Simple and practical algorithms for ℓ_p-norm low-rank approximation

Anastasios Kyrillidis
IBM T.J. Watson Research Center
Rice University
anastasios@rice.edu

Abstract

We propose practical algorithms for entrywise ℓ_p-norm low-rank approximation, for $p = 1$ or $p = \infty$. The proposed framework, which is non-convex and gradient-based, is easy to implement and typically attains better approximations, faster, than state of the art.

From a theoretical standpoint, we show that the proposed scheme can attain $(1 + \varepsilon)$-OPT approximations. Our algorithms are not hyperparameter-free: they achieve the desiderata only assuming algorithm's hyperparameters are known *apriori*—or are at least approximable. *I.e.*, our theory indicates what problem quantities need to be known, in order to get a good solution within polynomial time, and does not contradict to recent inapproximabilty results, as in [46].

1 INTRODUCTION

We focus on the following optimization problem:

$$\min_{U\in\mathbb{R}^{m\times r},V\in\mathbb{R}^{n\times r}} |M - UV^\top|_p, \quad p \in \{1, \infty\}. \qquad (1)$$

Here, $M \in \mathbb{R}^{m\times n}$ is a given input matrix of arbitrary rank, $r \leq \{m, n\}$ is the target rank, (U, V) represent the variables such that $\texttt{rank}(UV^\top) \leq r$, and $|\cdot|_p$ denotes the p-th, *entrywise*, matrix norm. In words, (1) is described as "finding the factors of the best rank-r approximation of M, with respect to the ℓ_p-norm". We denote such optimal factors $U^\star$ and $V^\star$, and their product $X^\star = U^\star V^{\star\top}$. We focus on $p \in \{1, \infty\}$, since these instances are the most common found in practice, beyond the classic $p = 2$ (Frobenius) norm; we will use the terms "Frobenius" and "ℓ_2" norm, interchangeably.

There are numerous applications where ℓ_1- / ℓ_∞-norm low rank approximations are useful in practice. First, the ℓ_1-norm is more robust than the ℓ_2-norm, and is suited in problem settings where Gaussian assumptions for noise models may not apply. ℓ_1-norm low rank applications include robust PCA applications [56, 6, 31, 32, 24, 57], computer vision tasks such as background subtraction and motion detection [52, 1, 38], detection of brain activation patterns [44], and detection of anomalous behavior in dynamic networks [44]. [1]

For the ℓ_∞-norm version of (1), the problem cases are only a few. [43] considers the special case of $m = n$ and $r = \min\{m, n\} - 1$ as the problem of distance to robust non-singularity. [22, 23] use the notion of ℓ_∞-norm low rank approximation for the maximal-volume concept in approximation, as well as for the skeleton approximation of a matrix. Finally, [17] identifies that (1) with $p = \infty$ can be used for the recovery of a low-rank matrix from a quantized M.

Despite the utility of (1), its solution is not straightforward. While (1) with ℓ_2-norm has a closed-form solution via the Singular Value Decomposition (SVD), the same does not hold for $p \in \{1, \infty\}$. Additionally, it has been proved that actually finding the exact solution to (1) can be exponentially complex:

[1] Closely related to the ℓ_1-norm low-rank approximation is the problem of ℓ_1-norm subspace recovery [30]. Briefly, it is well-known that, for $p = 2$ in (1), the SVD solution is also the solution to the dual problem: $U^\star = \text{argmax}_{U\in\mathbb{R}^{m\times r}} |U^\top M|_2$, subject to $U^\top U = I$. $V^\star$ is then set as $V^\star = U^{\star\top} M$; this can be easily proved due to the orthogonality of $U^\star$ [20]. Motivated by this dual formulation, ℓ_1-norm subspace recovery is defined as

$$U^\star = \underset{U\in\mathbb{R}^{m\times r}}{\text{argmax}} |U^\top M|_1, \text{ subject to } U^\top U = I.$$

Algorithmic solutions to this criterion are usually greedy [30], even combinatorial [36, 37]. However, in this case, $U^\star$ does not necessarily resemble with that of (1) with $p = 1$ (up to orthogonal rotations).

[19] show that ℓ_1-norm low rank matrix approximation is NP-hard, even for $r = 1$; further, under the exponential time hypothesis for 3SAT problems, [46] provide a $\left(1 + \frac{1}{\log^{1+\gamma}(\max\{m,n\})}\right)$-inapproximability result for some hard instances M, where $\gamma > 0$ is an arbitrary small constant. [17] proves the NP-completeness of (1) for $p = \infty$, using a reduction from not-all-equal-3SAT.

The above restrict research to only approximations of (1). To the best of our knowledge only the works in [9, 46] present polynomial and provably good approximation schemes: [46] focuses mostly on the case of ℓ_1-norm, and proves the existence of a $O(\log(\min\{m,n\}) \cdot \texttt{poly}(r))$-approximation scheme with $O(\texttt{nnz}(M) + (m+n)\texttt{poly}(r))$ computational complexity. [9] extends the ideas in [46] for ℓ_p-norms, where $p \in [1,\infty]$: there, the authors describe a $\texttt{poly}(r)$-approximation with $O\left(\texttt{poly}(m,n)(r\log\max\{m,n\})^r\right)$ computational complexity. Both approaches are based on numerical linear algebra and sketching techniques.

Apart from the above provable schemes, there are numerous heuristics proposed for (1), with no rigorous approximation guarantees. Starting with ℓ_1-norm, [38] propose a coordinate descent algorithm for (1), where a sequence of alternating scalar minimization sub-problems are solved using a (weighted) median filter; see also [29]. Previously to that work, [26, 27] follow a similar approach, where each sub-problem is solved using linear or quadratic programming[2]. Inspired by [55], [13] propose a ℓ_1-norm version of the Wiberg method; the resulting algorithm involves several matrix-matrix multiplications (even of size greater than the input matrix), and the solution of linear programming criteria, per iteration. Cabral et al. use Augmented Lagrange Multipliers (ALM) method and handle the weighted ℓ_1-norm low rank approximation problem in [5]; however, no non-asymptotic convergence guarantees are provided. We note that most of the above heuristics are designed to handle *missing data* in M or the case of *weighted* factorization; we plan to consider such cases for our future research directions. For the ℓ_∞-norm case, we mention the recent work of Gillis et al. [17] that proposes a block coordinate descent method that operates in an alternating minimization fashion over subsets of variables in (1).

Our approach and main contributions: Inspired by the recent advances on smooth non-convex optimization for matrix factorization [47, 58, 51, 4, 42, 16, 40, 41, 35, 34, 50, 54, 15, 33], we study the application of alternating gradient descent in (1). Despite its NP-hardness, this paper follows a more optimistic course and works towards deciphering the components/quantities that, if known a priori, could lead to a $(1+\varepsilon)$-approximation for (1).

Our approach is based on two techniques from optimization theory: (i) the smoothing technique for non-smooth convex optimization by Nesterov [39, 12] (Section 4), and (ii) the recent theoretical results on finding the global minimum of matrix factorization problems using non-convex smooth methods (Section 3); see also references above. Our theory relies on provably bounding the objective function in ℓ_1- or ℓ_∞-norm by its smoothing counterpart (Sections 4), using the provable performance of the non-convex algorithm (Section 3), and properly setting up the input parameters (Section 5). Our guarantees assume that we can at least approximate the optimal function value of (1), and that the optimal low-rank solution of the smoothed problem is well-conditioned; the latter assumption is required for a good initialization to be easily found. The above are summarized as:

- Under assumptions, we provide a polynomial approximation algorithm for $p = \{1,\infty\}$ in (1) that achieves a $(1+\epsilon)$-approximation guarantee.
- We experimentally show that our scheme outperforms in practice state-of-the-art approaches.

There are several questions that remain open and need further investigation. In Section 7, we discuss what are the advantages and disadvantages of our approach and point to possible future research directions.

2 NOTATION AND ASSUMPTIONS

Notation. For matrices $X, Y \in \mathbb{R}^{m\times n}$, $\langle X, Y\rangle = \textsc{Tr}\left(X^\top Y\right)$ represents their inner product and $X \odot Y$ their Hadamard product. We represent matrix norms as follows: $|X|_2 = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |X_{ij}|^2}$ denotes the Frobenius (or ℓ_2-) norm, $|X|_1 = \sum_{i=1}^m \sum_{j=1}^n |X_{ij}|$ denotes the entrywise ℓ_1-norm, and $|X|_\infty = \max_{i,j} |X_{ij}|$ denotes the entrywise ℓ_∞-norm. For the spectral norm, we use $\sigma_1(X)$; this also denotes the largest singular value of X. For vectors, we use $\|x\|_2$ to denote its Euclidean ℓ_2-norm. For a differentiable function $f(X)$ with $X = UV^\top$, the gradient of f w.r.t. U and V is $\nabla f(X)V$ and $\nabla f(X)^\top U$, respectively.

Assumptions. For our discussion, we will need two well-known notions of convex analysis: (restricted) *strong* convexity and (restricted) Lipschitz gradient continuity.

Definition 2.1. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be a convex differentiable function. Then, f is (resp. restricted) gradient Lip-*

[2] In [26, 27], there are some convergence guarantees for the alternating optimization scheme; however, there are no results w.r.t. whether we converge to a saddle point or local minimum, nor results on the convergence rate.

schitz continuous with parameter L if $\forall X, Y \in \mathbb{R}^{m\times n}$ (resp. $\forall X, Y \in \mathbb{R}^{m\times n}$ that are at most rank-r):

$$f(Y) \leq f(X)+\langle \nabla f(X), Y-X\rangle+\tfrac{L}{2}\left|Y-X\right|_2^2. \quad (2)$$

Definition 2.2. *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be convex and differentiable. Then, f is (resp. restricted) μ-strongly convex if $\forall X, Y \in \mathbb{R}^{m\times n}$ (resp. $\forall X, Y \in \mathbb{R}^{m\times n}$ that are at most rank-r):*

$$f(Y) \geq f(X)+\langle \nabla f(X), Y-X\rangle+\tfrac{\mu}{2}\left|Y-X\right|_2^2. \quad (3)$$

3 BFGD FOR SMOOTH OBJECTIVES

Let us first succinctly describe the Bi-Factored Gradient Descent (BFGD) algorithm [41], upon which our proposal is based. BFGD is a non-convex gradient descent scheme for *smooth* problems such as:

$$\min_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}} f(UV^\top), \quad (4)$$

where f is assumed to be convex, *differentiable*, and at least have Lipschitz continuous gradients. Observe that while f is convex w.r.t. to any input $\in \mathbb{R}^{m\times n}$, motions over U and V jointly lead to non-convex optimization. Such approaches have a long history and different variants have been proposed for (4).

For the rest of this section, we denote $X = UV^\top$ as the result of the factorization. Also, let $\widehat{X}^\star$ be the optimal point of (4): if $\texttt{rank}(\widehat{X}^\star) = r$, then $\widehat{X}^\star = \widehat{X}_r^\star$; otherwise, denote its best rank-r approximation (w.r.t. the ℓ_2-norm) as $\widehat{X}_r^\star$.

Algorithm 1 Bi-factored gradient descent (BFGD)

1: **Input:** r, T, γ (*e.g.*, $\frac{1}{4}$), $C > 0$ (*e.g.*, $C = 1$), $\widehat{L}$.
2: Compute $X_0 := {}^1\!/_{\widehat{L}} \cdot (-\nabla f(0_{m\times n}))$.
3: Set $U_0 \in \mathbb{R}^{m\times r}$, $V_0 \in \mathbb{R}^{n\times r}$ s.t. $X_0 = U_0V_0^\top$, via SVD.
4: **for** $i = 0$ to $T-1$ **do**
5: Set η such that: $\eta \leq \frac{C}{15\widehat{L}\left|\begin{bmatrix}U_i & V_i\end{bmatrix}^\top\right|_2^2+3\left|\nabla f(U_iV_i^\top)\right|_2}$.
6: • If f satisfies Definition 2.1: **Rule 1**

$$\begin{bmatrix}U_{i+1}\\ V_{i+1}\end{bmatrix} = \begin{bmatrix}U_i\\ V_i\end{bmatrix} - \eta\begin{bmatrix}\nabla f(U_iV_i^\top)\cdot V_i\\ \nabla f(U_iV_i^\top)^\top \cdot U_i\end{bmatrix}$$

• If f satisfies Definitions 2.1-2.2: **Rule 2**

$$\begin{bmatrix}U_{i+1}\\ V_{i+1}\end{bmatrix} = \begin{bmatrix}U_i\\ V_i\end{bmatrix} - \eta\begin{bmatrix}\nabla f(U_iV_i^\top)V_i+\gamma U_i(U_i^\top U_i - V_i^\top V_i)\\ \nabla f(U_iV_i^\top)^\top U_i-\gamma V_i(U_i^\top U_i - V_i^\top V_i)\end{bmatrix}$$

7: **end for**
8: **Output:** $\widehat{X} = U_TV_T^\top$.

The pseudocode for BFGD is provided in Algorithm 1 and obeys the following motions: (i) given a proper initialization $X_0 = U_0V_0^\top$, and (ii) a proper step size η,[3] BFGD applies iteratively **Rule 1** if f satisfies only Definition 2.1, or **Rule 2** if f also satisfies Definition 2.2. The algorithm assumes an approximation of L—say $\widehat{L}$ and see [4]—and a good initialization point (U_0, V_0). For a more complete discussion of initialization (U_0, V_0), we refer the reader to [4, 41]; we briefly discuss this issue in Section 5.

An important issue in optimizing f over (U, V) is the existence of non-unique possible factorizations for a given X. We need a notion of distance to the low-rank solution $\widehat{X}_r^\star$ over the factors. Similar to [51, 41], we focus on the set of "equally-footed" factorizations:

$$\widehat{\mathcal{X}}_r^\star = \Big\{\left(\widehat{U}^\star, \widehat{V}^\star\right) : \widehat{U}^\star \in \mathbb{R}^{m\times r}, \widehat{V}^\star \in \mathbb{R}^{n\times r}, \widehat{U}^\star\widehat{V}^{\star\top} = \widehat{X}_r^\star, \\ \sigma_i(\widehat{U}^\star) = \sigma_i(\widehat{V}^\star) = \sigma_i(\widehat{X}_r^\star)^{1/2}, \forall i \in [r]\Big\}. \quad (5)$$

Given a pair (U, V), we define the distance to $\widehat{X}_r^\star$ as:

$$\textsc{Dist}\left(U, V; \widehat{X}_r^\star\right) = \min_{(\widehat{U}^\star,\widehat{V}^\star)\in\widehat{\mathcal{X}}_r^\star}\left|\begin{bmatrix}U\\ V\end{bmatrix} - \begin{bmatrix}\widehat{U}^\star\\ \widehat{V}^\star\end{bmatrix}\right|_2.$$

Algorithm 1 has local convergence guarantees, when f is μ-strongly convex and has L-Lipschitz continuous gradients, according to the following theorem:[4]

Theorem 3.1 (Theorem 4.4 in [41])**.** *Let $\kappa = L/\mu$. If the initial point $X_0 = U_0V_0^\top$, satisfies $\textsc{Dist}(U_0, V_0; X_r^\star) \leq \frac{\sqrt{2}\cdot\sigma_r(X_r^\star)^{1/2}}{10}$, then BFGD converges with rate $O(1/T)$:*

$$f(U_TV_T^\top) - f(\widehat{U}^\star\widehat{V}^{\star\top}) \leq \tfrac{10\cdot\textsc{Dist}(U_0,V_0;\widehat{X}_r^\star)^2}{\eta T}$$

4 CHARBONNIER APPROXIMATION AND THE logsumexp FUNCTION

A key assumption in BFGD is that f is at least once *differentiable* and has Lipschitz continuous gradients.Therefore, to connect BFGD with our original objective in (1), we will first approximate both the ℓ_1 and ℓ_∞ entrywise matrix norms by smooth functions that have derivatives at least in two degrees. For similar approaches in optimization where non-smooth functions are substituted by smooth ones, we refer to the seminal paper of Nesterov [39] and follow-up works [12, 28].

[3] In this work, we do not focus on the most efficient step size selections: *e.g.*, the step size considered in this work varies per iteration, and it is less efficient than a constant step size selection as in [4, 41]. However, in all cases, we could bound the varying step size with one that is constant.

[4] In this work, we will borrow only the sublinear rate results in [41], since that result alone is sufficient to lead to polynomial algorithms for (1). Using the linear convergence rate result in [41] is left for the extension of this work.

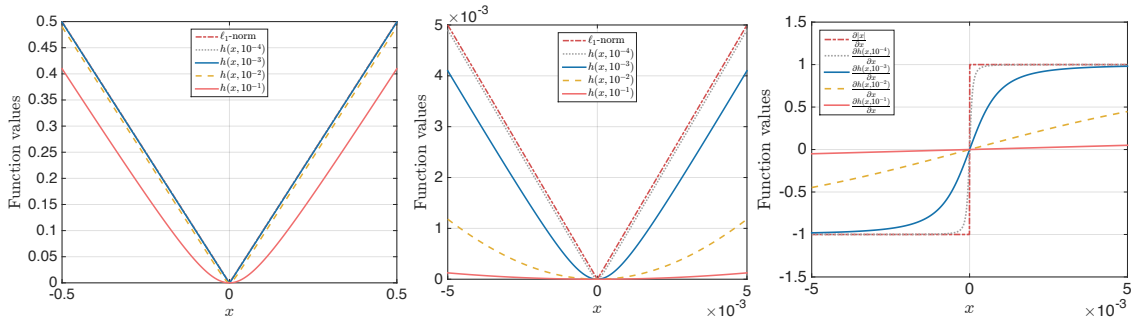

Figure 1: ℓ_1-norm and its Charbonnier smooth approximations. *Left and middle:* Function values vs. input variable. *Right:* Gradient approximation.

Approximating the entrywise ℓ_1-norm. For the approximation of the ℓ_1-norm, we will use the Charbonnier loss function [7, 3], parameterized as follows:

$$h(x,\tau) = \tau \cdot \left(\sqrt{\left(\frac{x}{\tau}\right)^2 + 1} - 1 \right). \tag{6}$$

To illustrate how a good approximation is (6) to the ℓ_1-norm, see Figure 1.

We now discuss about the matrix form of (6) and its properties. With a slight overload of notation, we define the matrix version of (6) as follows:

$$\begin{aligned} h(X,\tau) &= \sum_{i=1}^{m}\sum_{j=1}^{n} h(X_{ij},\tau) \\ &:= \tau \cdot \sum_{i=1}^{m}\sum_{j=1}^{n} \left(\sqrt{\left(\frac{X_{ij}}{\tau}\right)^2 + 1} - 1 \right). \end{aligned} \tag{7}$$

The distinction between scalar and matrix h will be apparent from the text. Gradient and Hessian information of h satisfy the following lemma; the proof is deferred to the supp. material:

Lemma 4.1. *For any $X \in \mathbb{R}^{m\times n}$:*

- $\nabla h(X,\tau) = \frac{1}{\tau} X \odot S \in \mathbb{R}^{m\times n}$, *where* $S \in \mathbb{R}^{m\times n}$ *and* $S_{ij} := \frac{2}{\sqrt{(X_{ij}/\tau)^2+1}}$,
- $\nabla^2 h(X,\tau) = \frac{1}{\tau} I \odot Q \in \mathbb{R}^{mn\times mn}$, *where* $Q \in \mathbb{R}^{mn\times mn}$ *and* $Q_{ij} := \frac{2}{\left((X_{ij}/\tau)^2+1\right)^{3/2}}$.

The above lead to the following lemma; the proof is provided in the supp. material:

Lemma 4.2. *Function h is a convex continuously differentiable function and it has Lipschitz continuous gradients with constant $\frac{2}{\tau}$. Moreover:*

$$|X|_1 - mn\tau \le h(X,\tau) \le |X|_1.$$

An alternative to the Charbonnier approximation is the Huber loss function with parameter τ [25]:

$$h(x,\tau) = \begin{cases} x^2/2\tau, & \text{if } |x| \le \tau \\ |x| - \tau/2, & \text{otherwise.} \end{cases} \tag{8}$$

Huber loss combines a ℓ_2-norm measure for small values of x and a ℓ_1-norm like measure for large x. Observe in (8) that it is only first-order differentiable; thus any computations involving second order derivatives cannot be applied. On the other hand, the Charbonnier loss function, which is also known as the "pseudo-Huber loss function", is a smooth approximation of the Huber loss that ensures that derivatives are continuous for all degrees. W.l.o.g., we focus on the Charbonnier function.

Approximating the entrywise ℓ_∞-norm. Following similar procedure for the entrywise matrix ℓ_∞-norm, we will use the `logsumexp` function, defined as follows:

$$\sigma(X,\tau) = \tau \cdot \log\left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} e^{X_{ij}/\tau} + e^{-X_{ij}/\tau}}{2mn} \right) \tag{9}$$

Define matrices $P, N \in \mathbb{R}^{m\times n}$ such that: $P_{ij} = e^{X_{ij}/\tau} + e^{-X_{ij}/\tau}$ and $N_{ij} = e^{X_{ij}/\tau} - e^{-X_{ij}/\tau}$. Then, the following lemma defines the gradient and Hessian information of the `logsumexp` function; see also the supp. material:

Lemma 4.3. *For any $X \in \mathbb{R}^{m\times n}$:*

- $\nabla\sigma(X,\tau) = \frac{1}{\mathrm{Tr}(\mathbb{1}\cdot P)} \cdot N \in \mathbb{R}^{m\times n}$,
- $\nabla^2\sigma(X,\tau) = \frac{\left(\mathtt{diag}(\mathtt{vec}(P)) - \frac{\mathtt{vec}(N)\mathtt{vec}(N)^\top}{\mathrm{Tr}(\mathbb{1}\cdot P)}\right)}{\tau\cdot\mathrm{Tr}(\mathbb{1}\cdot P)} \in \mathbb{R}^{mn\times mn}$

where $\mathtt{diag}(\cdot) : \mathbb{R}^{mn} \to \mathbb{R}^{mn\times mn}$ *turns the vector input to a diagonal matrix output,* $\mathtt{vec}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ *turns a matrix to a vector by "stacking" its columns, and $\mathbb{1}$ denotes the all-ones matrix.*

Similar to the Charbonnier approximation, we get the following lemma; the proof is in the supp. material:

Lemma 4.4. *The* `logsumexp` *function σ is a convex continuously differentiable function and it has Lipschitz continuous gradients with constant $\frac{1}{\tau}$. Moreover:*

$$|X|_\infty - \tau \log(2mn) \leq \sigma(X, \tau) \leq |X|_\infty.$$

5 AN APPROXIMATE SOLVER FOR ℓ_p-NORM LOW RANK APPROXIMATION

The proposed schemes are provided in Algorithms 2-3, and are based on Algorithm 1 as a sub-solver. In order to hope for a good initialization, we consider the smooth versions of (1), as described in Section 4, with the added twist that we regularize further the objective with a strongly convex component. *I.e.*, we approximate (1) for $p = 1$ with:

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}} h(M - UV^\top, \tau) + \frac{\lambda}{2}|UV^\top|_2^2, \quad (10)$$

and the case $p = \infty$ with

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}} \sigma(M - UV^\top, \tau) + \frac{\lambda}{2}|UV^\top|_2^2. \quad (11)$$

This modification asserts that both (10)-(11) are strongly convex w.r.t. X with parameter λ; see also the proof of Lemma 4.2. Observe that the smaller the λ parameter is, the less the "drift" from the original problem. We remind that the optimal factors of (1) are $U^\star$ and $V^\star$, and their product is denoted as $X^\star = U^\star V^{\star\top}$.

Algorithm 2 ℓ_1-norm low rank approximation solver

1: **Parameters:** r, OPT, values of $|X^\star|_2^2$ and $\sigma_r(\widehat{X}^\star)$, $\varepsilon > 0$.
2: Set $\tau = \frac{\varepsilon \cdot \mathrm{OPT}}{3mn}$.
3: Set function $T = O\left(\frac{\sigma_r(\widehat{X}_r^\star)}{\varepsilon \mathrm{OPT}}\right)$.
4: Set $\lambda = \frac{2\varepsilon \cdot \mathrm{OPT}}{3|X^\star|_2^2}$
5: Compute $\widehat{L} = (\frac{1}{\tau} + \lambda)$.
6: Set $f(UV^\top) := h(M - UV^\top, \tau) + \frac{\lambda}{2}|UV^\top|_2^2$.
7: Run Algorithm 1 $(U_T,\ V_T) = \mathtt{BFGD}(r, T, \frac{1}{4}, 1, \widehat{L})$.

Let us first focus on the case of ℓ_1-norm and Algorithm 2. The following theorem states that, under proper configuration of algorithm's hyperparameters, one can achieve $(1+\varepsilon)$-OPT approximation guarantee.

Theorem 5.1. *Let $\widehat{X} = U_T V_T^\top \in \mathbb{R}^{m \times n}$ be the solution of Algorithm 2. Let the optimal function value of* (1) *for $p = 1$ be denoted as* $\mathrm{OPT} := \min_{U,V} |M - UV^\top|_1$ *and assumed known, or at least be approximable. Also, assume we know $\sigma_r(\widehat{X}^\star)$ and $|X^\star|_2^2$. For user defined parameter $\varepsilon > 0$ and setting the Charbonnier parameter $\tau = \frac{\varepsilon \cdot \mathrm{OPT}}{3mn}$, and the strong convexity parameter as $\lambda = \frac{2\varepsilon \cdot \mathrm{OPT}}{3|X^\star|_2^2}$, the pair (U_T, V_T) of Algorithm 2 satisfies:*

$$|M - U_T V_T^\top|_1 \leq (1 + \varepsilon) \cdot \mathrm{OPT},$$

after $T = O\left(\sigma_r(\widehat{X}_r^\star)\left(\frac{mn}{(\varepsilon \mathrm{OPT})^2} + \frac{1}{\|X^\star\|_2^2}\right)\right)$ iterations.

The proof is provided in the appendix. In the case where OPT is only approximable, straightforward modifications lead to similar performance (where higher number of iterations required).

Analytical complexity: Let us denote the time to compute $\nabla f(\cdot)$ as t_{grad}. The initialization complexity of Algorithm 1, as well as its per iteration complexity, is $O(t_{\mathrm{grad}} + mnr)$, where the last term is due to either low-rank SVD calculation or matrix-matrix multiplication. Running Algorithm 1 for $T = O\left(\frac{\sigma_r(\widehat{X}_r^\star)}{\varepsilon \mathrm{OPT}}\right)$ iterations leads to an overall $O\left(\frac{\sigma_r(\widehat{X}_r^\star)}{\varepsilon \mathrm{OPT}} \cdot (t_{\mathrm{grad}} + mnr)\right)$ time complexity.

Similarly for the case of $p = \infty$, we use the `logsumexp` function in Algorithm 3 to smooth the objective, and we obtain the following guarantees:

Algorithm 3 ℓ_∞-norm low rank approximation solver

1: **Parameters:** r, OPT, values of $|X^\star|_2^2$ and $\sigma_r(\widehat{X}^\star)$, $\varepsilon > 0$.
2: Set $\tau = \frac{\varepsilon \cdot \mathrm{OPT}}{3\log(2mn)}$.
3: Set function $T = O\left(\frac{\sigma_r(\widehat{X}_r^\star)}{\varepsilon \mathrm{OPT}}\right)$.
4: Set $\lambda = \frac{2\varepsilon \cdot \mathrm{OPT}}{3|X^\star|_2^2}$
5: Compute $\widehat{L} = (\frac{1}{\tau} + \lambda)$.
6: Set $f(UV^\top) := \sigma(M - UV^\top, \tau) + \frac{\lambda}{2}|UV^\top|_2^2$.
7: Run Algorithm 1 $(U_T,\ V_T) = \mathtt{BFGD}(r, T, \frac{1}{4}, 1, \widehat{L})$.

Corollary 5.2. *Let $\widehat{X} = U_T V_T^\top \in \mathbb{R}^{m \times n}$ be the solution of Algorithm 2. Let the optimal function value of* (1) *for $p = \infty$ be denoted as* $\mathrm{OPT} := \min_{U,V} |M - UV^\top|_\infty$, *and assumed known, or be at least approximable. Also, assume we know $\sigma_r(\widehat{X}^\star)$ and $|X^\star|_2^2$. For user defined approximation parameter $\varepsilon > 0$ and setting the* `logsumexp` *parameter $\tau = \frac{\varepsilon \cdot \mathrm{OPT}}{3\log(2mn)}$, and the strong convexity parameter as $\lambda = \frac{2\varepsilon \cdot \mathrm{OPT}}{3|X^\star|_2^2}$, the pair (U_T, V_T) of Algorithm 2 satisfies:*

$$|M - U_T V_T^\top|_\infty \leq (1 + \varepsilon) \cdot \mathrm{OPT},$$

after $T = O\left(\sigma_r(\widehat{X}_r^\star)\left(\frac{\log(mn)}{(\varepsilon \mathrm{OPT})^2} + \frac{1}{\|X^\star\|_2^2}\right)\right)$ iterations.

Similar analytical complexity can be derived for Algorithm 3 and is omitted due to lack of space.

Results of similar flavor (and under similar assumptions) can be found in [28] for the problem of maximum flow. There, the authors consider non-Euclidean gradient descent algorithms for the minimization of ℓ_∞-norm over vectors, where the gradient step takes into consideration the geometry of the non-smooth objective with the use of *sharp* operators. We applied a similar approach for both $p \in \{1, \infty\}$ in our setting; however, the empirical performance was prohibitive to consider a similar approach here (despite the fact that one can still achieve $(1+\varepsilon)$-optimal approximation guarantees).

Some remarks regarding the above results.

Remark 1. *Both algorithms require the knowledge of three quantities:* OPT, $|X^\star|_2^2$ *and* $\sigma_r(\widehat{X}^\star)$. *While finding these values could be as difficult as the original problem* (1)*, these values do not need to be known exactly: in particular, the algorithms imply that "for sufficiently small* τ *and* λ *parameters, and for a sufficiently large number of iterations* T*, we can find a good approximation".*

Remark 2. *While finding the exact value of* OPT *is difficult, there are problem cases where this value could be easily upper bounded. E.g., consider the problem of low-rank matrix approximation from quantization, as noted in [17]: there, we know from structure that* $|M - X^\star|_\infty = \mathrm{OPT} \leq 0.5$.

Remark 3. *Finding a good initialization is a key assumption for Theorem 5.1 and its corollary. Such assumptions are made also in other non-convex matrix factorization results; see [47, 58, 51, 4, 42, 16, 40, 41, 35, 34, 54, 15]. From [41], it is known that we can easily compute such an initialization as the best rank-*r *approximation of* M *w.r.t. the* ℓ_2*-norm, via SVD. In particular, such an initialization satisfies* $\textsc{Dist}(U_0, V_0; \widehat{X}^\star) \leq \frac{\sqrt{2}\cdot\sigma_r(\widehat{X}^\star)^{1/2}}{10\sqrt{\kappa}}$*, as long as* f *is strongly convex with condition number* $\kappa \leq 1 + \frac{\sigma_r(\widehat{X}^\star)^2}{4608\cdot|\widehat{X}_r^\star|_2^2}$*. While this condition is not easily met in theory (i.e., since* $\kappa = \frac{\frac{1}{\tau}+\lambda}{\lambda}$*, this means that* τ *should be large enough compared to* λ*), our experiments show that such an initialization performs well.*

Remark 4. *As a continuation of the above remark, the reason we use the regularizer* $\frac{\lambda}{2}|UV^\top|_2^2$ *is to turn the smooth approximations into strongly convex functions (and thus borrow results for initialization). In practice, the proposed schemes work as well without the addition of the regularizer; and thus, knowing a priori the quantity* $|X^\star|_2^2$ *is not necessary in practice.*

Remark 5. *The approach we follow somewhat resembles with the approach proposed in [27]. There, the authors consider* (1) *for* $p = 1$ *and propose an alternating minimization scheme. Despite the similarities, there are differences with our approach: among which, we perform a single gradient descent step on* U *and* V *per iteration, for a smoothed version of* (1)*, instead of minimizing a quadratic programming formulation per each column of* U *and* V*. On the contrary, [27] handles empirically missing values and weighted low-rank matrix factorization cases; we leave this direction for future research.*

6 EXPERIMENTS

Our experiments include synthesized applications, in order to highlight the empirical performance of the proposed framework. We compare the algorithms in Section 5 (i) with the algorithms for ℓ_p-low rank approximation in [9], and (ii) with the recent heuristic in [17] for ℓ_∞-low rank approximation.

Similarly to [9, 17] and in order to guarantee fair comparison, we follow in practice the "folklore" advice for getting an initial estimate for the ℓ_p-norm problem in (1) by beginning with the optimum ℓ_2-norm solution (*i.e.*, with the low-rank SVD solution).

6.1 ℓ_1-norm approximation

We perform experiments on both real and synthetic datasets. At first, we generate data according to the recent ICML paper [9]: We use 20×30 random matrices M, where each entry is a uniformly random value in $[0, 1]$. Such constructions lead to full rank matrices with high-probability. We also construct matrices M of the same size with $\{\pm 1\}$ entries, each selected with 0.5 probability. For real datasets, similar to [9], we use the FIDAP dataset[5] and a word frequency dataset from UC Irvine[6]. The FIDAP matrix M is 27×27 with 279 real asymmetric non-zero entries. The word frequency matrix M is 3430×6906 with $353,160$ non-zero entries.

For the synthesized datasets, we perform 10 Monte Carlo instantiations and take the median error reported. For all datasets, we are interested in computing the best rank-r approximation of each M above, w.r.t. the ℓ_1-norm and for $r \in \{1, \ldots, 10\}$. To compare with [9], we use their suggestion and run a simplified version of Algorithm 2 in [9], where we repeatedly sample r columns, uniformly at random. We then run the ℓ_p-projection (see Lemma 1 in [9]) on each sampled set and finally select the solution with the smallest ℓ_p-error. For a fair contrast between the algorithms, we first run our algorithm and measure the required time; for approximately the same amount of time, we run [9].[7] To perform the ℓ_p-projection, we use

[5] `http://math.nist.gov/MatrixMarket/data/SPARSKIT/fidap/fidap005.html`

[6] `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

[7] In all our experiments, we make sure the algorithm in [9] runs at least the same time with our scheme.

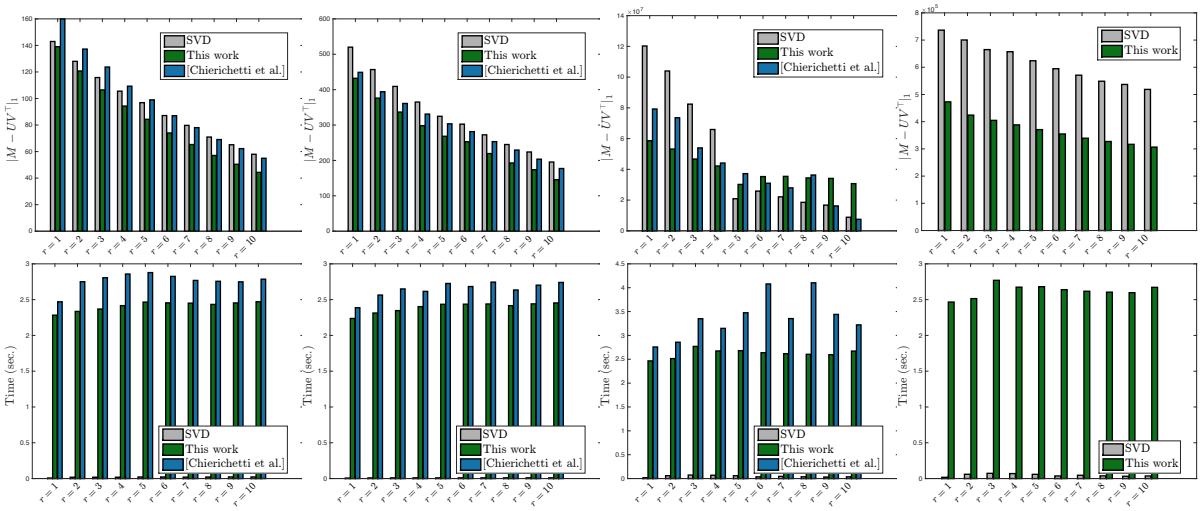

Figure 2: Top row: function value performance $|M - UV^\top|_1$; Bottom row: corresponding execution time. In all settings, we set problem (1) for $r = \{1, \ldots, 10\}$. First column: $M \in [0, 1]^{20\times 30}$ where each entry is randomly and independently generated. Second column: $M \in \{-1, 1\}^{20\times 30}$ where each entry is randomly and independently generated. Third column: $M \in \mathbb{R}^{27\times 27}$ is the FIDAP matrix. Fourth column: $M \in \mathbb{R}^{3430\times 6906}$ is the word-frequency matrix. In the latter case, the sub-solver for ℓ_p-projection was not able to complete the task, and thus the algorithm in [9] is omitted.

CVX package [14].[8]

In our algorithm, we set $\tau = \lambda = 10^{-3}$, and the maximum number of iterations as $T = 4 \cdot 10^4$. As mentioned above, we use the SVD initialization, and the step size is set according to Algorithm 1.

The results are provided in Figure 2. Some remarks: (i) for the synthetic cases (two leftmost columns), we observe that our approach attains a better objective function, faster, compared to [9]. Both our work and [9] is much slower than plain SVD; however, the latter gives a worse solution. (ii) for the real case (two rightmost columns), our approach is overall better in terms of objective function values; however, this is not universal; there are cases where [9] (or even SVD) gets to a better result within the same time, especially when r increases. For the large matrix case, [9] with CVX do not scale well; thus omitted.

6.2 ℓ_∞-norm approximation

In this experiment, we follow the experimental setting in [17]. We generate matrices $M \in \mathbb{R}^{100\times 75}$ as follows: We generate $\widetilde{M} = UV^\top$ where $U \in \mathbb{R}^{100\times r}$ and $V \in \mathbb{R}^{75\times r}$. Each U and V is generated i.i.d. from $N(0, 1)$. Given $\widetilde{M}$, we compute the *rounded* version of $\widetilde{M}$ such as $M = \texttt{round}(\widetilde{M})$. This procedure guarantees that, given M, there is a low-rank matrix $\widetilde{M}$ that satisfies $|M - \widetilde{M}|_\infty \leq 0.5$ (since this is an hard problem, this construction gives an idea how far/close we are to a good solution).

We repeat the above procedure for $r = \{1, \ldots, 10\}$ and for 10 Monte Carlo instances. We report the minimum, mean and median values of the objective function attained and the time required. We compare our algorithms with plain SVD and the heuristics in [17].

The results are reported in Table 1. Our findings show that both our work and the algorithm in [17] perform much better (in terms of quality of solution) than plain SVD (the full set of results can be found in the appendix). Further, the algorithm in [17] has time comparable to the implementation of SVD in Matlab, while our proposed algorithm is much slower; accelerating our proposed algorithm is considered future research direction. However, while our algorithm does not succeed to find solutions with small objective value (see minimum value in table and compare our work with [17]), the median value of objective function values over 10 problem instances is lower than that of [17]. *I.e.*, the "typical" achieved objective value is lower than that of [17].[9]

[8] We are not aware of another standardized package for ℓ_p-regression. To accelerate the execution of SeDuMi, we use the lowest precision set up in CVX.

[9] We ran the algorithm in [17] for more time (repeatedly within allowed time) and picked the best minimum result. However, this did not improve the results of [17].

	[17]	
	Time (sec.)	Error
Rank r	[min, mean, median]	
1	[6.81e-02, 2.24e-01, 2.28e-01]	[4.91e-01, 4.93e-01, 4.93e-01]
2	[1.55e-02, 2.75e-02, 2.31e-02]	[5.33e-01, 6.00e-01, 5.96e-01]
3	[2.42e-02, 5.89e-02, 4.59e-02]	[5.22e-01, 5.63e-01, 5.44e-01]
4	[2.69e-02, 4.61e-02, 4.04e-02]	[5.24e-01, 5.66e-01, 5.42e-01]
5	[4.67e-02, 3.36e-01, 1.48e-01]	[5.04e-01, 5.36e-01, 5.26e-01]
6	[6.72e-02, 6.24e-01, 1.34e-01]	[4.98e-01, 5.20e-01, 5.22e-01]
7	[5.46e-02, 8.91e-01, 5.47e-01]	[4.90e-01, 5.14e-01, 5.11e-01]
8	[1.36e-01, 1.66e+00, 5.39e-01]	[4.81e-01, 5.15e-01, 5.02e-01]
9	[1.90e-01, 2.91e+00, 2.56e+00]	[4.73e-01, 4.98e-01, 4.89e-01]
10	[2.30e-01, 9.60e+00, 4.25e+00]	[4.59e-01, 4.97e-01, 4.79e-01]

	This work	
	Time (sec.)	Error
Rank r	[min, mean, median]	
1	[2.57e-02, 4.32e+01, 5.44e+01]	[4.99e-01, 5.82e-01, 5.01e-01]
2	[2.60e-02, 4.95e+01, 5.44e+01]	[5.04e-01, 5.49e-01, 5.07e-01]
3	[5.20e+01, 5.43e+01, 5.42e+01]	[5.06e-01, 5.10e-01, 5.10e-01]
4	[1.55e-02, 3.67e+01, 5.15e+01]	[5.05e-01, 5.90e-01, 5.10e-01]
5	[4.17e-02, 7.92e+01, 8.93e+01]	[5.07e-01, 5.33e-01, 5.13e-01]
6	[7.27e+01, 8.03e+01, 7.76e+01]	[5.02e-01, 5.08e-01, 5.09e-01]
7	[1.62e-02, 5.11e+01, 6.52e+01]	[5.08e-01, 5.84e-01, 5.08e-01]
8	[5.51e+01, 6.55e+01, 6.73e+01]	[4.95e-01, 5.09e-01, 5.02e-01]
9	[5.36e+01, 5.89e+01, 5.77e+01]	[4.78e-01, 5.06e-01, 5.06e-01]
10	[1.69e-02, 3.86e+01, 5.23e+01]	[4.69e-01, 5.94e-01, 4.75e-01]

Table 1: Attained objective function values and execution time. Table includes minimum, mean and median values for 10 Monte Carlo instances.

7 CONCLUSION AND FUTURE WORK

We consider the problem of low-rank matrix approximation, w.r.t. (entrywise) ℓ_p-norms, and proposed two algorithms that lead to $(1+\varepsilon)$-OPT approximations. Our schemes combine ideas from smoothing techniques in convex optimization, as well as recent non-convex gradient descent algorithms. Key assumption is that problem-related quantities are known or at least are approximable. Our experiments show that our scheme performs (at least) competitively with state of the art.

We have provided several possible extensions of this work. A particularly interesting open problem is that of weighted low-rank matrix approximation:

$$\min_{U\in\mathbb{R}^{m\times r},V\in\mathbb{R}^{n\times r}} |W\odot\left(M-UV^\top\right)|_p, \quad p\in\{1,\infty\},$$

where different assumptions on W lead to different open research questions.

References

[1] H. Aanas, R. Fisker, K. Astrom, and J. Carstensen. Robust factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1215–1225, 2002.

[2] M. Asteris, A. Kyrillidis, D. Papailiopoulos, and A. Dimakis. Bipartite correlation clustering: Maximizing agreements. In *Artificial Intelligence and Statistics*, pages 121–129, 2016.

[3] J. Barron. A more general robust loss function. *arXiv preprint arXiv:1701.03077*, 2017.

[4] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. In *29th Annual Conference on Learning Theory*, pages 530–582, 2016.

[5] R. Cabral, F. De la Torre, J. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[6] E. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[7] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 2, pages 168–172. IEEE, 1994.

[8] K.-Y. Chiang, C.-J. Hsieh, and I. Dhillon. Robust principal component analysis with side information. In *International Conference on Machine Learning*, pages 2291–2299, 2016.

[9] F. Chierichetti, S. Gollapudi, R. Kumar, S. Lattanzi, R. Panigrahy, and D. Woodruff. Algorithms for ℓ_p low rank approximation. *arXiv preprint arXiv:1705.06730*, 2017.

[10] M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal components analysis to the exponential family. In *Advances in neural information processing systems*, pages 617–624, 2002.

[11] I. Csiszar and G. Tusnady. Information geometry and alternating minimization procedures. *Statistics and decisions*, 1984.

[12] A. d'Aspremont. Smooth optimization with approximate gradient. *SIAM Journal on Optimization*, 19(3):1171–1183, 2008.

[13] A. Eriksson and A. Van Den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the ℓ_1-norm. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010.

[14] Michael G. and Stephen B. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[15] R. Ge, C. Jin, and Y. Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. *arXiv preprint arXiv:1704.00708*, 2017.

[16] R. Ge, J. Lee, and T. Ma. Matrix completion has no spurious local minimum. *To appear in NIPS-16, arXiv preprint arXiv:1605.07272*, 2016.

[17] N. Gillis and Y. Shitov. Low-rank matrix approximation in the infinity norm. *arXiv preprint arXiv:1706.00078*, 2017.

[18] N. Gillis and S. Vavasis. Fast and robust recursive algorithmsfor separable nonnegative matrix factorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):698–714, 2014.

[19] N. Gillis and S. Vavasis. On the complexity of robust PCA and ell_1-norm low-rank matrix approximation. *arXiv preprint arXiv:1509.09236*, 2015.

[20] G. Golub and C. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[21] G. Gordon. Generalized^2 linear^2 models. In *Advances in neural information processing systems*, pages 593–600, 2003.

[22] S. Goreinov and E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 280:47–52, 2001.

[23] S. Goreinov and E. Tyrtyshnikov. Quasioptimality of skeleton approximation of a matrix in the Chebyshev norm. In *Doklady Mathematics*, volume 83, pages 374–375. Springer, 2011.

[24] Q. Gu, Z. W. Wang, and H. Liu. Low-rank and sparse structure pursuit via alternating minimization. In *Artificial Intelligence and Statistics*, pages 600–609, 2016.

[25] P. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[26] Q. Ke and T. Kanade. Robust subspace computation using ℓ_1-norm. 2003.

[27] Q. Ke and T. Kanade. Robust ℓ_1 factorization in the presence of outliers and missing data by alternative convex programming. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 739–746. IEEE, 2005.

[28] J. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 217–226. SIAM, 2014.

[29] E. Kim, M. Lee, C.-H. Choi, N. Kwak, and S. Oh. Efficient ℓ_1-norm-based low-rank matrix approximations for large-scale problems using alternating rectified gradient method. *IEEE transactions on neural networks and learning systems*, 26(2):237–251, 2015.

[30] N. Kwak. Principal component analysis based on ℓ_1-norm maximization. *IEEE transactions on pattern analysis and machine intelligence*, 30(9):1672–1680, 2008.

[31] A. Kyrillidis and V. Cevher. Matrix ALPS: Accelerated low rank and sparse matrix reconstruction. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 185–188. IEEE, 2012.

[32] A. Kyrillidis and V. Cevher. Matrix recipes for hard thresholding methods. *Journal of mathematical imaging and vision*, 48(2):235–265, 2014.

[33] A. Kyrillidis, A. Kalev, D. Park, S. Bhojanapalli, C. Caramanis, and S. Sanghavi. Provable quantum state tomography via non-convex methods. *arXiv preprint arXiv:1711.02524*, 2017.

[34] X. Li, Z. Wang, J. Lu, R. Arora, J. Haupt, H. Liu, and T. Zhao. Symmetry, saddle points, and global geometry of nonconvex matrix factorization. *arXiv preprint arXiv:1612.09296*, 2016.

[35] Y. Li, Y. Liang, and A. Risteski. Recovery guarantee of non-negative matrix factorization via alternating updates. In *Advances in Neural Information Processing Systems*, pages 4987–4995, 2016.

[36] P. Markopoulos, G. Karystinos, and D. Pados. Some options for ℓ_1-subspace signal processing. In *Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on*, pages 1–5. VDE, 2013.

[37] P. Markopoulos, G. Karystinos, and D. Pados. Optimal algorithms for ℓ_1-subspace signal processing. *IEEE Transactions on Signal Processing*, 62(19):5046–5058, 2014.

[38] D. Meng, Z. Xu, L. Zhang, and J. Zhao. A cyclic weighted median method for ℓ_1 low-rank matrix factorization with missing entries. In *AAAI*, volume 4, page 6, 2013.

[39] Y. Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, 2007.

[40] D. Park, A. Kyrillidis, S. Bhojanapalli, C. Caramanis, and S. Sanghavi. Provable Burer-Monteiro factorization for a class of norm-constrained matrix problems. *arXiv preprint arXiv:1606.01316*, 2016.

[41] D. Park, A. Kyrillidis, C. Caramanis, and S. Sanghavi. Finding low-rank solutions to matrix problems, efficiently and provably. *arXiv preprint arXiv:1606.03168*, 2016.

[42] D. Park, A. Kyrillidis, C. Caramanis, and S. Sanghavi. Non-square matrix sensing without spurious local minima via the Burer-Monteiro approach. *arXiv preprint arXiv:1609.03240*, 2016.

[43] S. Poljak and J. Rohn. Checking robust nonsingularity is NP-hard. *Mathematics of Control, Signals, and Systems (MCSS)*, 6(1):1–9, 1993.

[44] C. Qiu, N. Vaswani, B. Lois, and L. Hogben. Recursive robust PCA or recursive sparse recovery in large but structured noise. *IEEE Transactions on Information Theory*, 60(8):5007–5039, 2014.

[45] A. Singh and G. Gordon. A unified view of matrix factorization models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 358–373. Springer, 2008.

[46] Z. Song, D. Woodruff, and P. Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 688–701. ACM, 2017.

[47] R. Sun and Z.-Q. Luo. Guaranteed matrix completion via nonconvex factorization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 270–289, 2015.

[48] M. Tipping. Probabilistic visualisation of high-dimensional binary data. In *Advances in neural information processing systems*, pages 592–598, 1999.

[49] M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[50] Q. Tran-Dinh and Z. Zhang. Extended Gauss-Newton and Gauss-Newton-ADMM algorithms for low-rank matrix optimization. *arXiv preprint arXiv:1606.03358*, 2016.

[51] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via Procrustes flow. *arXiv preprint arXiv:1507.03566*, 2015.

[52] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[53] N. Veldt, A. Wirth, and D. Gleich. Correlation clustering with low-rank matrices. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1025–1034. International World Wide Web Conferences Steering Committee, 2017.

[54] L. Wang, X. Zhang, and Q. Gu. A universal variance reduction-based catalyst for nonconvex low-rank matrix recovery. *arXiv preprint arXiv:1701.02301*, 2017.

[55] T. Wiberg. Computation of principal components when data are missing. In *Proc. of Second Symp. Computational Statistics*, pages 229–236, 1976.

[56] H. Xu, C. Caramanis, and S. Sanghavi. Robust PCA via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.

[57] X. Yi, D. Park, Y. Chen, and C. Caramanis. Fast algorithms for robust PCA via gradient descent. In *Advances in neural information processing systems*, pages 4152–4160, 2016.

[58] T. Zhao, Z. Wang, and H. Liu. A nonconvex optimization framework for low rank matrix estimation. In *Advances in Neural Information Processing Systems*, pages 559–567, 2015.

[59] T. Zhou and D. Tao. GoDec: Randomized low-rank & sparse matrix decomposition in noisy case. In *International conference on machine learning*. Omnipress, 2011.