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 Computing 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 ℓ^1-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:

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

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 $\hat{A}$ 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:

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

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 λ 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 ℓ^1-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. Hence, even for matrix sizes as small as 800×800, the algorithm 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):

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

where $f(A,E) \doteq \frac{1}{2}\|D-A-E\|_F^2$ penalizes violations of the equality constraint, and $\mu > 0$ is a relaxation parameter. As μ 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 $Y_k \doteq (Y_k^A, Y_k^E)$.

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 $\mathcal{S}(.)$ in Algorithm 1 is defined as

$$\mathcal{S}_\varepsilon[x] \doteq \begin{cases} \text{sign}(x)\,(|x| - \varepsilon) & \text{if } |x| > \varepsilon, \\ 0, & \text{otherwise,} \end{cases} \qquad (4)$$

and extended to vectors and matrices by applying it elementwise.

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 μ, starting from a large initial value μ_0 and decreasing it geometrically [1] with each iteration until it reaches the floor $\bar{\mu}$. We observe that this greatly reduces the number of iterations and therefore, the number of SVD computations.

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

Since μ_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 $F(X) \equiv F(A,E) \doteq \bar{\mu}\|A\|_* + \bar{\mu}\lambda|E|_1 + \frac{1}{2}\|D-A-E\|_F^2$. Then, for all $k > k_0 \doteq \frac{C_1}{\log(\frac{1}{\eta})}$, we have

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

where $C_1 = \log\left(\frac{\mu_0}{\bar{\mu}}\right)$ 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 $D \in \mathbb{R}^{m\times n}$, λ.

$A_0, A_{-1} \leftarrow 0$; $E_0, E_{-1} \leftarrow 0$; $t_0, t_{-1} \leftarrow 1$; $\bar{\mu} \leftarrow 10^{-5}\,\mu_0$.
while not converged **do**
  $Y_k^A \leftarrow A_k + \frac{t_{k-1}-1}{t_k}(A_k - A_{k-1})$
  $Y_k^E \leftarrow E_k + \frac{t_{k-1}-1}{t_k}(E_k - E_{k-1})$.
  $G_k^A \leftarrow Y_k^A - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.
  $(U,S,V) \leftarrow \text{svd}(G_k^A)$, $A_{k+1} = U\mathcal{S}_{\frac{\mu_k}{2}}[S]V^T$.
  $G_k^E \leftarrow Y_k^E - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.
  $E_{k+1} = \mathcal{S}_{\frac{\lambda\mu_k}{2}}[G_k^E]$.
  $t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$.
  $\mu_{k+1} \leftarrow \max(\eta\,\mu_k, \bar{\mu})$.
  $k \leftarrow k+1$.
end while

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

III. The Dual Approach

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

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

where

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

$\|\cdot\|_2$ 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 $S = \{Y | J(Y) = 1\}$. 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

[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.[3] Then we may do line search along direction W_k by solving

$$\delta_k = \arg\max_{\delta \geq 0} \left\langle D, \frac{Y_k + \delta \cdot W_k}{J\left(Y_k + \delta \cdot W_k\right)} \right\rangle, \tag{8}$$

and updating the estimate of Y as

$$Y_{k+1} = \frac{Y_k + \delta_k \cdot W_k}{J\left(Y_k + \delta_k \cdot W_k\right)}, \tag{9}$$

where the scaling by $1/J\left(Y_k + \delta \cdot W_k\right)$ 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 δ_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 π_1 and π_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.

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

Algorithm 2 (Robust PCA via the Dual)

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

$Y_0 = \mathrm{sgn}(D)/J(\mathrm{sgn}(D))$; $k \leftarrow 0$.
while not converged **do**
  Compute the projection D_k of D onto $N(Y_k)$:
  if $\|Y_k\|_2 > \lambda^{-1}|Y_k|_\infty$ **then**
    $D_k \leftarrow \pi_2(D)$, $A \leftarrow D$, $E \leftarrow 0$.
  else if $\lambda^{-1}|Y_k|_\infty > \|Y_k\|_2$ **then**
    $D_k \leftarrow \pi_\infty(D)$, $A \leftarrow 0$, $E \leftarrow D$.
  else
    $A \leftarrow 0$, $E \leftarrow 0$.
    while not converged **do**
      $A \leftarrow \pi_2(D - E)$, $E \leftarrow \pi_\infty(D - A)$.
    end while
    $D_k \leftarrow A + E$.
  end if
  Do line search to determine a step size δ_k.
  $Y_{k+1} \leftarrow \frac{Y_k + \delta_k(D - D_k)}{J(Y_k + \delta_k(D - D_k))}$ and $k \leftarrow k + 1$.
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 800×800 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.[4] 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 $[-500, 500]$.[5] The matrix $D \doteq A_0 + E_0$ 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

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

[5]This is identical to the distribution used in [3].

m	$\frac{\mathrm{rank}(A_0)}{m}$	$\frac{\|E_0\|_0}{m^2}$	$\frac{\|\hat{A}-A_0\|_F}{\|A_0\|_F}$		no. of iterations		time (s)	
			APG	Dual	APG	Dual	APG	Dual
1,000	0.05	0.05	8.6×10^{-6}	1.0×10^{-4}	126	316	1,600	1,740
1,000	0.05	0.1	9.9×10^{-6}	9.3×10^{-5}	129	312	1,640	2,440
1,000	0.1	0.1	7.6×10^{-6}	1.1×10^{-4}	132	573	1,590	7,590
1,500	0.05	0.05	7.1×10^{-6}	6.9×10^{-5}	126	284	5,750	4,270
1,500	0.05	0.1	8.1×10^{-6}	8.1×10^{-5}	129	374	5,900	6,920
1,500	0.1	0.1	6.2×10^{-6}	8.7×10^{-5}	132	556	5,720	21,600
2,000	0.05	0.05	6.2×10^{-6}	4.6×10^{-5}	126	336	14,300	10,000
2,000	0.05	0.1	7.0×10^{-6}	1.2×10^{-4}	129	495	14,700	15,100
2,000	0.1	0.1	5.4×10^{-6}	7.4×10^{-5}	132	622	14,300	51,100

TABLE I

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.

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 $\mathrm{rank}(A_0)$ 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]). For the proximal gradient method, the thresholding steps in each iteration are inherently parallel. For the dual method, the computation of the principal singular spaces is much more readily parallelizable than the partial SVD. It remains to see which algorithm is more suitable for parallel or distributed implementation for solving large-scale Robust PCA problems.

REFERENCES

[1] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 1986.

[2] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 1936.

[3] J. Wright, A. Ganesh, S. Rao, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," submitted to *Journal of the ACM*, May 2009.

[4] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming (web page and software)," http://stanford.edu/∼boyd/cvx, June 2009.

[5] V. Chandrasekharan, S. Sanghavi, P. Parillo, and A. Wilsky, "Rank-sparsity incoherence for matrix decomposition," preprint, 2009.

[6] W. Yin, E. Hale, and Y. Zhang, "Fixed-point continuation for ℓ_1-minimization: methodology and convergence," preprint, 2008.

[7] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, Mar 2009.

[8] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing," *SIAM Journal on Imaging Sciences*, vol. 1, no. 1, pp. 143–168, 2008.

[9] J.-F. Cai, S. Osher, and Z. Shen, "Linearized Bregman iterations for compressed sensing," *Math. Comp.*, vol. 78, pp. 1515–1536, 2009.

[10] J. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," preprint, Sep 2008.

[11] B. Recht, M. Fazel, and P. Parillo, "Guaranteed minimum rank solution of matrix equations via nuclear norm minimization," submitted to *SIAM Review*, 2008.

[12] E. Candès and B. Recht, "Exact matrix completion via convex optimization," preprint, 2008.

[13] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," submitted to *SIAM Journal on Optimization*, May 2008.

[14] M. Fukushima and H. Mine, "A generalized proximal gradient algorithm for certain nonconvex minimization problems," *International Journal of Systems Science*, vol. 12, pp. 989–1000, 1981.

[15] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems," preprint, Apr 2009.

[16] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[17] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," University of Illinois Technical report UILU-ENG-09-2214, Aug 2009.

[18] R. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

[19] R. M. Larsen, "Lanczos bidiagonalization with partial reorthogonalization," Department of Computer Science, Aarhus University, Technical report, DAIMI PB-357, Sep 1998.

[20] T. Konda, M. Takata, M. Iwasaki, and Y. Nakamura, "A new singular value decomposition algorithm suited to parallelization and preliminary results," in *ACST'06: Proceedings of the 2nd IASTED Intl. Conf. on Advances in Computer Science and Technology*, 2006, pp. 79–84.