Provable Efficient Online Matrix Completion via Non-convex Stochastic Gradient Descent

Chi Jin
UC Berkeley
chijin@cs.berkeley.edu

Sham M. Kakade
University of Washington
sham@cs.washington.edu

Praneeth Netrapalli
Microsoft Research India
praneeth@microsoft.com

Abstract

Matrix completion, where we wish to recover a low rank matrix by observing a few entries from it, is a widely studied problem in both theory and practice with wide applications. Most of the provable algorithms so far on this problem have been restricted to the offline setting where they provide an estimate of the unknown matrix using all observations simultaneously. However, in many applications, the online version, where we observe one entry at a time and dynamically update our estimate, is more appealing. While existing algorithms are efficient for the offline setting, they could be highly inefficient for the online setting.

In this paper, we propose the first provable, efficient online algorithm for matrix completion. Our algorithm starts from an initial estimate of the matrix and then performs non-convex stochastic gradient descent (SGD). After every observation, it performs a fast update involving only one row of two tall matrices, giving near linear total runtime. Our algorithm can be naturally used in the offline setting as well, where it gives competitive sample complexity and runtime to state of the art algorithms. Our proofs introduce a general framework to show that SGD updates tend to stay away from saddle surfaces and could be of broader interests to other non-convex problems.

1 Introduction

Low rank matrix completion refers to the problem of recovering a low rank matrix by observing the values of only a tiny fraction of its entries. This problem arises in several applications such as video denoising [13], phase retrieval [3] and most famously in movie recommendation engines [15]. In the context of recommendation engines for instance, the matrix we wish to recover would be user-item rating matrix where each row corresponds to a user and each column corresponds to an item. Each entry of the matrix is the rating given by a user to an item. Low rank assumption on the matrix is inspired by the intuition that rating of an item by a user depends on only a few hidden factors, which are much fewer than the number of users or items. The goal is to estimate the ratings of all items by users given only partial ratings of items by users, which would then be helpful in recommending new items to users.

The seminal works of Candès and Recht [4] first identified regularity conditions under which low rank matrix completion can be solved in polynomial time using convex relaxation – low rank matrix completion could be ill-posed and NP-hard in general without such regularity assumptions [9]. Since then, a number of works have studied various algorithms under different settings for matrix completion: weighted and noisy matrix completion, fast convex solvers, fast iterative non-convex solvers, parallel and distributed algorithms and so on.

Most of this work however deals only with the offline setting where all the observed entries are revealed at once and the recovery procedure does computation using all these observations simultaneously. However in several applications [5, 18], we encounter the online setting where observations are

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

only revealed sequentially and at each step the recovery algorithm is required to maintain an estimate of the low rank matrix based on the observations so far. Consider for instance recommendation engines, where the low rank matrix we are interested in is the user-item rating matrix. While we make an observation only when a user rates an item, at any point of time, we should have an estimate of the user-item rating matrix based on all prior observations so as to be able to continuously recommend items to users. Moreover, this estimate should get better as we observe more ratings.

Algorithms for offline matrix completion can be used to solve the online version by rerunning the algorithm after every additional observation. However, performing so much computation for every observation seems wasteful and is also impractical. For instance, using alternating minimization, which is among the fastest known algorithms for the offline problem, would mean that we take several passes of the entire data for every additional observation. This is simply not feasible in most settings. Another natural approach is to group observations into batches and do an update only once for each batch. This however induces a lag between observations and estimates which is undesirable. To the best of our knowledge, there is no known *provable, efficient, online algorithm* for matrix completion.

On the other hand, in order to deal with the online matrix completion scenario in practical applications, several heuristics (with no convergence guarantees) have been proposed in literature [2, 19]. Most of these approaches are based on starting with an estimate of the matrix and doing fast updates of this estimate whenever a new observation is presented. One of the update procedures used in this context is that of stochastic gradient descent (SGD) applied to the following non-convex optimization problem

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{M} - \mathbf{U}\mathbf{V}^\top\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \mathbf{U} \in \mathbb{R}^{d_1 \times k}, \mathbf{V} \in \mathbb{R}^{d_2 \times k}, \tag{1}$$

where $\mathbf{M}$ is the unknown matrix of size $d_1 \times d_2$, k is the rank of $\mathbf{M}$ and $\mathbf{U}\mathbf{V}^\top$ is a low rank factorization of $\mathbf{M}$ we wish to obtain. The algorithm starts with some $\mathbf{U}_0$ and $\mathbf{V}_0$, and given a new observation $(\mathbf{M})_{ij}$, SGD updates the i^{th}-row and the j^{th}-row of the current iterates $\mathbf{U}_t$ and $\mathbf{V}_t$ respectively by

$$\begin{aligned} \mathbf{U}_{t+1}^{(i)} &= \mathbf{U}_t^{(i)} - 2\eta d_1 d_2 \left(\mathbf{U}_t\mathbf{V}_t^\top - \mathbf{M}\right)_{ij} \mathbf{V}_t^{(j)}, \text{ and,} \\ \mathbf{V}_{t+1}^{(j)} &= \mathbf{V}_t^{(j)} - 2\eta d_1 d_2 \left(\mathbf{U}_t\mathbf{V}_t^\top - \mathbf{M}\right)_{ij} \mathbf{U}_t^{(i)}, \end{aligned} \tag{2}$$

where η is an appropriately chosen stepsize, and $\mathbf{U}^{(i)}$ denote the i^{th} row of matrix $\mathbf{U}$. Note that each update modifies only one row of the factor matrices $\mathbf{U}$ and $\mathbf{V}$, and the computation only involves one row of $\mathbf{U}, \mathbf{V}$ and the new observed entry $(\mathbf{M})_{ij}$ and hence are extremely fast. These fast updates make SGD extremely appealing in practice. Moreover, SGD, in the context of matrix completion, is also useful for parallelization and distributed implementation [23].

1.1 Our Contributions

In this work we present the first provable efficient algorithm for online matrix completion by showing that SGD (2) with a good initialization converges to a true factorization of $\mathbf{M}$ at a geometric rate. Our main contributions are as follows.

- We provide the *first provable, efficient, online* algorithm for matrix completion. Starting with a good initialization, after each observation, the algorithm makes quick updates each taking time $O(k^3)$ and requires $O(\mu dk\kappa^4(k + \log \frac{\|\mathbf{M}\|_{\mathrm{F}}}{\epsilon}) \log d)$ observations to reach ϵ accuracy, where μ is the incoherence parameter, $d = \max(d_1, d_2)$, k is the rank and κ is the condition number of $\mathbf{M}$.
- Moreover, our result features both sample complexity and total runtime *linear in* d, and is *competitive* to even the best existing offline results for matrix completion. (either improve over or is incomparable, i.e., better in some parameters and worse in others, to these results). See Table 1 for the comparison.
- To obtain our results, we introduce *a general framework to show SGD updates tend to stay away from saddle surfaces*. In order to do so, we consider distances from saddle surfaces, show that they behave like sub-martingales under SGD updates and use martingale convergence techniques to conclude that the iterates stay away from saddle surfaces. While [24] shows that SGD updates stay away from saddle surfaces, the stepsizes they can handle are

Table 1: Comparison of sample complexity and runtime of our algorithm with existing algorithms in order to obtain Frobenius norm error ϵ. $\widetilde{O}(\cdot)$ hides $\log d$ factors. See Section 1.2 for more discussion.

Algorithm	Sample complexity	Total runtime	Online?
Nuclear Norm [22]	$\widetilde{O}(\mu dk)$	$\widetilde{O}(d^3/\sqrt{\epsilon})$	No
Alternating minimization [14]	$\widetilde{O}(\mu dk\kappa^8 \log \frac{1}{\epsilon})$	$\widetilde{O}(\mu dk^2\kappa^8 \log \frac{1}{\epsilon})$	No
Alternating minimization [8]	$\widetilde{O}\left(\mu dk^2\kappa^2 \left(k + \log \frac{1}{\epsilon}\right)\right)$	$\widetilde{O}\left(\mu dk^3\kappa^2 \left(k + \log \frac{1}{\epsilon}\right)\right)$	No
Projected gradient descent[12]	$\widetilde{O}(\mu dk^5)$	$\widetilde{O}(\mu dk^7 \log \frac{1}{\epsilon})$	No
SGD [24]	$\widetilde{O}(\mu^2 dk^7\kappa^6)$	poly$(\mu, d, k, \kappa) \log \frac{1}{\epsilon}$	Yes
Our result	$\widetilde{O}\left(\mu dk\kappa^4 \left(k + \log \frac{1}{\epsilon}\right)\right)$	$\widetilde{O}\left(\mu dk^4\kappa^4 \log \frac{1}{\epsilon}\right)$	Yes

quite small (scaling as $1/\text{poly}(d_1, d_2)$), leading to suboptimal computational complexity. Our framework makes it possible to establish the same statement for much larger step sizes, giving us near-optimal runtime. We believe these techniques may be applicable in other non-convex settings as well.

1.2 Related Work

In this section we will mention some more related work.

Offline matrix completion: There has been a lot of work on designing offline algorithms for matrix completion, we provide the detailed comparison with our algorithm in Table 1. The nuclear norm relaxation algorithm [22] has near-optimal sample complexity for this problem but is computationally expensive. Motivated by the empirical success of non-convex heuristics, a long line of works, [14, 8, 12, 24] and so on, has obtained convergence guarantees for alternating minimization, gradient descent, projected gradient descent etc. Even the best of these are suboptimal in sample complexity by poly(k, κ) factors. Our sample complexity is better than that of [14] and is incomparable to those of [8, 12]. To the best of our knowledge, the only provable online algorithm for this problem is that of Sun and Luo [24]. However the stepsizes they suggest are quite small, leading to suboptimal computational complexity by factors of poly(d_1, d_2). The runtime of our algorithm is linear in d, which makes poly(d) improvements over it.

Other models for online matrix completion: Another variant of online matrix completion studied in the literature is where observations are made on a column by column basis e.g., [16, 26]. These models can give improved offline performance in terms of space and could potentially work under relaxed regularity conditions. However, they do not tackle the version where only entries (as opposed to columns) are observed.

Non-convex optimization: Over the last few years, there has also been a significant amount of work in designing other efficient algorithms for solving non-convex problems. Examples include eigenvector computation [6, 11], sparse coding [20, 1] etc. For general non-convex optimization, an interesting line of recent work is that of [7], which proves gradient descent with noise can also escape saddle point, but they only provide polynomial rate without explicit dependence. Later [17, 21] show that without noise, the space of points from where gradient descent converges to a saddle point is a measure zero set. However, they do not provide a rate of convergence. Another related piece of work to ours is [10], proves global convergence along with rates of convergence, for the special case of computing matrix squareroot.

1.3 Outline

The rest of the paper is organized as follows. In Section 2 we formally describe the problem and all relevant parameters. In Section 3, we present our algorithms, results and some of the key intuition

behind our results. In Section 4 we give proof outline for our main results. We conclude in Section 5. All formal proofs are deferred to the Appendix.

2 Preliminaries

In this section, we introduce our notation, formally define the matrix completion problem and regularity assumptions that make the problem tractable.

2.1 Notation

We use $[d]$ to denote $\{1, 2, \cdots, d\}$. We use bold capital letters $\mathbf{A}, \mathbf{B}$ to denote matrices and bold lowercase letters $\mathbf{u}, \mathbf{v}$ to denote vectors. $\mathbf{A}_{ij}$ means the $(i,j)^{\text{th}}$ entry of matrix $\mathbf{A}$. $\|\mathbf{w}\|$ denotes the ℓ_2-norm of vector $\mathbf{w}$ and $\|\mathbf{A}\|/\|\mathbf{A}\|_{\text{F}}/\|\mathbf{A}\|_\infty$ denotes the spectral/Frobenius/infinity norm of matrix $\mathbf{A}$. $\sigma_i(\mathbf{A})$ denotes the i^{th} largest singular value of $\mathbf{A}$ and $\sigma_{\min}(\mathbf{A})$ denotes the smallest singular value of $\mathbf{A}$. We also let $\kappa(\mathbf{A}) = \|\mathbf{A}\| / \sigma_{\min}(\mathbf{A})$ denote the condition number of $\mathbf{A}$ (i.e., the ratio of largest to smallest singular value). Finally, for orthonormal bases of a subspace $\mathbf{W}$, we also use $\mathcal{P}_{\mathbf{W}} = \mathbf{W}\mathbf{W}^\top$ to denote the projection to the subspace spanned by $\mathbf{W}$.

2.2 Problem statement and assumptions

Consider a general rank k matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. Let $\Omega \subset [d_1] \times [d_2]$ be a subset of coordinates, which are sampled uniformly and independently from $[d_1] \times [d_2]$. We denote $\mathcal{P}_\Omega(\mathbf{M})$ to be the projection of $\mathbf{M}$ on set Ω so that:

$$[\mathcal{P}_\Omega(\mathbf{M})]_{ij} = \begin{cases} \mathbf{M}_{ij}, & \text{if } (i,j) \in \Omega \\ 0, & \text{if } (i,j) \notin \Omega \end{cases}$$

Low rank matrix completion is the task of recovering $\mathbf{M}$ by only observing $\mathcal{P}_\Omega(\mathbf{M})$. This task is ill-posed and NP-hard in general [9]. In order to make this tractable, we make by now standard assumptions about the structure of $\mathbf{M}$.

Definition 2.1. Let $\mathbf{W} \in \mathbb{R}^{d \times k}$ be an orthonormal basis of a subspace of $\mathbb{R}^d$ of dimension k. The coherence of $\mathbf{W}$ is defined to be

$$\mu(\mathbf{W}) \stackrel{\text{def}}{=} \frac{d}{k} \max_{1 \le i \le d} \|\mathcal{P}_{\mathbf{W}} \mathbf{e}_i\|^2 = \frac{d}{k} \max_{1 \le i \le d} \left\|\mathbf{e}_i^\top \mathbf{W}\right\|^2$$

Assumption 2.2 (μ-incoherence[4, 22])**.** *We assume $\mathbf{M}$ is μ-incoherent, i.e., $\max\{\mu(\mathbf{X}), \mu(\mathbf{Y})\} \le \mu$, where $\mathbf{X} \in \mathbb{R}^{d_1 \times k}$, $\mathbf{Y} \in \mathbb{R}^{d_2 \times k}$ are the left and right singular vectors of $\mathbf{M}$.*

3 Main Results and Intuition

In this section, we present our main result. We will first state result for a special case where $\mathbf{M}$ is a symmetric positive semi-definite (PSD) matrix, where the algorithm and analysis are much simpler. We will then discuss the general case.

3.1 Symmetric PSD Case

Consider the special case where $\mathbf{M}$ is symmetric PSD and let $d \stackrel{\text{def}}{=} d_1 = d_2$. Then, we can parametrize a rank k symmetric PSD matrix by $\mathbf{U}\mathbf{U}^\top$ where $\mathbf{U} \in \mathbb{R}^{d \times k}$. Our algorithm for this case is given in Algorithm 1. The following theorem provides guarantees on the performance of Algorithm 1. The algorithm starts by using an initial set of samples Ω_{init} to construct a crude approximation to the low rank of factorization of $\mathbf{M}$. It then observes samples from $\mathbf{M}$ one at a time and updates its factorization after every observation.

Theorem 3.1. *Let $\mathbf{M} \in \mathbb{R}^{d \times d}$ be a rank k, symmetric PSD matrix with μ-incoherence. There exist some absolute constants c_0 and c such that if $|\Omega_{init}| \ge c_0 \mu d k^2 \kappa^2(\mathbf{M}) \log d$, learning rate $\eta \le \frac{c}{\mu d k \kappa^3(\mathbf{M}) \|\mathbf{M}\| \log d}$, then with probability at least $1 - \frac{1}{d^8}$, we will have for all $t \le d^2$ that*[1]*:*

$$\|\mathbf{U}_t \mathbf{U}_t^\top - \mathbf{M}\|_F^2 \le \left(1 - \frac{1}{2}\eta \cdot \sigma_{\min}(\mathbf{M})\right)^t \left(\frac{1}{10}\sigma_{\min}(\mathbf{M})\right)^2.$$

[1] W.L.O.G, we can always assume $t < d^2$, otherwise we already observed the entire matrix.

Algorithm 1 Online Algorithm for PSD Matrix Completion.

Input: Initial set of uniformly random samples Ω_{init} of a symmetric PSD matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$, learning rate η, iterations T
Output: $\mathbf{U}$ such that $\mathbf{U}\mathbf{U}^\top \approx \mathbf{M}$
$\mathbf{U}_0\mathbf{U}_0^\top \leftarrow$ top k SVD of $\frac{d^2}{|\Omega_{\text{init}}|}\mathcal{P}_{\Omega_{\text{init}}}(\mathbf{M})$
for $t = 0, \cdots, T-1$ **do**
  Observe $\mathbf{M}_{ij}$ where $(i,j) \sim \text{Unif}([d]\times[d])$
  $\mathbf{U}_{t+1} \leftarrow \mathbf{U}_t - 2\eta d^2(\mathbf{U}_t\mathbf{U}_t^\top - \mathbf{M})_{ij}(\mathbf{e}_i\mathbf{e}_j^\top + \mathbf{e}_j\mathbf{e}_i^\top)\mathbf{U}_t$
end for
Return $\mathbf{U}_T$

Remarks:

- The algorithm uses an initial set of observations Ω_{init} to produce a warm start iterate $\mathbf{U}_0$, then enters the online stage, where it performs SGD.
- The sample complexity of the warm start phase is $O(\mu dk^2\kappa^2(\mathbf{M})\log d)$. The initialization consists of a top-k SVD on a sparse matrix, whose runtime is $O(\mu dk^3\kappa^2(\mathbf{M})\log d)$.
- For the online phase (SGD), if we choose $\eta = \frac{c}{\mu dk\kappa^3(\mathbf{M})\|\mathbf{M}\|\log d}$, the number of observations T required for the error $\|\mathbf{U}_T\mathbf{U}_T^\top - \mathbf{M}\|_{\text{F}}$ to be smaller than ϵ is $O(\mu dk\kappa(\mathbf{M})^4\log d\log\frac{\sigma_{\min}(\mathbf{M})}{\epsilon})$.
- Since each SGD step modifies two rows of $\mathbf{U}_t$, its runtime is $O(k)$ with a total runtime for online phase of $O(kT)$.

Our proof approach is to essentially show that the objective function is well-behaved (i.e., is smooth and strongly convex) in a local neighborhood of the warm start region, and then use standard techniques to show that SGD obtains geometric convergence in this setting. The most challenging and novel part of our analysis comprises of showing that the iterate does not leave this local neighborhood while performing SGD updates. Refer Section 4 for more details on the proof outline.

3.2 General Case

Let us now consider the general case where $\mathbf{M} \in \mathbb{R}^{d_1\times d_2}$ can be factorized as $\mathbf{U}\mathbf{V}^\top$ with $\mathbf{U} \in \mathbb{R}^{d_1\times k}$ and $\mathbf{V} \in \mathbb{R}^{d_2\times k}$. In this scenario, we denote $d = \max\{d_1, d_2\}$. We recall our remarks from the previous section that our analysis of the performance of SGD depends on the smoothness and strong convexity properties of the objective function in a local neighborhood of the iterates. Having $\mathbf{U} \neq \mathbf{V}$ introduces additional challenges in this approach since for any nonsingular k-by-k matrix $\mathbf{C}$, and $\mathbf{U}' \stackrel{\text{def}}{=} \mathbf{U}\mathbf{C}^\top, \mathbf{V}' \stackrel{\text{def}}{=} \mathbf{V}\mathbf{C}^{-1}$, we have $\mathbf{U}'\mathbf{V}'^\top = \mathbf{U}\mathbf{V}^\top$. Suppose for instance $\mathbf{C}$ is a very small scalar times the identity i.e., $\mathbf{C} = \delta\mathbf{I}$ for some small $\delta > 0$. In this case, $\mathbf{U}'$ will be large while $\mathbf{V}'$ will be small. This drastically deteriorates the smoothness and strong convexity properties of the objective function in a neighborhood of $(\mathbf{U}', \mathbf{V}')$.

Algorithm 2 Online Algorithm for Matrix Completion (Theoretical)

Input: Initial set of uniformly random samples Ω_{init} of $\mathbf{M} \in \mathbb{R}^{d_1\times d_2}$, learning rate η, iterations T
Output: $\mathbf{U}, \mathbf{V}$ such that $\mathbf{U}\mathbf{V}^\top \approx \mathbf{M}$
$\mathbf{U}_0\mathbf{V}_0^\top \leftarrow$ top k SVD of $\frac{d_1 d_2}{|\Omega_{\text{init}}|}\mathcal{P}_{\Omega_{\text{init}}}(\mathbf{M})$
for $t = 0, \cdots, T-1$ **do**
  $\mathbf{W}_U\mathbf{D}\mathbf{W}_V^\top \leftarrow \text{SVD}(\mathbf{U}_t\mathbf{V}_t^\top)$
  $\tilde{\mathbf{U}}_t \leftarrow \mathbf{W}_U\mathbf{D}^{\frac{1}{2}}, \quad \tilde{\mathbf{V}}_t \leftarrow \mathbf{W}_V\mathbf{D}^{\frac{1}{2}}$
  Observe $\mathbf{M}_{ij}$ where $(i,j) \sim \text{Unif}([d]\times[d])$
  $\mathbf{U}_{t+1} \leftarrow \tilde{\mathbf{U}}_t - 2\eta d_1 d_2(\tilde{\mathbf{U}}_t\tilde{\mathbf{V}}_t^\top - \mathbf{M})_{ij}\mathbf{e}_i\mathbf{e}_j^\top\tilde{\mathbf{V}}_t$
  $\mathbf{V}_{t+1} \leftarrow \tilde{\mathbf{V}}_t - 2\eta d_1 d_2(\tilde{\mathbf{U}}_t\tilde{\mathbf{V}}_t^\top - \mathbf{M})_{ij}\mathbf{e}_j\mathbf{e}_i^\top\tilde{\mathbf{U}}_t$
end for
Return $\mathbf{U}_T, \mathbf{V}_T$.

To preclude such a scenario, we would ideally like to renormalize after each step by doing $\tilde{\mathbf{U}}_t \leftarrow \mathbf{W}_U \mathbf{D}^{\frac{1}{2}}, \tilde{\mathbf{V}}_t \leftarrow \mathbf{W}_V \mathbf{D}^{\frac{1}{2}}$, where $\mathbf{W}_U \mathbf{D} \mathbf{W}_V^\top$ is the SVD of matrix $\mathbf{U}_t \mathbf{V}_t^\top$. This algorithm is described in Algorithm 2. However, a naive implementation of Algorithm 2, especially the SVD step, would incur $O(\min\{d_1, d_2\})$ computation per iteration, resulting in a runtime overhead of $O(d)$ over both the online PSD case (i.e., Algorithm 1) as well as the near linear time offline algorithms (see Table 1). It turns out that we can take advantage of the fact that in each iteration we only update a single row of $\mathbf{U}_t$ and a single row of $\mathbf{V}_t$, and do efficient (but more complicated) update steps instead of doing an SVD on $d_1 \times d_2$ matrix. The resulting algorithm is given in Algorithm 3. The key idea is that in order to implement the updates, it suffices to do an SVD of $\mathbf{U}_t^\top \mathbf{U}_t$ and $\mathbf{V}_t^\top \mathbf{V}_t$ which are $k \times k$ matrices. So the runtime of each iteration is at most $O(k^3)$. The following lemma shows the equivalence between Algorithms 2 and 3.

Algorithm 3 Online Algorithm for Matrix Completion (Practical)

Input: Initial set of uniformly random samples Ω_{init} of $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$, learning rate η, iterations T
Output: $\mathbf{U}, \mathbf{V}$ such that $\mathbf{U}\mathbf{V}^\top \approx \mathbf{M}$

$\mathbf{U}_0 \mathbf{V}_0^\top \leftarrow$ top k SVD of $\frac{d_1 d_2}{|\Omega_{\text{init}}|} \mathcal{P}_{\Omega_{\text{init}}}(\mathbf{M})$
for $t = 0, \cdots, T-1$ **do**
  $\mathbf{R}_U \mathbf{D}_U \mathbf{R}_U^\top \leftarrow \text{SVD}(\mathbf{U}_t^\top \mathbf{U}_t)$
  $\mathbf{R}_V \mathbf{D}_V \mathbf{R}_V^\top \leftarrow \text{SVD}(\mathbf{V}_t^\top \mathbf{V}_t)$
  $\mathbf{Q}_U \mathbf{D} \mathbf{Q}_V^\top \leftarrow \text{SVD}(\mathbf{D}_U^{\frac{1}{2}} \mathbf{R}_U^\top \mathbf{R}_V (\mathbf{D}_V^{\frac{1}{2}})^\top)$
  Observe $\mathbf{M}_{ij}$ where $(i, j) \sim \text{Unif}([d] \times [d])$
  $\mathbf{U}_{t+1} \leftarrow \mathbf{U}_t - 2\eta d_1 d_2 (\mathbf{U}_t \mathbf{V}_t^\top - \mathbf{M})_{ij} \mathbf{e}_i \mathbf{e}_j^\top \mathbf{V}_t \mathbf{R}_V \mathbf{D}_V^{-\frac{1}{2}} \mathbf{Q}_V \mathbf{Q}_U^\top \mathbf{D}_U^{\frac{1}{2}} \mathbf{R}_U^\top$
  $\mathbf{V}_{t+1} \leftarrow \mathbf{V}_t - 2\eta d_1 d_2 (\mathbf{U}_t \mathbf{V}_t^\top - \mathbf{M})_{ij} \mathbf{e}_j \mathbf{e}_i^\top \mathbf{U}_t \mathbf{R}_U \mathbf{D}_U^{-\frac{1}{2}} \mathbf{Q}_U \mathbf{Q}_V^\top \mathbf{D}_V^{\frac{1}{2}} \mathbf{R}_V^\top$
end for
Return $\mathbf{U}_T, \mathbf{V}_T$.

Lemma 3.2. *Algorithm 2 and Algorithm 3 are equivalent in the sense that: given same observations from* $\mathbf{M}$ *and other inputs, the outputs of Algorithm 2,* $\mathbf{U}, \mathbf{V}$ *and those of Algorithm 3,* $\mathbf{U}', \mathbf{V}'$ *satisfy* $\mathbf{U}\mathbf{V}^\top = \mathbf{U}'\mathbf{V}'^\top$.

Since the output of both algorithms is the same, we can analyze Algorithm 2 (which is easier than that of Algorithm 3), while implementing Algorithm 3 in practice. The following theorem is the main result of our paper which presents guarantees on the performance of Algorithm 2.

Theorem 3.3. *Let* $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$ *be a rank* k *matrix with* μ*-incoherence and let* $d \stackrel{def}{=} \max(d_1, d_2)$. *There exist some absolute constants* c_0 *and* c *such that if* $|\Omega_{init}| \geq c_0 \mu d k^2 \kappa^2(\mathbf{M}) \log d$, *learning rate* $\eta \leq \frac{c}{\mu d k \kappa^3(\mathbf{M}) \|\mathbf{M}\| \log d}$, *then with probability at least* $1 - \frac{1}{d^8}$, *we will have for all* $t \leq d^2$ *that:*

$$\|\mathbf{U}_t \mathbf{V}_t^\top - \mathbf{M}\|_F^2 \leq \left(1 - \frac{1}{2}\eta \cdot \sigma_{\min}(\mathbf{M})\right)^t \left(\frac{1}{10}\sigma_{\min}(\mathbf{M})\right)^2.$$

Remarks:

- Just as in the case of PSD matrix completion (Theorem 3.1), Algorithm 2 needs an initial set of observations Ω_{init} to provide a warm start $\mathbf{U}_0$ and $\mathbf{V}_0$ after which it performs SGD.
- The sample complexity and runtime of the warm start phase are the same as in symmetric PSD case. The stepsize η and the number of observations T to achieve ϵ error in online phase (SGD) are also the same as in symmetric PSD case.
- However, runtime of each update step in online phase is $O(k^3)$ with total runtime for online phase $O(k^3 T)$.

The proof of this theorem again follows a similar line of reasoning as that of Theorem 3.1 by first showing that the local neighborhood of warm start iterate has good smoothness and strong convexity properties and then use them to show geometric convergence of SGD. Proof of the fact that iterates do not move away from this local neighborhood however is significantly more challenging due to renormalization steps in the algorithm. Please see Appendix C for the full proof.

4 Proof Sketch

In this section we will provide the intuition and proof sketch for our main results. For simplicity and highlighting the most essential ideas, we will mostly focus on the symmetric PSD case (Theorem 3.1). For the asymmetric case, though the high-level ideas are still valid, a lot of additional effort is required to address the renormalization step in Algorithm 2. This makes the proof more involved.

First, note that our algorithm for the PSD case consists of an initialization and then stochastic descent steps. The following lemma provides guarantees on the error achieved by the initial iterate $\mathbf{U}_0$.

Lemma 4.1. *Let $\mathbf{M} \in \mathbb{R}^{d\times d}$ be a rank-k PSD matrix with μ-incoherence. There exists a constant c_0 such that if $|\Omega_{init}| \geq c_0\mu dk^2\kappa^2(\mathbf{M})\log d$, then with probability at least $1-\frac{1}{d^{10}}$, the top-k SVD of $\frac{d^2}{|\Omega_{init}|}\mathcal{P}_{\Omega_{init}}(\mathbf{M})$ (denote as $\mathbf{U}_0\mathbf{U}_0^\top$) satisfies:*

$$\|\mathbf{M}-\mathbf{U}_0\mathbf{U}_0^\top\|_F \leq \frac{1}{20}\sigma_{\min}(\mathbf{M}) \quad \text{and} \quad \max_j \left\|\mathbf{e}_j^\top\mathbf{U}_0\right\|^2 \leq \frac{10\mu k\kappa(\mathbf{M})}{d}\|\mathbf{M}\| \tag{3}$$

By Lemma 4.1, we know the initialization algorithm already gives $\mathbf{U}_0$ in the local region given by Eq.(3). Intuitively, stochastic descent steps should keep doing local search within this local region.

To establish linear convergence on $\|\mathbf{U}_t\mathbf{U}_t^\top-\mathbf{M}\|_{\mathrm{F}}^2$ and obtain final result, we first establish several important lemmas describing the properties of this local regions. Throughout this section, we always denote $\mathrm{SVD}(\mathbf{M}) = \mathbf{X}\mathbf{S}\mathbf{X}^\top$, where $\mathbf{X}\in\mathbb{R}^{d\times k}$, and diagnal matrix $\mathbf{S}\in\mathbb{R}^{k\times k}$. We postpone all the formal proofs in Appendix.

Lemma 4.2. *For function $f(\mathbf{U}) = \|\mathbf{M}-\mathbf{U}\mathbf{U}^\top\|_F^2$ and any $\mathbf{U}_1,\mathbf{U}_2\in\{\mathbf{U}|\,\|\mathbf{U}\|\leq\Gamma\}$, we have:*

$$\|\nabla f(\mathbf{U}_1)-\nabla f(\mathbf{U}_2)\|_F \leq 16\max\{\Gamma^2,\|\mathbf{M}\|\}\cdot\|\mathbf{U}_1-\mathbf{U}_2\|_F$$

Lemma 4.3. *For function $f(\mathbf{U}) = \|\mathbf{M}-\mathbf{U}\mathbf{U}^\top\|_F^2$ and any $\mathbf{U}\in\{\mathbf{U}|\sigma_{\min}(\mathbf{X}^\top\mathbf{U})\geq\gamma\}$, we have:*

$$\|\nabla f(\mathbf{U})\|_F^2 \geq 4\gamma^2 f(\mathbf{U})$$

Lemma 4.2 tells function f is smooth if spectral norm of $\mathbf{U}$ is not very large. On the other hand, $\sigma_{\min}(\mathbf{X}^\top\mathbf{U})$ not too small requires both $\sigma_{\min}(\mathbf{U}^\top\mathbf{U})$ and $\sigma_{\min}(\mathbf{X}^\top\mathbf{W})$ are not too small, where $\mathbf{W}$ is top-k eigenspace of $\mathbf{U}\mathbf{U}^\top$. That is, Lemma 4.3 tells function f has a property similar to strongly convex in standard optimization literature, if $\mathbf{U}$ is rank k in a robust sense ($\sigma_k(\mathbf{U})$ is not too small), and the angle between the top k eigenspace of $\mathbf{U}\mathbf{U}^\top$ and the top k eigenspace $\mathbf{M}$ is not large.

Lemma 4.4. *Within the region $\mathcal{D} = \{\mathbf{U}|\left\|\mathbf{M}-\mathbf{U}\mathbf{U}^\top\right\|_F \leq \frac{1}{10}\sigma_k(\mathbf{M})\}$, we have:*

$$\|\mathbf{U}\| \leq \sqrt{2\|\mathbf{M}\|}, \qquad \sigma_{\min}(\mathbf{X}^\top\mathbf{U}) \geq \sqrt{\sigma_k(\mathbf{M})/2}$$

Lemma 4.4 tells inside region $\{\mathbf{U}|\left\|\mathbf{M}-\mathbf{U}\mathbf{U}^\top\right\|_F \leq \frac{1}{10}\sigma_k(\mathbf{M})\}$, matrix $\mathbf{U}$ always has a good spectral property which gives preconditions for both Lemma 4.2 and 4.3, where $f(\mathbf{U})$ is both smooth and has a property very similar to strongly convex.

With above three lemmas, we already been able to see the intuition behind linear convergence in Theorem 3.1. Denote stochastic gradient

$$SG(\mathbf{U}) = 2d^2(\mathbf{U}\mathbf{U}^\top-\mathbf{M})_{ij}(\mathbf{e}_i\mathbf{e}_j^\top+\mathbf{e}_j\mathbf{e}_i^\top)\mathbf{U} \tag{4}$$

where $SG(\mathbf{U})$ is a random matrix depends on the randomness of sample (i,j) of matrix $\mathbf{M}$. Then, the stochastic update step in Algorithm 1 can be rewritten as:

$$\mathbf{U}_{t+1} \leftarrow \mathbf{U}_t - \eta SG(\mathbf{U}_t)$$

Let $f(\mathbf{U}) = \|\mathbf{M}-\mathbf{U}\mathbf{U}^\top\|_{\mathrm{F}}^2$, By easy caculation, we know $\mathbb{E}SG(\mathbf{U}) = \nabla f(\mathbf{U})$, that is $SG(\mathbf{U})$ is unbiased. Combine Lemma 4.4 with Lemma 4.2 and Lemma 4.3, we know within region $\mathcal{D}$ specified by Lemma 4.4, we have function $f(\mathbf{U})$ is $32\|\mathbf{M}\|$-smooth, and $\|\nabla f(\mathbf{U})\|_F^2 \geq 2\sigma_{\min}(\mathbf{M})f(\mathbf{U})$.

Let's suppose ideally, we always have $\mathbf{U}_0,\ldots,\mathbf{U}_t$ inside region $\mathcal{D}$, this directly gives:

$$\begin{aligned}
\mathbb{E}f(\mathbf{U}_{t+1}) &\leq \mathbb{E}f(\mathbf{U}_t) - \eta\mathbb{E}\langle\nabla f(\mathbf{U}_t), SG(\mathbf{U}_t)\rangle + 16\eta^2\|\mathbf{M}\|\cdot\mathbb{E}\|SG(\mathbf{U}_t)\|_{\mathrm{F}}^2 \\
&= \mathbb{E}f(\mathbf{U}_t) - \eta\mathbb{E}\|\nabla f(\mathbf{U}_t)\|_{\mathrm{F}}^2 + 16\eta^2\|\mathbf{M}\|\cdot\mathbb{E}\|SG(\mathbf{U}_t)\|_{\mathrm{F}}^2 \\
&\leq (1-2\eta\sigma_{\min}(\mathbf{M}))\mathbb{E}f(\mathbf{U}_t) + 16\eta^2\|\mathbf{M}\|\cdot\mathbb{E}\|SG(\mathbf{U}_t)\|_{\mathrm{F}}^2
\end{aligned}$$

One interesting aspect of our main result is that we actually show linear convergence under the presence of noise in gradient. This is true because for the second-order (η^2) term above, we can roughly see from Eq.(4) that $\|SG(\mathbf{U})\|_{\mathrm{F}}^2 \leq h(\mathbf{U}) \cdot f(\mathbf{U})$, where $h(\mathbf{U})$ is a factor depends on $\mathbf{U}$ and always bounded. That is, $SG(\mathbf{U})$ enjoys self-bounded property — $\|SG(\mathbf{U})\|_{\mathrm{F}}^2$ will goes to zero, as objective function $f(\mathbf{U})$ goes to zero. Therefore, by choosing learning rate η appropriately small, we can have the first-order term always dominate the second-order term, which establish the linear convergence.

Now, the only remaining issue is to prove that "$\mathbf{U}_0, \ldots, \mathbf{U}_t$ always stay inside local region $\mathcal{D}$". In reality, we can only prove this statement with high probability due to the stochastic nature of the update. This is also the most challenging part in our proof, which makes our analysis different from standard convex analysis, and uniquely required due to non-convex setting.

Our key theorem is presented as follows:

Theorem 4.5. *Let* $f(\mathbf{U}) = \left\|\mathbf{U}\mathbf{U}^\top - \mathbf{M}\right\|_F^2$ *and* $g_i(\mathbf{U}) = \left\|\mathbf{e}_i^\top \mathbf{U}\right\|^2$. *Suppose initial* $\mathbf{U}_0$ *satisfying:*

$$f(\mathbf{U}_0) \leq \left(\frac{\sigma_{\min}(\mathbf{M})}{20}\right)^2, \qquad \max_i g_i(\mathbf{U}_0) \leq \frac{10\mu k\kappa(\mathbf{M})^2}{d}\|\mathbf{M}\|$$

Then, there exist some absolute constant c *such that for any learning rate* $\eta < \frac{c}{\mu dk\kappa^3(\mathbf{M})\|\mathbf{M}\|\log d}$, *with at least* $1 - \frac{T}{d^{10}}$ *probability, we will have for all* $t \leq T$ *that:*

$$f(\mathbf{U}_t) \leq (1 - \frac{1}{2}\eta\sigma_{\min}(\mathbf{M}))^t \left(\frac{\sigma_{\min}(\mathbf{M})}{10}\right)^2, \qquad \max_i g_i(\mathbf{U}_t) \leq \frac{20\mu k\kappa(\mathbf{M})^2}{d}\|\mathbf{M}\| \tag{5}$$

Note function $\max_i g_i(\mathbf{U})$ indicates the incoherence of matrix $\mathbf{U}$. Theorem 4.5 guarantees if inital $\mathbf{U}_0$ is in the local region which is incoherent and $\mathbf{U}_0\mathbf{U}_0^\top$ is close to $\mathbf{M}$, then with high probability for all steps $t \leq T$, $\mathbf{U}_t$ will always stay in a slightly relaxed local region, and $f(\mathbf{U}_t)$ has linear convergence.

It is not hard to show that all saddle points of $f(\mathbf{U})$ satisfy $\sigma_k(\mathbf{U}) = 0$, and all local minima are global minima. Since $\mathbf{U}_0, \ldots, \mathbf{U}_t$ automatically stay in region $f(\mathbf{U}) \leq (\frac{\sigma_{\min}(\mathbf{M})}{10})^2$ with high probability, we know $\mathbf{U}_t$ also stay away from all saddle points. The claim that $\mathbf{U}_0, \ldots, \mathbf{U}_t$ stays incoherent is essential to better control the variance and probability 1 bound of $SG(\mathbf{U}_t)$, so that we can have large step size and tight convergence rate.

The major challenging in proving Theorem 4.5 is to both prove $\mathbf{U}_t$ stays in the local region, and achieve good sample complexity and running time (linear in d) in the same time. This also requires the learning rate η in Algorithm 1 to be relatively large. Let the event $\mathfrak{E}_t$ denote the good event where $\mathbf{U}_0, \ldots, \mathbf{U}_t$ satisfies Eq.(5). Theorem 4.5 is claiming that $P(\mathfrak{E}_T)$ is large. The essential steps in the proof is contructing two supermartingles related to $f(\mathbf{U}_t)1_{\mathfrak{E}_t}$ and $g_i(\mathbf{U}_t)1_{\mathfrak{E}_t}$ (where $1_{(\cdot)}$ denote indicator function), and use Bernstein inequalty to show the concentration of supermartingales. The $1_{\mathfrak{E}_t}$term allow us the claim all previous $\mathbf{U}_0, \ldots, \mathbf{U}_t$ have all desired properties inside local region.

Finally, we see Theorem 3.1 as a immediate corollary of Theorem 4.5.

5 Conclusion

In this paper, we presented the first provable, efficient online algorithm for matrix completion, based on nonconvex SGD. In addition to the online setting, our results are also competitive with state of the art results in the offline setting. We obtain our results by introducing a general framework that helps us show how SGD updates self-regulate to stay away from saddle points. We hope our paper and results help generate interest in online matrix completion, and our techniques and framework prompt tighter analysis for other nonconvex problems.

References

[1] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. *arXiv preprint arXiv:1503.00778*, 2015.

[2] Matthew Brand. Fast online SVD revisions for lightweight recommender systems. In *SDM*, pages 37–46. SIAM, 2003.

[3] Emmanuel J Candes, Yonina C Eldar, Thomas Strohmer, and Vladislav Voroninski. Phase retrieval via matrix completion. *SIAM Review*, 57(2):225–251, 2015.

[4] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, December 2009.

[5] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

[6] Christopher De Sa, Kunle Olukotun, and Christopher Ré. Global convergence of stochastic gradient descent for some non-convex matrix problems. *arXiv preprint arXiv:1411.1134*, 2014.

[7] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. *arXiv preprint arXiv:1503.02101*, 2015.

[8] Moritz Hardt. Understanding alternating minimization for matrix completion. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 651–660. IEEE, 2014.

[9] Moritz Hardt, Raghu Meka, Prasad Raghavendra, and Benjamin Weitz. Computational limits for matrix completion. In *COLT*, pages 703–725, 2014.

[10] Prateek Jain, Chi Jin, Sham M Kakade, and Praneeth Netrapalli. Computing matrix squareroot via non convex local search. *arXiv preprint arXiv:1507.05854*, 2015.

[11] Prateek Jain, Chi Jin, Sham M Kakade, Praneeth Netrapalli, and Aaron Sidford. Matching matrix bernstein with little memory: Near-optimal finite sample guarantees for oja's algorithm. *arXiv preprint arXiv:1602.06929*, 2016.

[12] Prateek Jain and Praneeth Netrapalli. Fast exact matrix completion with finite samples. *arXiv preprint arXiv:1411.1087*, 2014.

[13] Hui Ji, Chaoqiang Liu, Zuowei Shen, and Yuhong Xu. Robust video denoising using low rank matrix completion. In *CVPR*, pages 1791–1798. Citeseer, 2010.

[14] Raghunandan Hulikal Keshavan. *Efficient algorithms for collaborative filtering*. PhD thesis, STANFORD UNIVERSITY, 2012.

[15] Yehuda Koren. The BellKor solution to the Netflix grand prize. *Netflix prize documentation*, 81:1–10, 2009.

[16] Akshay Krishnamurthy and Aarti Singh. Low-rank matrix and tensor completion via adaptive sampling. In *Advances in Neural Information Processing Systems*, pages 836–844, 2013.

[17] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *University of California, Berkeley*, 1050:16, 2016.

[18] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan 2003.

[19] Xin Luo, Yunni Xia, and Qingsheng Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271–280, 2012.

[20] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[21] Ioannis Panageas and Georgios Piliouras. Gradient descent converges to minimizers: The case of non-isolated critical points. *arXiv preprint arXiv:1605.00405*, 2016.

[22] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(Dec):3413–3430, 2011.

[23] Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

[24] Ruoyu Sun and Zhi-Quan Luo. Guaranteed matrix completion via nonconvex factorization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 270–289. IEEE, 2015.

[25] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

[26] Se-Young Yun, Marc Lelarge, and Alexandre Proutiere. Streaming, memory limited matrix completion with noise. *arXiv preprint arXiv:1504.03156*, 2015.