Robust Principal Component Analysis Based on Low-Rank and Block-Sparse Matrix Decomposition

Gongguo Tang and Arye Nehorai
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO 63130-1127
Email: {gt2, nehorai}@ese.wustl.edu

***Abstract*—In this paper, we propose a convex program for low-rank and block-sparse matrix decomposition. Potential applications include outlier detection when certain columns of the data matrix are outliers. We design an algorithm based on the augmented Lagrange multiplier method to solve the convex program. We solve the subproblems involved in the augmented Lagrange multiplier method using the Douglas/Peaceman-Rachford (DR) monotone operator splitting method. Numerical simulations demonstrate the accuracy of our method compared with the robust principal component analysis based on low-rank and sparse matrix decomposition.**

***Index Terms*—augmented Lagrange multiplier method, low-rank and block-sparse matrix decomposition, operator splitting method, robust principal component analysis**

In this work, we design algorithms for low-rank and block-sparse matrix decomposition as a robust version of principal component analysis insensitive to column/row outliers. Many data sets arranged in matrix formats are of low-rank due to the correlations and connections within the data samples. The ubiquity of low-rank matrices is also suggested by the success and popularity of principal component analysis in many applications [1]. Examples of special interest are network traffic matrices [2], and the data matrix formed in face recognition [3], where the columns correspond to vectorized versions of face images. When a few columns of the data matrix are generated by mechanisms different from the rest of the columns, the existence of these outlying columns tends to destroy the low-rank structure of the data matrix. A decomposition that enforces the low-rankness of one part and the block-sparsity of the other part would separate the principal components from the outliers. While robust principal component analysis (RPCA) [4] can perform low-rank and sparse matrix decomposition, it does not yield good results when the sparse pattern involves entire columns.

We use a convex program to separate the low-rank part and block-sparse part of the observed matrix. The decomposition involves the following model:

$$D \quad = \quad A + E, \tag{1}$$

where D is the observation matrix with a low-rank component A and block-sparse component E. The block-sparse matrix E contains mostly zero columns, with several non-zero ones corresponding to outliers. In order to eliminate ambiguity, the columns of the low-rank matrix A corresponding to the outlier columns are assumed to be zeros. We demonstrate that the following convex program recovers the low-rank matrix A and the block-sparsity matrix E:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \kappa(1-\lambda)\|A\|_{2,1} + \kappa\lambda\|E\|_{2,1} \\ \text{subject to } & D = A + E, \end{aligned} \tag{2}$$

where $\|\cdot\|_*, \|\cdot\|_{\mathrm{F}}, \|\cdot\|_{2,1}$ denote respectively the nuclear norm, the Frobinius norm, and the ℓ_1 norm of the vector formed by taking the ℓ_2 norms of the columns of the underlying matrix. Note that the extra term $\kappa(1-\lambda)\|A\|_{2,1}$ actually ensures the recovered A has exact zero columns corresponding to the outliers.

We design efficient algorithms to solve the convex program (2) based on the augmented Lagrange multiplier (ALM) method. The ALM method was employed to successfully perform low-rank and sparse matrix decomposition for large scale problems [5]. The challenges here are the special structure of the $\|\cdot\|_{2,1}$ norm, and the existence of the extra term $\kappa(1-\lambda)\|A\|_{2,1}$. We demonstrate the validity of (2) in decomposition and compare the performance of the ALM algorithm using numerical simulations. In future work, we will test the algorithms by applying them to face recognition and network traffic anomaly detection problems. Due to the ubiquity of low-rank matrices and the importance of outlier detection, we expect low-rank and block-sparse matrix decomposition to have wide applications, especially in computer vision and data mining.

The paper is organized as follows. In Section I, we introduce notations and the problem setup. Section II is devoted to the methods of augmented Lagrange multipliers for solving the convex program (2). We provide implementation details in Section III. Numerical experiments are used to demonstrate the effectiveness of our method and the results are reported in Section IV. Section V summarizes our conclusions.

I. Notations and Problem Setup

In this section, we introduce notations and the problem setup used throughout the paper. For any matrix $A \in \mathbb{R}^{n\times p}$, we use A_{ij} to denote its ijth element, and A_j to denote its jth column. The usual ℓ_p norm is denoted by $\|\cdot\|_p, p \geq 1$ when

This work was supported by the ONR Grant N000140810849, and the National Science Foundation, Grant No. CCF-1014908.

978-1-4244-9848-2/11$26.00©2011 IEEE

the argument is a vector. The nuclear norm $\|\cdot\|_*$, the Frobinius norm $\|\cdot\|_{\rm F}$, and the $\ell_{2,1}$ norm $\|\cdot\|_{2,1}$ of matrix A are defined as follows:

$$\|A\|_* = \sum_i \sigma_i(A), \quad (3)$$

$$\|A\|_{\rm F} = \sqrt{\sum_{i,j} A_{ij}^2}, \quad (4)$$

$$\|A\|_{2,1} = \sum_j \|A_j\|_2, \quad (5)$$

where $\sigma_i(A)$ is the ith largest singular value of A. We consider the following model:

$$D = A + E. \quad (6)$$

Here E has at most s non-zero columns. These non-zero columns are considered as outliers in the entire data matrix D. The corresponding columns of A are assumed to be zeros. In addition, A has low rank, *i.e.*, $\mathrm{rank}(A) \leq r$.

We use the following optimization to recover A and E from the observation D:

$$\min_{A,E} \ \|A\|_* + \kappa(1-\lambda)\|A\|_{2,1} + \kappa\lambda\|E\|_{2,1} \quad \text{subject to } D = A + E. \quad (7)$$

This procedure separates the outliers in E and the principal components in A. As we will see in the numerical examples, simply enforcing the sparsity of E and the low-rankness of A would not separates A and E well.

II. The Methods of Augmented Lagrange Multipliers

The Augmented Lagrange Multipliers (ALM) method is a general procedure to solve the following equality constrained optimization problem:

$$\min f(\boldsymbol{x}) \text{ subject to } \ h(\boldsymbol{x}) = 0, \quad (8)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$. The procedure defines the augmented Lagrangian function as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}; \mu) = f(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, h(\boldsymbol{x})\rangle + \frac{\mu}{2}\|h(\boldsymbol{x})\|_2^2, \quad (9)$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the Lagrange multiplier vector, and μ is a positive scalar. The ALM iteratively solves $\boldsymbol{x}$ and the Lagrangian multiplier vector $\boldsymbol{\lambda}$ as shown in Table I.

To apply the general ALM to problem (7), we define

$$f(X) = \|A\|_* + \kappa(1-\lambda)\|A\|_{2,1} + \kappa\lambda\|E\|_{2,1}, \quad (10)$$

$$h(X) = D - A - E \quad (11)$$

with $X = (A, E)$. The corresponding augmented Lagrangian function is

$$\mathcal{L}(A, E, Y; \mu) = \|A\|_* + \kappa(1-\lambda)\|A\|_{2,1} + \kappa\lambda\|E\|_{2,1} + \langle Y, D - A - E\rangle + \frac{\mu}{2}\|D - A - E\|_{\rm F}^2, \quad (12)$$

where we use Y to denote the Lagrange multiplier. Given $Y = Y_k$ and $\mu = \mu_k$, one key step in applying the general ALM method is to solve $\min_{A,E} \mathcal{L}(A, E, Y_k; \mu_k)$. We adopt an alternating procedure: for fixed $E = E_k$, solve

$$A_{k+1} = \mathrm{argmin}_A \mathcal{L}(A, E_k, Y_k; \mu_k), \quad (13)$$

and for fixed $A = A_{k+1}$, solve

$$E_{k+1} = \mathrm{argmin}_E \mathcal{L}(A_{k+1}, E, Y_k; \mu_k). \quad (14)$$

We first address (14), which is equivalent to solving

$$\min_E \kappa\lambda\|E\|_{2,1} + \frac{\mu_k}{2}\left\|D - A_{k+1} + \frac{1}{\mu_k}Y_k - E\right\|_{\rm F}^2. \quad (15)$$

Denote $G^E = D - A_{k+1} + \frac{1}{\mu_k}Y_k$. The following lemma gives the closed-form solution of the minimization with respect to E, whose proof is given in Appendix A.

Lemma 1 *The solution to*

$$\min_E \eta\|E\|_{2,1} + \frac{1}{2}\|E - G^E\|_{\rm F}^2 \quad (16)$$

is given by

$$E_j = G_j^E \max\left(0, 1 - \frac{\eta}{\|G_j^E\|_2}\right) \quad (17)$$

for $j = 1, \ldots, p$.

We denote the operator solving (16) as $\mathcal{T}_\eta(\cdot)$. So $E = \mathcal{T}_\eta(G)$ sets the columns of E to be zero vectors if the ℓ_2 norms of the corresponding columns of G are less than η, and scales down the columns otherwise by a factor $1 - \frac{\eta}{\|G_j\|_2}$.

Now we turn to solve

$$A_{k+1} = \mathrm{argmin}_A \mathcal{L}(A, E_k, Y_k; \mu_k) \quad (18)$$

which can be rewritten as

$$A_{k+1} = \mathrm{argmin}_A \left\{\|A\|_* + \kappa(1-\lambda)\|A\|_{2,1} + \frac{\mu_k}{2}\|G^A - A\|_{\rm F}^2\right\}, \quad (19)$$

where $G^A = D - E_k + \frac{1}{\mu_k}Y_k$. We know that without the extra $\kappa(1-\lambda)\|A\|_{2,1}$ term, a closed form solution is simply given by the soft-thresholding operator [6]. Unfortunately, a closed form solution to (19) is not available. We use the Douglas/Peaceman-Rachford (DR) monotone operator splitting method [7]–[9] to iteratively solve (19). Define $f_1(A) = \kappa(1-\lambda)\|A\|_{2,1} + \frac{\mu_k}{2}\|G^A - A\|_{\rm F}^2$ and $f_2(A) = \|A\|_*$. For any $\beta > 0$ and a sequence $\alpha_t \in (0, 2)$, the DR itration for (19) is expressed as

$$A^{(j+1/2)} = \mathrm{prox}_{\beta f_2}(A^{(j)}), \quad (20)$$

$$A^{(j+1)} = A^{(j)} + \alpha_j\left(\mathrm{prox}_{\beta f_1}(2A^{(j+1/2)} - A^{(j)}) - A^{(j+1/2)}\right). \quad (21)$$

Here for any proper, lower semi-continuous, convex function f, the proximity operators $\mathrm{prox}_f(\cdot)$ gives the unique point

978-1-4244-9848-2/11$26.00©2011 IEEE

TABLE I: General augmented Lagrange multiplier method

1. **initialize:** Given $\rho > 1, \mu_0 > 0$, starting point $\boldsymbol{x}_0^{\mathrm{s}}$ and $\boldsymbol{\lambda}_0$; $k \leftarrow 0$
2. **while** not converged **do**
3. Approximately solve $\boldsymbol{x}_{k+1} = \mathrm{argmin}_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_k; \mu_k)$ using starting point $\boldsymbol{x}_k^{\mathrm{s}}$.
4. Set starting point $\boldsymbol{x}_{k+1}^{\mathrm{s}} = \boldsymbol{x}_{k+1}$.
5. Update the Lagrange multiplier vector $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu_k h(\boldsymbol{x}_{k+1})$.
6. Update $\mu_{k+1} = \rho \mu_k$.
7. **end while**
8. **output:** $\boldsymbol{x} \leftarrow \boldsymbol{x}_k, \boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}_k$.

$\mathrm{prox}_f(\boldsymbol{x})$ achieving the infimum of the function

$$\boldsymbol{z} \mapsto \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + f(\boldsymbol{z}). \tag{22}$$

The following lemmas gives the explicit expression for the proximity operator involved in our DR iteration when $f(A) = \|A\|_*$ and $f(A) = \kappa(1-\lambda)\|A\|_{2,1} + \frac{\mu_k}{2}\|G^A - A\|_{\mathrm{F}}^2$.

Lemma 2 *For $f(\cdot) = \|\cdot\|_*$, suppose the singular value decomposition of A is $A = U\Sigma V^T$, then the proximity operator is*

$$\mathrm{prox}_{\beta f}(A) = U\mathcal{S}_\beta(\Sigma)V^T, \tag{23}$$

where the nonnegative soft-thresholding operator is defined as

$$\mathcal{S}_\nu(x) = \max(0, x - \nu), x \geq 0, \nu > 0, \tag{24}$$

for a scalar x and extended entry-wisely to vectors and matrices.

For $f(\cdot) = \eta\|\cdot\|_{2,1} + \frac{\mu}{2}\|G - \cdot\|_{\mathrm{F}}^2$, the proximity operator is

$$\mathrm{prox}_{\beta f}(A) = \mathcal{T}_{\frac{\beta\eta}{1+\beta\mu}}\left(\frac{A + \beta\mu G}{1 + \beta\mu}\right). \tag{25}$$

With these preparations, we present the algorithm for solving (7) using the ALM, named as RPCA-LBD, in Table II. Note that instead of alternating between (13) and (14) many times until convergence, the RPCA-LBD algorithm in Table II executes them only once. This inexact version greatly reduces the computational burden and yields sufficiently accurate results for appropriately tuned parameters. The strategy was also adopted in [5] to perform the low-rank and sparse matrix decomposition.

III. Implementation

We provide some implementation details in this section.

Parameter selection and initialization: The tuning parameters in (7) are chosen to be $\kappa = 1.1$ and $\lambda = 0.61$. We do not have a systematic way to select these parameters. But these empirical values work well for all tested cases. For the DR iteration (20) and (21), we have $\beta = 0.2$ and $\alpha_j \equiv 1$. The parameters for the ALM method are $\mu_0 = 30/\|\mathrm{sign}(D)\|_2$ and $\rho = 1.1$. The low-rank matrix A, the block-sparse matrix E, and the Lagrange multiplier Y are initialized respectively to D, $\mathbf{0}$, and $\mathbf{0}$. The outer loop in Table II terminates when it reaches the maximal iteration number 500 or the error tolerance 10^{-7}. The error in the outer loop is computed by $\|D - A_k - E_k\|_{\mathrm{F}}/\|D\|_{\mathrm{F}}$. The inner loop for the DR iteration has maximal iteration number 20 and tolerance error 10^{-6}. The error in the inner loop is the Frobenius norm of the difference between successive $A_k^{(j)}$s.

Performing SVD: The major computational cost in the RPCA-LBD algorithm is the singular value decomposition (SVD) in the DR iteration (Line 07 in Table II). We usually do not need to compute the full SVD because only those that are greater than β are used. We use the PROPACK [10] to compute the first (largest) few singular values. In order to both save computation and ensure accuracy, we need to predict the number of singular values of $A_{k+1}^{(j)}$ that exceed β for each iteration. We adopt the following rule [5]:

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

where $\mathrm{sv}_0 = 10$ and

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

Here $d = \min(n, p)$, sv_k is the predicted number of singular values that are greater than β, and svp_k is the number of singular values in the sv_k singular values that are larger than β. In addition, maxgap_k and maxid_k are respectively the largest ratio between successive singular values of $A_{k+1}^{(j)}$ when arranged in a decreasing order and the corresponding index.

IV. Numerical Simulations

In this section, we perform numerical simulations and compare the results of our algorithm and those of the classical RPCA solved by the ALM algorithm proposed in [5]. The RPCA decomposes the low-rank matrix and the sparse matrix through the following optimization problem:

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \frac{1}{\sqrt{n}}\|E\|_{2,1} \\ \text{subject to } & D = A + E, \end{aligned} \tag{26}$$

All the numerical experiments in this section were conducted on a desktop computer with a Pentium D CPU@3.40GHz, 2GB RAM, and Windows XP operating system, and the computations were running single-core.

We consider only square matrices with $n = p$. The low-rank matrix L is constructed as the product of two Gaussian matrices with dimensions $n \times r$ and $r \times n$, where r is the

978-1-4244-9848-2/11$26.00©2011 IEEE

TABLE II: Block-sparse and low-rank matrix decomposition via ALM

RPCA-LBD(D, κ, λ)

Input: Data matrix $D \in \mathbb{R}^{n\times p}$ and the parameters κ, λ.

Output: The low-rank part A and the block-sparse part E.

initialize: $A_0 \leftarrow D$; $E_0 \leftarrow 0$; $\mu_0 = 30/\|\text{sign}(D)\|_2$; $\rho > 0, \beta > 0, \alpha \in (0,1)$, $k \leftarrow 0$
while not converged **do**
  \\ Lines 4 - 11 solve $A_{k+1} = \text{argmin}_A \mathcal{L}(A, E_k, Y_k; \mu_k)$.
  $G^A = D - E_k + \frac{1}{\mu_k} Y_k$.
  $j \leftarrow 0$, $A_{k+1}^{(0)} = G^A$.
  while not converged **do**
    $A_{k+1}^{(j+1/2)} = U\mathcal{S}_\beta(\Sigma)V^T$ where $A_{k+1}^{(j)} = U\Sigma V^T$ is the SVD of $A_{k+1}^{(j)}$.
    $A_{k+1}^{(j+1)} = A_{k+1}^{(j)} + \alpha\left(\mathcal{T}_{\frac{\beta\kappa(1-\lambda)}{1+\beta\mu_k}}\left(\frac{2A_{k+1}^{(j+1/2)} - A_{k+1}^{(j)} + \beta\mu_k G^A}{1+\beta\mu_k}\right) - A_{k+1}^{(j+1/2)}\right)$.
    $j \leftarrow j + 1$.
  end while
  $A_{k+1} = A_{k+1}^{(j+1/2)}$.
  $G^E = D - A_{k+1} + \frac{1}{\mu_k} Y_k$.
  $E_{k+1} = \mathcal{T}_{\frac{\kappa\lambda}{\mu_k}}\left(G^E\right)$.
  $Y_{k+1} = Y_k + \mu_k * (D - A_{k+1} - E_{k+1})$.
  $\mu_{k+1} = \rho\mu_k$.
  $k \leftarrow k + 1$.
end while
output: $A \leftarrow A_k, E \leftarrow E_k$.

rank of L. The columns of L are normalized to have unit lengths. For the block sparse matrix B, first a support S of size s is generated as a random subset of $\{1, \ldots, n\}$, then the columns of B corresponding to the support S are sampled from the Gaussian distribution. The non-zero columns of B are also normalized to have unit lengths. The columns of L corresponding to S are set to zeros. Finally, $D = L + B$ is formed.

In Figure 1, we show the original L, B and those recovered by the RPCA-LBD and the classical RPCA. Here $n = 80$, $r = \text{round}(0.04n)$ and $s = \text{round}(0.3n)$. We can see that our algorithm RPCA-LBD recovers the low-rank matrix and the block sparse matrix near perfectly, while the RPCA performs badly. Actually, the relative error for L recovered by the RPCA-LBD is 3.1991×10^{-4} while that for the RPCA is 0.4863.

In Table III, we compare various characteristics of the RPCA-LBD and the RPCA. We consider $n = 100, 300, 500, 700$ and 900. The rank of the low-rank matrix L is $r = \text{round}(0.04n)$ and the number of the non-zero columns of the block sparse matrix B is $s = \text{round}(0.04n)$. We see that the RPCA-LBD can accurately recover the low-rank matrix L (and hence the block sparse matrix as $B = D - L$.) The relative recovery error is at the order of 10^{-8}, while that for the RPCA is at the order of 10^{-1}. The rank of L is also correctly recovered by the RPCA-LBD as shown in the fourth column. However, the RPCA-LBD takes significantly more time than the ALM implementation for RPCA. The extra time is spent on the DR iteration, which is not needed for the RPCA. In addition, the RPCA seems to be more robust than the RPCA-LBD

TABLE III: Comparison of our algorithm (RPCA-LBD) and the RPCA.

n	algorithm	$\frac{\|\hat{L}-L\|_F}{\|L\|_F}$	$\text{rank}(\hat{L})$	#iter	time (s)
100	RPCA-LBD	2.60e-008	4	479	9.66
	RPCA	1.25e-001	6	32	0.51
300	RPCA-LBD	1.31e-008	12	614	28.69
	RPCA	1.41e-001	19	29	2.09
500	RPCA-LBD	7.74e-008	20	651	99.14
	RPCA	1.48e-001	31	29	6.67
700	RPCA-LBD	1.57e-008	28	709	218.61
	RPCA	1.52e-001	44	29	16.23
900	RPCA-LBD	8.27e-008	36	745	409.60
	RPCA	1.55e-001	57	28	30.07

V. Conclusions

In this work we proposed a convex program for accurate low-rank and block-sparse matrix decomposition. We solved the convex program using the augmented Lagrange multiplier method. We used the Douglas/Peaceman-Rachford (DR) monotone operator splitting method to solve a subproblem involved in the augmented Lagrange multiplier method. We demonstrated the accuracy of our program and compared its results with those given by the robust principal component

978-1-4244-9848-2/11$26.00©2011 IEEE

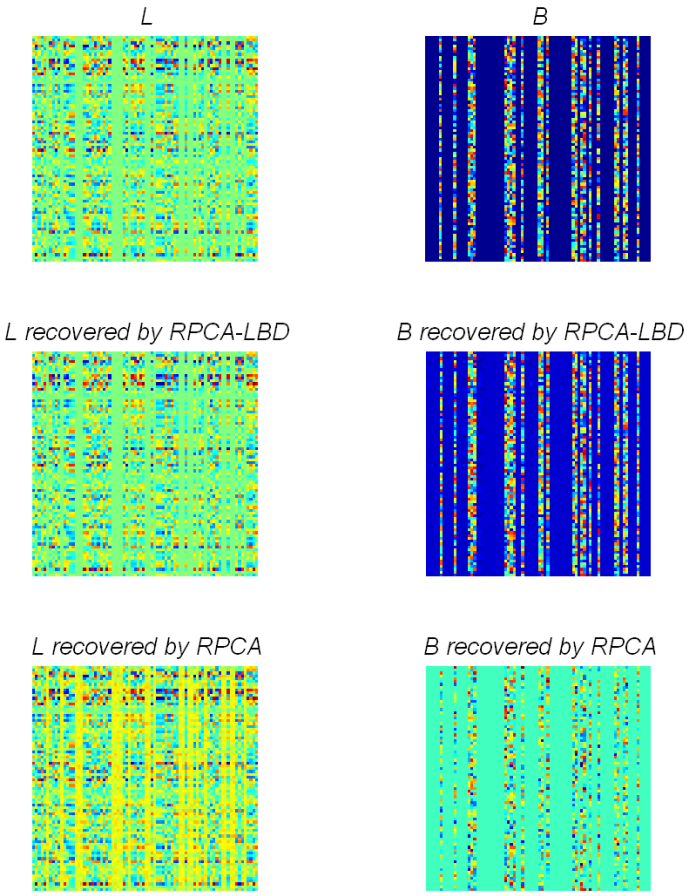

Fig. 1: L and B recovered by our RPCA-LBD algorithms and RPCA.

analysis. The proposed algorithm is potentially useful for outlier detection.

APPENDIX

Proof: Note the objective function expands as

$$\begin{aligned} & \eta\|E\|_{2,1} + \frac{1}{2}\|E - G^E\|_{\mathrm{F}}^2 \\ = & \sum_{j=1}^{n} \left(\eta\|E_j\|_2 + \frac{1}{2}\|E_j - G_j^E\|_2^2 \right). \end{aligned} \tag{27}$$

Now we can minimize with respect to E_j separately. Denote

$$\begin{aligned} h(\boldsymbol{e}) &= \eta\|\boldsymbol{e}\|_2 + \frac{1}{2}\|\boldsymbol{e} - \boldsymbol{g}\|_2^2 \\ &= \eta\|\boldsymbol{e}\|_2 + \frac{1}{2}\left[\|\boldsymbol{e}\|_2^2 - 2\langle \boldsymbol{e}, \boldsymbol{g}\rangle + \|\boldsymbol{g}\|_2^2\right]. \end{aligned} \tag{28}$$

First consider $\|\boldsymbol{g}\|_2 \leq \eta$. Cauchy-Schwarz inequality leads to

$$\begin{aligned} h(\boldsymbol{e}) &\geq \eta\|\boldsymbol{e}\|_2 + \frac{1}{2}\left[\|\boldsymbol{e}\|_2^2 - 2\|\boldsymbol{e}\|_2\|\boldsymbol{g}\|_2 + \|\boldsymbol{g}\|_2^2\right] \\ &= \frac{1}{2}\|\boldsymbol{e}\|_2^2 + (\eta - \|\boldsymbol{g}\|_2)\|\boldsymbol{e}\|_2 + \frac{1}{2}\|\boldsymbol{g}\|_2^2 \qquad (29) \\ &\geq \frac{1}{2}\|\boldsymbol{g}\|_2^2, \qquad (30) \end{aligned}$$

where for the last inequality we used (29) is an increasing function on $[0, \infty)$ if $\|\boldsymbol{g}\|_2 \leq \eta$. Apparently, $h(\boldsymbol{e}) = \|\boldsymbol{g}\|_2/2$ is achieved by $\boldsymbol{e} = 0$ which is also unique. Therefore, if $\|\boldsymbol{g}\|_2 \leq \eta$, then the minimum of $h(\boldsymbol{e})$ is achieved by the unique solution $\boldsymbol{e} = 0$

In the second case when $\|\boldsymbol{g}\|_2 > \eta$. Setting the derivative of $h(\boldsymbol{e})$ with respect to $\boldsymbol{e}$ to zero yields

$$\boldsymbol{e}\left(\frac{\eta}{\|\boldsymbol{e}\|_2} + 1\right) = \boldsymbol{g}. \tag{31}$$

Taking the ℓ_2 norm of both sides yields

$$\|\boldsymbol{e}\|_2 = \|\boldsymbol{g}\|_2 - \eta > 0. \tag{32}$$

Plugging $\|\boldsymbol{e}\|_2$ into (31) gives

$$\boldsymbol{e} = \boldsymbol{g}\left(1 - \frac{\eta}{\|\boldsymbol{g}\|_2}\right). \tag{33}$$

In conclusion, the minimum of (16) is achieved by E with

$$E_j = G_j^E \max\left(0, \left(1 - \frac{\eta}{\|G_j^E\|_2}\right)\right) \tag{34}$$

for $j = 1, 2, \ldots, p$. ■

REFERENCES

[1] I. T. Jolliffe, *Principal component analysis*, Springer series in statistics. Springer, 2002.

[2] A. Abdelkefi, Y. Jiang, W. Wang, A. Aslebo, and O. Kvittem, "Robust traffic anomaly detection with principal component pursuit," in *Proceedings of the ACM CoNEXT Student Workshop*, New York, NY, USA, 2010, CoNEXT '10 Student Workshop, pp. 10:1–10:2, ACM.

[3] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, Feb. 2009.

[4] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," submitted for publication, 2010.

[5] Z. Lin, M. Chen, L. Wu, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," UIUC Technical Report UILU-ENG-09-2215, Nov. 2009.

[6] J-F Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2008.

[7] P. L. Combettes and J. . Pesquet, "A Douglas-rachford splittting approach to nonsmooth convex variational signal recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 564–574, 2007.

[8] M. J. Fadili, J. L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *The Computer Journal*, vol. 52, pp. 64–79, 2007.

[9] M.J. Fadili and J.-L. Starck, "Monotone operator splitting for optimization problems in sparse recovery," in *Inter. Conf. Image Processing (ICIP 2009)*, 2009, pp. 1461 –1464.

[10] R. M. Larsen, "Lanczos bidiagonalization with partial reorthogonalization," Department of computer science, Aarhus University, Technical report, DAIMI PB-357, code available at http://soi.stanford.edu/ rmunk/PROPACK/, 1998.

978-1-4244-9848-2/11$26.00©2011 IEEE