Fast Algorithms for Recovering a Corrupted Low-Rank Matrix

Arvind Ganesh§, Zhouchen Lin†, John Wright†, Leqin Wu‡, Minming Chen¶, Yi Ma§,†

§ECE Department
University of Illinois, Urbana-Champaign
†Microsoft Research Asia
¶Institute of Computational Technology
Chinese Academy of Sciences
‡Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences

Abstract—This paper studies algorithms for solving the problem of recovering a low-rank matrix with a fraction of its entries arbitrarily corrupted. This problem can be viewed as a robust version of classical PCA, and arises in a number of application domains, including image processing, web data ranking, and bioinformatic data analysis. It was recently shown that under surprisingly broad conditions, it can be exactly solved via a convex programming surrogate that combines nuclear norm minimization and ℓ1-norm minimization. This paper develops and compares two complementary approaches for solving this convex program. The first is an accelerated proximal gradient algorithm directly applied to the primal; while the second is a gradient algorithm applied to the dual problem. Both are several orders of magnitude faster than the previous state-of-the-art algorithm for this problem, which was based on iterative thresholding. Simulations demonstrate the performance improvement that can be obtained via these two algorithms, and clarify their relative merits.

I. INTRODUCTION

Principal Component Analysis (PCA) is a popular tool for high-dimensional data analysis, with applications ranging across a wide variety of scientific and engineering fields [1]. It relies on the basic assumption that the given high-dimensional data lie near a much lower-dimensional linear subspace. Correctly estimating this subspace is crucial for reducing the dimension of the data and facilitating tasks such as processing, analyzing, compressing, or visualizing the data [1], [2].

More formally, suppose that the given data are arranged as the columns of a large matrix $D \in \mathbb{R}^{m \times n}$. Suppose that $D = A + E$, where $A$ is a rank-$r$ matrix and $E$ is a matrix whose entries are i.i.d. Gaussian random variables. In this setting, PCA seeks an optimal estimate of $A$, via the following constrained optimization:

$$\min_{A,E} \|E\|_F, \quad \text{subject to} \quad \text{rank}(A) \leq r, \quad D = A + E,$$  

where $\|\cdot\|_F$ is the Frobenius norm.

Although PCA offers the optimal estimate of the subspace when the data are corrupted by small Gaussian noise, it breaks down under large corruption, even if that corruption affects only a few of the observations. For example, even with just a single entry corrupted, the estimated $A$ obtained by classical PCA can be arbitrarily far from the true $A$. This undesirable behavior has motivated the study of the problem of recovering a low-rank matrix $A$ from a corrupted data matrix $D = A + E$, where some entries of $E$ may be of arbitrarily large magnitude.

Recently, [3] showed that under surprisingly broad conditions, one can exactly recover the low-rank matrix $A$ from $D = A + E$ with gross but sparse errors $E$ by solving the following convex optimization problem:

$$\min_{A,E} \|A\|_F + \lambda \|E\|_1, \quad \text{subject to} \quad D = A + E. \quad (2)$$

Here, $\|\cdot\|_*$ represents the nuclear norm of a matrix (the sum of its singular values), $\|\cdot\|_1$ denotes the sum of the absolute values of matrix entries, and $\lambda$ is a positive weighting parameter. In [3], this optimization is dubbed Robust PCA (RPCA), because it enables one to correctly recover underlying low-rank structure in the data, even in the presence of gross errors or outlying observations. This optimization can be easily recast as a semidefinite program and solved by an off-the-shelf interior point solver (e.g., [4]), see also [5]. However, although interior point methods offer superior convergence rates, the complexity of computing the step direction is $O(m^6)$. So they do not scale well with the size of the matrix.

In recent years, the search for more scalable algorithms for high-dimensional convex optimization problems has prompted a return to first-order methods. One striking example of this is the current popularity of iterative thresholding algorithms for ℓ1-norm minimization problems arising in compressed sensing [6]–[9]. Similar iterative thresholding techniques [10] can be applied to the problem of recovering a low-rank matrix from an incomplete (but clean) subset of its entries [11], [12]. This optimization is closely related to the RPCA problem (2), and the algorithm and convergence proof extend quite naturally to RPCA [3]. However, the iterative thresholding scheme proposed in [3] exhibits extremely slow convergence: solving one instance requires about $10^4$ iterations, each of which has the same cost as one singular value decomposition: solving one instance requires more than 8 hours on a typical PC.

In this paper, we propose two fast and scalable algorithms to solve (2). In section II, we propose a first-order accelerated
proximal gradient algorithm to directly solve the primal problem. Section III develops an entirely new algorithm to solve problem (2) via its dual. We believe that the two algorithms presented in Sections II and III represent the fastest algorithms known today for Robust PCA. We compare both algorithms in Section IV with extensive simulations on randomly generated matrices. Finally in Section V, we discuss future directions of research that could further boost the performance of the proposed algorithms.

II. THE ACCELERATED PROXIMAL GRADIENT APPROACH

We consider the following relaxed version of the RPCA optimization problem (2):

$$\min_{A,E} \mu \|A\|_* + \mu \lambda |E|_1 + \frac{1}{2} \|D - A - E\|_F^2,$$  \hspace{1cm} (3)

where $f(A, E) = \frac{1}{2} \|D - A - E\|_F^2$ penalizes violations of the equality constraint, and $\mu > 0$ is a relaxation parameter. As $\mu$ approaches 0, any solution to (3) approaches the solution set of (2). Since $f(A, E)$ is convex, smooth, and has a Lipschitz continuous gradient (with Lipschitz constant 2), the optimization problem (3) is amenable to efficient optimization by a family of optimization algorithms known as proximal gradient algorithms [7], [13], [14]. These algorithms iteratively form separable quadratic approximations to the smooth penalty term $f(A, E)$ at specially chosen points $Y_k \approx (Y_k^A, Y_k^E)$.

The main motivation for forming the separable quadratic approximation is that in many cases of interest, the iterates $(A_k, E_k)$ have a simple, or even closed-form expression. In [3], this property was exploited to give an iterative thresholding algorithm for RPCA. However, the iterative thresholding algorithm proposed there requires a very large number of iterations to converge, and hence has only limited applicability. Recently, [15] demonstrated significant improvements for matrix completion [11], [12] by combining the judicious choice of $Y_k$ suggested by [7], [16] with continuation techniques.

Here, we will see that a very similar iterative thresholding scheme, summarized as Algorithm 1, can achieve dramatically better performance, in some cases cutting the number of iterations by a factor of almost 100. The soft-thresholding scalar operator $S_\epsilon(\cdot)$ in Algorithm 1 is defined as

$$S_\epsilon(x) = \begin{cases} \text{sign}(x) (|x| - \epsilon) & \text{if } |x| > \epsilon, \\ 0, & \text{otherwise}, \end{cases}$$  \hspace{1cm} (4)

and extended to vectors and matrices by applying it element-wise.

Two key factors enable this performance gain. The first is formulating the problem within the proximal gradient framework and using the smoothed computation of $Y_k$ suggested by [7], [16]. The second is the use of continuation techniques: rather than applying the proximal gradient algorithm directly to (3), we vary $\mu$, starting from a large initial value $\mu_0$ and decreasing it geometrically $^1$ with each iteration until it reaches the floor $\mu$. We observe that this greatly reduces the number of iterations and therefore, the number of SVD computations.

Since $\mu_k$ converges to $\bar{\mu} > 0$, the proof of convergence of Algorithm 1 is very similar to the one provided for FISTA in [7]. We summarize the main result below:

**Theorem 2.1:** Let $F(X) \equiv F(A, E) \equiv \bar{\mu} \|A\|_* + \bar{\mu} \lambda |E|_1 + \frac{1}{2} \|D - A - E\|_F^2$. Then, for all $k > k_0 \geq \frac{C_1}{\log(\frac{1}{\epsilon})}$, we have

$$F(X_k) - F(X^*) \leq \frac{4 \|X_{k_0} - X^*\|_F^2}{(k - k_0 + 1)^2},$$  \hspace{1cm} (5)

where $C_1 = \log \left( \frac{\bar{\mu}}{\mu} \right)$ and $X^*$ is any solution to (3).

Thus, for any $\epsilon > 0$, when $k > k_0 + \frac{2\|X_{k_0} - X^*\|_F}{\sqrt{\epsilon}}$, we can guarantee that $F(X_k) < F(X^*) + \epsilon$.

**Algorithm 1 (Robust PCA via Accelerated Proximal Gradient)**

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

1: $A_0, A_{-1} \leftarrow 0$; $E_0, E_{-1} \leftarrow 0$; $t_0, t_{-1} \leftarrow 1$; $\bar{\mu} \leftarrow 10^{-5} \mu_0$.

2: while not converged do

3: $Y_{k}^A \leftarrow A_k + t_k^{-1} \left( A_k - A_{k-1} \right)$

4: $Y_{k}^E \leftarrow E_k + t_k^{-1} \left( E_k - E_{k-1} \right)$

5: $G_k^A \leftarrow Y_k^A - \frac{1}{2} \left( Y_k^A + Y_k^E - D \right)$

6: $G_k^E \leftarrow Y_k^E - \frac{1}{2} \left( Y_k^A + Y_k^E - D \right)$

7: $E_{k+1} \leftarrow S_{\mu_k}(G_k^E)$

8: $t_{k+1} \leftarrow 1 + \frac{\sqrt{t_k^2 + 1}}{t_k}$

9: $\mu_{k+1} \leftarrow \max(\eta \mu_k, \bar{\mu})$

10: $k \leftarrow k + 1$.

11: end while

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

III. THE DUAL APPROACH

We now consider the following dual problem of (2),

$$\max_Y \langle D, Y \rangle, \text{ subject to } J(Y) \leq 1,$$  \hspace{1cm} (6)

where

$$\langle A, B \rangle = \text{tr}(A^T B), \quad J(Y) = \max \left( \|Y\|_2, \lambda^{-1} \|Y\|_\infty \right).$$  \hspace{1cm} (7)

$\|\cdot\|_2$ represents the spectral norm, and $\|\cdot\|_\infty$ is the maximum absolute value of the matrix entries.

**Constrained Steepest Ascent:** Notice that since $J(Y)$ is positive and homogeneous and the objective function is linear, the optimal solution must lie on the manifold $S = \{Y \mid J(Y) = 1\}$. We can therefore replace the inequality constraint with an equality constraint, leading to an optimization problem on a nonlinear and non-smooth manifold, which can be solved by steepest ascent.$^2$

More formally, let $Y_k$ denote our estimate of $Y$ at iteration $k$. The steepest ascent direction $W_k$ at $Y_k$ can be obtained by projecting the gradient $D$ of the objective function (6) onto

$^1$From our experiments, we observe that $\eta = 0.9$ is a good choice.

$^2$Note that the proximal gradient algorithm cannot be directly applied to the dual problem because the sub-problem to solve is identical to the dual problem.
the tangent cone of $S$. Then we may do line search along direction $W_k$ by solving

$$\delta_k = \arg \max_{\delta \geq 0} \left\langle D, \frac{Y_k + \delta \cdot W_k}{J(Y_k + \delta \cdot W_k)} \right\rangle,$$

(8)

and updating the estimate of $Y$ as

$$Y_{k+1} = \frac{Y_k + \delta_k \cdot W_k}{J(Y_k + \delta_k \cdot W_k)},$$

(9)

where the scaling by $1/J(Y_k + \delta \cdot W_k)$ ensures that the iterate $Y_{k+1}$ lies on the manifold $S$. This yields an algorithm that provably terminates at the optimum of the dual problem.

**Theorem 3.1:** If the maximizing $\delta_k$ in (8) is equal to zero at some point $Y_k$, then $Y_k$ is the optimal solution to the dual problem (6).

**Proof:** See [17].

The key step to solve the dual problem is to find the steepest ascent direction $W_k$. To this end, we have the following result.

**Lemma 3.2:** If two cones $C_1$ and $C_2$ are polar cones to each other, and $\pi_1$ and $\pi_2$ are the projection operators onto $C_1$ and $C_2$, respectively, then for all points $P$, we have $\pi_1(P) + \pi_2(P) = P$.

**Proof:** See [17].

Based on this lemma, we may first find the projection $D_k$ of $D$ onto the normal cone $N(Y_k)$ of $S$ and obtain the steepest ascent direction $W_k$ as $W_k = D - D_k$.

**Projection onto the Normal Cone:** Thus, to solve the dual problem by steepest ascent, the remaining problem is to compute the projection $D_k$ of $D$ onto the normal cone $N(Y_k)$. By [18] Corollary 23.7.1, the normal cone $N(Y_k) = \{aX : a \geq 0, X \in \partial J(Y_k)\}$, where $\partial J(.)$ represents the subgradient of $J$. A detailed description of the projection algorithm can be found in [17]. It is interesting to note that computing the projection only requires the principal singular space that is associated with the largest singular value of $Y_k$.

**Back to the Primal Problem:** With the solution $\hat{Y}$ to (6) in hand, the two remaining KKT conditions for the primal problem (2) are $\hat{Y} \in \partial ||A||_*$ and $\lambda^{-1} \hat{Y} \in \partial |E|_1$. It is easy to see, either from the definition of these two subgradients or from more general duality considerations, that if $||\hat{Y}||_2 < 1$, then the primal problem has a degenerate solution $A = 0$ and $E = D$. Similarly, if $\lambda^{-1} ||\hat{Y}||_\infty < 1$, the solution is $A = D$ and $E = 0$. In the remaining case when $||\hat{Y}||_2 = \lambda^{-1} ||\hat{Y}||_\infty = 1$, we have the following result.

**Theorem 3.3:** Let $\hat{Y}$ be the solution to the dual problem (6), and suppose that $||\hat{Y}||_2 = \lambda^{-1} ||\hat{Y}||_\infty = 1$. Then any pair of accumulation points $\hat{A}, \hat{E}$ generated by projecting $D$ onto $N(\hat{Y})$ solve the primal problem (2).

**Proof:** See [17].

The complete optimization procedure is summarized as Algorithm 2.

**Algorithm 2 (Robust PCA via the Dual)**

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

1: $Y_0 = \text{sgn}(D)/J(\text{sgn}(D)); k \leftarrow 0$

2: while not converged do

3: Compute the projection $D_k$ of $D$ onto $N(Y_k)$:

4: if $||Y_k||_2 > \lambda^{-1} ||Y_k||_\infty$ then

5: $D_k \leftarrow \pi_2(D)$, $A \leftarrow D, E \leftarrow 0$

6: else if $\lambda^{-1} ||Y_k||_\infty > ||Y_k||_2$ then

7: $D_k \leftarrow \pi_\infty(D), A \leftarrow 0, E \leftarrow D$

8: else

9: $A \leftarrow 0, E \leftarrow 0$

10: while not converged do

11: $A \leftarrow \pi_2(D - E), E \leftarrow \pi_\infty(D - A)$

12: end while

13: $D_k \leftarrow A + E$

14: end if

15: Do line search to determine a step size $\delta_k$

16: $Y_{k+1} \leftarrow \frac{Y_k + \delta_k(D - D_k)}{J(Y_k + \delta_k(D - D_k))}$ and $k \leftarrow k + 1$

17: end while

**Output:** $(A, E)$.

**IV. SIMULATIONS**

In this section, using numerical simulations, we compare the two proposed algorithms. While the iterative thresholding algorithm proposed in [3] takes about 8 hours to recover a $800 \times 800$ matrix, the proposed algorithms can recover a $1,000 \times 1,000$ matrix in less than an hour under similar conditions.

**Simulation Conditions:** We use randomly generated square matrices for our simulations. We denote the true solution by the ordered pair $(A_0, E_0) \in \mathbb{R}^{m \times m} \times \mathbb{R}^{m \times m}$. We generate the rank-$r$ matrix $A_0$ as a product $UV^T$, where $U$ and $V$ are independent $m \times r$ matrices whose elements are i.i.d. Gaussian random variables with zero mean and unit variance. We generate $E_0$ 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 $D = A_0 + E_0$ 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. All the simulations are conducted and timed on the same Mac Pro computer with a 2.8 GHz processor, eight cores, and 10 GB of memory. We present in Table I the comparison between the proposed proximal gradient algorithm and the dual method on their ability to scale up with large matrices.

**Observations and Comparisons:** We observe that as the dimension of the problem increases, the dual approach scales better than the proximal gradient approach. This difference is mainly due to the fact that the proximal gradient algorithm does one full SVD computation per iteration, as against a partial SVD (e.g., using PROPACK [19]) done by the dual

---

3The tangent cone and the normal cone in the following should be defined for the convex set $S = \{Y | J(Y) \leq 1\}$. However, for simplicity we do not make such distinction.

4It can be shown that $A_0$ is distributed according to the random orthogonal model of rank $r$, as defined in [12].

5This is identical to the distribution used in [3].
method. This is because the rank of \( A_k \) during the iteration of Algorithm 1 does not always increase monotonically, making it difficult to predict the number of singular values that need to be computed. On the other hand, the number of iterations taken by the proximal gradient algorithm to optimality is less vulnerable to changes in dimension and to the setting of the problem (almost always around 128 iterations), which might be attributed to its \( O(k^{-2}) \) convergence rate. We also see in Table I that the average number of iterations taken by the dual approach is significantly increased when \( \|E_0\|_0 \) or rank(\( A_0 \)) increases.

V. CONCLUSION

In this paper, we have proposed two first-order algorithms for solving the Robust PCA problem (2), one for the primal and the other for the dual. The proposed algorithms and techniques can be easily modified for solving the slightly simpler matrix completion problem \([11],[12]\). Both algorithms are significantly faster than the previous state-of-the-art algorithms based on iterative thresholding \([3]\). Nevertheless, the dual method is potentially more scalable than the proximal gradient method as in principle it does not require a full SVD computation at each iteration. In addition, although the proximal gradient algorithm cannot be applied to the dual method in its current form, it is possible that the proximal gradient technique and the continuation technique, in other forms, can be applied to speed up the dual method too.

The proposed algorithms are already sufficient to solve robust PCA problems that arise in image processing and computer vision, which typically involve matrices of sizes up to a few thousand, in a reasonable amount of time. However, both algorithms may be unsuitable for direct use in data mining applications (like web search and ranking), involving much larger matrices (say \( m \geq 10^5 \)). Problems of such large scale demand better hardware and possibly an efficient distributed algorithm to solve (2). There has been significant progress recently in developing parallelized algorithms for SVD computations \((e.g.,[20])\).

### Table I

<table>
<thead>
<tr>
<th>( m )</th>
<th>( \text{rank}(A_k) )</th>
<th>( |E_0|_0 )</th>
<th>( |A_k-A_0|_F )</th>
<th>no. of iterations</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>0.05</td>
<td>0.05</td>
<td>8.6 ( \times 10^{-6} )</td>
<td>1.0 ( \times 10^{-4} )</td>
<td>126</td>
</tr>
<tr>
<td>1,000</td>
<td>0.05</td>
<td>0.1</td>
<td>9.9 ( \times 10^{-6} )</td>
<td>9.3 ( \times 10^{-6} )</td>
<td>129</td>
</tr>
<tr>
<td>1,000</td>
<td>0.1</td>
<td>0.1</td>
<td>7.6 ( \times 10^{-6} )</td>
<td>1.1 ( \times 10^{-4} )</td>
<td>132</td>
</tr>
<tr>
<td>1,500</td>
<td>0.05</td>
<td>0.05</td>
<td>7.1 ( \times 10^{-6} )</td>
<td>6.9 ( \times 10^{-6} )</td>
<td>126</td>
</tr>
<tr>
<td>1,500</td>
<td>0.05</td>
<td>0.1</td>
<td>8.1 ( \times 10^{-6} )</td>
<td>8.1 ( \times 10^{-6} )</td>
<td>129</td>
</tr>
<tr>
<td>1,500</td>
<td>0.1</td>
<td>0.1</td>
<td>6.2 ( \times 10^{-6} )</td>
<td>8.7 ( \times 10^{-6} )</td>
<td>132</td>
</tr>
<tr>
<td>2,000</td>
<td>0.05</td>
<td>0.05</td>
<td>6.2 ( \times 10^{-6} )</td>
<td>4.6 ( \times 10^{-6} )</td>
<td>126</td>
</tr>
<tr>
<td>2,000</td>
<td>0.05</td>
<td>0.1</td>
<td>7.0 ( \times 10^{-6} )</td>
<td>1.2 ( \times 10^{-5} )</td>
<td>129</td>
</tr>
<tr>
<td>2,000</td>
<td>0.1</td>
<td>0.1</td>
<td>5.4 ( \times 10^{-6} )</td>
<td>7.4 ( \times 10^{-6} )</td>
<td>132</td>
</tr>
</tbody>
</table>

Scalability of the APG and the Dual Methods. For all the instances, the matrix \( A \) is correctly estimated with relatively high accuracy, and the rank is correctly recovered.

### References