Big Data and Information Analytics

https://www.aimspress.com/journal/BDIA

BDIA, 10: 409–431.
DOI:10.3934/bdia.2026021
Received: 09 November 2026
Accepted: 10 November 2026
Published: 23 November 2026

Research article

The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices

Zhouchen Lin[1,*], **Minming Chen**[2] **and Yi Ma**[3]

[1] State Key Lab of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing, China

[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

[3] School of Computing and Data Science and the Musketeers Foundation Institute of Data Science, The University of Hong Kong, Hong Kong SAR, China

* **Correspondence:** Email: zlin@pku.edu.cn; Tel: +86-10-62753313.

Abstract: This paper proposes scalable and fast algorithms for solving the robust principle component analysis (RPCA) problem, namely recovering a low-rank matrix with an unknown fraction of its entries being arbitrarily corrupted. This problem arises in many applications, such as image processing, web data ranking, and bioinformatic data analysis. It was shown that under surprisingly broad conditions, the RPCA problem can be exactly solved via convex optimization that minimizes a combination of the nuclear norm and the ℓ^1-norm . In this paper, we apply the method of augmented Lagrange multipliers (ALM) to solve this convex program. As the objective function is non-smooth, we show how to extend the classical analysis of ALM to such new objective functions, prove the optimality of the proposed algorithms, and characterize their convergence rate. Empirically, the proposed new algorithms can be more than five times faster than the previous state-of-the-art algorithms for RPCA, such as the accelerated proximal gradient (APG) algorithm. Moreover, the new algorithms achieve higher precision, yet being less storage/memory demanding. We also show that the ALM technique can be used to solve the (related but somewhat simpler) matrix completion (MC) problem and obtain rather promising results too. The Matlab code of all algorithms discussed is available at: https://zhouchenlin.github.io/RPCA+MC_codes.zip.

Keywords: low-rank matrix recovery or completion; robust principal component analysis; nuclear norm minimization; ℓ^1-norm minimization; proximal gradient algorithms; augmented Lagrange multipliers

1. Introduction

Principal component analysis (PCA), as a popular tool for high-dimensional data processing, analysis, compression, and visualization, has wide applications in scientific and engineering fields [1]. It assumes that the given high-dimensional data lie near a much lower-dimensional linear subspace. To large extent, the goal of PCA is to efficiently and accurately estimate this low-dimensional subspace.

Suppose that the given data are arranged as the columns of a large matrix $D \in \mathbb{R}^{m\times n}$. The mathematical model for estimating the low-dimensional subspace is to find a low rank matrix A, such that the discrepancy between A and D is minimized, leading to the following constrained optimization:

$$\min_{A,E} \quad \|E\|_F, \quad \text{subject to} \qquad \operatorname{rank}(A) \le r, \quad D = A + E, \tag{1}$$

where $r \ll \min(m, n)$ is the target dimension of the subspace and $\|\cdot\|_F$ is the Frobenius norm, which corresponds to assuming that the data are corrupted by i.i.d. Gaussian noise. This problem can be conveniently solved by first computing the singular value decomposition (SVD) of D and then projecting the columns of D onto the subspace spanned by the r principal left singular vectors of D [1].

As PCA gives the optimal estimate when the corruption is caused by additive i.i.d. Gaussian noise, it works well in practice as long as the magnitude of noise is small. However, it breaks down under large corruption, even if that corruption affects only very few of the observations. In fact, even if only one entry of A is arbitrarily corrupted, the estimated $\hat{A}$ obtained by classical PCA can be arbitrarily far from the true A. Therefore, it is necessary to investigate whether a low-rank matrix A can still be efficiently and accurately recovered from a corrupted data matrix $D = A + E$, where some entries of the additive errors E may be arbitrarily large.

In 2011, Candès et al. [2] showed that under rather broad conditions, the answer is affirmative: As long as the error matrix E is sufficiently sparse (relative to the rank of A), one can exactly recover the low-rank matrix A from $D = A + E$ by solving the following convex optimization problem:

$$\min_{A,E} \quad \|A\|_* + \lambda \|E\|_1, \quad \text{subject to} \qquad D = A + E, \tag{2}$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix (i.e., the sum of its singular values), $\|\cdot\|_1$ denotes the sum of the absolute values of matrix entries, and λ is a positive weighting parameter. Due to the ability to exactly recover underlying low-rank structure in the data, even in the presence of large errors or outliers, this optimization is referred to as robust PCA (RPCA) in [2] (a popular term that has been used by a long line of work that aims to render PCA robust to outliers and gross corruption). Several applications of RPCA, e.g., background modeling and removing shadows and specularities from face images, have been demonstrated in [3] to show the advantage of RPCA.

The optimization (2) can be treated as a general convex optimization problem and solved by any off-the-shelf interior point solver (e.g., CVX [4]), after being reformulated as a semidefinite program [5]. However, although interior point methods normally take very few iterations to converge, they have difficulty in handling large matrices because the complexity of computing the step direction

is $O(m^6)$, where m is the dimension of the matrix. As a result, on a typical personal computer (PC), generic interior point solvers cannot handle matrices with dimensions larger than $m = 10^2$. In contrast, applications in image and video processing often involve matrices of dimension $m = 10^4$ to 10^5; applications in web search and bioinformatics can easily involve matrices of dimension $m = 10^6$ and beyond. So the generic interior point solvers are too limited for RPCA to be practical for many real applications.

That the interior point solvers do not scale well for large matrices is because they rely on second-order information of the objective function. To overcome the scalability issue, we should use the first-order information only and fully harness the special properties of this class of convex optimization problems. For example, it has been shown that the (first-order) iterative thresholding (IT) algorithms can be very efficient for ℓ^1-norm minimization problems arising in compressed sensing [6–9]. It has also been shown in [10] that the same techniques can be used to minimize the nuclear norm for the matrix completion (MC) problem, namely recovering a low-rank matrix from an incomplete but clean subset of its entries [11, 12].

As the matrix recovery (RPCA) problem (2) involves minimizing a combination of both the ℓ^1-norm and the nuclear norm, in the original paper [2], the authors also adopted the IT technique to solve (2) and obtained similar convergence and scalability properties. However, the IT scheme proposed in [2] converges extremely slowly. Typically, it requires about 10^4 iterations to converge, with each iteration having the same cost as one SVD. As a result, even for matrices with dimensions as small as $m = 800$, the algorithm has to run eight hours on a typical PC. To alleviate the slow convergence of the IT method [2], Lin et al. [13] proposed two new algorithms for solving the problem (2), which in some sense are complementary to each other: The first one is an accelerated proximal gradient (APG) algorithm applied to the primal, which is a direct application of the fast iterative shrinkage-thresholding algorithm (FISTA) framework introduced by [7], coupled with a fast continuation technique*; the second one is a gradient-ascent algorithm applied to the dual of the problem (2). From simulations with matrices of dimension up to $m = 1000$, both methods are at least 50 times faster than the IT method (see [13] for more details).

In this paper, we present novel algorithms for matrix recovery which utilize techniques of augmented Lagrange multipliers (ALM). The exact ALM (EALM) method to be proposed here is proven to have a pleasing Q-linear convergence speed, while the APG is in theory only sub-linear. A slight improvement over the EALM leads an inexact ALM (IALM) method, which converges practically as fast as the EALM, but the required number of partial SVDs is significantly less. Experimental results show that IALM is at least five times faster than APG, and its precision is also higher. In particular, the number of non-zeros in E computed by IALM is much more accurate (actually, often exact) than that by APG, which often leaves many small non-zero terms in E.

In the rest of the paper, for completeness, we will first sketch the previous work in Section 2. Then we present our new ALM-based algorithms and analyze their convergence properties in Section 3 (while leaving all technical proofs to Appendix A). We will also quickly illustrate how the same ALM method can be easily adapted to solve the (related but somewhat simpler) MC problem. We will then discuss some implementation details of our algorithms in Section 4. Next in Section 5, we compare the new algorithms and other existing algorithms for both matrix recovery and MC, using extensive simulations on randomly generated matrices. Finally we give some concluding remarks in Section 6.

*Similar techniques have been applied to the MC problem by [14].

2. Previous algorithms for matrix recovery

In this section, for completeness as well as purpose of comparison, we briefly introduce and summarize other existing algorithms for solving the matrix recovery problem (2).

2.1. The iterative thresholding approach

The IT approach proposed in [2] solves a relaxed convex problem of (2):

$$\min_{A,E} \quad \|A\|_* + \lambda \|E\|_1 + \frac{1}{2\tau}\|A\|_F^2 + \frac{1}{2\tau}\|E\|_F^2, \quad \text{subject to} \quad A + E = D, \tag{3}$$

where τ is a large positive scalar so that the objective function is only perturbed slightly. By introducing a Lagrange multiplier Y to remove the equality constraint, one has the Lagrangian function of (3):

$$L(A, E, Y) = \|A\|_* + \lambda \|E\|_1 + \frac{1}{2\tau}\|A\|_F^2 + \frac{1}{2\tau}\|E\|_F^2 + \frac{1}{\tau}\langle Y, D - A - E\rangle. \tag{4}$$

Then the IT approach updates A, E, and Y iteratively. It updates A and E by minimizing $L(A, E, Y)$ with respect to A and E, with Y fixed. Then the amount of violation of the constraint $A + E = D$ is used to update Y.

For convenience, we introduce the following soft-thresholding (shrinkage) operator:

$$\mathcal{S}_\varepsilon[x] \doteq \begin{cases} x - \varepsilon, & \text{if } x > \varepsilon, \\ x + \varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $x \in \mathbb{R}$ and $\varepsilon > 0$. This operator can be extended to vectors and matrices by applying it element-wise. Then the IT approach works as described in Algorithm 1, where the thresholdings directly follow from the well-known analysis [6, 10]:

$$U\mathcal{S}_\varepsilon[S]V^T = \arg\min_X \varepsilon\|X\|_* + \frac{1}{2}\|X - W\|_F^2, \quad \mathcal{S}_\varepsilon[W] = \arg\min_X \varepsilon\|X\|_1 + \frac{1}{2}\|X - W\|_F^2, \tag{6}$$

where USV^T is the SVD of W. Although being extremely simple and provably correct, the IT algorithm requires a very large number of iterations to converge, and it is difficult to choose the step size δ_k for speedup, hence its applicability is limited.

2.2. The accelerated proximal gradient approach

A general theory of the APG approach can be found in [7, 15, 16]. To solve the following unconstrained convex problem:

$$\min_{X \in \mathcal{H}} \quad F(X) \doteq g(X) + f(X), \tag{7}$$

where $\mathcal{H}$ is a real Hilbert space endowed with an inner product $\langle \cdot, \cdot \rangle$ and a corresponding norm $\| \cdot \|$, both g and f are convex, and f is further Lipschitz continuous: $\|\nabla f(X_1) - \nabla f(X_2)\| \le L_f \|X_1 - X_2\|$, one may approximate $f(X)$ locally as a quadratic function and solve

$$X_{k+1} = \arg\min_{X \in \mathcal{H}} Q(X, Y_k) \doteq f(Y_k) + \langle \nabla f(Y_k), X - Y_k\rangle + \frac{L_f}{2}\|X - Y_k\|^2 + g(X), \tag{8}$$

Algorithm 1 (RPCA via IT)

Input: Observation matrix $D \in \mathbb{R}^{m\times n}$, weights λ and τ.

while not converged **do**
  $(U, S, V) = \text{svd}(Y_{k-1})$,
  $A_k = U\mathcal{S}_\tau[S]V^T$,
  $E_k = \mathcal{S}_{\lambda\tau}[Y_{k-1}]$,
  $Y_k = Y_{k-1} + \delta_k(D - A_k - E_k)$.
end while

Output: $A \leftarrow A_k$, $E \leftarrow E_k$.

which is assumed to be easy, to update the solution X. The convergence behavior of this iteration depends strongly on the points Y_k at which the approximations $Q(X, Y_k)$ are formed. The natural choice $Y_k = X_k$ (proposed, e.g., by [17]) can be interpreted as a gradient algorithm and results in a convergence rate no worse than $O(k^{-1})$ [7]. However, for smooth g, Nesterov showed that instead setting $Y_k = X_k + \frac{t_{k-1}-1}{t_k}(X_k - X_{k-1})$ for a sequence $\{t_k\}$ satisfying $t_{k+1}^2 - t_{k+1} \le t_k^2$ can improve the convergence rate to $O(k^{-2})$ [16]. In 2009, Beck and Teboulle extended this scheme to the nonsmooth g, again demonstrating a convergence rate of $O(k^{-2})$, in a sense that $F(X_k) - F(X^*) \le Ck^{-2}$ [7].

The above APG approach can be directly applied to a relaxed version of the RPCA problem by identifying

$$X = (A, E), \quad f(X) = \frac{1}{\mu}\|D - A - E\|_F^2, \quad \text{and} \quad g(X) = \|A\|_* + \lambda\|E\|_1,$$

where μ is a small positive scalar. A continuation technique [14], which varies μ, starting from a large initial value μ_0 and decreasing it geometrically with each iteration until it reaches the floor $\bar{\mu}$, can greatly speed up the convergence. The APG approach for RPCA is described in Algorithm 2 (for details, see [3, 13]).

Algorithm 2 (RPCA via APG)

Input: Observation matrix $D \in \mathbb{R}^{m\times n}$, λ.

$A_0 = A_{-1} = 0$; $E_0 = E_{-1} = 0$; $t_0 = t_{-1} = 1$; $\bar{\mu} > 0$; $\eta < 1$.
while not converged **do**
  $Y_k^A = A_k + \frac{t_{k-1}-1}{t_k}(A_k - A_{k-1})$, $Y_k^E = E_k + \frac{t_{k-1}-1}{t_k}(E_k - E_{k-1})$.
  $G_k^A = Y_k^A - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.
  $(U, S, V) = \text{svd}(G_k^A)$, $A_{k+1} = U\mathcal{S}_{\frac{\mu_k}{2}}[S]V^T$.
  $G_k^E = Y_k^E - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.
  $E_{k+1} = \mathcal{S}_{\frac{\lambda\mu_k}{2}}[G_k^E]$.
  $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$; $\mu_{k+1} = \max(\eta\mu_k, \bar{\mu})$.
  $k \leftarrow k + 1$.
end while

Output: $A \leftarrow A_k$, $E \leftarrow E_k$.

2.3. The dual approach

The dual approach proposed in our earlier work [13] tackles the problem (2) via its dual. That is, one first solves the dual problem

$$\max_{Y} \langle D, Y\rangle, \quad \text{subject to} \quad J(Y) \le 1 \tag{9}$$

for the optimal Lagrange multiplier Y, where

$$\langle A, B\rangle = \mathrm{tr}(A^T B), \quad J(Y) = \max\left(\|Y\|_2, \lambda^{-1}\|Y\|_\infty\right) \tag{10}$$

and $\|\cdot\|_\infty$ is the maximum absolute value of the matrix entries. A steepest ascend algorithm constrained on the surface $\{Y|J(Y) = 1\}$ can be adopted to solve (9), where the constrained steepest ascend direction is obtained by projecting D onto the tangent cone of the convex body $\{Y|J(Y) \le 1\}$. It turns out that the optimal solution to the primal problem (2) can be obtained during the process of finding the constrained steepest ascend direction. For details of the final algorithm, one may refer to [13].

A merit of the dual approach is that only the principal singular space associated to the largest singular value 1 is needed. In theory, computing this special principal singular space should be easier than computing the principal singular space associated to the unknown leading singular values. So the dual approach is promising if an efficient method for computing the principal singular space associated to the known largest singular value can be obtained.

3. The methods of ALM

In [18], the general method of ALM is introduced for solving constrained optimization problems of the kind

$$\min f(X), \quad \text{subject to} \quad h(X) = 0, \tag{11}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$. One may define the augmented Lagrangian function

$$L(X, Y, \mu) = f(X) + \langle Y, h(X)\rangle + \frac{\mu}{2}\|h(X)\|_F^2, \tag{12}$$

where μ is a positive scalar, and then the optimization problem can be solved via the method of ALM, outlined as in Algorithm 3 (see [19] for more details).

Algorithm 3 (General method of ALM)

1: $\rho \ge 1$.
2: **while** not converged **do**
3: Solve $X_{k+1} = \arg\min_X L(X, Y_k, \mu_k)$.
4: $Y_{k+1} = Y_k + \mu_k h(X_{k+1})$;
5: Update μ_k to μ_{k+1}.
6: **end while**

Output: X_k.

Under some rather general conditions, when $\{\mu_k\}$ is an increasing sequence and both f and h are continuously differentiable functions, it has been proven in [18] that the Lagrange multipliers Y_k

produced by Algorithm 3 converge Q-linearly to the optimal solution when $\{\mu_k\}$ is bounded and super-Q-linearly when $\{\mu_k\}$ is unbounded. This superior convergence property of ALM makes it very attractive. Another merit of ALM is that the optimal step size to update Y_k is proven to be the chosen penalty parameter μ_k, making the parameter tuning much easier than the IT algorithm. A third merit of ALM is that the algorithm converges to the exact optimal solution, even without requiring μ_k to approach infinity [18]. In contrast, strictly speaking, both the IT and the APG approaches mentioned earlier only find approximate solutions for the problem. Finally, the analysis (of convergence) and the implementation of the ALM algorithms are relatively simple, as we will demonstrate on both the matrix recovery and the MC problems.

3.1. Two ALM algorithms for robust PCA (matrix recovery)

For the RPCA problem (2), we may apply the ALM method by identifying

$$X = (A, E), \quad f(X) = \|A\|_* + \lambda\|E\|_1, \quad \text{and} \quad h(X) = D - A - E.$$

Then the Lagrangian function is

$$L(A, E, Y, \mu) \doteq \|A\|_* + \lambda\|E\|_1 + \langle Y, D - A - E\rangle + \frac{\mu}{2}\|D - A - E\|_F^2, \tag{13}$$

and the ALM method for solving the RPCA problem can be described in Algorithm 4, which we will refer to as the EALM method for reasons that will soon become clear.

The initialization $Y_0^* = \text{sgn}(D)/J(\text{sgn}(D))$ in the algorithm is inspired by the dual problem (9) as it is likely to make the objective function value $\langle D, Y_0^*\rangle$ reasonably large.

Although the objective function of the RPCA problem (2) is non-smooth and hence the results in [18] do not directly apply here, we can still prove that Algorithm 4 has the same excellent convergence property. More precisely, we have established the following statement.

Theorem 1. *For Algorithm 4, any accumulation point (A^*, E^*) of (A_k^*, E_k^*) is an optimal solution to the RPCA problem, and the convergence rate is at least $O(\mu_k^{-1})$ in the sense that*

$$\left|\|A_k^*\|_* + \lambda\|E_k^*\|_1 - f^*\right| = O(\mu_{k-1}^{-1}),$$

where f^* is the optimal value of the RPCA problem.

Proof. See Appendix A.3.

From Theorem 1, we see that if μ_k grows geometrically, the EALM method will converge Q-linearly; if μ_k grows faster, the EALM method will also converge faster. However, numerical tests show that for larger μ_k, the IT approach to solve the sub-problem $(A_{k+1}^*, E_{k+1}^*) = \arg\min_{A,E} L(A, E, Y_k^*, \mu_k)$ will converge slower. As the SVD accounts for the majority of the computational load, the choice of $\{\mu_k\}$ should be judicious so that the total number of SVDs is minimal.

Fortunately, as it turns out, we do not have to solve the sub-problem

$$(A_{k+1}^*, E_{k+1}^*) = \arg\min_{A,E} L(A, E, Y_k^*, \mu_k)$$

exactly. Rather, updating A_k and E_k once when solving this sub-problem is sufficient for A_k and E_k to converge to the optimal solution of the RPCA problem. This leads to an IALM method, described in Algorithm 5.

The validity and optimality of Algorithm 5 is guaranteed by the following theorem.

Algorithm 4 (RPCA via the EALM method)

Input: Observation matrix $D \in \mathbb{R}^{m\times n}$, λ.

1: $Y_0^* = \mathrm{sgn}(D)/J(\mathrm{sgn}(D))$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
2: **while** not converged **do**
3: // Lines 4–12 solve $(A_{k+1}^*, E_{k+1}^*) = \arg\min_{A,E} L(A, E, Y_k^*, \mu_k)$.
4: $A_{k+1}^0 = A_k^*$, $E_{k+1}^0 = E_k^*$, $j = 0$;
5: **while** not converged **do**
6: // Lines 7–8 solve $A_{k+1}^{j+1} = \arg\min_{A} L(A, E_{k+1}^j, Y_k^*, \mu_k)$.
7: $(U, S, V) = \mathrm{svd}(D - E_{k+1}^j + \mu_k^{-1} Y_k^*)$;
8: $A_{k+1}^{j+1} = U\mathcal{S}_{\mu_k^{-1}}[S]V^T$;
9: // Line 10 solves $E_{k+1}^{j+1} = \arg\min_{E} L(A_{k+1}^{j+1}, E, Y_k^*, \mu_k)$.
10: $E_{k+1}^{j+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1}^{j+1} + \mu_k^{-1} Y_k^*]$;
11: $j \leftarrow j + 1$.
12: **end while**
13: $Y_{k+1}^* = Y_k^* + \mu_k(D - A_{k+1}^* - E_{k+1}^*)$.
14: Update μ_k to μ_{k+1}.
15: $k \leftarrow k + 1$.
16: **end while**

Output: (A_k^*, E_k^*).

Theorem 2. *For Algorithm 5, if $\{\mu_k\}$ is nondecreasing and $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$, then (A_k, E_k) converges to an optimal solution (A^*, E^*) to the RPCA problem.*

Proof. See Appendix A.5.

We can further prove that the condition $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$ is also necessary to ensure the convergence, as stated in the following theorem.

Theorem 3. *If $\sum_{k=1}^{+\infty} \mu_k^{-1} < +\infty$, then the sequence $\{(A_k, E_k)\}$ produced by Algorithm 5 may not converge to the optimal solution of the RPCA problem.*

Proof. See Appendix A.6.

Note that, unlike Theorem 1 for the EALM method, Theorem 2 only guarantees convergence but does not specify the rate of convergence for the IALM method. The condition $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$ implies that μ_k cannot grow too fast. However, in practice we may still choose geometrically growing μ_k at the beginning and then switch to slowly growing μ_k when $\|D - A_k - E_k\|_F$ is below a threshold. Our numerical tests show that with our initialization as presented in Algorithm 5, the switch to slowly growing μ_k is often unnecessary if the geometric μ_k's at the beginning do not grow too fast. These will be discussed in more detail in Section 4.

Algorithm 5 (RPCA via the IALM method)

Input: Observation matrix $D \in \mathbb{R}^{m\times n}$, λ.

1: $Y_0 = D/J(D)$; $E_0 = 0$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
2: **while** not converged **do**
3: // Lines 4–5 solve $A_{k+1} = \arg\min_A L(A, E_k, Y_k, \mu_k)$.
4: $(U, S, V) = \text{svd}(D - E_k + \mu_k^{-1} Y_k)$;
5: $A_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[S]V^T$.
6: // Line 7 solves $E_{k+1} = \arg\min_E L(A_{k+1}, E, Y_k, \mu_k)$.
7: $E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu_k^{-1} Y_k]$.
8: $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$.
9: Update μ_k to μ_{k+1}.
10: $k \leftarrow k + 1$.
11: **end while**

Output: (A_k, E_k).

3.2. An ALM algorithm for matrix completion

The MC problem can be viewed as a special case of the matrix recovery problem, where one has to recover the missing entries of a matrix, given limited number of known entries. Such a problem is ubiquitous, e.g., in machine learning [20–22], control [23], and computer vision [24]. In many applications, it is reasonable to assume that the matrix to recover is of low rank. Candès and Recht [12] proved that most matrices A of rank r can be perfectly recovered by solving the following optimization problem:

$$\min_A \|A\|_*, \quad \text{subject to} \quad A_{ij} = D_{ij}, \quad \forall (i, j) \in \Omega, \tag{14}$$

provided that the number p of samples obeys $p \geq Crn^{6/5} \ln n$ for some positive constant C, where Ω is the set of indices of samples. This bound has since been improved by the work of several others. The state-of-the-art algorithms to solve the MC problem (14) include the APG approach [14] and the singular value thresholding (SVT) approach [10]. As the RPCA problem is closely connected to the MC problem, it is natural to believe that the ALM method can be similarly effective on the MC problem.

We may formulate the MC problem as follows:

$$\min_A \|A\|_*, \quad \text{subject to} \quad A + E = D, \quad \pi_\Omega(E) = 0, \tag{15}$$

where $\pi_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a linear operator that keeps the entries in Ω unchanged and sets those outside Ω (i.e., in $\bar{\Omega}$) as zeros. As E will compensate for the unknown entries of D, the unknown entries of D are simply set as zeros. Then the partial augmented Lagrangian function (Section 2.4 of [18]) of (15) is

$$L(A, E, Y, \mu) = \|A\|_* + \langle Y, D - A - E\rangle + \frac{\mu}{2}\|D - A - E\|_F^2. \tag{16}$$

Then similarly we can have the EALM and the IALM approaches for the MC problem, where for updating E, the constraint $\pi_\Omega(E) = 0$ should be enforced when minimizing $L(A, E, Y, \mu)$. The IALM approach is described in Algorithm 6.

Algorithm 6 (Matrix completion via the IALM method)

Input: Observation samples D_{ij}, $(i, j) \in \Omega$, of matrix $D \in \mathbb{R}^{m\times n}$.

1: $Y_0 = 0$; $E_0 = 0$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
2: **while** not converged **do**
3: // Lines 4–5 solve $A_{k+1} = \arg\min_A L(A, E_k, Y_k, \mu_k)$.
4: $(U, S, V) = \text{svd}(D - E_k + \mu_k^{-1} Y_k)$;
5: $A_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[S]V^T$.
6: // Line 7 solves $E_{k+1} = \arg\min_{\pi_\Omega(E)=0} L(A_{k+1}, E, Y_k, \mu_k)$.
7: $E_{k+1} = \pi_{\bar{\Omega}}(D - A_{k+1} + \mu_k^{-1} Y_k)$.
8: $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$.
9: Update μ_k to μ_{k+1}.
10: $k \leftarrow k + 1$.
11: **end while**

Output: (A_k, E_k).

Note that due to the choice of E_k, $\pi_{\bar{\Omega}}(Y_k) = 0$ holds throughout the iteration, i.e., the values of Y_k at unknown entries are always zeros. Theorems 1 and 2 are also true for the MC problem. As the proofs are similar, we hence omit them here. However, Theorem 3 is no longer true for the MC problem, because $\{Y_k\}$ (see Lemma 1) may not be bounded. This implies that the convergence condition for the MC problem is more relaxed than that for the RPCA problem.

4. Implementation details

Predicting the dimension of principal singular space. It is apparent that computing the full SVD for the RPCA and MC problems is unnecessary: We only need those singular values that are larger than a particular threshold and their corresponding singular vectors. So a software package, PROPACK [25], has been widely recommended in the community. To use PROPACK, one has to predict the dimension of the principal singular space whose singular values are larger than a given threshold. For Algorithm 5, the prediction is relatively easy as the rank of A_k is observed to be monotonically increasing and becomes stable at the true rank. So the prediction rule is

$$\text{sv}_{k+1} = \begin{cases} \text{svp}_k + 1, & \text{if } \text{svp}_k < \text{sv}_k, \\ \min(\text{svp}_k + \text{round}(0.05d), d), & \text{if } \text{svp}_k = \text{sv}_k, \end{cases} \tag{17}$$

where $d = \min(m, n)$, sv_k is the predicted dimension, svp_k is the number of singular values in the sv_k singular values that are larger than μ_k^{-1}, and $\text{sv}_0 = 10$. Algorithm 4 also uses the above prediction strategy for the inner loop that solves (A^*_{k+1}, E^*_{k+1}). For the outer loop, the prediction rule is simply $\text{sv}_{k+1} = \min(\text{svp}_k + \text{round}(0.1d), d)$. As for Algorithm 6, the prediction is much more difficult as the ranks of A_k are often oscillating. It is also often that for small k's, the ranks of A_k are close to d and then gradually decrease to the true rank, making the partial SVD inefficient[†]. To remedy this issue, we

[†]Numerical tests show that when we want to compute more than $0.2d$ principal singular vectors/values, using PROPACK is often slower than computing the full SVD.

initialize both Y and A as zero matrices and adopt the following truncation strategy which is similar to that in [14]:

$$\mathrm{sv}_{k+1} = \begin{cases} \mathrm{svn}_k + 1, & \text{if } \mathrm{svn}_k < \mathrm{sv}_k, \\ \min(\mathrm{svn}_k + 10, d), & \text{if } \mathrm{svn}_k = \mathrm{sv}_k, \end{cases} \tag{18}$$

where $\mathrm{sv}_0 = 5$ and

$$\mathrm{svn}_k = \begin{cases} \mathrm{svp}_k, & \text{if } \mathrm{maxgap}_k \leq 2, \\ \min(\mathrm{svp}_k, \mathrm{maxid}_k), & \text{if } \mathrm{maxgap}_k > 2, \end{cases} \tag{19}$$

in which maxgap_k and maxid_k are the largest ratios between successive singular values (arranging the computed sv_k singular values in a descending order) and the corresponding index, respectively. We utilize the gap information because we have observed that the singular values are separated into two groups quickly, with a large gap between them, making the rank revealing fast and reliable. With the above prediction scheme, the rank of A_k becomes monotonically increasing and is stable at the true rank.

Order of updating A and E. Although in theory updating whichever of A and E first does not affect the convergence rate, numerical tests show that this does result in slightly different number of iterations to achieve the same accuracy. Considering the huge complexity of SVD for large dimensional matrices, such slight difference should also be considered. Via extensive numerical tests, we suggest updating E first in Algorithms 4 and 5. What is equally important, updating E first also makes the rank of A_k much more likely to be monotonically increasing, which is critical for the partial SVD to be effective, as having been elaborated in the previous paragraph.

Memory saving for Algorithm 6. In the real implementation of Algorithm 6, sparse matrices are used to store D and Y_k, and as done in [14] A is represented as $A = LR^T$, where both L and R are matrices of size $m \times \mathrm{svp}_k$. E_k is not explicitly stored by observing

$$E_{k+1} = \pi_{\bar{\Omega}}(D - A_{k+1} + \mu_k^{-1} Y_k) = \pi_{\Omega}(A_{k+1}) - A_{k+1}. \tag{20}$$

In this way, only $\pi_\Omega(A_k)$ is required to compute Y_k and $D - E_k + \mu_k^{-1} Y_k$. So much memory can be saved due to the small percentage of samples.

Stopping criteria. For the RPCA problem, the Karush–Ckuhn–CTucker (KKT) conditions are

$$D - A^* - E^* = 0, \quad Y^* \in \partial \|A^*\|_*, \quad Y^* \in \partial(\|\lambda E^*\|_1). \tag{21}$$

The last two conditions hold if and only if $\partial\|A^*\|_* \cap \partial(\|\lambda E^*\|_1) \neq \emptyset$. So we may take the following conditions as the stopping criteria for Algorithms 4 and 5:

$$\|D - A_k - E_k\|_F / \|D\|_F < \varepsilon_1 \quad \text{and} \quad \mathrm{dist}(\partial\|A_k\|_*, \partial(\|\lambda E_k\|_1))/\|D\|_F < \varepsilon_2, \tag{22}$$

where $\mathrm{dist}(X, Y) = \min\{\|x - y\|_F | x \in X, y \in Y\}$. For Algorithm 4, the second condition is always guaranteed by the inner loop. So we only have to check the first condition. For Algorithm 5,

unfortunately, it is expensive to compute $\mathrm{dist}(\partial\|A_k\|_*, \partial(\|\lambda E_k\|_1))$ as the projection onto $\partial\|A_k\|_*$ is costly. So we may estimate $\mathrm{dist}(\partial\|A_k\|_*, \partial(\|\lambda E_k\|_1))$ by $\|\hat{Y}_k - Y_k\|_F = \mu_{k-1}\|E_k - E_{k-1}\|_F$ since $\hat{Y}_k \in \partial\|A_k\|_*$ and $Y_k \in \partial(\|\lambda E_k\|_1)$.

Similarly, for the MC problem, we may take the following conditions as the stopping criteria for Algorithm 6:

$$\|D - A_k - E_k\|_F/\|D\|_F < \varepsilon_1 \quad \text{and} \quad \mathrm{dist}(\partial\|A_k\|_*, S)/\|D\|_F < \varepsilon_2, \tag{23}$$

where $S = \{Y|Y_{ij} = 0 \text{ if } (i,j) \notin \Omega\}$. Different from the RPCA problem, here $\mathrm{dist}(\partial\|A_k\|_*, S)$ has a closed-form solution thanks to the fact that S is a linear subspace. More specifically, the closed-form solution is as follows.

Proposition 1. *If the skinny SVD of A_k is $A_k = U_k\Lambda_k V_k^T$, then $\mathrm{dist}(\partial\|A_k\|_*, S) = \|\pi_{\bar{\Omega}}(U_k V_k^T)\|_F$.*

Proof. See Appendix A.7.

Note that the SVD of A_k is readily available during the alternating minimization process of ALM. So it is very convenient to compute $\mathrm{dist}(\partial\|A_k\|_*, S)$.

Updating μ_k. For the RPCA problem, the updating rule is

$$\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \mu_k\|E_{k+1} - E_k\|_F/\|D\|_F < \varepsilon_2, \\ \mu_k, & \text{otherwise,} \end{cases} \tag{24}$$

where $\rho > 1$. One can easily see that this updating rule is consistent with Theorem 2. For the MC problem, the updating rule is

$$\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \|D - A_{k+1} - E_{k+1}\|_F/\|D\|_F \geq \varepsilon_1 \text{ or } \mathrm{dist}(\partial\|A_k\|_*, S)/\|D\|_F < \varepsilon_2, \\ \mu_k, & \text{otherwise.} \end{cases} \tag{25}$$

Choosing parameters. For Algorithm 4, we set $\mu_0 = 0.5/\|\operatorname{sgn}(D)\|_2$ and $\rho = 6$. The stopping criterion for the inner loop is $\|A_k^{j+1} - A_k^j\|_F/\|D\|_F < 10^{-6}$ and $\|E_k^{j+1} - E_k^j\|_F/\|D\|_F < 10^{-6}$. The stopping criterion for the outer iteration is $\|D - A_k^* - E_k^*\|_F/\|D\|_F < 10^{-7}$. For Algorithm 5, we set $\mu_0 = 1.25/\|D\|_2$ and $\rho = 1.6$. And the parameters in the stopping criteria are $\varepsilon_1 = 10^{-7}$ and $\varepsilon_2 = 10^{-5}$. For Algorithm 6, we set $\mu_0 = 1/\|D\|_2$ and $\rho = 1.2172 + 1.8588\rho_s$, where $\rho_s = |\Omega|/(mn)$ is the sampling density and the relation between ρ and ρ_s is obtained by regression. And the parameters in the stopping criteria are $\varepsilon_1 = 10^{-7}$ and $\varepsilon_2 = 10^{-6}$.

5. Simulations

In this section, using numerical simulations, for the RPCA problem we compare the proposed ALM algorithms with the APG algorithm proposed in [13]; for the MC problem, we compare the IALM algorithm with the SVT algorithm [10] and the APG algorithm [14]. All the simulations are conducted and timed on the same workstation with an Intel Xeon E5540 2.53GHz CPU that has 4 cores and 24GB memory[‡], running Windows 7 and Matlab (version 7.7).[§]

[‡]But on a Win32 system only 3GB can be used by each thread.

[§]Matlab code for all the algorithms compared are available at: https://zhouchenlin.github.io/RPCA+MC_codes.zip.

I. Comparison on the robust PCA problem. For the RPCA problem, we use randomly generated square matrices for our simulations. We denote the true solution by the ordered pair $(A^*, E^*) \in \mathbb{R}^{m\times m} \times \mathbb{R}^{m\times m}$. We generate the rank-$r$ matrix A^* as a product LR^T, where L and R are independent $m\times r$ matrices whose elements are i.i.d. Gaussian random variables with zero mean and unit variance.[¶] We generate E^* as a sparse matrix whose support is chosen uniformly at random, and whose non-zero entries are i.i.d. uniformly in the interval $[-500, 500]$. The matrix $D \doteq A^* + E^*$ is the input to the algorithm, and $(\hat{A}, \hat{E})$ denotes the output. We choose a fixed weighting parameter $\lambda = m^{-1/2}$ for a given problem.

We use the latest version of the code for Algorithm 2, provided by the authors of [13], and also apply the prediction rule (17), with $\mathrm{sv}_0 = 5$, to it so that the partial SVD can be utilized[‖]. With the partial SVD, APG is faster than the dual approach in Section 2.3. So we need not involve the dual approach for comparison.

A brief comparison of the three algorithms is presented in Tables 1 and 2. We can see that both APG and IALM algorithms stop at relatively constant iteration numbers and IALM is at least five times faster than APG. Moreover, the accuracies of EALM and IALM are higher than that of APG. In particular, APG often overestimates $\|E^*\|_0$, the number of non-zeros in E^*, quite a bit, while the estimated $\|E^*\|_0$ by EALM and IALM are always extremely close to the ground truth.

II. Comparison on the matrix completion problem. For the MC problem, the true low-rank matrix A^* is first generated as that for the RPCA problem. Then we sample p elements uniformly from A^* to form the known samples in D. A useful quantity for reference is $d_r = r(2m - r)$, which is the number of degrees of freedom in an $m \times m$ matrix of rank r [14].

The SVT and APGL (APG with line search[**]) codes are provided by the authors of [10] and [14], respectively. A brief comparison of the three algorithms is presented in Table 3. One can see that IALM is always faster than SVT. It is also advantageous over APGL when the sampling density p/m^2 is relatively high, e.g., $p/m^2 > 10\%$. This phenomenon is actually consistent with the results on the RPCA problem, where most samples of D are assumed accurate, although the positions of accurate samples are not known a priori.

6. Conclusions

In this paper, we have proposed two ALM-based algorithms, namely EALM and IALM, for solving the RPCA problem (2). Both algorithms are faster than the previous state-of-the-art APG algorithm [13]. In particular, in all simulations IALM is consistently over five times faster than APG.

We have also applied the method of ALM to the MC problem. The corresponding IALM algorithm is considerably faster than the famous SVT algorithm [10]. It is also faster than the state-of-the-art APGL algorithm [14] when the percentage of available entries is not too low, say $> 10\%$.

Compared to APG-based methods, ALM-based algorithms are simpler to analyze and easier to implement. Moreover, they are also of much higher accuracy as the iterations are proven to converge to the exact solution of the problem, even if the penalty parameter does not approach infinity [18]. In contrast, APG methods normally find a close approximation to the solution by solving a relaxed

[¶]It can be shown that A^* is distributed according to the random orthogonal model of rank r, as defined in [12].

[‖]Such a prediction scheme was not proposed in [13]. So the full SVD was used therein.

[**]For the MC problem, APGL is faster than APG without line search. However, for the RPCA problem, APGL is not faster than APG [13].

Table 1. Comparison between APG, EALM, and IALM on the RPCA problem. We present typical running times for randomly generated matrices. Corresponding to each triplet $\{m, \text{rank}(A^*), \|E^*\|_0\}$, the RPCA problem is solved for the same data matrix D using three different algorithms. For APG and IALM, the number of SVDs is equal to the number of iterations.

m	algorithm	$\frac{\|\hat{A}-A^*\|_F}{\|A^*\|_F}$	rank($\hat{A}$)	$\|\hat{E}\|_0$	#SVD	time (s)
rank(A^*) = 0.05 m, $\|E^*\|_0 = 0.05\, m^2$						
500	APG	1.12×10^{-5}	25	12,542	127	11.01
	EALM	3.99×10^{-7}	25	12,499	28	4.08
	IALM	5.21×10^{-7}	25	12,499	20	1.72
800	APG	9.84×10^{-6}	40	32,092	126	37.21
	EALM	1.47×10^{-7}	40	32,002	29	18.59
	IALM	3.29×10^{-7}	40	31,999	21	5.87
1000	APG	8.79×10^{-6}	50	50,082	126	57.62
	EALM	7.85×10^{-8}	50	50,000	29	33.28
	IALM	2.67×10^{-7}	50	49,999	22	10.13
1500	APG	7.16×10^{-6}	75	112,659	126	163.80
	EALM	7.55×10^{-8}	75	112,500	29	104.97
	IALM	1.86×10^{-7}	75	112,500	22	30.80
2000	APG	6.27×10^{-6}	100	200,243	126	353.63
	EALM	4.61×10^{-8}	100	200,000	30	243.64
	IALM	9.54×10^{-8}	100	200,000	22	68.69
3000	APG	5.20×10^{-6}	150	450,411	126	1106.22
	EALM	4.39×10^{-8}	150	449,998	30	764.66
	IALM	1.49×10^{-7}	150	449,993	22	212.34
rank(A^*) = 0.05 m, $\|E^*\|_0 = 0.10\, m^2$						
500	APG	1.41×10^{-5}	25	25,134	129	14.35
	EALM	8.72×10^{-7}	25	25,009	34	4.75
	IALM	9.31×10^{-7}	25	25,000	21	2.52
800	APG	1.12×10^{-5}	40	64,236	129	37.94
	EALM	2.86×10^{-7}	40	64,002	34	20.30
	IALM	4.87×10^{-7}	40	64,000	24	6.69
1000	APG	9.97×10^{-6}	50	100,343	129	65.41
	EALM	6.07×10^{-7}	50	100,002	33	30.63
	IALM	3.78×10^{-7}	50	99,996	22	10.77
1500	APG	8.18×10^{-6}	75	225,614	129	163.36
	EALM	1.45×10^{-7}	75	224,999	33	109.54
	IALM	2.79×10^{-7}	75	224,996	23	35.71
2000	APG	7.11×10^{-6}	100	400,988	129	353.30
	EALM	1.23×10^{-7}	100	400,001	34	254.77
	IALM	3.31×10^{-7}	100	399,993	23	70.33
3000	APG	5.79×10^{-6}	150	901,974	129	1110.76
	EALM	1.05×10^{-7}	150	899,999	34	817.69
	IALM	2.27×10^{-7}	150	899,980	23	217.39

problem. Finally, ALM algorithms require less storage/memory than APG for both the RPCA and the MC problems[††]. For large-scale applications, such as web data analysis, this could prove to be a big advantage for ALM-type algorithms.

[††]By smart reuse of intermediate matrices (and accordingly the codes become hard to read), for the RPCA problem, APG still needs one more intermediate (dense) matrix than IALM; for the MC problem, APG needs two more low-rank matrices (for representing A_{k-1}) and one more sparse matrix than IALM. Our numerical simulation testifies this too: For the MC problem, on our workstation IALM was able to handle A^* with size $10^4 \times 10^4$ and rank 10^2, while APG could not.

Table 2. Comparison between APG, EALM and IALM on the Robust PCA problem. Continued from Table 1 with different parameters of $\{m, \text{rank}(A^*), \|E^*\|_0\}$.

m	algorithm	$\frac{\|\hat{A}-A^*\|_F}{\|A^*\|_F}$	rank($\hat{A}$)	$\|\hat{E}\|_0$	#SVD	time (s)
rank(A^*) = $0.10\,m$, $\|E^*\|_0 = 0.05\,m^2$						
500	APG	9.36×10^{-6}	50	13,722	129	13.99
	EALM	5.53×10^{-7}	50	12,670	41	7.35
	IALM	6.05×10^{-7}	50	12,500	22	2.32
800	APG	7.45×10^{-6}	80	34,789	129	67.54
	EALM	1.13×10^{-7}	80	32,100	40	30.56
	IALM	3.08×10^{-7}	80	32,000	22	10.81
1000	APG	6.64×10^{-6}	100	54,128	129	129.40
	EALM	4.20×10^{-7}	100	50,207	39	50.31
	IALM	2.61×10^{-7}	100	50,000	22	20.71
1500	APG	5.43×10^{-6}	150	121,636	129	381.52
	EALM	1.22×10^{-7}	150	112,845	41	181.28
	IALM	1.76×10^{-7}	150	112,496	24	67.84
2000	APG	4.77×10^{-6}	200	215,874	129	888.93
	EALM	1.15×10^{-7}	200	200,512	41	423.83
	IALM	2.49×10^{-7}	200	199,998	23	150.35
3000	APG	3.98×10^{-6}	300	484,664	129	2923.90
	EALM	7.92×10^{-8}	300	451,112	42	1444.74
	IALM	1.30×10^{-7}	300	450,000	23	485.70
rank(A^*) = $0.10\,m$, $\|E^*\|_0 = 0.10\,m^2$						
500	APG	9.78×10^{-6}	50	27,478	133	13.90
	EALM	1.14×10^{-6}	50	26,577	52	9.46
	IALM	7.64×10^{-7}	50	25,000	25	2.62
800	APG	8.66×10^{-6}	80	70,384	132	68.12
	EALM	3.59×10^{-7}	80	66,781	51	41.33
	IALM	4.77×10^{-7}	80	64,000	25	11.88
1000	APG	7.75×10^{-6}	100	109,632	132	130.37
	EALM	3.40×10^{-7}	100	104,298	49	77.26
	IALM	3.73×10^{-7}	100	99,999	25	22.95
1500	APG	6.31×10^{-6}	150	246,187	132	383.28
	EALM	3.55×10^{-7}	150	231,438	49	239.62
	IALM	5.42×10^{-7}	150	224,998	24	66.78
2000	APG	5.49×10^{-6}	200	437,099	132	884.86
	EALM	2.81×10^{-7}	200	410,384	51	570.72
	IALM	4.27×10^{-7}	200	399,999	24	154.27
3000	APG	4.50×10^{-6}	300	980,933	132	2915.40
	EALM	2.02×10^{-7}	300	915,877	51	1904.95
	IALM	3.39×10^{-7}	300	899,990	24	503.05

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgements

We thank Leqin WU for contributing to the second part of the proof of Theorem 1 and helping draft the early version of the paper. We thank the authors of [14] for kindly sharing with us their code of APG and APGL for matrix completion. We also also like to thank Dr. Arvind Ganesh and Prof. John Wright for providing the code of APG for matrix recovery.

Table 3. Comparison between SVT, APG, and IALM on the MC problem. We present typical running times for randomly generated matrices. Corresponding to each triplet {m, rank(A^*), p/d_r}, the MC problem was solved for the same data matrix D using the three different algorithms.

m	r	p/d_r	p/m^2	algorithm	#iter	rank($\hat{A}$)	time (s)	$\frac{\|\hat{A}-A^*\|_F}{\|A^*\|_F}$
1000	10	6	0.12	SVT	208	10	18.23	1.64×10^{-6}
				APGL	69	10	4.46	3.16×10^{-6}
				IALM	69	10	3.73	1.40×10^{-6}
1000	50	4	0.39	SVT	201	50	126.18	1.61×10^{-6}
				APGL	76	50	24.54	4.31×10^{-6}
				IALM	38	50	12.68	1.53×10^{-6}
1000	100	3	0.57	SVT	228	100	319.93	1.71×10^{-6}
				APGL	81	100	70.59	4.40×10^{-6}
				IALM	41	100	42.94	1.54×10^{-6}
3000	10	6	0.04	SVT	218	10	70.14	1.77×10^{-6}
				APGL	88	10	15.63	2.33×10^{-6}
				IALM	131	10	27.18	1.41×10^{-6}
3000	50	5	0.165	SVT	182	50	370.13	1.58×10^{-6}
				APGL	78	50	101.04	5.74×10^{-6}
				IALM	57	50	82.68	1.31×10^{-6}
3000	100	4	0.26	SVT	204	100	950.01	1.68×10^{-6}
				APGL	82	100	248.16	5.18×10^{-6}
				IALM	50	100	188.22	1.52×10^{-6}
5000	10	6	0.024	SVT	231	10	141.88	1.79×10^{-6}
				APGL	81	10	30.52	5.26×10^{-6}
				IALM	166	10	68.38	1.37×10^{-6}
5000	50	5	0.10	SVT	188	50	637.97	1.62×10^{-6}
				APGL	88	50	208.08	1.93×10^{-6}
				IALM	79	50	230.73	1.30×10^{-6}
5000	100	4	0.158	SVT	215	100	2287.72	1.72×10^{-6}
				APGL	98	100	606.82	4.42×10^{-6}
				IALM	64	100	457.79	1.53×10^{-6}
8000	10	6	0.015	SVT	230	10	283.94	1.86×10^{-6}
				APGL	87	10	66.45	5.27×10^{-6}
				IALM	235	10	186.73	2.08×10^{-6}
8000	50	5	0.06	SVT	191	50	1095.10	1.61×10^{-6}
				APGL	100	50	509.78	6.16×10^{-6}
				IALM	104	50	559.22	1.36×10^{-6}
10,000	10	6	0.012	SVT	228	10	350.20	1.80×10^{-6}
				APGL	89	10	96.10	5.13×10^{-6}
				IALM	274	10	311.46	1.96×10^{-6}
10,000	50	5	0.05	SVT	192	50	1582.95	1.62×10^{-6}
				APGL	105	50	721.96	3.82×10^{-6}
				IALM	118	50	912.61	1.32×10^{-6}

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. Jolliffe IT, (1986) *Principal Component Analysis*. Springer-Verlag. https://doi.org/10.1007/978-1-4757-1904-8

2. Candès EJ, Li X, Ma Y, Wright J, (2011) Robust principal component analysis?. *J ACM* 58: 1–37. https://doi.org/10.1145/1970392.1970395
3. Wright J, Peng Y, Ma Y, Ganesh A, Rao S, (2009) Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Adv Neural Inf Process Syst* 2009: 22.
4. Grant M, Boyd S, CVX: Matlab Software for Disciplined Convex Programming (Web Page and Software), 2009. Available from: http://stanford.edu/~boyd/cvx.
5. Chandrasekharan V, Sanghavi S, Parillo P, Wilsky A, (2011) Rank-sparsity incoherence for matrix decomposition. *SIAM J Optim* 21: 572–596. https://doi.org/10.1137/090761793
6. Yin W, Hale E, Zhang Y, (2008) Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM J Optim* 19: 1107–1130. https://doi.org/10.1137/070698920
7. Beck A, Teboulle M, (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J Imaging Sci* 2: 183–202. https://doi.org/10.1137/080716542
8. Yin W, Osher S, Goldfarb D, Darbon J, (2008) Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM J Imaging Sci* 1: 143–168. https://doi.org/10.1137/070703983
9. Cai JF, Osher S, Shen Z, (2009) Linearized Bregman iterations for compressed sensing. *Math Comput* 78: 1515–1536. https://doi.org/10.1090/S0025-5718-08-02189-3
10. Cai J, Candès E, Shen Z, (2010) A singular value thresholding algorithm for matrix completion. *SIAM J Optim* 20: 1956–1982. https://doi.org/10.1137/080738970
11. Recht B, Fazel M, Parillo P, (2010) Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev* 52: 471–501. https://doi.org/10.1137/070697835
12. Candès E, Recht B, (2009) Exact matrix completion via convex optimization. *Found Comput Math* 9: 717–772. https://doi.org/10.1007/s10208-009-9045-5
13. Lin Z, Ganesh A, Wright J, Wu L, Chen M, Ma Y, Fast Convex Optimization Algorithms for Exact Recovery of a Corrupted Low-Rank Matrix. Technical Report UILU-ENG-09-2214, UIUC, August 2009. Available from: https://people.eecs.berkeley.edu/ yima/psfile/rpca_algorithms.pdf.
14. Toh KC, Yun S, (2010) An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pac J Optim* 6: 615–640.
15. Tseng P, On Accelerated Proximal Gradient Methods for Convex-Concave Optimization, 2008. Available from: https://www.mit.edu/ dimitrib/PTseng/papers/apgm.pdf.
16. Nesterov Y, (1983) A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Sov Math Dokl* 27: 372–376.
17. Fukushima M, Mine H, (1981) A generalized proximal gradient algorithm for certain nonconvex minimization problems. *Int J Syst Sci* 12: 989–1000. https://doi.org/10.1080/00207728108963798
18. Bertsekas D, (1982) *Constrained Optimization and Lagrange Multiplier Method*, Academic Press.
19. Bertsekas D, (1999) *Nonlinear Programming*. Athena Scientific.
20. Abernethy J, Bach F, Evgeniou T, Vert JP, Low-Rank Matrix Factorization with Attributes, Ecole des Mines de Paris, N24/06/MM, 2006.

21. Amit Y, Fink M, Srebro N, Ullman S, (2007) Uncovering shared structures in multiclass classification, In: *Proceedings of the Twenty-fourth International Conference on Machine Learning*, 17–24. https://doi.org/10.1145/1273496.1273499

22. Argyriou A, Evgeniou T, Pontil M, (2006) Multi-task feature learning, In: *Proceedings of Advances in Neural Information Processing Systems*, 19. https://doi.org/10.7551/mitpress/7503.003.0010

23. Mesbahi M, Papavassilopoulos GP, (1997) On the rank minimization problem over a positive semidefinite linear matrix inequality. *IEEE Trans Autom Control* 42: 239–243. https://doi.org/10.1109/9.554402

24. Tomasi C, Kanade T, (1992) Shape and motion from image streams under orthography: A factorization method. *Int J Comput Vis* 9: 137–154. https://doi.org/10.1007/BF00129684

25. Larsen RM, Lanczos Bidiagonalization with Partial Reorthogonalization, Department of Computer Science, Aarhus University, DAIMI PB-357, 1998. Available from: http://soi.stanford.edu/~rmunk/PROPACK/.

Appendices

A. Proofs and technical details for Section 3

In this appendix, we provide the mathematical details in Section 3. To prove Theorems 1 and 2, we have to prepare some results in Sections A.1 and A.2.

A.1. Relationship between primal and dual norms

Our convergence theorems require the boundedness of some sequences, which results from the following theorem.

Theorem 4. *Let $\mathcal{H}$ be a real Hilbert space endowed with an inner product $\langle\cdot,\cdot\rangle$ and a corresponding norm $\|\cdot\|$, and $y \in \partial\|x\|$, where $\partial f(x)$ is the subgradient of $f(x)$. Then $\|y\|^* = 1$ if $x \neq 0$, and $\|y\|^* \leq 1$ if $x = 0$, where $\|\cdot\|^*$ is the dual norm of $\|\cdot\|$.*

Proof. As $y \in \partial\|x\|$, we have

$$\|w\| - \|x\| \geq \langle y, w - x\rangle, \quad \forall\, w \in \mathcal{H}. \tag{A.1}$$

If $x \neq 0$, choosing $w = 0, 2x$, we can deduce that

$$\|x\| = \langle y, x\rangle \leq \|x\|\|y\|^*. \tag{A.2}$$

So $\|y\|^* \geq 1$. On the other hand, we have

$$\|w - x\| \geq \|w\| - \|x\| \geq \langle y, w - x\rangle, \quad \forall\, w \in \mathcal{H}. \tag{A.3}$$

So

$$\left\langle y, \frac{w - x}{\|w - x\|}\right\rangle \leq 1, \quad \forall\, w \neq x.$$

Therefore $\|y\|^* \leq 1$. Then we conclude that $\|y\|^* = 1$.

If $x = 0$, then (A.1) is equivalent to

$$\langle y, w\rangle \le 1, \quad \forall\, \|w\| = 1. \tag{A.4}$$

By the definition of dual norm, this means that $\|y\|^* \le 1$.

A.2. Boundedness of some sequences

With Theorem 4, we can prove the following lemmas.

Lemma 1. *The sequences $\{Y_k^*\}$, $\{Y_k\}$, and $\{\hat{Y}_k\}$ are all bounded, where $\hat{Y}_k = Y_{k-1} + \mu_{k-1}(D - A_k - E_{k-1})$.*

Proof. By the optimality of A_{k+1}^* and E_{k+1}^* we have that

$$0 \in \partial_A L(A_{k+1}^*, E_{k+1}^*, Y_k^*, \mu_k), \quad 0 \in \partial_E L(A_{k+1}^*, E_{k+1}^*, Y_k^*, \mu_k), \tag{A.5}$$

i.e.,

$$\begin{array}{l} 0 \in \partial\|A_{k+1}^*\|_* - Y_k^* - \mu_k(D - A_{k+1}^* - E_{k+1}^*), \\ 0 \in \partial\left(\|\lambda E_{k+1}^*\|_1\right) - Y_k^* - \mu_k(D - A_{k+1}^* - E_{k+1}^*). \end{array} \tag{A.6}$$

So we have that

$$Y_{k+1}^* \in \partial\|A_{k+1}^*\|_*, \quad Y_{k+1}^* \in \partial\left(\|\lambda E_{k+1}^*\|_1\right). \tag{A.7}$$

Then by Theorem 4, the sequence $\{Y_k^*\}$ is bounded[‡‡] by observing the fact that the dual norms of $\|\cdot\|_*$ and $\|\cdot\|_1$ are $\|\cdot\|_2$ and $\|\cdot\|_\infty$ [10, 13], respectively. The boundedness of $\{Y_k\}$ and $\{\hat{Y}_k\}$ can be proved similarly.

A.3. Proof of Theorem 1

Proof. By

$$\begin{array}{rcl} L(A_{k+1}^*, E_{k+1}^*, Y_k^*, \mu_k) & = & \min\limits_{A,E} L(A, E, Y_k^*, \mu_k) \\ & \le & \min\limits_{A+E=D} L(A, E, Y_k^*, \mu_k) \\ & = & \min\limits_{A+E=D} (\|A\|_* + \lambda\|E\|_1) = f^*, \end{array} \tag{A.8}$$

we have

$$\begin{array}{rcl} & & \|A_{k+1}^*\|_* + \lambda\|E_{k+1}^*\|_1 \\ & = & L(A_{k+1}^*, E_{k+1}^*, Y_k^*, \mu_k) - \dfrac{1}{2\mu_k}\left(\|Y_{k+1}^*\|_F^2 - \|Y_k^*\|_F^2\right) \\ & \le & f^* - \dfrac{1}{2\mu_k}\left(\|Y_{k+1}^*\|_F^2 - \|Y_k^*\|_F^2\right). \end{array} \tag{A.9}$$

By the boundedness of $\{Y_k^*\}$, we see that

$$\|A_{k+1}^*\|_* + \lambda\|E_{k+1}^*\|_1 \le f^* + O(\mu_k^{-1}). \tag{A.10}$$

By letting $k \to +\infty$, we have that

$$\|A^*\|_* + \lambda\|E^*\|_1 \le f^*. \tag{A.11}$$

[‡‡]A stronger result is that $\|Y_k^*\|_2 = \lambda^{-1}\|Y_k^*\|_\infty = 1$ if $A_k^* \neq 0$ and $E_k^* \neq 0$.

As $D - A^*_{k+1} - E^*_{k+1} = \mu_k^{-1}(Y^*_{k+1} - Y^*_k)$, by the boundedness of Y^*_k and letting $k \to +\infty$, we see that

$$A^* + E^* = D. \tag{A.12}$$

Therefore, (A^*, E^*) is an optimal solution to the RPCA problem.

On the other hand, by the triangular inequality of norms,

$$\begin{array}{rcl} \|A^*_{k+1}\|_* + \lambda\|E^*_{k+1}\|_1 & \geq & \|D - E^*_{k+1}\|_* + \lambda\|E^*_{k+1}\|_1 - \|D - A^*_{k+1} - E^*_{k+1}\|_* \\ & \geq & f^* - \|D - A^*_{k+1} - E^*_{k+1}\|_* \\ & = & f^* - \mu_k^{-1}\|Y^*_{k+1} - Y^*_k\|_*. \end{array} \tag{A.13}$$

So

$$\|A^*_{k+1}\|_* + \lambda\|E^*_{k+1}\|_1 \geq f^* - O(\mu_k^{-1}). \tag{A.14}$$

This together with (A.10) proves the convergence rate.

A.4. Key lemmas

First, we have the following identity.

Lemma 2.

$$\begin{array}{rl} & \|E_{k+1} - E^*\|_F^2 + \mu_k^{-2}\|Y_{k+1} - Y^*\|_F^2 \\ = & \|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2 - \|E_{k+1} - E_k\|_F^2 - \mu_k^{-2}\|Y_{k+1} - Y_k\|_F^2 \\ & -2\mu_k^{-1}(\langle Y_{k+1} - Y_k, E_{k+1} - E_k\rangle + \langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle + \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle), \end{array} \tag{A.15}$$

where (A^*, E^*) and Y^* are the optimal solutions to the RPCA problem (2) and the dual problem (9), respectively.

Proof. The identity can be routinely checked. Using $A^* + E^* = D$ and $D - A_{k+1} - E_{k+1} = \mu_k^{-1}(Y_{k+1} - Y_k)$, we have

$$\begin{array}{rl} & \mu_k^{-1}\langle Y_{k+1} - Y_k, Y_{k+1} - Y^*\rangle \\ = & -\langle A_{k+1} - A^*, Y_{k+1} - Y^*\rangle - \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle \\ = & \langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y_{k+1}\rangle - \langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle - \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle \\ = & \mu_k\langle A_{k+1} - A^*, E_{k+1} - E_k\rangle - \langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle - \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle. \end{array} \tag{A.16}$$

Then

$$\begin{array}{rl} & \|E_{k+1} - E^*\|_F^2 + \mu_k^{-2}\|Y_{k+1} - Y^*\|_F^2 \\ = & (\|E_k - E^*\|_F^2 - \|E_{k+1} - E_k\|_F^2 + 2\langle E_{k+1} - E^*, E_{k+1} - E_k\rangle) \\ & +\mu_k^{-2}(\|Y_k - Y^*\|_F^2 - \|Y_{k+1} - Y_k\|_F^2 + 2\langle Y_{k+1} - Y^*, Y_{k+1} - Y_k\rangle) \\ = & \|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2 - \|E_{k+1} - E_k\|_F^2 - \mu_k^{-2}\|Y_{k+1} - Y_k\|_F^2 \\ & +2\langle E_{k+1} - E^*, E_{k+1} - E_k\rangle + 2\mu_k^{-2}\langle Y_{k+1} - Y^*, Y_{k+1} - Y_k\rangle \\ = & \|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2 - \|E_{k+1} - E_k\|_F^2 - \mu_k^{-2}\|Y_{k+1} - Y_k\|_F^2 \\ & +2\langle E_{k+1} - E^*, E_{k+1} - E_k\rangle + 2\langle A_{k+1} - A^*, E_{k+1} - E_k\rangle \\ & -2\mu_k^{-1}(\langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle + \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle) \end{array}$$

$$
\begin{aligned}
&= \|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2 - \|E_{k+1} - E_k\|_F^2 - \mu_k^{-2}\|Y_{k+1} - Y_k\|_F^2 \\
&\quad +2\langle A_{k+1} + E_{k+1} - D, E_{k+1} - E_k\rangle \\
&\quad -2\mu_k^{-1}(\langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle + \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle) \\
&= \|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2 - \|E_{k+1} - E_k\|_F^2 - \mu_k^{-2}\|Y_{k+1} - Y_k\|_F^2 \\
&\quad -2\mu_k^{-1}\langle Y_{k+1} - Y_k, E_{k+1} - E_k\rangle - 2\mu_k^{-1}(\langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle + \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle).
\end{aligned}
$$

We then quote a classic result.

Lemma 3. *The subgradient of a convex function is a monotone operator. Namely, if f is a convex function, then*

$$\langle x_1 - x_2, g_1 - g_2\rangle \geq 0, \quad \forall g_i \in \partial f(x_i), i = 1, 2.$$

Proof. By the definition of subgradient, we have

$$f(x_1) - f(x_2) \geq \langle x_1 - x_2, g_2\rangle, \quad f(x_2) - f(x_1) \geq \langle x_2 - x_1, g_1\rangle.$$

Adding the above two inequalities proves the lemma.

Then we have the following result.

Lemma 4. *If μ_k is nondecreasing, then each entry of the following series is nonnegative, and its sum is finite:*

$$
\begin{aligned}
&\sum_{k=1}^{+\infty} \mu_k^{-1}(\langle Y_{k+1} - Y_k, E_{k+1} - E_k\rangle + \langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle + \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle) \\
&< +\infty.
\end{aligned} \tag{A.17}
$$

Proof. (A^*, E^*, Y^*) is a saddle point of the Lagrangian function

$$L(A, E, Y) = \|A\|_* + \lambda\|E\|_1 + \langle Y, D - A - E\rangle \tag{A.18}$$

of the RPCA problem (2). So we have

$$Y^* \in \partial\|A^*\|_*, \quad Y^* \in \partial(\|\lambda E^*\|_1). \tag{A.19}$$

Then by Lemma 3, $\hat{Y}_{k+1} \in \partial\|A_{k+1}\|_*$, and $Y_{k+1} \in \partial\left(\|\lambda E_{k+1}\|_1\right)$ (cf. Section A.2), we have

$$
\begin{aligned}
\langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle &\geq 0, \\
\langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle &\geq 0, \\
\langle E_{k+1} - E_k, Y_{k+1} - Y_k\rangle &\geq 0.
\end{aligned} \tag{A.20}
$$

By the above together with $\mu_{k+1} \geq \mu_k$ and (A.15), we have that $\{\|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2\}$ is non-increasing and

$$
\begin{aligned}
&2\mu_k^{-1}(\langle Y_{k+1} - Y_k, E_{k+1} - E_k\rangle + \langle A_{k+1} - A^*, \hat{Y}_{k+1} - Y^*\rangle + \langle E_{k+1} - E^*, Y_{k+1} - Y^*\rangle) \\
&\leq (\|E_k - E^*\|_F^2 + \mu_k^{-2}\|Y_k - Y^*\|_F^2) - (\|E_{k+1} - E^*\|_F^2 + \mu_{k+1}^{-2}\|Y_{k+1} - Y^*\|_F^2).
\end{aligned} \tag{A.21}
$$

So (A.17) is proven.

A.5. Proof of Theorem 2

Proof. When $\{\mu_k\}$ is upper bounded, the convergence of Algorithm 5 is already proved by He et al. In the following, we assume that $\mu_k \to +\infty$.

Similar to the proof of Lemma 4, we have

$$\sum_{k=1}^{+\infty} \mu_k^{-2} \|Y_{k+1} - Y_k\|_F^2 < +\infty.$$

So

$$\|D - A_k - E_k\|_F = \mu_k^{-1} \|Y_k - Y_{k-1}\|_F \to 0.$$

Then any accumulation point of (A_k, E_k) is a feasible solution.

On the other hand, denote the optimal objective value of the RPCA problem by f^*. As $\hat{Y}_k \in \partial \|A_k\|_*$ and $Y_k \in \partial(\lambda \|E_k\|_1)$, we have

$$\begin{aligned} & \|A_k\|_* + \lambda \|E_k\|_1 \\ \leq\ & \|A^*\|_* + \lambda \|E^*\|_1 - \langle \hat{Y}_k, A^* - A_k \rangle - \langle Y_k, E^* - E_k \rangle \\ =\ & f^* + \langle Y^* - \hat{Y}_k, A^* - A_k \rangle + \langle Y^* - Y_k, E^* - E_k \rangle - \langle Y^*, A^* - A_k + E^* - E_k \rangle \\ =\ & f^* + \langle Y^* - \hat{Y}_k, A^* - A_k \rangle + \langle Y^* - Y_k, E^* - E_k \rangle - \langle Y^*, D - A_k - E_k \rangle. \end{aligned} \tag{A.22}$$

From Lemma 4,

$$\sum_{k=1}^{+\infty} \mu_k^{-1} (\langle A_k - A^*, \hat{Y}_k - Y^* \rangle + \langle E_k - E^*, Y_k - Y^* \rangle) < +\infty. \tag{A.23}$$

As $\sum_{k=1}^{+\infty} \mu_k^{-1} = +\infty$, there must exist a subsequence (A_{k_j}, E_{k_j}) such that

$$\langle A_{k_j} - A^*, \hat{Y}_{k_j} - Y^* \rangle + \langle E_{k_j} - E^*, Y_{k_j} - Y^* \rangle \to 0.$$

Then we see that

$$\lim_{j \to +\infty} \|A_{k_j}\|_* + \lambda \|E_{k_j}\|_1 \leq f^*.$$

So (A_{k_j}, E_{k_j}) approaches to an optimal solution (A^*, E^*) to the RPCA problem. As $\mu_k \to +\infty$ and $\{Y_k\}$ are bounded, we have that $\{\|E_{k_j} - E^*\|_F^2 + \mu_{k_j}^{-2} \|Y_{k_j} - Y^*\|_F^2\} \to 0$.

On the other hand, in the proof of Lemma 4, we have shown that $\{\|E_k - E^*\|_F^2 + \mu_k^{-2} \|Y_k - Y^*\|_F^2\}$ is non-increasing. So $\|E_k - E^*\|_F^2 + \mu_k^{-2} \|Y_k - Y^*\|_F^2 \to 0$, and we have that $\lim_{k \to +\infty} E_k = E^*$. As $\lim_{k \to +\infty} D - A_k - E_k = 0$ and $D = A^* + E^*$, we see that $\lim_{k \to +\infty} A_k = A^*$.

A.6. Proof of Theorem 3

Proof. We have proved that both $\{Y_k\}$ and $\{\hat{Y}_k\}$ are bounded sequences. So there exists a constant C such that

$$\|Y_k\|_F \leq C \quad \text{and} \quad \|\hat{Y}_k\|_F \leq C.$$

Then

$$\|E_{k+1} - E_k\|_F = \mu_k^{-1} \|\hat{Y}_{k+1} - Y_{k+1}\|_F \leq 2C\mu_k^{-1}.$$

As $\sum_{k=0}^{\infty} \mu_k^{-1} < +\infty$, we see that $\{E_k\}$ is a Cauchy sequence, hence it has a limit E_∞. Then

$$\begin{aligned} \|E_\infty - E^*\|_F &= \left\| E_0 + \sum_{k=0}^{\infty} (E_{k+1} - E_k) - E^* \right\|_F \\ &\geq \|E_0 - E^*\|_F - \sum_{k=0}^{\infty} \|E_{k+1} - E_k\|_F \\ &\geq \|E_0 - E^*\|_F - 2C \sum_{k=0}^{\infty} \mu_k^{-1}. \end{aligned} \tag{A.24}$$

So if Algorithm 5 is badly initialized such that

$$\|E_0 - E^*\|_F > 2C \sum_{k=0}^{\infty} \mu_k^{-1},$$

then E_k will not converge to E^*.

A.7. Proof of Proposition 1

Proof. It is well known [10] that if the skinny SVD of A_k is $A_k = U_k \Lambda_k V_k^T$ and $U_k^\perp$ and $V_k^\perp$ are the orthogonal complements of U_k and V_k, respectively, then

$$\partial \|A_k\|_* = \{U_k V_k^T + U_k^\perp W (V_k^\perp)^T \,|\, \|W\|_2 \leq 1\}.$$

So

$$\operatorname{dist}(\partial \|A_k\|_*, S) = \min_{\|W\|_2 \leq 1} \operatorname{dist}(U_k V_k^T + U_k^\perp W (V_k^\perp)^T, S) = \min_{\|W\|_2 \leq 1} \|\pi_{\hat{\Omega}}(U_k V_k^T + U_k^\perp W (V_k^\perp)^T)\|_F.$$

Proposition 2. *Suppose the skinny SVD of A_k is $A_k = U_k \Lambda_k V_k^T$ and $U_k^\perp$ and $V_k^\perp$ are the orthogonal complements of U_k and V_k, respectively. Let the SVD of $(U_k^\perp)^T X V_k^\perp$ be $P_k \Sigma_k Q_k^T$, then*

$$\pi_{\partial \|A_k\|_*}(X) = U_k V_k^T + U_k^\perp \hat{W} (V_k^\perp)^T,$$

where $\hat{W} = P_k \min(\Sigma_k, 1) Q_k^T$.

Proof. To find $\pi_{\partial \|A_k\|_*}(X)$, we have to find $\hat{W}$ such that

$$\hat{W} = \arg \min_{\|W\|_2 \leq 1} \|X - U_k V_k^T - U_k^\perp W (V_k^\perp)^T\|_F.$$

One may check that

$$\begin{aligned} \arg \min_{\|W\|_2 \leq 1} \|X - U_k V_k^T - U_k^\perp W (V_k^\perp)^T\|_F &= \arg \min_{\|W\|_2 \leq 1} \|(U_k^\perp)^T (X - U_k V_k^T) V_k^\perp - W\|_F \\ &= \arg \min_{\|W\|_2 \leq 1} \|(U_k^\perp)^T X V_k^\perp - W\|_F. \end{aligned} \tag{A.25}$$

So $\hat{W} = P_k \min(\Sigma_k, 1) Q_k^T$.

AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)