2)$ (Zheng and Lafferty, 2015, 2016), thus the dominant term in the per-iteration complexity of all three methods in Table 1 are $\mathcal{T}_g$. Therefore, the per-iteration complexities of all three methods are in the same order.

4.3 Extension to Asymmetric Lower-Rank Matrix factorization

The low-rank matrix estimation problem (1.1) focuses on the condition that $\mathbf{M}$ is symmetric and positive semidefinite. However, our method can be easily applied to the following asymmetric matrix estimation problem:

$$\min_{\mathbf{M}\in\mathbb{R}^{d_1\times d_2}} \mathcal{L}(\mathbf{M}), \text{ subject to } \mathrm{rank}(\mathbf{M}) \le r, \tag{4.1}$$

where $\mathcal{L}(\cdot)$ is L-smooth and μ-strongly convex over the set of matrices with rank less than or equal to $r > 0$. Following Park et al. (2016); Wang et al. (2017a); Li and Lin (2017), we factorize $\mathbf{M} = \mathbf{U}_1\mathbf{U}_2^\top$ where $\mathbf{U}_1 \in \mathbb{R}^{d_1\times r}$, $\mathbf{U}_2 \in \mathbb{R}^{d_2\times r}$ and consider the asymmetric counterpart of (1.2):

$$\min_{\mathbf{U}_1\in\mathbb{R}^{d_1\times r},\mathbf{U}_2\in\mathbb{R}^{d_2\times r}} \mathcal{L}(\mathbf{U}_1\mathbf{U}_2^\top) + \frac{\mu}{8}\|\mathbf{U}_1^\top\mathbf{U}_1 - \mathbf{U}_2^\top\mathbf{U}_2\|_F^2, \tag{4.2}$$

where the regularizer $(\mu/8)\cdot\|\mathbf{U}_1^\top\mathbf{U}_1 - \mathbf{U}_2^\top\mathbf{U}_2\|_F^2$ forces the optimal solution to be balanced since otherwise $(\xi\mathbf{U}_1, \xi^{-1}\mathbf{U}_2,\ \xi\neq 0$ gives exactly the same objective function value as $(\mathbf{U}_1, \mathbf{U}_2)$. Meanwhile, it has been proved in Zhu et al. (2018) that the global optimal solution obtained by solving (4.2) is the same as the global optimal solution to (4.1). To apply AFGD in Algorithm 1 onto (4.2), we first define $\mathbf{Z}$ and $\widetilde{\mathbf{M}}$ as follows:

$$\mathbf{Z} = \begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{pmatrix}, \widetilde{\mathbf{M}} = \mathbf{Z}\mathbf{Z}^\top = \begin{pmatrix}\mathbf{U}_1\mathbf{U}_1^\top, \mathbf{U}_1\mathbf{U}_2^\top\\ \mathbf{U}_2\mathbf{U}_1^\top, \mathbf{U}_2\mathbf{U}_2^\top\end{pmatrix}.$$

Note that $\widetilde{\mathbf{M}}$ can be represented as follows:

$$\widetilde{\mathbf{M}} = \begin{pmatrix}\widetilde{\mathbf{M}}_{1,1}, \widetilde{\mathbf{M}}_{1,2}\\ \widetilde{\mathbf{M}}_{2,1}, \widetilde{\mathbf{M}}_{2,2}\end{pmatrix},$$

where $\widetilde{\mathbf{M}}_{i,j} = \mathbf{U}_i\mathbf{U}_j^\top$. We now define $\widetilde{\mathcal{L}}(\cdot)$ as follows

$$\widetilde{\mathcal{L}}(\widetilde{\mathbf{M}}) = \mathcal{L}(\widetilde{\mathbf{M}}_{1,2}) + \frac{\mu}{8}\Big[\|\widetilde{\mathbf{M}}_{1,1}\|_F^2 + \|\widetilde{\mathbf{M}}_{2,2}\|_F^2 - 2\|\widetilde{\mathbf{M}}_{1,2}\|_F^2\Big],$$

It is easy to check that

$$\widetilde{\mathcal{L}}(\widetilde{\mathbf{M}}) = \mathcal{L}(\mathbf{U}_1\mathbf{U}_2^\top) + \frac{\mu}{8}\|\mathbf{U}_1^\top\mathbf{U}_1 - \mathbf{U}_2^\top\mathbf{U}_2\|_F^2,$$

and $\widetilde{\mathcal{L}}(\cdot)$ is a restricted $(L+\mu/2)$-smooth and $(\mu/4)$-strongly convex function over positive semi-definite matrices with rank less than or equal to r. We further define $\widetilde{\mathcal{G}}(\mathbf{Z}) = \widetilde{\mathcal{L}}(\mathbf{Z}\mathbf{Z}^\top)$. Thus we have converted problem (4.2) into the form of (1.2), and our AFGD algorithm as well as its theoretical guarantee are directly applicable.

5 Experiments

In this section, we evaluate our proposed algorithm by applying it to matrix regression (i.e., matrix sensing) and matrix completion on synthetic data. Due to space limit, we defer the experiment results for matrix regression to Appendix E. We compare AFGD with two baseline algorithms: FGD (Bhojanapalli et al., 2015) and vanilla Nesterov's AGD (Nesterov, 2004), as discussed in (3.1). Note that there is no theoretical guarantee of AGD for solving (1.2). We did not compare with AGD-AC (Li and Lin, 2017), because it is not a practical algorithm, as we explained before. We notice that Li and Lin (2017) implemented a heuristic version of AGD-AC in their experiments, which does not have theoretical guarantee but has almost the same empirical performance as AGD. We did not include this heuristic version of AGD-AC for comparison here, because it provides little additional insight beyond the comparison with vanilla AGD. For FGD, AGD, AFGD, the step size is selected by grid search over $\{10^{-k}, 5\cdot 10^{-k}\}, k = 0, 1, 2, 3, 4$, and γ is selected from $\{10^{-k}, 5\cdot 10^{-k}\}, k = 0, 1, 2, 3, 4$. For ACCPROJ, we set the iteration number to $T = 10$.

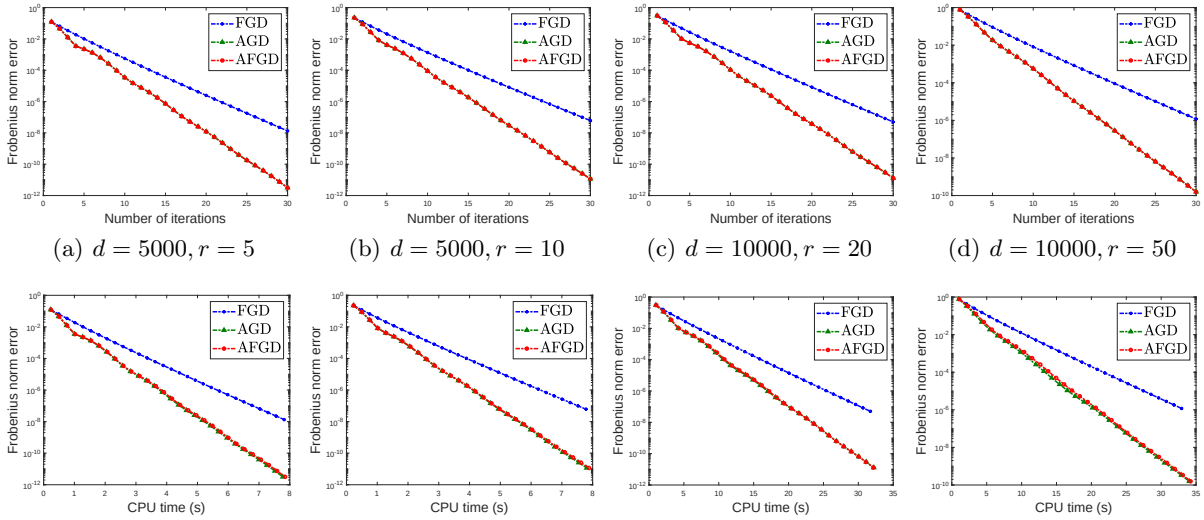
5.1 Matrix Completion

In this section we present the experimental results for matrix completion. In matrix completion we want to estimate an underlying low rank positive semidefinite matrix $\mathbf{M}^* \in \mathbb{R}^{d\times d}$ from partial observations. In detail, we partially observe entries of $\mathbf{M}^*$ over a subset $\Omega \in [d]\times[d]$. We assume a uniform sampling model such that each entry of $\mathbf{M}^*$ is observed with probability p, i.e., $(i,j)\in\Omega$ with probability p. We consider the noiseless case where the observed entries $\mathbf{Y}_{i,j} = \mathbf{M}^*_{i,j}$, then the matrix completion problem is formulated as

$$\min_{\mathbf{M}\succeq\mathbf{0},\ \mathrm{rank}(\mathbf{M})\le r} \frac{1}{2}\sum_{(i,j)\in\Omega}(\mathbf{M}_{i,j} - \mathbf{Y}_{i,j})^2.$$

We reparameterize $\mathbf{M}$ as $\mathbf{M} = \mathbf{U}\mathbf{U}^\top$, and consider the following low-rank matrix factorization problem:

$$\min_{\mathbf{U}\in\mathbb{R}^{d\times r}} \frac{1}{2}\sum_{(i,j)\in\Omega}[(\mathbf{U}\mathbf{U}^\top)_{i,j} - \mathbf{Y}_{i,j}]^2.$$



(a) $d = 5000, r = 5$ (b) $d = 5000, r = 10$ (c) $d = 10000, r = 20$ (d) $d = 10000, r = 50$

(e) $d = 5000, r = 5$ (f) $d = 5000, r = 10$ (g) $d = 10000, r = 20$ (h) $d = 10000, r = 50$

Figure 1: Comparison of FGD, AGD and AFGD for matrix completion. Plots of squared error $\|\mathbf{U}\mathbf{U}^\top - \mathbf{M}^*\|_F^2$ versus number of iterations ((a)-(d)) and CPU time ((e)-(h)).

We consider the following four settings: $d = 5000, r = 5$; $d = 5000, r = 10$; $d = 10000, r = 20$; $d = 10000, r = 50$. First, we generate the unknown matrix $\mathbf{M}^* = \mathbf{U}^*(\mathbf{U}^*)^\top$, with $\mathbf{U}^*$ being randomly generated from standard Gaussian distribution. We use uniform observation model to obtain data matrix $\mathbf{Y}$ with observation probability 0.2. All algorithms use the same initialization method proposed in Bhojanapalli et al. (2015) (See Appendix F for more details). All results are averaged over 20 trials. To illustrate the convergence rate, we report the squared error $\|\mathbf{U}\mathbf{U}^\top - \mathbf{M}^*\|_F^2$ in log scale. We compare different algorithms with respect to both number of iterations and CPU time and plot them in Figure 1. The results for $d = 5000, r = 5$ are presented in Figures 1(a) and 1(e), the results for $d = 5000, r = 10$ are shown in Figures 1(b) and 1(f), the results for $d = 10000, r = 20$ are shown in Figures 1(c) and 1(g), the results for $d = 10000, r = 50$ are shown in Figures 1(d) and 1(h).

We can see that AFGD converges faster than FGD with respect to both number of iterations and CPU time, which corroborates our theory. We also observe that vanilla AGD performs almost the same as AFGD in all settings, while in theory its accelerated convergence rate for low-rank matrix factorization is not proved. Our proposed AFGD enjoys an accelerated convergence rate both in theory and practice, and its projection step (solved by Algorithm 2) does not introduce significant computational overhead. This again strengthens the superiority of our algorithm for matrix factorization.

6 Conclusions and Future Work

In this work, we proposed a novel accelerated factored gradient descent method for low-rank matrix factorization. We proved that our algorithm achieves better iteration complexity and computational complexity locally compared with other existing baseline algorithms. Our algorithm and theory can be easily extended to rectangle matrices. Yet there are still some open questions. Can our algorithm also achieve an accelerated convergence rate with only random initialization? It has been shown in Chen et al. (2018) that gradient descent with random initialization can achieve a globally linear convergence rate for phase retrieval problem. We will explore whether a globally accelerated convergence rate can be proved for AFGD for solving low-rank matrix factorization.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. This research was sponsored in part by the National Science Foundation IIS-1906169. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

Agarwal, N., Allen-Zhu, Z., Bullins, B., Hazan, E. and Ma, T. (2017). Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM.

Bhojanapalli, S., Kyrillidis, A. and Sanghavi, S. (2015). Dropping convexity for faster semi-definite optimization. *arXiv preprint* .

Burer, S. and Monteiro, R. D. (2003). A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming* **95** 329–357.

Byrd, R. H., Nocedal, J. and Schnabel, R. B. (1994). Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming* **63** 129–156.

Cabral, R. S., De la Torre, F., Costeira, J. P. and Bernardino, A. (2011). Matrix completion for multi-label image classification. In *NIPS*, vol. 201.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on* **56** 2053–2080.

Carmon, Y., Duchi, J. C., Hinder, O. and Sidford, A. (2018). Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization* **28** 1751–1772.

Chen, Y., Chi, Y., Fan, J. and Ma, C. (2018). Gradient descent with random initialization: Fast global convergence for nonconvex phase retrieval. *Mathematical Programming* 1–33.

Chen, Y. and Wainwright, M. J. (2015). Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025* .

Ge, R., Huang, F., Jin, C. and Yuan, Y. (2015). Escaping from saddle points-online stochastic gradient for tensor decomposition. In *COLT*.

Ghadimi, S. and Lan, G. (2016). Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming* **156** 59–99.

Gui, H., Han, J. and Gu, Q. (2016). Towards faster rates and oracle property for low-rank matrix estimation. In *International Conference on Machine Learning*.

Hardt, M. (2014). Understanding alternating minimization for matrix completion. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE.

Hardt, M. and Price, E. (2014). The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems*.

Jain, P., Kar, P. et al. (2017). Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning* **10** 142–336.

Jain, P., Netrapalli, P. and Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM.

Jin, C., Netrapalli, P. and Jordan, M. I. (2018). Accelerated gradient descent escapes saddle points faster than gradient descent. In *Conference On Learning Theory*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Li, H. and Lin, Z. (2017). Provable accelerated gradient method for nonconvex low rank optimization. *arXiv preprint arXiv:1702.04959* .

Li, R.-C. (1995). New perturbation bounds for the unitary polar factor. *SIAM Journal on Matrix Analysis and Applications* **16** 327–332.

Lin, H., Mairal, J. and Harchaoui, Z. (2015). A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*.

Liu, Y., Shang, F., Cheng, J., Cheng, H. and Jiao, L. (2017). Accelerated first-order methods for geodesically convex optimization on riemannian manifolds. In *Advances in Neural Information Processing Systems*.

Negahban, S. and Wainwright, M. J. (2011). Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics* 1069–1097.

Negahban, S. and Wainwright, M. J. (2012). Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *Journal of Machine Learning Research* **13** 1665–1697.

Negahban, S., Yu, B., Wainwright, M. J. and Ravikumar, P. K. (2009). A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*.

Nesterov, Y. (1988). On an approach to the construction of optimal methods of smooth convex functions. *Ékonom. i. Mat. Metody* **24** 509–517.

Nesterov, Y. (2004). *Introductory lectures on convex optimization: A Basic Course*. Springer Science & Business Media.

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. Akad. Nauk SSSR*, vol. 269.

Paquette, C., Lin, H., Drusvyatskiy, D., Mairal, J. and Harchaoui, Z. (2018). Catalyst for gradient-based nonconvex optimization. In *AISTATS 2018-21st International Conference on Artificial Intelligence and Statistics*.

Park, D., Kyrillidis, A., Caramanis, C. and Sanghavi, S. (2016). Finding low-rank solutions to matrix problems, efficiently and provably. *Preprint* .

Recht, B., Fazel, M. and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review* **52** 471–501.

Srebro, N., Rennie, J. and Jaakkola, T. S. (2004). Maximum-margin matrix factorization. In *Advances in neural information processing systems*.

Sun, R. and Luo, Z.-Q. (2015). Guaranteed matrix completion via nonconvex factorization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE.

Tu, S., Boczar, R., Simchowitz, M., Soltanolkotabi, M. and Recht, B. (2016). Low-rank solutions of linear matrix equations via procrustes flow. In *International Conference on Machine Learning*.

Wang, L., Zhang, X. and Gu, Q. (2017a). A unified computational and statistical framework for nonconvex low-rank matrix estimation. In *Artificial Intelligence and Statistics*.

Wang, L., Zhang, X. and Gu, Q. (2017b). A unified variance reduction-based framework for nonconvex low-rank matrix recovery. In *International Conference on Machine Learning*.

Zhang, H. and Sra, S. (2018). An estimate sequence for geodesically convex optimization. In *Conference On Learning Theory*.

Zhang, X., Wang, L. and Gu, Q. (2018). A unified framework for nonconvex low-rank plus sparse matrix recovery. In *International Conference on Artificial Intelligence and Statistics*.

Zhao, T., Wang, Z. and Liu, H. (2015). A nonconvex optimization framework for low rank matrix estimation. In *Advances in Neural Information Processing Systems*.

Zheng, Q. and Lafferty, J. (2015). A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements. In *Advances in Neural Information Processing Systems*.

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

Zhu, Z., Li, Q., Tang, G. and Wakin, M. B. (2018). Global optimality in low-rank matrix optimization. *IEEE Transactions on Signal Processing* **66** 3614–3628.