FAST CONVEX OPTIMIZATION ALGORITHMS FOR EXACT RECOVERY OF A CORRUPTED LOW-RANK MATRIX

ZHOUCHEN LIN[†], ARVIND GANESH[‡], JOHN WRIGHT[†],
LEQIN WU[§], MINMING CHEN[¶], AND YI MA[†‡]

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.

Key words. Principal Component Analysis, Convex optimization, Nuclear norm minimization, Duality, Proximal gradient algorithms.

AMS subject classifications. 15A03, 15A60, 90C25

1. 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. 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 [12, 7].

More formally, suppose that the given data are arranged as the columns of a large matrix $D \in \mathbb{R}^{m\times n}$. Classical PCA assumes that this data matrix was generated by perturbing a matrix $A \in \mathbb{R}^{m\times n}$ whose columns lie on a subspace of dimension $r \ll \min(m,n)$. In other words, $D = A + E$, 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:

$$\min_{A,E} \|E\|_F, \quad \text{subject to} \quad \operatorname{rank}(A) \le r, \quad D = A + E, \tag{1.1}$$

where $\|\cdot\|_F$ is the Frobenius norm. It is well-known that this problem can be efficiently solved by simply computing the Singular Value Decomposition (SVD) of D. The optimal estimate of the low-rank matrix A is simply the projection of the columns of D onto the subspace spanned by the r principal left singular vectors of D [12].

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

[†]Microsoft Research Asia

[‡]ECE Department, University of Illinois at Urbana-Champaign

[§]Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences

[¶]Institute of Computing Technology, Chinese Academy of Sciences

Corresponding author: Arvind Ganesh, Room 146 Coordinated Science Laboratory, 1308 West Main Street, Urbana, IL 61801. Email: abalasu2@illinois.edu

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, [21] 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:

$$\min_{A,E} \|A\|_* + \lambda |E|_1, \quad \text{subject to} \quad D = A + E. \tag{1.2}$$

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 [21], 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.,* [10]), see also [6]. 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. On a typical PC, generic interior point solvers are currently limited to matrices of dimension $m \approx 100$. Modern data processing applications demand solutions to much larger scale problems. For instance, applications in image and video processing often involve matrices of dimension $m = 10^3$ to 10^4; applications in web search and bioinformatics can involve matrices of dimension $m = 10^6$ and beyond.

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 [22, 1, 23, 4]. Similar iterative thresholding techniques [3] can be applied to the problem of recovering a low-rank matrix from an incomplete (but clean) subset of its entries [17, 5]. This optimization is closely related to the RPCA problem (1.2), and the algorithm and convergence proof extend quite naturally to RPCA [21]. However, the iterative thresholding scheme proposed in [21] 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, our goal is to develop faster and more scalable algorithms, by further studying the convex optimization problem (1.2) associated with Robust PCA. In Section 2, we propose a first-order accelerated proximal gradient algorithm for this problem. The proposed algorithm is a direct application of the FISTA framework introduced by [1], coupled with a fast continuation technique.[1]

Section 3 develops an entirely new algorithm to solve problem (1.2) via its dual. This algorithm efficiently solves the dual problem and subsequently computes the solution to the primal. Given that in the literature most first-order methods for solving this type of optimization problems are now based on proximal gradient, this new dual algorithm certainly offers some new perspectives and ideas, at least to the specific problem at hand. In fact, unlike the proximal gradient approach, the new

[1] Similar techniques have been applied to the matrix completion problem by [19].

dual algorithm does not depend on computing full SVD and hence in theory could be more scalable.

We believe that the two algorithms presented in Sections 2 and 3 represent the fastest algorithms known today for Robust PCA. We compare both algorithms in Section 4 with extensive simulations on randomly generated matrices. Finally in Section 5, we discuss future directions of research that could further boost the performance of the proposed algorithms.

2. The Accelerated Proximal Gradient Approach. The Robust PCA optimization (1.2) is a special case of a more general family of optimization problems of the form

$$\min_{X\in\mathcal{H}} \quad g(X), \quad \text{subject to} \quad \mathcal{A}(X) = b, \tag{2.1}$$

where $\mathcal{H}$ is a real Hilbert space equipped with a norm $\|\cdot\|$, g is a continuous convex function, $\mathcal{A}$ is a linear map, and b is an observation. It is often computationally expedient to relax the equality constraint in (2.1) and instead solve

$$\min_{X\in\mathcal{H}} \quad F(X) \doteq \mu\, g(X) + f(X), \tag{2.2}$$

where $f(X) \doteq \frac{1}{2}\|\mathcal{A}(X) - b\|^2$ penalizes violations of the equality constraint and $\mu > 0$ is a relaxation parameter. As μ approaches 0, any solution to (2.2) approaches the solution set of (2.1). The penalty function $f(\cdot)$ is convex and smooth, with Lipschitz continuous gradient: $\|\nabla f(X_1) - \nabla f(X_2)\| \le L_f\|X_1 - X_2\|$, where ∇f is the Fréchet derivative of f, identifiable as an element in $\mathcal{H}$. The Lipschitz constant L_f is simply the square of the operator norm of the linear map $\mathcal{A}$. Because of this Lipschitz property, (2.2) is amenable to efficient optimization by a family of optimization algorithms known as *proximal gradient algorithms* [20, 1].

2.1. General Formulation. Instead of directly minimizing $F(X)$, proximal gradient algorithms minimize a sequence of separable quadratic approximations to $F(X)$, denoted as $Q(X, Y)$, formed at specially chosen points Y:

$$Q(X, Y) \doteq f(Y) + \langle \nabla f(Y), X - Y\rangle + \frac{L_f}{2}\|X - Y\|^2 + \mu g(X). \tag{2.3}$$

It is easy to show that for any Y, $Q(X, Y)$ upper bounds $F(X)$. Moreover, if we define $G \doteq Y - \frac{1}{L_f}\nabla f(Y)$, then

$$\arg\min_X\, Q(X, Y) = \arg\min_X \left\{\mu g(X) + \frac{L_f}{2}\,\|X - G\|^2\right\}. \tag{2.4}$$

To solve (2.2), one may repeatedly set $X_{k+1} = \arg\min_X Q(X, Y_k)$, with Y_k chosen based on $X_0, \cdots, X_k$. 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 [8]) can be interpreted as a gradient algorithm, and results in a convergence rate no worse than $O(k^{-1})$ [1]. However, in the smooth case $g(X) \equiv 0$, [15] 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 $O(k^{-2})$. Recently, [1] extended this scheme to the nonsmooth setting ($g(X) \not\equiv 0$), again demonstrating a convergence rate of $O(k^{-2})$. Both variants can be considered special cases of a general proximal gradient algorithm, stated more precisely as Algorithm 1.

Algorithm 1 (General Proximal Gradient Algorithm)

1: **while** not converged **do**
2: $Y_k \leftarrow X_k + \frac{t_{k-1}-1}{t_k}\,(X_k - X_{k-1})$.
3: $G_k \leftarrow Y_k - \frac{1}{L_f}\nabla f(Y_k)$.
4: $X_{k+1} \leftarrow \arg\min_X \left\{\mu g(X) + \frac{L_f}{2}\,\|X - G_k\|^2\right\}$.
5: $t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$, $k \leftarrow k+1$.
6: **end while**

2.2. Robust PCA by Accelerated Proximal Gradient. The main motivation for forming the separable quadratic approximation in Algorithm 1 is that in many cases of interest, the minimizer X_{k+1} has a simple, or even closed-form expression. For example, when $\mathcal{H}$ is an Euclidean space and $g(\cdot)$ is the ℓ^1 norm, X_{k+1} is given by soft-thresholding the entries of G_k. More formally, for $x \in \mathbb{R}$ and $\varepsilon > 0$, let

$$\mathcal{S}_\varepsilon[x] \doteq \begin{cases} x-\varepsilon, & \text{if } x > \varepsilon, \\ x+\varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{otherwise,} \end{cases} \tag{2.5}$$

and extend this operator to vectors and matrices by applying it elementwise. Then in this notation, $X_{k+1} = \mathcal{S}_{\frac{\mu}{L_f}}[G_k]$. This property has been widely exploited in the compressed sensing literature [22, 1, 23, 4].

If instead, $\mathcal{H}$ is the space of same-sized matrices endowed with the Frobenius norm $\|\cdot\|_F$ (we keep this assumption in the sequel) and $g(\cdot)$ is the matrix nuclear norm, then X_{k+1} can still be efficiently computed, now by soft-thresholding the singular values. If $G_k = U\Sigma V^T$ is the SVD of G_k, then

$$X_{k+1} = U\mathcal{S}_{\frac{\mu}{L_f}}(\Sigma)V^T. \tag{2.6}$$

Based on this property, [3] introduced an iterative thresholding algorithm for matrix completion. Recently, [19] demonstrated significant improvements by combining the judicious choice of Y_k suggested by [15, 1] with continuation techniques.

The Robust PCA problem (1.2) combines aspects of both of these lines of work. Here, our iterates X_k are ordered pairs $(A_k, E_k) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n}$, and $g(X_k) = \|A_k\|_* + \lambda|E_k|_1$. The relaxation (2.2) is then

$$\min_{A,E} \quad F(X) \;\doteq\; \mu\,\|A\|_* + \mu\,\lambda\,|E|_1 + \frac{1}{2}\,\|D - A - E\|_F^2. \tag{2.7}$$

Here, again, the iterate X_{k+1} has a simple expression. Write $G_k \doteq (G_k^A, G_k^E) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n}$, and let USV^T be the singular value decomposition of G_k^A. Then (noticing that for this problem the Lipschitz constant $L_f = 2$)

$$A_{k+1} = U\mathcal{S}_{\frac{\mu}{2}}[S]V^T \quad \text{and} \quad E_{k+1} = \mathcal{S}_{\frac{\lambda\mu}{2}}[G_k^E]. \tag{2.8}$$

In [21], 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.

Here, we will see that a very similar iterative thresholding scheme, summarized as Algorithm 2, can achieve dramatically better performance, in some cases cutting

the number of iterations by a factor of almost 100. 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 [15, 1]. The second is the use of continuation techniques: rather than applying the proximal gradient algorithm directly to (2.7), we vary μ, starting from a large initial value μ_0 and decreasing it geometrically 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. Since μ_k converges to $\bar{\mu} > 0$, the proof of convergence of Algorithm 2 is very similar to the one provided for FISTA in [1]. 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* $k > k_0 \doteq \frac{C_1}{\log\left(\frac{1}{\eta}\right)}$, *we have*

$$F(X_k) - F(X^*) \leq \frac{4\,\|X_{k_0} - X^*\|_F^2}{(k - k_0 + 1)^2}, \tag{2.9}$$

where $C_1 = \log\left(\frac{\mu_0}{\bar{\mu}}\right)$ *and* X^* *is any solution to* (2.7).

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 2 (Robust PCA via Accelerated Proximal Gradient)

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

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

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

2.3. Implementation Issues.

Stopping Criterion. The stopping criterion for Algorithm 2 is identical to the one proposed in [19]. We define

$$S_{k+1}^A \doteq 2\,\left(Y_k^A - A_{k+1}\right) + \left(A_{k+1} + E_{k+1} - Y_k^A - Y_k^E\right), \tag{2.10}$$

$$S_{k+1}^E \doteq 2\,\left(Y_k^E - E_{k+1}\right) + \left(A_{k+1} + E_{k+1} - Y_k^A - Y_k^E\right). \tag{2.11}$$

We terminate the iteration when $\|S_{k+1}\|$ is less than some pre-defined tolerance, where $S_{k+1} \doteq (S_{k+1}^A, S_{k+1}^E)$ and $\|S_{k+1}\|^2 = \|S_{k+1}^A\|_F^2 + \|S_{k+1}^A\|_F^2$. The interpretation here is that $\|S_{k+1}\|$ is an upper bound to the distance between the origin and the set of subgradients of the cost function in (2.7) at (A_{k+1}, E_{k+1}).

Effect of Line Search Techniques. The worst-case iteration complexity predicted by Theorem 2.1 is no better than that for the case when the sequence μ_k is constant ($= \bar{\mu}$) for all k. However, in practice, we observe that implementing the continuation technique outlined in step 9 of Algorithm 2 greatly reduces the number of iterations. [19] also proposes a linesearch-like procedure to further accelerate their matrix completion algorithm. Our experiments suggest that for RPCA, this does not always reduce the overall computation time, since the line-search often increases the number of SVD computations per iteration.

Details of the Continuation. The sequence (A_k, E_k) generated by Algorithm 2 gets arbitrarily close to the optimal solution set of (2.7). In addition, the smaller the $\bar{\mu}$, the closer is our solution to the optimal solution set of (1.2). We find empirically that a choice of $\mu_0 = 0.99\,\|D\|_2$ and $\delta \leq 10^{-5}$ is sufficient for most practical purposes, where $\|\cdot\|_2$ is the spectral norm.

Empirically, we find that convergence is typically very slow for $\eta \in (0, 0.5)$. The reason for this is that once $\mu_k = \bar{\mu}$, for very small $\bar{\mu}$, the thresholding operator $\mathcal{S}_{\frac{\bar{\mu}}{2}}(\cdot)$ is close to the identity operator. Thus, subsequent iterations converge very slowly to the optimal solution. From experiments, we find that $\eta = 0.9$ is a good choice.

Computing the SVD. The key computational bottleneck of Algorithm 2 is the SVD required by every iteration. However, due to the singular value thresholding step at every iteration, we observe that a full SVD computation is not always necessary, especially in the first few iterations when μ_k is quite large. The algorithm can be potentially sped up by computing a partial SVD, instead of the full SVD, using publicly available packages like PROPACK [14]. Our current implementation does not adopt partial SVD because the rank of A_k during the iteration of Algorithm 2 does not always increase monotonically, making it difficult to predict how many singular values need to be computed.

3. The Dual Approach. Although the dependence of the accelerated proximal gradient algorithm on the full SVD can be relieved by exploring a workable prediction strategy and then using the partial SVD, our numerical tests show that the partial SVD by PROPACK [14] may still be slower than the full SVD, when the number of singular values/vectors to compute exceeds some relatively small threshold, say $0.15m$. It is therefore interesting to develop algorithms that could be truly independent of computing the SVD.

3.1. Robust PCA via the Dual. We observe that the dual norm of the nuclear norm is the spectral norm $\|\cdot\|_2$, which only depends on the largest singular value/vectors and in principle can be computed without the SVD. This observation leads us to consider the dual problem of (1.2),

$$\max_Y \langle D, Y\rangle, \quad \text{subject to} \quad J(Y) \leq 1, \tag{3.1}$$

where

$$\langle A, B\rangle = \mathrm{tr}(A^T B), \quad J(Y) = \max\left(\|Y\|_2, \lambda^{-1}|Y|_\infty\right), \tag{3.2}$$

and $|\cdot|_\infty$ is the maximum absolute value of the matrix entries. Problem (3.1) actually finds the dual D^* of D with respect to the matrix norm $\|X\|_J = J(X)$.

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 (3.1) onto the tangent cone of S.[3] Then we may do line search along direction W_k by solving

$$\delta_k = \arg\max_{\delta \geq 0} \left\langle D, \frac{Y_k + \delta \cdot W_k}{J\left(Y_k + \delta \cdot W_k\right)} \right\rangle, \tag{3.3}$$

and updating the estimate of Y as

$$Y_{k+1} = \frac{Y_k + \delta_k \cdot W_k}{J\left(Y_k + \delta_k \cdot W_k\right)}, \tag{3.4}$$

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* (3.3) *is equal to zero at some point Y_k, then Y_k is the optimal solution to the dual problem* (3.1).

Proof. See Appendix A.2. ∎

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

PROPOSITION 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 point P*

$$\pi_1(P) + \pi_2(P) = P. \tag{3.5}$$

Proof. See Appendix A.1. ∎

Based on this proposition, 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 is determined by the subgradient of J:

$$N(Y_k) = \left\{aX : a \geq 0, X \in \partial J(Y_k)\right\}. \tag{3.6}$$

Because $J(Y)$ is the maximum of two convex functions, we have [2]:

$$\partial J(Y_k) = \begin{cases} \partial||Y_k||_2, & \text{if } \partial||Y_k||_2 > \lambda^{-1}|Y_k|_\infty, \\ \partial(\lambda^{-1}|Y_k|_\infty), & \text{if } \partial||Y_k||_2 < \lambda^{-1}|Y_k|_\infty, \\ \text{co}\{\partial||Y_k||_2, \partial(\lambda^{-1}|Y_k|_\infty)\}, & \text{if } \partial||Y_k||_2 = \lambda^{-1}|Y_k|_\infty. \end{cases} \tag{3.7}$$

where "co" denotes the convex hull. Thus, in the first case we must compute the projection $\pi_2(D)$ of D onto the cone $N_2(Y_k)$ generated by $\partial\|\cdot\|_2$ at Y_k. In the

[2]Note that the proximal gradient algorithm (Algorithm 1) cannot be directly applied to the dual problem because the sub-problem to solve is identical to the dual problem.

[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.

second case, we must compute the projection onto the cone $N_\infty(Y_k)$ generated by the subgradient of $|\cdot|_\infty$ at Y_k. These projections are both straightforward and efficient to compute; readers unfamiliar with their details may consult Appendix A.3. Note that computing $\pi_2(D)$ only requires the principal singular space that is associated to the *largest* singular value of Y_k, which is *known* to be 1.

In the third, and most complicated, case, $N(Y_k) = N_2(Y_k) + N_\infty(Y_k)$. The projection of D onto $N(Y_k)$ can be accomplished by alternating between projection onto $N_2(Y_k)$ and projection onto $N_\infty(Y_k)$. We initialize $E_0 \leftarrow 0$ and $i \leftarrow 0$ and then repeatedly set

$$\begin{array}{rcl} A_{i+1} & \leftarrow & \pi_2(D - E_i), \\ E_{i+1} & \leftarrow & \pi_\infty(D - A_{i+1}), \\ i & \leftarrow & i + 1. \end{array} \tag{3.8}$$

The above alternating projection algorithm is guaranteed to yield D_k thanks to the following theorem:

THEOREM 3.3. *The sequence $A_i + E_i$ converges to the projection of D onto $N(Y_k)$.*

Proof. See Appendix A.4. ∎

Note that the alternating projection algorithm (3.8) is valid for projecting onto the sum of general cones.

Back to the Primal Problem. Theorems 3.1 and 3.3 give a solution $\hat{Y}$ to the dual problem (3.1). With the solution $\hat{Y}$ in hand, the two remaining KKT conditions for the primal problem (1.2) are

$$\hat{Y} \in \partial\|A\|_* \quad \text{and} \quad \lambda^{-1}\hat{Y} \in \partial|E|_1. \tag{3.9}$$

It is easy to see, either from the definition of these two subgradients or from more general duality considerations in Theorem A.2 of Appendix A.5, 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

THEOREM 3.4. *Let $\hat{Y}$ be the solution to the dual problem* (3.1)*, 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 $N(\hat{Y})$ via the alternating projection algorithm* (3.8) *solve the primal problem* (1.2)*.*

Proof. By Theorem 3.1, the steepest ascent terminates at a point $\hat{Y}$ at which $D \in N(\hat{Y})$. Theorem 3.3 showed that every pair of accumulation points satisfies $\hat{A} + \hat{E} = D$, and hence satisfies the equality constraint for the primal problem (1.2). Observe that

$$\hat{A} \in N_2(\hat{Y}) \quad \text{and} \quad \hat{E} \in N_\infty(\hat{Y})$$

implies

$$\hat{Y} \in \partial\|\hat{A}\|_* \quad \text{and} \quad \lambda^{-1}\hat{Y} \in \partial|\hat{E}|_1 \tag{3.10}$$

by Theorem A.2, (3.6), and $\|\hat{Y}\|_2 = \lambda^{-1}|\hat{Y}|_\infty = 1$. So $(\hat{A}, \hat{E})$ is a solution to problem (1.2). ∎

Thus, in the course of solving the dual problem, we also obtain the solution to the primal problem. The complete optimization procedure is summarized as Algorithm 3 below.

Algorithm 3 (Robust PCA via the Dual)

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

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

3.2. Implementation Issues.

Evaluating J. When computing $J(\cdot)$, the computation of the 2-norm accounts for most of the time. As $J(\cdot)$ will be evaluated many times, computing the 2-norm efficiently is necessary. Although the power method [9] is often advocated in the literature, it suffers from slow convergence when the gap between the largest and the second largest singular values is small, which usually happens in practice. So we choose to use PROPACK [14] to output the largest singular value. It turns out to be consistently faster than the power method. Note that PROPACK can be easily specialized for computing the largest singular value *only*.

Determining Equality of the Two Norms. When finding the steepest ascent direction, we have to decide whether $\|Y_k\|_2$ is equal to $\lambda^{-1}|Y_k|_\infty$. This is fulfilled by checking whether the discrepancy between them exceeds 10^{-4}.

Computing the Principal Singular Spaces. Currently, we compute the principal singular spaces associated with the largest singular value 1 of Y_k by letting PROPACK output a predicted number of leading singular vectors. The prediction is possible because we have observed that the dimension of the principal singular spaces associated with the largest singular value 1 of Y_k is always increasing [4]. However, we have to emphasize that computing the principal singular spaces associated to the *known* largest singular value should be an easier problem than the partial SVD that computes the principal singular space associated to *unknown* and possibly *different* leading singular values. We are currently exploring better methods to replace PROPACK.

[4] Although currently we do not prove this, an intuitive explanation is possible: the manifold S has a lot of "ridges" and "corners" and the objective function is linear; so it is very likely that the solution is at some "ridges" or "corners"; each line search moves Y_k to a better estimate, which should be at "ridges" or "corners" of higher singularity; the singularity is measurable by the complexity of the normal cone, which is directly related to the dimension of the principal singular space.

Line Search. It is well known that gradient ascent/descent produces zigzagged solution trajectories [16]. Therefore, it is unnecessary to do exact line search as suggested by (3.3). So we adopt the gradient method recommended in [16] (page 304), which is based on Armijo's rule. Moreover, it is also unnecessary to compute the exact steepest ascent direction. So we also perform inexact projection onto the normal cone: when projecting onto $N_\infty(Y_k)$, $|(Y_k)_{ij}|$ is deems to be $|Y_k|_\infty$ if $|(Y_k)_{ij}| \geq 0.95|Y_k|_\infty$; when projecting onto $N_2(Y_k)$, the singular values no less than 0.99 are deemed to be 1. Our experiments show that such inexact treatments indeed speed up convergence.

Stopping Criteria. We terminate the steepest ascent iteration when $\|D - D_k\|_F < 2 \times 10^{-5}\|D\|_F$ and the alternating projection when $\|A_i - A_{i-1}\|_F < 10^{-8}\|D\|_F$ and $\|E_i - E_{i-1}\|_F < 10^{-8}\|D\|_F$.

4. Simulations. In this section, using numerical simulations, we compare the two proposed algorithms with the iterative thresholding algorithm proposed in [21], and also highlight the differences between the proximal gradient approach and the dual approach.

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.[5] 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 $[-500, 500]$.[6] 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.

We set $\tau = 10,000$ and step size $\delta_k = 0.5$, $\forall\, k$, for the iterative thresholding algorithm (see [21] for details). 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. A brief comparison of the three algorithms is presented in Table 4.1. We also present in Table 4.2 the comparison between the proposed proximal gradient algorithm and the dual method on their ability to scale up with larger sized matrices.

Observations and Comparisons. It is clear from Table 4.1 that for dimension up to $m = 800$, the proposed proximal gradient approach (Algorithm 2) and the dual approach (Algorithm 3) are at least 50 times faster[7] than the iterative thresholding scheme proposed in [21], and achieve comparable accuracy in terms of relative error in the estimate of the low-rank matrix A. Though all three algorithms involve computing a SVD per iteration[8], we observe that the proposed algorithms take much fewer iterations (than the iterative thresholding method) to converge to the optimal solution.

Between the two algorithms proposed in this paper, 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 done by the dual method. On the other hand, the number of iterations taken by the proximal gradient algorithm to optimality is less vulnerable to changes in dimension

[5] It can be shown that A_0 is distributed according to the random orthogonal model of rank r, as defined in [5].

[6] This is identical to the distribution used in [21].

[7] We observe that the iterative thresholding algorithm [21] can occasionally converge slightly faster, but still about 10–20 times slower than the proposed algorithms.

[8] The dual approach only needs a partial SVD computation.

m	$\frac{\|\hat{A}-A_0\|_F}{\|A_0\|_F}$	$\text{rank}(\hat{A})$	$\|\hat{E}\|_0$	no. of iterations	time (s)
$\text{rank}(A_0) = 0.05\,m,\ \|E_0\|_0 = 0.05\,m^2$					
Accelerated Proximal Gradient (Algorithm 2)					
100	2.6×10^{-5}	5	508	127	3.1
200	1.9×10^{-5}	10	2,014	127	16.3
400	1.4×10^{-5}	20	8,035	126	108
800	9.8×10^{-6}	40	32,070	126	744
Dual Method (Algorithm 3)					
100	2.9×10^{-4}	5	507	247	7.7
200	4.4×10^{-4}	10	2,001	170	20.6
400	1.0×10^{-4}	20	7,999	209	92.2
800	7.9×10^{-5}	40	31,976	268	677
Iterative Thresholding [21]					
100	2.7×10^{-4}	5	511	10,000	136
200	2.3×10^{-5}	10	2,024	2,814	228
400	1.5×10^{-5}	20	8,016	10,000	6,310
800	9.9×10^{-6}	40	32,095	10,000	48,700
$\text{rank}(A_0) = 0.05\,m,\ \|E_0\|_0 = 0.1\,m^2$					
Accelerated Proximal Gradient (Algorithm 2)					
100	3.4×10^{-5}	5	1,021	129	3.1
200	2.2×10^{-5}	10	4,023	129	16.8
400	1.6×10^{-5}	20	16,122	129	111
800	1.1×10^{-5}	40	64,226	129	766
Dual Method (Algorithm 3)					
100	1.2×10^{-3}	5	1,024	339	13.6
200	3.1×10^{-4}	10	4,009	264	32.3
400	1.7×10^{-4}	20	16,014	263	131
800	1.2×10^{-4}	40	63,964	385	1,170
Iterative Thresholding [21]					
100	6.2×10^{-5}	5	1,041	5,346	76.4
200	3.2×10^{-5}	10	4,018	8,668	727
400	2.8×10^{-5}	24	16,120	10,000	6,720
800	1.4×10^{-5}	74	64,560	10,000	51,900

TABLE 4.1

Comparison of the Three Algorithms. *We present typical running times for randomly generated matrices. Corresponding to each triplet $\{m,\ \text{rank}(A_0),\ \|E_0\|_0\}$, the RPCA problem was solved for the same data matrix D using three different algorithms. The proposed algorithms are about 50–100 times faster than the iterative thresholding algorithm proposed in [21].*

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 4.1 that the average number of iterations taken by the dual approach is significantly increased when $\|E_0\|_0$ is doubled.

We corroborate the above observations with some more examples, and simultane-

m	$\frac{\text{rank}(A_0)}{m}$	$\frac{\|E_0\|_0}{m^2}$	$\frac{\|\hat{A}-A_0\|_F}{\|A_0\|_F}$		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 4.2

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. Notice how stable the number of iterations is for APG by comparing with those in Table 4.1.*

ously verify the scalability of the algorithms with matrices of higher dimensions. We see from Table 4.2 that both the proposed algorithms can efficiently handle matrices of over a million entries in less than 5 hours. However, the two algorithms scale differently. The dual algorithm scales well with the size of the problem but takes more iterations to converge to the optimal solution when the error matrix E_0 becomes less sparse. The proximal gradient algorithm, on the other hand, is slower than the dual method when E_0 is very sparse, but takes about the same number of iterations when the error sparsity is varied or the dimension is increased.

5. Discussions. In this paper, we have proposed two first-order algorithms for solving the Robust PCA problem (1.2), 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 [17, 5]. Both algorithms are significantly faster than the previous state-of-the-art algorithms based on iterative thresholding [21]. For matrices with the range of dimensions ($m \leq 10^4$) that can be handled by a single PC, the performance of the two algorithms are more or less comparable.

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. Furthermore, from our experience with the proximal gradient algorithm, one can significantly cut down the number of iterations by some other choices of the sequence μ_k, although there is currently a lack of rigorous justification for such choices. It is also important to understand why the number of iterations of the proximal gradient algorithm is so stable with respect to the change of dimension and problem settings. It might be possible to develop a much better hybrid algorithm by harnessing these good properties of both the primal and the dual solutions.

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 (1.2). For hardware improvements, we could resort to a more powerful machine with multiple cores and GPUs or even cluster of many machines. There has also been significant progress recently in developing parallelized algorithms for SVD computations [24, 13, 11]. 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. For example, we may run PROPACK (or any other algorithms devoted to computing the leading singular spaces) with different random initial vectors simultaneously, each thread outputting a fraction of the singular vectors, and then choose an orthonormal basis among the collection of all output singular vectors. In comparison, parallel partial SVD is less studied and running multiple threads of partial SVDs simultaneously is not as effective. It remains to see which algorithm is more suitable for parallel or distributed implementation for solving large-scale Robust PCA problems.

Appendix A. Proofs and Details for the Dual Method.

In this appendix, we provide additional mathematical details supporting the arguments in Section 3. Section A.1 proves Proposition 3.2 on the decomposition onto mutually polar cones. Section A.2 proves Theorem 3.1 on the convergence of the steepest ascent algorithm. Section A.3 describes how to compute the projection onto the normal cones N_2 and N_∞ used in Algorithm 3. Section A.4 proves the correctness of the iteration used in Algorithm 3 to project onto $N_2 + N_\infty$. Finally, Section A.5 proves a result on norm duality that we used in moving from the dual solution $\hat{Y}$ to the primal solution $\hat{A}, \hat{E}$.

A.1. Decomposition onto Polar Cones. Here we prove Proposition 3.2.

Proof. It suffices to prove that

$$P - \pi_2(P) = \pi_1(P),$$

i.e.,

$$P - \pi_2(P) \in C_1, \quad \text{and} \tag{A.1}$$

$$\langle P-(P-\pi_2(P)), c_1-(P-\pi_2(P))\rangle = \langle \pi_2(P), c_1-(P-\pi_2(P))\rangle \leq 0, \quad \forall c_1 \in C_1. \tag{A.2}$$

Indeed, as $\pi_2(P)$ is the projection of P onto C_2, we have

$$\langle P - \pi_2(P), c_2 - \pi_2(P)\rangle \leq 0, \quad \forall c_2 \in C_2. \tag{A.3}$$

By letting $c_2 = \alpha\pi_2(P)$, we see that

$$\langle P - \pi_2(P), \pi_2(P)\rangle = 0. \tag{A.4}$$

So (A.3) reduces to

$$\langle P - \pi_2(P), c_2\rangle \leq 0, \quad \forall c_2 \in C_2.$$

This proves (A.1). On the other hand, from C_1 and C_2 being polar to each other, we have

$$\langle \pi_2(P), c_1\rangle \leq 0, \quad \forall c_1 \in C_1. \tag{A.5}$$

Summing (A.4) and (A.5) gives (A.2). ∎

A.2. Convergence of Steepest Ascent. Before proving Theorem 3.1, we need a lemma.

LEMMA A.1. $\lim_{\delta\downarrow 0} \dfrac{J(Y_k+\delta(D-D_k))-J(Y_k)}{\delta} = 0$

Proof. It is well established in convex analysis that [2]:

$$\lim_{\delta\downarrow 0} \frac{J(Y_k+\delta(D-D_k))-J(Y_k)}{\delta} = \max_{P\in\partial J(Y_k)} \langle P, D-D_k\rangle. \tag{A.6}$$

So we only have to prove that

$$\max_{P\in\partial J(Y_k)} \langle P, D-D_k\rangle = 0. \tag{A.7}$$

Since D_k is the projection of D onto $N(Y_k)$, it follows that for all $P \in N(Y_k)$, $\langle P-D_k, D-D_k\rangle \le 0$. By choosing $P = \alpha D_k$ ($\alpha \ge 0$), we see that

$$\langle D_k, D-D_k\rangle = 0, \tag{A.8}$$

$$\langle P, D-D_k\rangle \le 0, \quad \forall\ P \in N(Y_k). \tag{A.9}$$

By (3.6), $\partial J(Y_k) \subset N(Y_k)$, and so

$$\max_{P\in\partial J(Y_k)} \langle P, D-D_k\rangle \le \max_{P\in N(Y_k)} \langle P, D-D_k\rangle \le 0. \tag{A.10}$$

This together with (A.8) gives (A.7). ∎

Now we are ready to prove Theorem 3.1.

Proof. By Proposition 3.2, we may write

$$Y_k(\delta) = \frac{Y_k+\delta(D-D_k)}{J\left(Y_k+\delta(D-D_k)\right)}. \tag{A.11}$$

Notice that the optimal step $\delta_k = 0$ if and only if

$$\langle D, Y_k(\delta)-Y_k\rangle \le 0, \quad \forall\ \delta > 0. \tag{A.12}$$

By plugging the expression of $Y_k(\delta)$ into (A.12) and observing that $J(Y_k) = 1$, it is easy to show that (A.12) occurs if and only if

$$\langle D, D-D_k\rangle \le \frac{J(Y_k+\delta(D-D_k))-J(Y_k)}{\delta}\langle D, Y_k\rangle, \quad \forall\ \delta > 0, \tag{A.13}$$

Let $\delta \downarrow 0$ and applying Lemma A.1 shows that $\langle D, D-D_k\rangle \le 0$. This together with (A.8) gives $\|D-D_k\|_F^2 \le 0$, and so $D = D_k$, which implies $D \in N(Y_k)$. ∎

A.3. Formulae for Projection onto the Normal Cone.

A.3.1. Projecting onto N_2. It is known that

$$\partial\|Y_k\|_2 = \mathrm{co}\{uv^T \,|\, \|u\|_2 = \|v\|_2 = 1, u^T Y_k v = \sigma_{\max}(Y_k)\}. \tag{A.14}$$

Suppose v_i, $i = 1,\ldots,d$, is the basis of the principal right singular space of Y_k (i.e., the set of left singular vectors associated to the *largest* singular value $\sigma_{\max}(Y_k)$) and $u_i = Y_k v_i/\sigma_{\max}(Y_k)$ is the corresponding basis of principal left singular space. Then any generator uv^T of $\partial\|Y_k\|_2$ can be written as

$$u = \frac{\sum_{i=1}^d a_i u_i}{\left\|\sum_{i=1}^d a_i u_i\right\|} \quad \text{and} \quad v = \frac{\sum_{i=1}^d a_i v_i}{\left\|\sum_{i=1}^d a_i v_i\right\|}.$$

So by (3.6), the normal cone $N_2(Y_k)$ is

$$N_2(Y_k) = \text{co}\{U_d a a^T V_d^T | a \in \mathbb{R}^d\}, \tag{A.15}$$

where $U_d = [u_1, \cdots, u_d]$, $V_d = [v_1, \cdots, v_d]$ and $a = [a_1, \cdots, a_d]^T$. Notice that aa^T can represent any rank-1 positive semi-definite (PSD) matrix. So we have

$$N_2(Y_k) = \{U_d X V_d^T | X \text{ is a PSD matrix}\}. \tag{A.16}$$

To calculate the projection D_k of D onto $N_2(Y_k)$, we need to solve the following optimization problem

$$\min_X \ \|D - U_d X V_d^T\|_F, \quad \text{subject to} \quad X \succeq 0. \tag{A.17}$$

It is equivalent to solve

$$\min_X \ \|\bar{D} - X\|_F, \quad \text{subject to} \quad X \succeq 0, \tag{A.18}$$

where $\bar{D} = U_d^T D V_d$. Notice that

$$\begin{aligned} 2\|\bar{D} - X\|_F^2 &= \|\bar{D} - X\|_F^2 + \|\bar{D}^T - X\|_F^2 \\ &= 2\left\|\frac{\bar{D}^T + \bar{D}}{2} - X\right\|_F^2 + 2\left\|\frac{\bar{D}^T - \bar{D}}{2}\right\|_F^2. \end{aligned} \tag{A.19}$$

So we have an equivalent problem:

$$\min_X \ \|D' - X\|_F, \quad \text{subject to} \quad X \succeq 0, \tag{A.20}$$

where $D' = \frac{1}{2}(\bar{D}^T + \bar{D})$ is a symmetric matrix. Let $D' = Q\Lambda Q^T$ be its eigenvalue decomposition, where $\Lambda = \text{diag}(\lambda_i)$, $i = 1, \cdots, d$. Then the solution to (A.20) (and hence (A.18)) is $X = Q[\Lambda]_+ Q^T$, and the projection of D onto $N(Y_k)$ is $D_k = U_d X V_d^T$. Since D' is of size $d \times d$, its eigenvalue decomposition is of low cost.

A.3.2. Projecting onto N_∞. Denote E_{ij} as the $m \times n$ matrix which has only one nonzero entry 1 at (i, j). Then the subgradient of $\lambda^{-1}|Y_k|_\infty$ is

$$\partial(\lambda^{-1}|Y_k|_\infty) = \text{co}(\{E_{ij} | (Y_k)_{ij} = |Y_k|_\infty\}, \{-E_{ij} | (Y_k)_{ij} = -|Y_k|_\infty\}). \tag{A.21}$$

So the normal cone is:

$$N_\infty(Y_k) = \left\{ X \,\middle|\, X_{ij} \begin{cases} \leq 0, & \text{if } (Y_k)_{ij} = -|Y_k|_\infty, \\ \geq 0, & \text{if } (Y_k)_{ij} = |Y_k|_\infty, \\ = 0, & \text{if } |(Y_k)_{ij}| < |Y_k|_\infty. \end{cases} \right\}. \tag{A.22}$$

Then the projection of D onto $N(Y_k)$ is

$$(D_k)_{ij} = \begin{cases} \max(D_{ij}, 0), & \text{if } (Y_k)_{ij} = |Y_k|_\infty, \\ \min(D_{ij}, 0), & \text{if } (Y_k)_{ij} = -|Y_k|_\infty, \\ 0, & \text{otherwise.} \end{cases} \tag{A.23}$$

A.4. Convergence of Projection onto the Normal Cone. In this section we prove Theorem 3.3.

Proof. We will prove that the sequence $\{D_i' = A_i + E_i\}$ has only one accumulation point D_k.

By the definition of projections, we have that

$$\begin{array}{ll} \langle D - A_i - E_{i-1}, A' - A_i \rangle \leq 0, & \forall A' \in N_2(Y_k), \\ \langle D - A_i - E_i, E' - E_i \rangle \leq 0, & \forall E' \in N_\infty(Y_k). \end{array} \tag{A.24}$$

Then we have

$$\begin{aligned} &\|D - A_{i-1} - E_{i-1}\|_F^2 \\ &= \|D - A_i - E_{i-1}\|_F^2 + \|A_i - A_{i-1}\|_F^2 - 2\langle D - A_i - E_{i-1}, A_{i-1} - A_i \rangle \\ &\geq \|D - A_i - E_{i-1}\|_F^2 + \|A_i - A_{i-1}\|_F^2, \\ &\|D - A_i - E_{i-1}\|_F^2 \\ &= \|D - A_i - E_i\|_F^2 + \|E_i - E_{i-1}\|_F^2 - 2\langle D - A_i - E_i, E_{i-1} - E_i \rangle \\ &\geq \|D - A_i - E_i\|_F^2 + \|E_i - E_{i-1}\|_F^2. \end{aligned}$$

Hence,

$$\|D - A_{i-1} - E_{i-1}\|_F^2 \geq \|D - A_i - E_i\|_F^2 + \|A_i - A_{i-1}\|_F^2 + \|E_i - E_{i-1}\|_F^2. \tag{A.25}$$

So $\{\|D - A_i - E_i\|_F\}$ is a decreasing sequence. As it is also lower bounded by 0, it has a limit. Moreover,

$$\lim_{i \to \infty} \|A_i - A_{i-1}\|_F = \lim_{i \to \infty} \|E_i - E_{i-1}\|_F = 0. \tag{A.26}$$

Let $\hat{D} = \lim_{j \to \infty} D'_{k_j}$ be an accumulation point of $\{D_i'\}$. Without loss of generality, we may assume that $\lim_{j \to \infty} A_{k_j} = \hat{A}$ and $\lim_{j \to \infty} E_{k_j} = \hat{E}$. By letting $i = k_j$ in (A.24) and $j \to \infty$ and observing (A.26), we have that

$$\begin{array}{ll} \left\langle D - \hat{D}, A' - \hat{A} \right\rangle \leq 0, & \forall A' \in N_2(Y_k), \\ \left\langle D - \hat{D}, E' - \hat{E} \right\rangle \leq 0, & \forall E' \in N_\infty(Y_k). \end{array} \tag{A.27}$$

As $0 \in N_2(Y_k)$ and $0 \in N_\infty(Y_k)$, we have

$$\begin{array}{ll} \left\langle D - \hat{D}, -\hat{A} \right\rangle \leq 0, & \forall A' \in N_2(Y_k), \\ \left\langle D - \hat{D}, -\hat{E} \right\rangle \leq 0, & \forall E' \in N_\infty(Y_k). \end{array} \tag{A.28}$$

Adding the above to (A.27) appropriately results in

$$\begin{array}{ll} \left\langle D - \hat{D}, A' - \hat{D} \right\rangle \leq 0, & \forall A' \in N_2(Y_k), \\ \left\langle D - \hat{D}, E' - \hat{D} \right\rangle \leq 0, & \forall E' \in N_\infty(Y_k). \end{array} \tag{A.29}$$

This implies that $\hat{D} = D_k$ by the uniqueness of projection. This completes the proof. ∎

A.5. Relationship between Primal and Dual Norms. We used the following general result on norm duality to move from a solution to the dual problem back to a solution to the primal problem:

THEOREM A.2. *In a real Hilbert space $\mathcal{H}$, if $x \neq 0$ and $y \in \partial\|x\|$, then $\|y\|^* = 1$ and $x \in \|x\|\partial\|y\|^*$, where $\|\cdot\|^*$ is the dual norm of $\|\cdot\|$.*

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

$$\|w\| - \|x\| \geq \langle y, w - x\rangle, \quad \forall\, w \in \mathcal{H}. \tag{A.30}$$

Choosing $w = 0, 2x$, we can deduce that

$$\|x\| = \langle y, x\rangle \leq \|x\|\|y\|^*. \tag{A.31}$$

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

$$\|w - x\| \geq \|w\| - \|x\| \geq \langle y, w - x\rangle, \quad \forall\, w \in \mathcal{H}. \tag{A.32}$$

So

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

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

Next, by the definition of dual norm and (A.31) we have

$$\begin{aligned} \left\langle \frac{x}{\|x\|}, w - y \right\rangle &= \left\langle \frac{x}{\|x\|}, w \right\rangle - \frac{\langle x, y\rangle}{\|x\|} \\ &\leq \|w\|^* - 1 \\ &= \|w\|^* - \|y\|^*, \quad \forall\, w \in \mathcal{H}. \end{aligned} \tag{A.33}$$

This implies $\dfrac{x}{\|x\|} \in \partial\|y\|^*$, establishing the result. ∎

REFERENCES

[1] A. BECK AND M. TEBOULLE, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202.

[2] DIMITRI P. BERTSEKAS, *Convex Analysis and Optimization*, Athena Scientific, 2003.

[3] J. CAI, E. CANDÈS, AND Z. SHEN, *A singular value thresholding algorithm for matrix completion*, preprint, (2008).

[4] J.-F. CAI, S. OSHER, AND Z. SHEN, *Linearized Bregman iterations for compressed sensing*, Math. Comp., 78 (2009), pp. 1515–1536.

[5] E. CANDÈS AND B. RECHT, *Exact matrix completion via convex optimization*, preprint, (2008).

[6] V. CHANDRASEKHARAN, S. SANGHAVI, P. PARILLO, AND A. WILSKY, *Rank-sparsity incoherence for matrix decomposition*, preprint, (2009).

[7] C. ECKART AND G. YOUNG, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[8] M. FUKUSHIMA AND H. MINE, *A generalized proximal gradient algorithm for certain nonconvex minimization problems*, International Journal of Systems Science, 12 (1981), pp. 989–1000.

[9] G.H. GOLUB AND C.F.V. LOAN, *Matrix Computation, 3rd Edition*, The John Hopkins University Press, 1996.

[10] M. GRANT AND S. BOYD, *CVX: Matlab software for disciplined convex programming (web page and software)*. http://stanford.edu/∼boyd/cvx, June 2009.

[11] V. HERNANDEZ, J.E. ROMAN, AND V. VIDAL, *SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Trans. Math. Softw., 31 (2005), pp. 351–362.

[12] I. T. JOLLIFFE, *Principal Component Analysis*, Springer-Verlag, 1986.

[13] 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 international conference on Advances in computer science and technology, 2006, pp. 79–84.

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

[15] Y. Nesterov, *A method of solving a convex programming problem with convergence rate* $O(1/k^2)$, Soviet Mathematics Doklady, 27 (1983), pp. 372–376.

[16] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, 1971.

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

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

[19] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, preprint, (2009).

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

[21] 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, (2009).

[22] W. Yin, E. Hale, and Y. Zhang, *Fixed-point continuation for* ℓ_1*-minimization: methodology and convergence*, preprint, (2008).

[23] 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, 1 (2008), pp. 143–168.

[24] B.B. Zhou and R.P. Brent, *A parallel ring ordering algorithm for efficient one-sided Jacobi SVD computations*, in in Proc. Sixth IASTED/ISMM Internat. Conf. on Parallel and Distributed Computing and Systems, 1994, pp. 369–372.