Noname manuscript No.
(will be inserted by the editor)

Exact Low-Rank Matrix Completion from Sparsely Corrupted Entries via Adaptive Outlier Pursuit

Ming Yan · Yi Yang · Stanley Osher

Received: date / Accepted: date

Abstract Recovering a low-rank matrix from some of its linear measurements is a popular problem in many areas of science and engineering. One special case of it is the matrix completion problem, where we need to reconstruct a low-rank matrix from incomplete samples of its entries. A lot of efficient algorithms have been proposed to solve this problem and they perform well when Gaussian noise with a small variance is added to the given data. But they can not deal with the sparse random-valued noise in the measurements. In this paper, we propose a robust method for recovering the low-rank matrix with adaptive outlier pursuit when part of the measurements are damaged by outliers. This method will detect the positions where the data is completely ruined and recover the matrix using correct measurements. Numerical experiments show the accuracy of noise detection and high performance of matrix completion for our algorithms compared with other algorithms.

Keywords Robust Matrix Completion · Adaptive Outlier Pursuit · Random-valued Noise · Grassmann Manifold

Mathematics Subject Classification (2000) 15A83 · 90C26 · 90C27

1 Introduction

Nowadays, a lot of real world models can be categorized as matrix completion (MC) problems, such as video denoising [17], data mining and pattern recognitions [12], model reduction [13], low-dimensional embedding [21] etc. The

This work is supported by NSF Grant DMS-0714945, Center for Domain-Specific Computing (CDSC) under the NSF Expeditions in Computing Award CCF-0926127, ONR Grant N00014-11-1-0719, N00014-08-1-119 and an ARO MURI subcontract from Rice University.

M. Yan · Y. Yang · S. Osher
Department of Mathematics, University of California Los Angeles, Portola Plaza, Los Angeles CA 90095, USA.
E-mail: yanm@math.ucla.edu, yyang@math.ucla.edu, sjo@math.ucla.edu

general form of the MC problem is:

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \text{rank}(X), \text{ s.t. } X_{i,j} = M_{i,j}\ \forall (i,j)\in\Omega, \tag{1.1}$$

where we are given some entries of a matrix X with index set Ω and we want to recover it with its rank as low as possible. $\text{rank}(X)$ is defined as the number of nonzero singular values of X. However, solving (1.1) is often numerically expensive. Hence people tend to consider its relaxation:

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \|X\|_*, \text{ s.t. } X_{i,j} = M_{i,j}\ \forall (i,j)\in\Omega. \tag{1.2}$$

Here $\|X\|_*$ stands for the nuclear norm of X, which is the L_1 norm of the singular values $\sigma_i(X)$, i.e. $\|X\|_* = \sum_{i=1}^{r}\sigma_i(X)$ where $r = \text{rank}(X)$. It has been shown in [7,8,24] that, under certain reasonable conditions, (1.2) and (1.1) share the same solution. [24] also did further study about the recovery for general linear operator $\mathcal{A}:\ \mathbf{R}^{m\times n}\to\mathbf{R}^p$.

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \|X\|_*, \text{ s.t. } \mathcal{A}(X) = y. \tag{1.3}$$

Different types of algorithms have been proposed to solve (1.2), such as linearized Bregman method [4], fixed point and Bregman iterative methods [22] and accelerated proximal gradient algorithm [25]. [25] solved an unconstrained version of (1.2):

$$\underset{X\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \mu\|X\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(M)\|_F^2. \tag{1.4}$$

Here μ is a properly tuned parameter. $\mathcal{P}_\Omega$ stands for the projection onto the subspace of matrices with nonzeros restricted to the index subset Ω, and $\|\cdot\|_F$, the Frobenius norm, is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|A_{i,j}|^2}. \tag{1.5}$$

for any matrix $A = (A_{i,j})_{m\times n}$. Most of the existing MC algorithms require singular value decomposition (SVD) in each iteration, which is the main time cost in these algorithms. In order to get rid of SVD and accelerate the algorithm, the authors in [28] proposed a new method LMaFit (low-rank matrix fitting) which solves a slightly modified version of (1.2):

$$\underset{U,W,Z}{\text{minimize}} \quad \|UW - Z\|_F^2 \text{ s.t. } \ Z_{i,j} = M_{i,j},\ \forall (i,j)\in\Omega. \tag{1.6}$$

Here $U\in\mathbf{R}^{m\times k}$, $W\in\mathbf{R}^{k\times n}$, and $Z\in\mathbf{R}^{m\times n}$, where k is a predicted rank bound. With an appropriate k, it could give us the same result as (1.2). U, W, Z can be updated in an alternating fashion. Following the idea of nonlinear successive over relaxation (SOR) technique, [28] used weighted average between this update and the previous iterate and achieved a faster convergence.

Recently, people have optimized this model and derived other efficient algorithms, such as RTRMC (Riemannian trust-region for matrix completion) [3]. Other approaches include [19,15,5]. We refer the readers to these references for more details.

In practice, there will always be noise in the measurements during data acquisition, therefore, a robust method for solving MC problem is strongly needed. Almost all the existing algorithms can deal with additive Gaussian noise with a relatively small variance, but they can not perform stably when the given data is corrupted by outliers [20], another type of noise which often appears in application. For example, the problem of anticipating people's preference is gaining more and more attention nowadays. We are often asked to rate various kinds of products, such as movies, books, games, or even jokes. This problem is to use incomplete rankings provided by users on some of the products to predict their preference on other unrated products. It is typically treated as a low-rank MC problem. However, as the data collection process often lacks control and sometimes a few people may not be willing to provide their true opinions, the acquired data may contain some outliers. Therefore, applying the regular MC algorithm on this corrupted data may not lead to satisfactory result.

In order to deal with this case, we propose a method using adaptive outlier pursuit (AOP) which adaptively detects the damaged data location with high accuracy. Without the effect of wrong measurements, the reconstruction performance can be improved a lot. This AOP technique has been applied to image denoising in [29] and robust 1-bit compressive sensing in [30] and performed remarkably well. Combining this technique with the existing MC algorithm, our method is able to reconstruct the exact matrix even from sparsely corrupted entries. Here we also want to mention that besides AOP, people have proposed several methods to deal with outliers in the given data, such as [6, 11,16].

This paper is organized as follows. We will describe our algorithm together with other popular methods for robust matrix completion in section 2. Section 3 focuses on the connection between our problem and another robust low-rank matrix approximation model. We also provide extensive study in section 4 on the case when we only have limited information about the noise. The performance of the algorithms is shown in section 5. We will end this work by a short conclusion.

2 Algorithm description

From now on, let us assume that the rank r is given in advance, i.e. the rank estimate k is set to be r. According to massive experiments, the model (1.6) proves to be a quite efficient way to deal with MC problems when some information about the rank is known in advance. One drawback about this formulation is that the solution (U, W) is not unique. As a matter of fact, for

any $r \times r$ invertible matrix A, $(UA, A^{-1}W)$ is another pair of solution. Many people have devoted to improve this model, such as [9,10,18,19,23,2,26].

The author in [3] combined the ideas in these work and proposed the following model and the associated algorithm RTRMC:

$$\underset{\mathcal{U}\in\mathcal{G}(m,r),W\in\mathbf{R}^{r\times n}}{\text{minimize}} \frac{1}{2}\sum_{(i,j)\in\Omega} C_{i,j}^2((UW)_{i,j}-M_{i,j})^2+\frac{\lambda^2}{2}\sum_{(i,j)\notin\Omega}(UW)_{i,j}^2. \quad (2.1)$$

Here r is the given rank, $U \in \mathbf{R}^{m\times r}$ is any matrix such that its column space $\mathcal{U}$ belongs to the Grassmann manifold $\mathcal{G}(m,r)$. The confidence index $C_{i,j} > 0$ is introduced for each observation, and λ is a weighted parameter. A Riemannian trust-region method, GenRTR [1] was used to solve the above optimization problem on the Grassmannian. According to the numerical experiments, RTRMC outperforms other state-of-the-art algorithms on a wide range of problem instances. It is especially efficient for rectangular matrices and achieves a much smaller relative error.

However, its performance will be ruined when sparse random-valued noise is introduced to the measurements. In order to obtain better result, adaptively finding the error locations and reconstructing the matrix can be combined together as in [29,30]. Here we will plant this idea into the existing model.

We define K as the number of error terms in the given data and derive the following revised model:

$$\begin{cases} \underset{\mathcal{U},W,\Lambda}{\text{minimize}} & \frac{1}{2}\sum_{(i,j)\in\Omega} C_{i,j}^2\Lambda_{i,j}((UW)_{i,j}-M_{i,j})^2+\frac{\lambda^2}{2}\sum_{(i,j)\notin\widetilde{\Omega}}(UW)_{i,j}^2 \\ \text{s.t.} & \sum_{(i,j)\in\Omega}(1-\Lambda_{i,j})\le K,\ \Lambda_{i,j}\in\{0,1\}. \end{cases} \quad (2.2)$$

Here $\Lambda \in \mathbf{R}^{m\times n}$ is a binary matrix denoting the "correct" data:

$$\Lambda_{i,j} = \begin{cases} 1, & \text{if } (i,j)\in\Omega,\ M_{i,j} \text{ is "correct"}, \\ 0, & \text{otherwise.} \end{cases} \quad (2.3)$$

and $\widetilde{\Omega}$ is a subspace of Ω such that $\Lambda_{i,j} = 1$ for all the indices in $\widetilde{\Omega}$. In this work, we only consider the case with sparsely corrupted measurements, and the other measurements are assumed to be correct. The parameter λ is set to be 10^{-8}, i.e., the term $\sum_{(i,j)\notin\widetilde{\Omega}}(UW)_{i,j}^2$ can be neglected. All the entries of the confidence matrix C are chosen to be 1. Hence the model can be simplified into:

$$\begin{cases} \underset{\mathcal{U},W,\Lambda}{\text{minimize}} & \sum_{(i,j)\in\Omega}\Lambda_{i,j}((UW)_{i,j}-M_{i,j})^2, \\ \text{s.t.} & \sum_{(i,j)\in\Omega}(1-\Lambda_{i,j})\le K,\ \Lambda_{i,j}\in\{0,1\}. \end{cases} \quad (2.4)$$

In order to solve this non-convex problem, we use alternating minimization method, which splits the energy minimization over Λ and U, W into two steps:

– Fix Λ and update U, W. We need to solve the following sub-problem:

$$\underset{U,W}{\text{minimize}} \sum_{(i,j)\in\widetilde{\Omega}} ((UW)_{i,j} - M_{i,j})^2. \tag{2.5}$$

This can be solved with RTRMC.

– Fix U, W and update Λ. This time we are solving:

$$\begin{cases} \underset{\Lambda}{\text{minimize}} & \sum_{(i,j)\in\Omega} \Lambda_{i,j}((UW)_{i,j} - M_{i,j})^2, \\ \text{s.t.} & \sum_{(i,j)\in\Omega} (1-\Lambda_{i,j}) \le K,\ \Lambda_{i,j} \in \{0,1\}. \end{cases} \tag{2.6}$$

This problem is to choose $|\Omega|-K$ elements with least sum from $\{((UW)_{i,j}-M_{i,j})^2,\ (i,j)\in\Omega\}$. Here $|\Omega|$ stands for the number of elements in set Ω. Defining τ as the value of the K^{th} largest term in that set, Λ can then be calculated by

$$(\Lambda)_{i,j} = \begin{cases} 1, & \text{if } (i,j)\in\Omega,\ ((UW)_{i,j} - M_{i,j})^2 < \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

If the K^{th} and $(K+1)^{th}$ larges terms have the same value, then we can choose any Λ such that $\sum_{(i,j)\in\Omega}(1-\Lambda_{i,j}) = K$ and

$$\min_{(i,j)\notin\widetilde{\Omega}} ((UW)_{i,j} - M_{i,j})^2 \ge \max_{(i,j)\in\widetilde{\Omega}} ((UW)_{i,j} - M_{i,j})^2. \tag{2.8}$$

In each iteration, we use Λ to identify the location of outliers based on the newly constructed U and W. This outlier detection technique, defined as adaptive outlier pursuit (AOP), was firstly used in [29,30]. Our algorithm is as follows:

Algorithm 1 RTRMC with AOP

Input: Ω, $\mathcal{P}_\Omega(M)$, Miter > 0, $r > 0$, $K \ge 0$, C, λ.
Initialization: $k = 0$, $\Lambda_{i,j} = 1$ for $(i,j)\in\Omega$ and 0 otherwise, $\widetilde{\Omega} = \Omega$.
while $k \le$ Miter **do**
  Replace Ω in (2.1) with $\widetilde{\Omega}$ and update U^k and W^k with RTRMC.
  Update Λ^k with (2.7).
  Update $\widetilde{\Omega}$ to be the indices in Ω where $\Lambda^k_{i,j} = 1$.
  If this new $\widetilde{\Omega}$ is the same as the old $\widetilde{\Omega}$, break.
  $k = k + 1$.
end while
return $U^k W^k$.

This algorithm, together with other two methods, SpaRCS (sparse and low-rank decomposition via compressive sensing) [27] and GRASTA (Grassmannian robust adaptive subspace tracking algorithm) [14], will be studied and compared with extensive numerical experiments. SpaRCS is a recently

proposed algorithm which aims at recovering low rank and sparse matrices from compressive measurements with the following model:

$$\underset{L,S}{\text{minimize}} \quad \|y-\mathcal{A}(L+S)\|_2, \text{ s.t. } \operatorname{rank}(L)\leq r, \|S\|_0\leq K. \tag{2.9}$$

Here $\|S\|_0$ stands for the number of nonzero entries of the matrix S. In our test this linear transformation $\mathcal{A}(L+S)$ is defined as the vector formed by the entries of $(L+S)$ in Ω. This model can be applied to solve MC problem with sparsely corrupted entries. We can form the vector y with the given noisy data. S can be treated as the matrix recording outliers, and L is the low-rank matrix we want to recover. GRASTA, a robust subspace tracking algorithm, is designed to tackle the following model:

$$\underset{S,W,\mathcal{U}}{\text{minimize}} \quad |\mathcal{P}_\Omega(S)|_1, \text{ s.t. } \mathcal{P}_\Omega(UW+S)=\mathcal{P}_\Omega(M),\ \mathcal{U}\in\mathcal{G}(m,r). \tag{2.10}$$

It alternates between estimating the subspace $\mathcal{U}$ with Grassmannian and finding the optimal W, S with augmented Lagrangian function. According to the numerical experiments in that paper, it can efficiently recover a low-rank matrix from partial measurements, even if the partially observed entries are corrupted by gross outliers.

3 Connection between (2.4) and (2.9)

In this section, we will show the connection between our problem and (2.9) with specially defined $\mathcal{A}(\cdot)$, i.e.

$$\underset{L,S\in\mathbf{R}^{m\times n}}{\text{minimize}} \quad \|\mathcal{P}_\Omega(M-L-S)\|_F, \text{ s.t. } \operatorname{rank}(L)\leq r, \|S\|_0\leq K. \tag{3.1}$$

We can change $\|\cdot\|_F$ in the above problem to $\|\cdot\|_F^2$ while still getting the same solution. Basically, for (3.1) we are given partial entries of a matrix ($M_{i,j}$ with $(i,j)\in\Omega$) and we want to represent this M by the sum of a low-rank matrix L and a sparse matrix S.

If a matrix pair (L,S) satisfies the constraints of problem (3.1), we can define

$$\Lambda_{i,j}=\begin{cases}1, \text{ if } (i,j)\in\Omega,\ S_{i,j}=0,\\ 0, \text{ otherwise.}\end{cases} \tag{3.2}$$

Hence for any $(i,j)\in\Omega$, we have $M_{i,j}=L_{i,j}+S_{i,j}$ if $\Lambda_{i,j}=0$, since we can simply set $S_{i,j}=M_{i,j}-L_{i,j}$ for fixed L without violating the constraint on S. If $\Lambda_{i,j}=1$, we have $S_{i,j}=0$, thus $M_{i,j}-(L_{i,j}+S_{i,j})=M_{i,j}-L_{i,j}$. Therefore, (3.1) is equivalent to

$$\begin{cases}\underset{L,S,\Lambda}{\text{minimize}} & \displaystyle\sum_{(i,j)\in\Omega}\Lambda_{i,j}(M_{i,j}-L_{i,j})^2\\ \text{s.t. } \operatorname{rank}(L)\leq r, \|S\|_0\leq K,\ \Lambda \text{ satisfies (3.2).}\end{cases} \tag{3.3}$$

From the relation of Λ and S in (3.2) and the constraint $\|S\|_0 \leq K$, we know the constraints for S and Λ in the above equation can be replaced by the constraint on Λ in (2.4). On the other hand, we know any matrix L with size $m \times n$ and $\text{rank}(L) \leq r$ can be written as the product of two matrices U and W, where $U \in \mathbf{R}^{m\times r}$ and $W \in \mathbf{R}^{r\times n}$. The global optimal solution of one problem is equivalent to the global optimal solution of the other problem. Since these two problems are non-convex problems, a local optimal solution of one problem may not be equivalent to a local optimal solution of the other problem. However, the problem of minimizing the function of L only by eliminating S and Λ for both problems are the same.

4 The K study

In practice, the exact number of corrupted entries K may be unknown. If K is underestimated, some damaged entries will still be used to solve the matrix completion problem, which will induce error for the reconstructed matrix. On the other hand, if K is overestimated, too many entries are removed and the reconstructed matrix may not be unique if the "new" sampling set is not large enough. We first focus on the case when K is overestimated.

When K is overestimated, we can always find more than one solution of problem (2.4) with the objective function value being zero. If K is overestimated by a small number, the product of U and W will be the same for all the solutions and we are able to recover the original low-rank matrix. When the difference is greater than a certain number, UW may not be unique. The following theorem provides a sufficient condition for the non-uniqueness of the matrix UW.

Theorem 1 *Suppose we are given p entries of an $m \times n$ matrix M, where the locations of these data are chosen randomly. We know in advance that K of the given entries are corrupted. Define the difference between our overestimated K value and the real K value as ΔK. If ΔK satisfies $\Delta K > (p-K)/\max(m,n) - r > 0$, then the reconstructed matrix will be non-unique.*

The above theorem provides a necessary condition for K estimate in order to have uniqueness of the problem. In practice, what we care about is how much we can overestimate K without sacrificing the accuracy of our algorithm. Since K is closely related to the location of the known data, how the given entries are distributed over the matrix will be important for our study. In the following theorem, we assume that whether the value at each entry is given or not is independent and identically distributed. Let us define k_i^r as the number of given entries in the i^{th} row and k_j^c as the number of given entries in the j^{th} column. $k_{\min}$ denotes the minimum of all the k_i^r and k_j^c. Through extensive numerical tests, we notice that the distribution of $k_{\min}$ is similar to the conditional distribution of $k_{\min}$ given the total number of known entries. For simplicity we use the former one to replace the conditional distribution and arrive at the following theorem.

Theorem 2 *Suppose that the probability of each entry being given is* $q = (p-K)/(mn)$*, and whether the entry is known or not is independent of other entries. For any small number* $P_0 \in (0,1)$*, let us define*

$$K_1 = \min\left(nq - \sqrt{\frac{-n}{2}\log(1-(1-P_0/2)^{1/m})}, \right. \\ \left. mq - \sqrt{\frac{-m}{2}\log(1-(1-P_0/2)^{1/n})}\right) \tag{4.1}$$

$$K_2 = \min\left(nq - \sqrt{-2nq\log(1-(1-P_0/2)^{1/m})}, \right. \\ \left. mq - \sqrt{-2mq\log(1-(1-P_0/2)^{1/n})}\right). \tag{4.2}$$

Then with at most P_0 *probability there exists one row or column in this matrix with at most* $\max(K_1, K_2)$ *given entries.*

The proof of the above two theorems can be found in the appendix. As we know, the uniqueness of MC problem with outliers depends on a lot of subtle factors. Here we want to derive an empirical upper bound for ΔK such that when ΔK is bounded by this value, with high probability our algorithm can recover the exact matrix. Considering the revised sampling set (the set with $(p-K)$ entries), in order to study the relationship between this upper bound and the number of given entries in each row and column, we design the following experiment. We first fix the matrix size 512×512. For each rank r, the sample ratio sr, defined as $p/(mn)$, is chosen as the smallest number which could guarantee the exact matrix recovery when the real K value is used as input. For more details about the sr value, we refer the readers to the phase transition charts in Section 5. Labeling the minimum of all the k_i^r and k_j^c from the revised sampling set as $k_{\min}$, we randomly choose 10 positive integers bounded above by $(k_{\min} - r)$ and treat them as ΔK. For each input $(K+\Delta K)$, the error of the recovered matrix M_r is calculated with the following expression:

$$\text{Err} = \max_{i,j} |M_{i,j} - (M_r)_{i,j}|. \tag{4.3}$$

If the error is less than 10^{-4}, we say the recovery is successful. Otherwise, we label it as a failure. The results displayed in Table 1 come from the average of 20 different tests for each setting.

Through massive experiments, we can see that if ΔK is smaller than $(k_{\min} - r)$, our algorithm can find the correct matrix with extremely high probability. Hence we come up with the following conclusion: for any small number P_0, we can find two values K_1 and K_2 according to Theorem 2 such that with at most P_0 probability there exists one row or column with at most $\max(K_1, K_2)$ given entries. Then, if ΔK is less than $(\max(K_1, K_2) - r)$, with high probability our algorithm will return the exact matrix with this overestimated K input.

rank & sample rate	avg K/p	avg $k_{\min}$	success percentage
rank=5, sr=0.15	0.01	50.75	100%
	0.03	49.15	100%
	0.05	49.00	100%
	0.07	47.05	100%
	0.09	45.35	100%
rank=10, sr=0.20	0.01	72.25	100%
	0.03	70.70	100%
	0.05	68.40	100%
	0.07	67.65	100%
	0.09	66.30	100%
rank=15, sr=0.25	0.01	95.25	100%
	0.03	93.20	100%
	0.05	92.75	100%
	0.07	89.25	100%
	0.09	85.40	100%
rank=20, sr=0.30	0.01	119.55	100%
	0.03	116.70	100%
	0.05	114.50	100%
	0.07	111.30	100%
	0.09	107.25	100%
rank=25, sr=0.35	0.01	143.30	100%
	0.03	139.20	100%
	0.05	136.15	100%
	0.07	134.10	100%
	0.09	129.30	100%

Table 1: The K study. For each matrix 10 different K inputs are chosen and 20 trials are conducted for each matrix setting.

In application, when K is overestimated, according to the above conclusion we just need to construct a strategy to update it such that ΔK can be bounded by $(k_{\min}-r)$. Let us define $\widetilde{K}$ as the estimated K value. One intuitive idea is to check the value of the $\widetilde{K}^{th}$ largest term in set $S_\Omega = \{((UW)_{i,j}-M_{i,j})^2,\ (i,j) \in \Omega\}$ in each iteration. If this value is less than the tolerance K_{tol}, it is possible that some of the deleted data are not outliers, and we can update $\widetilde{K}$ to be the number of terms in this set which are greater than K_{tol}. When our algorithm reaches a certain stage, we can calculate the minimum number of entries in one row or column from the sampling set with $(p-\widetilde{K})$ elements (label as $\widetilde{k}_{\min}$). If it is less than r, we update $\widetilde{K} = \widetilde{K} + \widetilde{k}_{\min} - r$. Here we just choose the smallest decrease in $\widetilde{K}$. In fact, we may pick a larger decrease in order to reduce the number of outer iterations. On the other hand, when K is underestimated, we need to increase its value in order to remove all the outliers. As we know, when there are outliers in the given data, it is quite possible that we are not

able to find a low-rank matrix with entries equalling the given data at these locations. Based on the fast convergence of our algorithm, if at certain iteration the difference between the entries of the recovered matrix at those $(p - \widetilde{K})$ locations and the associated input data are greater than tolerance ϵ, we can update $\widetilde{K}$ to be $\rho_1\widetilde{K}$ with $\rho_1 > 1$. ρ_1 should be chosen properly. When it is too small, we need a lot of steps to make $\widetilde{K}$ larger than the exact K, however, when ρ_1 is too large, $\widetilde{K}$ may go far above the exact K, and more iterations are required to decrease its value. Therefore, we arrive at the following algorithm.

Algorithm 2 RTRMC-AOP with K update

Input: Ω, $\mathcal{P}_\Omega(M)$, InnerMiter, OuterMiter > 0, $r > 0$, $\widetilde{K} \geq 0$, C, λ, K_{tol}, ρ_1, ϵ .
Initialization: $k, l = 0$, $\Lambda_{i,j} = 1$ for $(i,j) \in \Omega$ and 0 otherwise, $\widetilde{\Omega} = \Omega$.
while $l \leq$ OuterMiter **do**
  while $k \leq$ InnerMiter **do**
    Replace Ω in (2.1) with $\widetilde{\Omega}$ and update U^k and W^k with RTRMC.
    Find the $\widetilde{K}^{th}$ largest term in set S_Ω.
    If it is less than K_{tol}, update $\widetilde{K}$ to be the number of elements in S_Ω that are greater than K_{tol}.
    Update Λ^k with (2.7).
    Update $\widetilde{\Omega}$ to be the indices in Ω where $\Lambda^k_{i,j} = 1$.
    If this new $\widetilde{\Omega}$ is the same as the old $\widetilde{\Omega}$, break.
    $k = k + 1$.
  end while
  Calculate $\widetilde{k}_{\min}$ with the updated $\widetilde{K}$ value.
  If $\widetilde{k}_{\min} < r$, $\widetilde{K} = \widetilde{K} + \widetilde{k}_{\min} - r$.
  If $\widetilde{k}_{\min} \geq r$ and the function value of (2.5) is less than ϵ, break.
  If $\widetilde{k}_{\min} \geq r$ and the function value is greater than ϵ, $\widetilde{K} = \rho_1\widetilde{K}$.
  $k = 0$, $l = l + 1$.
end while
return U^kW^k, $\widetilde{K}$.

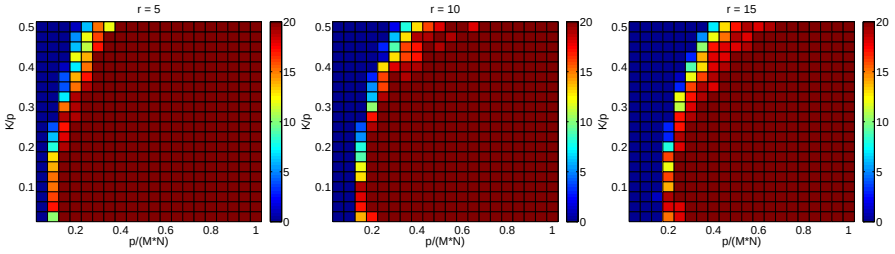
5 Numerical results

In this section we use some numerical experiments to demonstrate the effectiveness of our algorithm (AOP for short). AOP, together with SpaRCS and GRASTA are studied and compared.

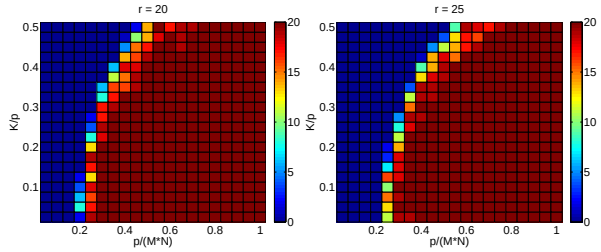
In each experiment, the original matrix M is generated by the product of $U \in \mathbf{R}^{m\times r}$ and $W \in \mathbf{R}^{r\times n}$, whose entries follow independent and identically distributed (i.i.d.) Gaussian distribution with variance 1. We denote the largest and smallest value of M as m_L and m_S. p entries are chosen randomly from M and their locations are recorded in Ω. We then pick K locations randomly from Ω and replace the values at these locations by a random number from $[m_S, m_L]$. The corrupted p entries and Ω are used as input in our code.

In the first experiment, we investigated the empirical performance of Algorithm 1 by testing it under different circumstances. Here the size of the matrix is chosen to be $m = n = 512$ and different values of r, p and K are considered. For each small rectangle in Figure 1, we fix the value of r, p and

K, and applied AOP to recover the low-rank matrix. If the relative error, i.e. $\|M_r - M\|_F / \|M\|_F$, is smaller than 10^{-3}, we denote the test as "successful". 20 different tests are conducted for each setting and the probability of successful recovery are recorded. Here red indicates recovery success and blue indicates failure. As expected, the performance gets worse when we increase r or K or decrease p.



(a) r=5 (b) r=10 (c) r=15

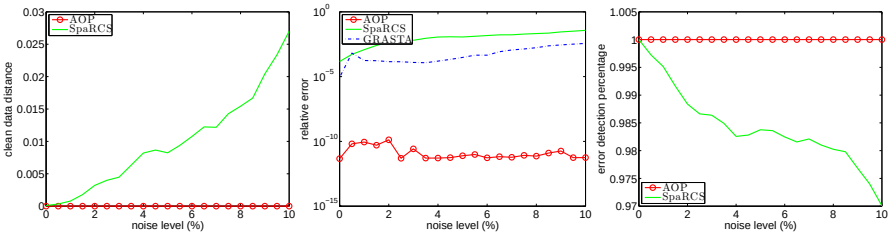


(d) r=20 (e) r=25

Fig. 1: Phase transitions for a recovery problem of size 512×512. Aggregate results are shown over 20 Monte-Carlo runs at each specification of r, K and p. Red indicates recovery success and blue indicates failure.

In the following two tests, the results of SpaRCS and GRASTA are also shown in the figures. Let us assume the K value is known in advance, i.e. Algorithm 1 is used in the comparison. We will compare these three items (since GRASTA solves a different problem, only the relative error is compared):
1) distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$, i.e. $\|P_{\widetilde{\Omega}}(M_r) - P_{\widetilde{\Omega}}(M)\|_F$;
2) relative error;
3) the probability of correct detections of corrupted data in the noisy measurements.

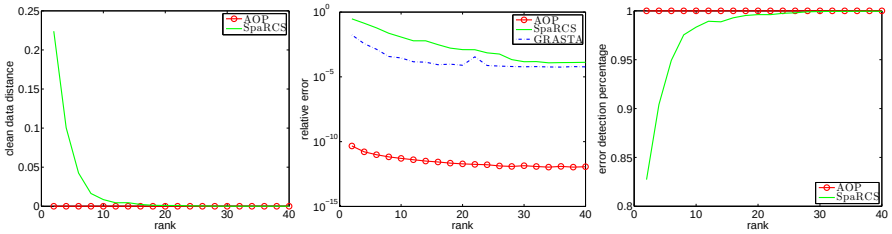
In our second experiment, we set $m = n = 500$, $r = 10$, and examine the performance of these algorithms on data with different noise levels. Here the noise level is defined as K/p. 21 noise levels are chosen by $5 \cdot i \cdot 10^{-3}$ with i ranging from 0 to 20. p is calculated by $6r(m + n - r)$. 20 trials are performed for each noise level and the mean of the above three items are recorded in



(a) Distance between clean data (b) Relative error (c) Error detection

Fig. 2: Algorithm comparison on corrupted data with different noise levels. (a) Distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$ vs noise level, (b) Relative error vs noise level, (c) Error location detection vs noise level. AOP proves to be more robust to contaminated data compared with others.

Figure 2. The plots demonstrate that in these comparisons AOP outperforms the other two greatly for all noise levels. According to the relative error plot, the result from our method is always around 10^{-10} while the result gained by the other two algorithms is bounded below by 10^{-4}. Compared with SpaRCS, GRASTA is slightly more stable when the given data is ruined by gross outliers. In plot (c), we record the probabilities of correct error locations detection. From the graph we can see that AOP algorithms can find all the positions of corrupted data with probability 1, while in comparison the performance of SpaRCS detection is not very satisfying.



(a) Distance between clean data (b) Relative error (c) Error detection

Fig. 3: Algorithm comparison on corrupted data with different rank. (a) Distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$ vs rank, (b) Relative error vs rank, (c) Error location detection vs rank. AOP yields a remarkable improvement in reducing the relative error and finding the correct error locations compared with others.

Next we fix $m = n = 500$ and the noise level 5% and change r between 2 and 40. p is still calculated by $6r(m + n - r)$. The results in Figure 3 come from the average of 20 tests. We can see that the distance between $P_{\widetilde{\Omega}}(M_r)$ and $P_{\widetilde{\Omega}}(M)$ and the relative error tend to decrease as the rank of M increases. Although all the algorithms show the same trend, AOP series always return a much better result with relative error staying around 10^{-11}, and it detects the true error locations with probability 1 in all the tests. The relative error from GRASTA is bounded below by 10^{-5} and when the rank is extremely low, the relative error could be as high as 10^{-2}.

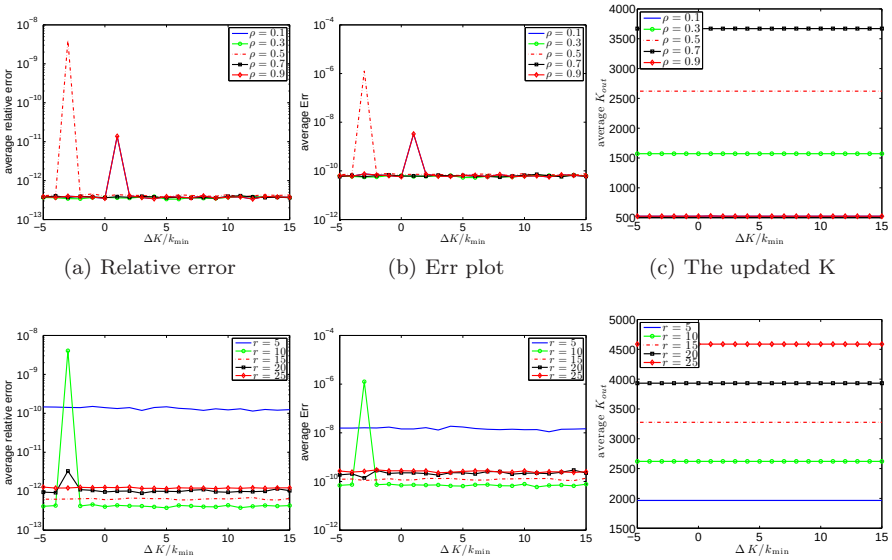
In the previous experiments, we assume the exact K value is always given. Now let us study the case when the input K is just an estimate of the actual number of errors. This time we fix $m = n = 512$, and examine the performance of Algorithm 2 under different settings. The relative error, Err defined by (4.3) and the updated K value will be displayed here. In Figure (a)-(c), we set $r = 10$ and pick 5 different noise levels between 0.01 and 0.09. For each setting, we calculate $k_{\min}$, and choose 21 different ΔK between $-5k_{\min}$ and $15k_{\min}$. Then each $(K + \Delta K)$ is used as the input K value. In Figure (d)-(f), we fix the noise level to be 0.05 and vary r from 5 to 25. Still, 21 different ΔK values are selected. All the p values in these tests are chosen the same as Table 1. The following results come from the average of 20 different trials. We can see that in all the cases our AOP with K update algorithm can detect the correct number of outliers with high probability even when the input K differs a lot from its real value. The relative error plot and Err plot suggest that this method always recovers the matrices with extremely high accuracy.

6 Conclusion

In this paper, we propose a method for exact low-rank matrix completion from sparsely corrupted data via adaptive outlier pursuit. By iteratively detecting the damaged measurements and recovering the matrix from "correct" measurements, this method can obtain better results in both finding the noisy measurements and recovering the exact matrix when random-valued noise is introduced in the measurements. Our algorithm is implemented and compared with SpaRCS and GRASTA in the numerical experiments. It has better performance in many aspects compared to the other two, especially in detecting all the outlier locations. When the exact value of the number of outliers is not provided, the AOP with K update algorithm can always detect the correct number of outliers and recover the exact matrix in all the cases with high probability. Our next step is to study the case when the given data is corrupted by Gaussian noise as well as sparse random-valued noise.

7 Appendix

This appendix provides the mathematical proofs of the theoretical results in Section 4.



(a) Relative error (b) Err plot (c) The updated K

(d) Relative error (e) Err plot (f) The updated K

Fig. 4: The K study. For (a)-(c) we fix the rank and vary the noise level. (a) relative error vs different K input. (b) Err vs different K input. (c) K output vs different K input. For (d)-(f) we fix the noise level and vary the rank of the matrix. (d) relative error vs different K input. (e) Err vs different K input. (f) K output vs different K input.

7.1 Proof of Theorem 1

Proof When we overestimate K, the K outliers will be found to make the objective function 0. Therefore, we only need to consider the $(p-K)$ correct entries. In the mean time, some non-outliers (the overestimated ΔK entries) are also considered as outliers and will not be used for matrix completion. As we know, if the number of given entries in one row (or column) is less than r, the reconstructed matrix is not unique. Since ΔK of the $(p-K)$ known entries will not be used in reconstructing the matrix, when ΔK is large enough to make the number of known entries in one row (or column) less than r, we will have more than one solution. It is easily seen that the smallest number of known entries in one row (or column) of the matrix is less than or equal to $\lfloor (p-K)/m \rfloor$ (or $\lfloor (p-K)/n \rfloor$), here $\lfloor x \rfloor$ is the largest integer that does not exceed x. Without loss of generality, let us assume column j has the smallest number of known entries. It is obvious that this number is bounded by $\min(\lfloor (p-K)/m \rfloor, \lfloor (p-K)/n \rfloor)$. To make the number of known entries in this column no less than r, the smallest number of entries to be deleted should not exceed $\min(\lfloor (p-K)/m \rfloor, \lfloor (p-K)/n \rfloor) - r + 1$. Thus if ΔK is greater

than $\min(\lfloor (p-K)/m \rfloor, \lfloor (p-K)/n \rfloor) - r$, the reconstructed matrix will not be unique.

7.2 Proof of Theorem 2

Proof Since the probability that a certain location is chosen is fixed and equals $q = (p-K)/(mn)$. In addition, whether one entry is chosen or not is independent of other entries. The number of known entries in each row (or column) of this matrix follows binomial distribution. For each row, the cumulative distribution can be expressed as

$$F(x, n, q) = P(X \leq x) = \sum_{i=0}^{\lfloor x \rfloor} \binom{n}{i} q^i (1-q)^{n-i}, \tag{7.1}$$

where X is the number of known entries in this row.

From the Hoeffding's inequality, we have

$$F(k, n, q) \leq \exp\left(-2\frac{(nq-k)^2}{n}\right), \tag{7.2}$$

for any integer $k \leq nq$. Since the distribution of the number of known entries in each row is independent, we can find the upper bound for the probability that there exists one row with at most k given entries:

$$P(\min(k_1^r, k_2^r, \cdots, k_m^r) \leq k) \leq 1 - \left(1 - \exp\left(-2\frac{(nq-k)^2}{n}\right)\right)^m := P_1. \tag{7.3}$$

Here k_i^r stands for the number of given entries in the i^{th} row. Similarly the upper bound for the probability that there exists one column with at most k given entries can be expressed as follows:

$$P(\min(k_1^c, k_2^c, \cdots, k_n^c) \leq k) \leq 1 - \left(1 - \exp(-2\frac{(mq-k)^2}{m})\right)^n := P_2. \tag{7.4}$$

where k_j^c is defined as the number of given entries in the j^{th} column. Combing these two together, we have

$$P(\min(k_1^r, k_2^r, \cdots, k_m^r, k_1^c, k_2^c, \cdots, k_n^c) \leq k) \leq P_1 + P_2 \leq 2\max(P_1, P_2), \tag{7.5}$$

which means the probability that there exists one row or column with at most k given entries can be bounded by $2\max(P_1, P_2)$.

Let us first assume $P_1 > P_2$. Defining

$$P_0 = 2\left(1 - \left(1 - \exp\left(-2\frac{(nq-k)^2}{n}\right)\right)^m\right), \tag{7.6}$$

we then have

$$\exp\left(-2\frac{(nq-k)^2}{n}\right) = 1-\left(1-\frac{P_0}{2}\right)^{1/m} \tag{7.7}$$

$$\Longrightarrow k = nq - \sqrt{\frac{-n}{2}\log(1-(1-P_0/2)^{1/m})}. \tag{7.8}$$

When $P_1 < P_2$, we have

$$k = mq - \sqrt{\frac{-m}{2}\log(1-(1-P_0/2)^{1/n})}. \tag{7.9}$$

We define K_1 as the minimal of these two values. Hence given P_0, the probability of having one row or column with at most K_1 entries is less than P_0.

Besides the Hoeffding's inequality, we also have Chernoff's inequality,

$$F(k,n,q) \leq \exp\left(-\frac{1}{2q}\frac{(nq-k)^2}{n}\right). \tag{7.10}$$

In this case

$$P_1 = 1-\left(1-\exp\left(-\frac{1}{2p}\frac{(nq-k)^2}{n}\right)\right)^m \tag{7.11}$$

$$P_2 = 1-\left(1-\exp\left(-\frac{1}{2p}\frac{(mq-k)^2}{m}\right)\right)^n. \tag{7.12}$$

After similar calculation, we have

$$k = nq - \sqrt{-2nq\log(1-(1-P_0/2)^{1/m})} \tag{7.13}$$

for $P_1 > P_2$, and

$$k = mq - \sqrt{-2mq\log(1-(1-P_0/2)^{1/n})} \tag{7.14}$$

when $P_1 < P_2$. Similarly we define K_2 to be the smaller one of these two values, and the probability of having one row or column with at most K_2 entries is less than P_0. Combining the results from two inequalities together, we know that with at most P_0 probability there exists one row or column with at most $\max(K_1, K_2)$ given entries.

References

1. Absil, P.A., Baker, C.G., Gallivan, K.A.: Trust-region methods on Riemannian manifolds. Foundations of Computational Mathematics **7**(3), 303–330 (2006) 4
2. Balzano, L., Nowak, R., Recht, B.: Online identification and tracking of subspaces from highly incomplete information. In: Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on, pp. 704 –711 (2010) 4
3. Boumal, N., Absil, P.A.: RTRMC: A Riemannian trust-region method for matrix completion. In: Twenty-Fifth Annual Conference on Neural Information Processing Systems (2011) 3, 4
4. Cai, J., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM J. on Optimization **20**, 1956–1982 (2010) 2
5. Cai, J., Osher, S.: Fast singular value thresholding without singular value decomposition. In: UCLA CAM Report 10-24 (2010) 3
6. Candès, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? Journal of ACM **58**(1) (2009) 3
7. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Foundations of Computational Mathematics **9**, 717–772 (2008) 2
8. Candès, E.J., Tao, T.: The power of convex relaxation: Near-optimal matrix completion. Information Theory, IEEE Transactions on **56**(5), 2053 –2080 (2010) 2
9. Dai, W., Kerman, E., Milenkovic, O.: A geometric approach to low-rank matrix completion. Information Theory, IEEE Transactions on **58**(1), 237 –247 (2012) 4
10. Dai, W., Milenkovic, O., Kerman, E.: Subspace evolution and transfer (SET) for low-rank matrix completion. Signal Processing, IEEE Transactions on **59**(7), 3120 –3132 (2011) 4
11. Dong, B., Ji, H., Li, J., Shen, Z., Xu, Y.: Wavelet frame based blind image inpainting. Applied and Computational Harmonic Analysis **32**(2) (2012) 3
12. Elden, L.: Matrix Methods in Data Mining and Pattern Recognition. Society for Industrial and Applied Mathematics, Philadephia, PA (2007). DOI DOI:10.1137/1.9780898718867 1
13. Fazel, M., Hindi, H., Boyd, S.P.: A rank minimization heuristic with application to minimum order system approximation. Proceedings of the 2001 American Control Conference Cat No01CH37148 **6**(2), 4734–4739 (2001) 1
14. He, J., Balzano, L., Lui, J.C.S.: Online robust subspace tracking from partial information. CoRR **abs/1109.3827** (2011) 5
15. Jain, P., Meka, R., Dhillon, I.: Guaranteed rank minimization via singular value projection. In: J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R. Zemel, A. Culotta (eds.) Advances in Neural Information Processing Systems 23, pp. 937–945 (2010) 3
16. Ji, H., Huang, S., Shen, Z., Xu, Y.: Robust video restoration by joint sparse and low rank matrix approximation. SIAM Journal on Imaging Sciences **4**(4) (2012) 3
17. Ji, H., Liu, C., Shen, Z., Xu, Y.: Robust video denoising using low rank matrix completion. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference (2010) 1
18. Keshavan, R., Montanari, A.: Regularization for matrix completion. In: Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on, pp. 1503 –1507 (2010) 4
19. Keshavan, R., Montanari, A., Oh, S.: Matrix completion from a few entries. Information Theory, IEEE Transactions on **56**(6), 2980 –2998 (2010) 3, 4
20. Keshavan, R.H., Montanari, A., Oh, S.: Low-rank matrix completion with noisy observations: a quantitative comparison. In: Proceedings of the 47th annual Allerton conference on Communication, control, and computing, Allerton'09, pp. 1216–1222. IEEE Press (2009) 3
21. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica **15**, 215–245 (1995) 1
22. Ma, S., Goldfarb, D., Chen, L.: Fixed point and Bregman iterative methods for matrix rank minimization. Mathematical Programming **128**(1-2) (2011) 2
23. Meyer, G., Bonnabel, S., Sepulchre, R.: Linear regression under fixed-rank constraints: a Riemannian approach. In: 28th International Conference on Machine Learning (ICML) (2011) 4

24. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Review **52**(3) (2010) 2
25. Toh, K.c., Yun, S.: An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. Engineering **117543**(3), 1–31 (2009) 2
26. Vandereycken, B.: Low-rank matrix completion by Riemannian optimization. submitted (2011) 4
27. Waters, A.E., Sankaranarayanan, A.C., Baraniuk, R.G.: Sparcs: Recovering low-rank and sparse matrices from compressive measurements. In: Neural Information Processing Systems (NIPS). Granada, Spain (2011) 5
28. Wen, Z., Yin, W., Zhang, Y.: Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. Rice University CAAM Technical Report TR1007 Submitted pp. 1–24 (2010) 2
29. Yan, M.: Restoration of images corrupted by impulse noise using blind inpainting and ℓ_0 norm. submitted (2011) 3, 4, 5
30. Yan, M., Yang, Y., Osher, S.: Robust 1-bit compressive sensing using adaptive outlier pursuit. To appear in Signal Processing, IEEE Transactions on (2012) 3, 4, 5