



An algorithm for low-rank matrix factorization and its applications

Baiyu Chen^a, Zi Yang^b, Zhouwang Yang^{a,*}

^a University of Science and Technology of China, Hefei, China

^b University of California, San Diego, USA

ARTICLE INFO

Article history:

Received 2 January 2017

Revised 7 August 2017

Accepted 14 September 2017

Available online 21 September 2017

Communicated by Dr Yiming Ying

Keywords:

Matrix factorization

Sparseness

Low-rank representation

Augmented Lagrangian method

ABSTRACT

This paper proposes a valid and fast algorithm for low-rank matrix factorization. There are multiple applications for low-rank matrix factorization, and numerous algorithms have been developed to solve this problem. However, many algorithms do not use rank directly; instead, they minimize a nuclear norm by using Singular Value Decomposition (SVD), which requires a huge time cost. In addition, these algorithms often fix the dimension of the factorized matrix, meaning that one must first find an optimum dimension for the factorized matrix in order to obtain a solution. Unfortunately, the optimum dimension is unknown in many practical problems, such as matrix completion and recommender systems. Therefore, it is necessary to develop a faster algorithm that can also estimate the optimum dimension. In this paper, we use the Hidden Matrix Factorized Augmented Lagrangian Method to solve low-rank matrix factorizations. We also add a tool to dynamically estimate the optimum dimension and adjust it while simultaneously running the algorithm. Additionally, in the era of Big Data, there will be more and more large, sparse data. In face of such highly sparse data, our algorithm has the potential to be more effective than other algorithms. We applied it to some practical problems, e.g. Low-Rank Representation(LRR), and matrix completion with constraint. In numerical experiments, it has performed well when applied to both synthetic data and real-world data.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

As the dataset that we need to analyze grows rapidly, along with the rapidly developing network, data-mining algorithms should be designed to handle large-scale data. Additionally, this huge data often has some noteworthy features, such as sparsity. A lot of High-Dimension data in practical application is sparse, so it can be expressed in the form of structured sparsity, such as low rank to a matrix. Research on low-rank matrix has received much attention in recent years, as it is an essential part of analyzing massive datasets. Low-rank matrix factorization [13] occurs in many areas, such as machine learning [18], computer vision [16] and motion segmentation [12]. Especially in intelligent video surveillance (e.g. background subtraction [14], visual tracking [8–10]), sparse and low-rank representation play an important part.

The purpose of low-rank factorization is to factorize the matrix into a product of two matrices with low dimensions. The low dimension constrains the rank of the original matrix. Low-rank matrix factorization is an effective tool for analyzing dyadic data in

order to discover the interactions between two entries. Successful applications include keyword searches and recommender systems. Matrix factorization is also applied to matrix completion problems. Generally, it is only possible to access small parts of the data from the whole. Furthermore, we also want to recover the original data from the observed data. Infinite choices exist in the original data, but in practice, we are only interested in the data with the lowest ranking. This is due to the low-rank structure of the original data. Low-rank matrix factorization has been proven to be an effective method to solve problem. Additionally, in some problems, such as Low-Rank Representation (LRR) and Principle Analyze, we need to restrict the rank of the objective matrix. In [5], researchers presented that the nuclear norms [7] of matrices can under some conditions be used to represent the matrix rank. Based on nuclear norms, there are several potential applications. [3] performs a soft-thresholding operation on the singular values of the matrix to approximate nuclear norms. Wei and Lin [20] propose the Robust Shape Interaction Method, which removes noise by using column-sparse robust PCA and then applying LRR to the denoised data. [25] presents an LRR-based discriminative projection method (LRR-DP) for robust feature extraction. Some practical uses for facial recognition have been put forward wherein robust LatLRR is employed [24] and present kernel LRR is used to implement the kernel trick [19]. The nuclear norm that these papers mention is

* Corresponding author.

E-mail address: yangzw@ustc.edu.cn (Z. Yang).

problematic because computing a nuclear norm often requires Singular Value Decomposition (SVD). In practical applications, SVD is too slow to be used when a dataset is large enough.

Our algorithm for matrix factorization can be applied to avoid resorting to SVD. By factorizing the objective matrix into a product of two matrix with low dimensions, its rank is constrained by the factorized matrix. This removes the need for a nuclear norm and thus avoids SVD. In order to solve the transformed problem of Hidden Matrix Factorization, we utilize techniques from the Augmented Lagrangian Method (ALM).

We call the resultant algorithm the Hidden Matrix Factorized Augmented Lagrangian Method (HMFALM), and it can directly solve matrix rankings instead of using nuclear norms. Numerical experiments show that HMFALM is faster than other algorithms without sacrificing accuracy. Especially when the sparsity of High-Dimension data is very high, many algorithms perform worse or even fail completely, but our algorithm still works efficiently. When we solve a low-rank matrix factorization problem, the dimensions of a factorized matrix are always fixed. But in practical problems, the optimum dimensions remain unknown; as a result, the estimated dimensions may be larger or smaller than what is optimal. Bad estimations may lead to slower convergences, over-fitting, poor results, and even divergence. In this paper, we propose a simple but efficient method to find the optimum dimension. Our procedure can be applied to practical problems and shows positive results in various numerical experiments.

The rest of this paper is organized as follows. Section 2 describes the low-rank matrix factorization problem and proposes our algorithm, HMFALM. Section 3 expounds the algorithm's applications in low-rank representation and matrix completion. Numerical experimental results are given in Section 4. Finally, Section 5 concludes this paper.

2. Algorithm

In this section, we present the Hidden Matrix Factorized Augmented Lagrangian Method (HMFALM) that can solve the low-rank matrix factorization problem.

2.1. Basic model

We focus on the matrix factorization problem with the following constraint:

$$\min_{E,P,Q} r_E(E) + r_1(P) + r_2(Q) \quad (1)$$

$$\text{s.t. } X = PQ + E$$

where $X \in \mathbb{R}^{m \times n}$ is the observed matrix corrupted by errors $E \in \mathbb{R}^{m \times n}$, $P \in \mathbb{R}^{m \times r}$, $Q \in \mathbb{R}^{r \times n}$ are the factorized matrix, and r is the estimated rank. The choice of r_E depends on the type of error, for example l_0 norm (number of nonzero entries) would be used to characterize the random and sparse corruptions.

In some particular problems, the factorized matrices P and Q may have some constraints, e.g. they have to be non-negative or P has to be sparse. In the model, these constraints are represented by functions r_1 and r_2 . In different problems, we can choose the appropriate functions, r_1 and r_2 , to satisfy our demands.

2.2. A solver using augmented Lagrangian method

The problem (1) can be solved by the Augmented Lagrangian Method (ALM). ALM is a particular optimization technique that is well-suited for the constrained minimization problem. The general method of ALM is introduced for solving constrained optimization problems of the following kind:

$$\min_x f(x) \quad (2)$$

$$\text{s.t. } c(x) = 0$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}$ and $c: \mathbb{R}^n \rightarrow \mathbb{R}^m$. One may define the augmented Lagrangian function:

$$L(x, Y, \beta) = f(x) + \langle Y, c(x) \rangle + \frac{\beta}{2} \|c(x)\|_F^2 \quad (3)$$

where Y is the Lagrange Multiplier, and β is the penalty parameter.

When ALM is applied to solve problem (1), the augmented Lagrangian function of problem (1) is

$$\begin{aligned} L(P, Q, E, Y, \beta) &= r_1(P) + r_2(Q) + r_E(E) + \langle Y, PQ + E - X \rangle \\ &\quad + \frac{\beta}{2} \|PQ + E - X\|_F^2 \\ &= r_1(P) + r_2(Q) + r_E(E) + \frac{\beta}{2} \left\| PQ + E - X + \frac{Y}{\beta} \right\|_F^2 \\ &\quad - \frac{\|Y\|^2}{2\beta} \end{aligned} \quad (4)$$

Inspired by classical ALM, we approximate it (1) by an Augmented Lagrangian subproblem at each outer iteration to solve the above problem. The form of the subproblem is

$$\min_{P, Q, E} L(P, Q, E, Y, \beta) \quad (5)$$

Within the algorithm, solving subproblem (5) is the focus. Observe that E and (P, Q) are separable, so that we can update E and (P, Q) separately

$$(P^{k+1}, Q^{k+1}) = \arg \min_{(P, Q)} r_1(P) + r_2(Q) + \frac{\beta^k}{2} \left\| PQ + E^k - X + \frac{Y^k}{\beta^k} \right\|_F^2 \quad (6)$$

$$E^{k+1} = \arg \min_E r_E(E) + \frac{\beta^k}{2} \left\| P^{k+1}Q^{k+1} + E - X + \frac{Y^k}{\beta^k} \right\|_F^2 \quad (7)$$

It is difficult to solve (6) directly; thus, we present an inner iterative method to obtain the approximation

$$P_{l+1}^k = \arg \min_P r_1(P) + \frac{\beta^k}{2} \left\| PQ_l^k + E^k - X + \frac{Y^k}{\beta^k} \right\|_F^2 \quad (8)$$

$$Q_{l+1}^k = \arg \min_Q r_2(Q) + \frac{\beta^k}{2} \left\| P_l^k Q + E^k - X + \frac{Y^k}{\beta^k} \right\|_F^2 \quad (9)$$

l is an iterative step in inner the iteration. These forms of P and Q are still difficult to solve; Inspire by [11], we linearize the quadratic term and adding a proximal term to obtain the approximation

$$\begin{aligned} P_{l+1}^k &= \arg \min_P r_1(P) + \frac{\beta^k \zeta_P^k}{2} \|P - P_l^k\|_F^2 \\ &\quad + \mathcal{A}^* \left(P_l^k Q_l^k + E^k - X + \frac{Y^k}{\beta^k} \right) / \zeta_P^k \\ &= \arg \min_P r_1(P) + \frac{\beta^k \zeta_P^k}{2} \|P - P_l^k\|_F^2 \\ &\quad + \left(P_l^k Q_l^k + E^k - X + \frac{Y^k}{\beta^k} \right) (Q_l^k)^T / \zeta_P^k \end{aligned} \quad (10)$$

where $\mathcal{A}(P) = PQ^k$, \mathcal{A}^* is the adjoint of \mathcal{A} and $\zeta_P^k > 0$ is a parameter (we set $\zeta_P^k = 1.02\sigma_{\max}^2(Q_l^k)$ in numerical experiments, which is similar to that in [11]).

Similarly, (9) can be approximated by

$$\begin{aligned} Q_{l+1}^k = \arg \min_Q r_2(Q) + \frac{\beta^k \zeta_Q^k}{2} \|Q - Q_l^k + (P_l^k)^T (P_l^k Q_l^k + E \\ - X + \frac{Y^k}{\beta^k}) / \zeta_Q^k\|_F^2 \end{aligned} \quad (11)$$

As (10) and (11) continue iterating and l grows larger, the iterations stop when the change of (P_l^k, Q_l^k) is not too great. At this time, $(P^k, Q^k) = (P_l^k, Q_l^k)$, which is the approximate solution of (6).

In order to avoid ALM converging to an infeasible point, we adopt the strategies proposed by Lu and Zhang [17]. For convenience, we write the above optimization problem in the form of a general optimization, so we set $f(x) = r_1(P) + r_2(Q) + r_E(E)$ and $c(x) = PQ + E - X$ with $x = (P, Q, E)$.

In this problem, we could easily find one feasible solution denoted by x^0 in the initialization. In this paper, we use singular value decomposition and orthogonal triangular decomposition of the matrix X to get the suitable $P^0 \in \mathbb{R}^{m \times r}, Q^0 \in \mathbb{R}^{n \times r}$. We now present the integral algorithm for solving problem (1).

Algorithm 1

Input: Observed Data X , Estimated Rank r .

Initialize: $E^0 = \mathbf{0}, Y^0 = \mathbf{0}, P^0, Q^0$ s.t. $\text{rank}(P^0) = \text{rank}(Q^0) = r, \beta_0 > 0, \rho > 1, \eta \in (0, 1), \tau \in (0, 1), \{\epsilon_k\} \geq 0$ with $\lim_{k \rightarrow \infty} \epsilon_k = 0$, and a sufficiently large constant $T > \max\{f(x), L(x, Y, \beta)\}$. set $k = 0$

While not converged **do**

1. With $Y = Y^k, \beta = \beta_k$, update (P^{k+1}, Q^{k+1}) according to (10) and (11) to find an approximate solution of (6). Then update E^{k+1} according to (7), so we can find an approximate point $x^{k+1} = (P^{k+1}, Q^{k+1}, E^{k+1})$ s.t.

$$\|\nabla_x L(x, Y^k, \beta^k)\| \leq \epsilon_{k+1}, \quad L(x^{k+1}, Y^k, \beta_k) < T \quad (12)$$

2. Set

$$Y^{k+1} = Y^k + \beta_k(E^k + P^k Q^k - X). \quad (13)$$

3. If $k > 0$ and

$$\|E^{k+1} + P^{k+1} Q^{k+1} - X\| \leq \eta \|E^k + P^k Q^k - X\|, \quad (14)$$

then set $\beta_{k+1} = \beta_k$. Otherwise, set

$$\beta_{k+1} = \max\{\rho \beta_k, \|Y^{k+1}\|^{1+\tau}\}. \quad (15)$$

4. Set $k \leftarrow k + 1$.

End while

2.3. Convergence Analysis

Notice that the problem of hidden matrix factorization, is non-convex. To ensure that local minimizer x^* for problem (1) is a KTT point, a constraint qualification is necessary so that it suits the Relaxed Constant Positive Linear Dependence (RCPLD) condition.

Here we consider the more general form

$$\begin{aligned} \min_x f(x) \\ \text{s.t. } c(x) = 0 \end{aligned} \quad (16)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a lower semi-continuous function and $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is continuously differentiable function.

Definition 1. For the above problem 16. Let \mathcal{F} be the feasible region, $x^* \in \mathcal{F}$, and $\{\nabla c_i(x^*) | i \in \mathcal{I}\}$ is a basis for $\text{span}\{\nabla c_i(x^*) | i = 1, \dots, m\}$ with $\mathcal{I} \subset \{1, \dots, m\}$. We say that RCPLD holds for the system $c(x) = 0$ at x^* if $\delta > 0$ such that $\{\nabla c_i(x) | i = 1, \dots, m\}$ has the same rank for each $x \in \mathcal{B}_\delta(x^*)$.

Note that when the RCPLD condition holds at a given feasible point, it is therefore true that it holds at every feasible point in neighborhood of the aforementioned point. Then, we present the convergence analysis under the RCPLD condition.

Theorem 2.3.1. For problem (1) let $x^k = \{P^k, Q^k, E^k, Y^k\}$ be a sequence generated by Algorithm 1 and let x^* be an accumulation point of $\{x^k\}$. Suppose that the function $f(x)$ is bounded; then, the following statements holds:

- (a) $c(x^k) \rightarrow \mathbf{0}$ as $k \rightarrow \infty$.
- (b) x^* is a feasible point of problem (1)
- (c) if RCPLD holds at x^* for the system $c(x) = 0$, then x^* is a KTT point of problem (1)

The proof is presented in the appendix.

3. Applications

In this section, we apply our algorithm to two models and propose a corresponding algorithm for each problem separately.

3.1. Low-rank representation

3.1.1. Basic model

Recently, a spectral subspace clustering method called Low-Rank Representation (LRR) has been proposed in papers [12] and [13]. LRR aims to represent the data matrix X as a linear combination of the basis in dictionary A , that $X = AZ$. Obviously, Z has various choices, and we always choose the lowest ranked one, which is also called the low-rank representation of matrix X on dictionary A . Based on a convex formulation, LRR is also robust to noise in the sampled data. The dictionary matrix is always chosen to be the data matrix X .

Low-Rank Representation (LRR) considers the following rank minimization problem

$$\begin{aligned} \min_{Z, E} \text{rank}(Z) + \mu \|E\|_1 \\ \text{s.t. } X = AZ + E \end{aligned} \quad (17)$$

where X is the given observation matrix corrupted by errors E , A is a dictionary that linearly spans the data space, and $\mu > 0$ is a parameter. The choice of $\|\cdot\|_1$ depends on the type of error. In this article, we use the $l_{2,1}$ norm to approximate the $l_{2,0}$ norm, which is used for dealing with sample-specific corruptions.

Regarding optimization, several algorithms [23], [11], and [12] are proposed to precisely solve the LRR problem. However, these algorithms all use the nuclear norm to replace the rank function in order to transform the original problem into a convex problem. Then they have to do singular value decomposition (SVD) in each iteration, which takes a long time when the dimension of matrix is large. Motivated by the strategy in [22], we rewrite the LRR model and use hidden matrix factorization instead of the nuclear norm to constrain the rank of the low-rank representation Z .

3.1.2. A hidden factor based reformulation of LRR

We assume an optimum point of the LRR problem from (17) is (Z^*, E^*) , $r = \text{rank}(Z^*)$, and the image space of $A \in \mathbb{R}^{m \times k}$ is full. It is well known that Z^* can be written into a matrix product form $Z^* = NQ$, where $N \in \mathbb{R}^{k \times r}$ and $Q \in \mathbb{R}^{r \times n}$. Therefore, Z^* can be revealed by solving the following problem

$$\begin{aligned} \min_{N, Q, E} \|E\|_{2,1} \\ \text{s.t. } X = ANQ + E \end{aligned} \quad (18)$$

Assuming the optimum point is (N^*, Q^*) , we have $Z^* = N^*Q^*$. It is difficult to solve this problem directly, so we can write $AN = P$ because the image space of A is full. Completing the aforementioned

procedure is equivalent to solving the following problem:

$$\begin{aligned} \min_{P,Q,E} & \|E\|_{2,1} \\ \text{s.t. } & X = PQ + E \end{aligned} \quad (19)$$

Assuming the optimum point is (P^*, Q^*) , Z can be revealed by $Z = A^+ P^* Q^*$ according to the assumptions that the image space of $A \in \mathbb{R}^{m \times k}$ is full and theorems in [12] and [13]. Hence, it is clear that $\text{rank}(Z) \leq \min(\text{rank}(P), \text{rank}(Q)) \leq r$. In other words, we use the dimensions of P and Q instead of the nuclear norm to replace the rank function.

3.1.3. Algorithm for LRR

The form of (19) is the same as the basic model (1). Therefore, [Algorithm 1](#) can be used to solve the problem.

When updating E in the LRR problem

$$E^{k+1} = \arg \min_E \|E\|_{2,1} + \frac{\beta_k}{2} \left\| P^{k+1} Q^{k+1} + E - X + \frac{Y^k}{\beta_k} \right\|_F^2$$

we have to solve a subproblem of the form $\alpha \|W\|_{2,1} + \frac{1}{2} \|W - Q\|_F^2$. It can be solved via the following lemma:

Lemma 3.1.1. If Q is a given matrix, and W^* is the optimum point of the following problem:

$$\min_W \alpha \|W\|_{2,1} + \frac{1}{2} \|W - Q\|_F^2$$

then the i^{th} column of W^* satisfies the problem below

$$[W^*]_{:,i} = \begin{cases} \frac{\|Q\|_{:,i} - \alpha}{\|Q\|_{:,i}} [Q]_{:,i} & \text{if } \|Q\|_{:,i} > \alpha \\ 0 & \text{otherwise} \end{cases}$$

where $[Q]_{:,i}$ refers to the i^{th} column of the matrix Q .

The algorithm still runs slowly because of the process of calculating the pseudo-inverse. We notice that only the product of PQ is important, and thus we can update P, Q in another way while ensuring the product remains unchanged. For convenience, in this part we use P_k, Q_k instead of P^k, Q^k .

Now, we consider the product $M_{k+1} := P_{k+1}Q_{k+1}$ (The method used here is proposed in [22]). By some properties of MoorePenrose pseudo-inverse, we have the following equations:

$$P_{k+1} = M_k Q_k^+ \equiv M_k Q_k^T (Q_k Q_k^T)^+ \quad (20)$$

$$Q_{k+1} = P_{k+1}^+ M_k \equiv (P_{k+1}^T P_{k+1})^+ (P_{k+1}^T M_k) \quad (21)$$

Using the preceding equation, we can get

$$P_{k+1} Q_{k+1} = P_{k+1} (P_{k+1}^T P_{k+1})^+ P_{k+1}^T M_k = C_{P_{k+1}} M_k \quad (22)$$

where $C_A = A(A^T A)^+ A^T = K K^T$ is an orthogonal projection onto the range space $\mathcal{R}(A)$ of A , and $K := \text{orth}(A)$ is an orthonormal basis for $\mathcal{R}(A)$. We can verify that $\mathcal{R}_{P_{k+1}} = \mathcal{R}_{M_k Q_{k+1}^T}$. In fact, if we let $Q_{k+1} = USV^T$ be the economy-form SVD of Q_{k+1} , then $P_{k+1} = M_k V S^+ U^T$ and $M_k Q_{k+1}^T = M_k V S U^T$ imply that $\mathcal{R}_{P_{k+1}} = \mathcal{R}_{M_k Q_{k+1}^T}$. Therefore, $C_{P_{k+1}} = C_{M_k Q_{k+1}^T}$ and $P_{k+1} Q_{k+1} = C_{M_k Q_{k+1}^T} M_k$.

Here, we provide another way to update P_{k+1}, Q_{k+1} . We only care about the product $P_{k+1} Q_{k+1}$, and different values of P_{k+1} and Q_{k+1} are essentially equivalent, as long as they give the same product $P_{k+1} Q_{k+1}$. The variant computes the orthogonal projection $C_{M_k Q_{k+1}^T} = V_k V_k^T$, where $V_k = \text{orth}(M_k Q_k^T)$. Hence, the product $P_{k+1} Q_{k+1}$ can be rewritten as $P_{k+1} Q_{k+1} = V_k V_k^T M_k$. The procedure updating P_{k+1} and Q_{k+1} is transformed into

$$P_{k+1} = V_k \quad (23)$$

$$Q_{k+1} = V_k^T M_k \quad (24)$$

The updating of other variables remains unchanged. In practical experiments and applications, we present a stop criteria, such as $\|E^{k+1} + P^{k+1} Q^{k+1} - X\|_F < \epsilon$. Then, the full algorithm is described in [Algorithm 2](#).

Algorithm 2

Input: index set data X , rank r

Initialize: $E^0 = \mathbf{0}$, $Y^0 = \mathbf{0}$, P^0, Q^0 s.t. $\text{rank}(P^0) = \text{rank}(Q^0) = r$. $\beta_0 > 0$, $\rho > 1$, $\eta \in (0, 1)$, $\tau \in (0, 1)$, $\{\epsilon_k\} \geq 0$ with $\lim_{k \rightarrow \infty} \epsilon_k = 0$. Set $k = 0$

While not converged **do**

$$\begin{aligned} 1. \quad M_k &= X - E^k - Y^k / \beta_k, & V_k &= \text{orth}(M_k (Q^k)^T), & P^{k+1} &= V_k, \\ Q^{k+1} &= V_k^T M_k \end{aligned}$$

$$2. \quad E^{k+1} = \arg \min_E \|E\|_{2,1} + \frac{\beta_k}{2} \|E + P^{k+1} Q^{k+1} - X + \frac{Y^k}{\beta_k}\|$$

$$3. \quad \text{Set } Y^{k+1} = Y^k + \beta_k (E^k + P^k Q^k - X).$$

4. if $k > 0$ and

$$\|E^{k+1} + P^{k+1} Q^{k+1} - X\| \leq \eta \|E^k + P^k Q^k - X\|,$$

then set $\beta_{k+1} = \beta_k$. Otherwise, set

$$\beta_{k+1} = \max\{\rho \beta_k, \|Y^{k+1}\|^{1+\tau}\}.$$

$$5. \quad \text{Set } k \leftarrow k + 1. \quad \text{Check the convergence condition: } \|E^{k+1} + P^{k+1} Q^{k+1} - X\|_F < \epsilon$$

end While

3.1.4. Time complexity

In [Algorithm 3](#), the most time-consuming steps in each iteration are updating V_k , E_{k+1} and conducting matrix multiplication. Here we let r be the fixed rank and $m \times n$ be the dimension of the matrix X . In particular, to update V_k we carry out the Gram-Schmidt process in each iteration when the time complexity is $O(r^2 m)$. For updating E_{k+1} , the time complexity depends on the norm type. When the norm is $l_{2,1}$ or l_1 , the time complexity for updating E_{k+1} is $O(mn)$. The matrix multiplication demands a time usage of $O(rmn)$ in each iteration. In summary, the time complexity for [Algorithm 2](#) is $O(rmn + r^2 m + mn)$. Because r is always much smaller than m and n , the time complexity of [Algorithm 2](#) is $O(rmn)$. There are other matrix multiplication algorithms, such as sub-cubic algorithms, that can reduce the multiplication complexity to $O(n^{\log_2 7})$ when multiplying two $n \times n$ matrices. Thus, the algorithm can still speed up when using efficient multiplication algorithms.

3.1.5. Tricks for finding optimum rank

In many practical problems, such as the matrix completion problem and the hidden factorization problem, convergence can speed up if we are aware of the optimum rank, r . In some cases we can estimate r by using some properties of the practical problem; however, in most cases, r is unknown. Therefore, it is a significant complication to find an efficient way to search for the optimum rank, r .

The simplest method is to test each possible rank and choose the best one as the final result, but clearly this method is inefficient. If we were to set $\text{rank} = r$, the optimum target function value would be $F(r) = \text{rank}(Z) + \mu \|E\|$. In experiments, we find that when the rank r used in our algorithm is slightly larger than the optimum rank r^* our result also performs well. Therefore, it is not necessary for us to test all of the possible ranks one by one.

We can first choose a proper interval d , then run the algorithm on the ranks $1, d+1, 2d+1, \dots, kd+1, \dots$, and stop it when the results begin to worsen. Thus, we could determine that the optimum rank is near the stopping point and then search through the options one by one to find the optimum rank. For example, if $F((k+1)d+1) > F(kd+1)$, the optimum rank is around $kd+1$. Then, we compute $F((k-1)d+1), F((k-1)d+2), \dots, F((k+1)d+1)$ one by one and find the best rank. It is of course important to choose a proper d to speed up our algorithm. In practice, the rank of result that we want is always relatively much smaller than the size of matrix. Suppose the matrix is $n \times n$, then we can choose d as σn where σ can be a small number for example 0.01, 0.02 etc.. Usually the smaller d is, the better the result will be. For instance, the LRR problem is not sensitive to the rank that we choose, so we can choose a relatively large one $0.025n$ to speed up the algorithm while keep the result accurate. However, for the matrix completion problem we choose d to be $0.01n$ because small change of rank can influence our result a lot. In order to get better result, we have to keep such d small enough. In practice, we can set d larger at first and then adjust it if it is not proper. When you think d is too big to get a satisfied result, you can decrease d to get better result with using more time. This trick remarkably improves the algorithm's efficiency, and the following theorem ensures its effectiveness.

Theorem 3.1.1. For problem (17), when r is fixed, using Algorithm 3 can obtain the optimum target function value $F(r) = \text{rank}(Z(r)) + \mu \|E(r)\|_l$. When r is a little larger than the optimum rank r^* , we can claim that $F(r) \geq F(r^*)$.

The proof is presented in the appendix. Due to this theorem, the algorithm is able to stop when r is larger than the optimum rank r^* .

3.2. Matrix completion

Matrix completion refers to the task of filling in the missing entries of a partially observed matrix. Many applications for this procedure are summarized in [4], such as collaborative filtering, system identification, and global positioning.

3.2.1. Basic model

In the matrix completion problem, we only have partial information from the whole matrix, and thus we want to recover the original matrix from what we already know. The problem can be written as

$$\begin{aligned} \min_{P, Q, E} & \|\mathcal{P}_\Omega(E)\|_l \\ \text{s.t. } & X = PQ + E \end{aligned} \quad (25)$$

where $P \in \mathbb{R}^{m \times r}$, $Q \in \mathbb{R}^{r \times n}$, $E \in \mathbb{R}^{m \times n}$, Ω is a index subset. \mathcal{P}_Ω is defined as

$$\mathcal{P}_\Omega(A)_{i,j} = \begin{cases} A_{i,j} & (i, j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

3.2.2. Algorithm for matrix completion

By using the strategy from the LRR problem, we obtain the algorithm to solve the problem.

Additionally, the previous method we used to search for the optimum rank can also be applied to the matrix completion problem.

3.2.3. Applications for recommender systems

Matrix Completion could be used in recommender systems, which can generate significant economic effects. As networks are widely applied to various parts of our life, electronic retailers offer us a huge selection of content and products. Matching consumers'

need with the most appropriate products is key to enhancing user satisfaction and loyalty. Therefore, more electronic retailers have become interested in recommender systems. Through a good recommender system, a retailer can provide consumers with personalized recommendations to suit the specific user.

Recommender systems are particularly essential for entertainment products, such as music, movies and so on. If consumers are willing to indicate their level of satisfaction with a particular movie or musical selection, such a system can analyze the data and make predictions based on what a user has previously liked. Without doubt, correct recommendations can encourage customers to use the software much more often.

Algorithm 3

Input: index set Ω , data $\mathcal{P}_\Omega(X)$, rank r

Initialize: $E^0 = \mathbf{0}$, $Y^0 = \mathbf{0}$, P^0, Q^0 s.t. $\text{rank}(P^0) = \text{rank}(Q^0) = r$. $\beta_0 > 0$, $\rho > 1$, $\eta \in (0, 1)$, $\tau \in (0, 1)$, $\{\epsilon_k\} \geq 0$ with $\lim_{k \rightarrow \infty} \epsilon_k = 0$. Set $k = 0$

While not converged **do**

$$\begin{aligned} 1. \quad M_k &= X - E^k - Y^k / \beta_k, & V_k &= \text{orth}(M_k(Q^k)^T), & P^{k+1} &= V_k, \\ Q^{k+1} &= V_k^T M_k \end{aligned}$$

$$2. \quad E_{k+1} = \arg \min_E \quad \|\mathcal{P}_\Omega(E)\|_l + \frac{\beta_k}{2} \|E + P_{k+1}Q_{k+1} - X + Y_k / \beta_k\|_F^2$$

$$3. \quad \text{Set } Y^{k+1} = Y^k + \beta_k(E^k + P^kQ^k - X).$$

4. If $k > 0$ and

$$\|E^{k+1} + P^{k+1}Q^{k+1} - X\| \leq \eta \|E^k + P^kQ^k - X\|, \quad (26)$$

then set $\beta_{k+1} = \beta_k$. Otherwise, set

$$\beta_{k+1} = \max\{\rho\beta_k, \|Y^{k+1}\|^{1+\tau}\}.$$

$$5. \quad \text{Set } k \leftarrow k+1. \quad \text{Check the convergence condition: } \|E^{k+1} + P^{k+1}Q^{k+1} - X\|_F < \epsilon$$

end While

In the Netflix Prize competition [2], matrix factorization models are proven to be superior to classic nearest-neighbor techniques for producing product recommendations. The basic model can be written as follows:

$$\begin{aligned} \min_{P, Q, E} & \|\mathcal{P}_\Omega(E)\|_F^2 + \mu_P \|P\|_l + \mu_Q \|Q\|_l \\ \text{s.t. } & X = PQ + E \end{aligned} \quad (27)$$

where $P \in \mathbb{R}^{m \times r}$, $Q \in \mathbb{R}^{r \times n}$, $E \in \mathbb{R}^{m \times n}$; x_{ij} in matrix X is the score that the i th customer awards to the j th item, and Ω is an index subset where the customer rates the item. We assume that there are r latent main factors that affect a customer's rating. Accordingly, p_{ij} in matrix P shows the level of importance of the j th factor to the i th customer, and q_{ij} in matrix Q is the score of the j th item's i th factor. Therefore, the score that the i th customer awards to the j th item can be expressed as

$$x_{ij} = \sum_{k=1}^r p_{ik} q_{kj}. \quad (28)$$

Model (27) aims to find the latent factorized matrices P and Q through the observed data. Then, we can use their product PQ to forecast customers' preferences. Additionally, our main purpose is to forecast the future, so that overfitting can be avoided. Thus, regularized term $\mu_P \|P\|_l + \mu_Q \|Q\|_l$ is added to the objective function. The choice of norms $\|\cdot\|_l$ and $\|\cdot\|_F$ is dependent on practical

problems. For instance, one item is supposed to have only a few factors. Therefore, each column of Q should be sparse and $\|\cdot\|_{l_0}$ can be the l_1 norm.

3.2.4. Algorithm for recommender systems

Similar to matrix completion, the target is to fill in the missing parts of the matrix, but there are bonuses for $\mu_P\|P\|_{l_2}$ and $\mu_Q\|Q\|_{l_1}$. Consequently, we can update E according to (26) in [Algorithm 3](#) and use the same method as in [Algorithm 1](#) to update P and Q .

Therefore, the update for E is

$$E_{k+1} = \arg \min_E \|\mathcal{P}_\Omega(E)\|_F^2 + \frac{\beta_k}{2} \|E + P_{k+1}Q_{k+1} - X + Y_k/\beta_k\|_F^2 \quad (29)$$

It is simple to solve the above subproblem because every element for E is separable.

The update for P, Q is transformed into the following form

$$\begin{aligned} P^{k+1} = \arg \min_P & \mu_P\|P\|_{l_2} + \frac{\beta_1^k \eta_P^k}{2} \|P - P^k + (P^k Q^k + E^k \\ & - X + Y_1^k/\beta_1^k)(Q^k)^T/\eta_P^k\|_F^2 \end{aligned} \quad (30)$$

$$\begin{aligned} Q^{k+1} = \arg \min_Q & \mu_Q\|Q\|_{l_1} + \frac{\beta_1^k \eta_Q^k}{2} \|Q - Q^k + (P^{k+1})^T(P^{k+1}Q^k \\ & + E^k - X + Y_1^k/\beta_1^k)/\eta_Q^k\|_F^2 \end{aligned} \quad (31)$$

There is a shrinkage operator to solve these two subproblems directly.

Algorithm 4

Input: index set Ω , data $\mathcal{P}_\Omega(X)$, rank r
Initialize: $E^0 = \mathbf{0}$, $Y^0 = \mathbf{0}$, P^0, Q^0 s.t. $\text{rank}(P^0) = \text{rank}(Q^0) = r$, $\beta_0 > 0$, $\rho > 1$, $\eta \in (0, 1)$, $\tau \in (0, 1)$, $\{\epsilon_k\} \geq 0$ with $\lim_{k \rightarrow \infty} \epsilon_k = 0$. Set $k = 0$

While not converged **do**

1. Update P^{k+1}, Q^{k+1} according to (30) and (31).
2. Update E^{k+1} according to (30).
3. Set $Y^{k+1} = Y^k + \beta_k(E^k + P^k Q^k - X)$.
4. If $k > 0$ and

$$\|E^{k+1} + P^{k+1}Q^{k+1} - X\| \leq \eta \|E^k + P^k Q^k - X\|, \quad (32)$$

then set $\beta_{k+1} = \beta_k$. Otherwise, set

$$\beta_{k+1} = \max\{\rho \beta_k, \|Y^{k+1}\|^{1+\tau}\}.$$

5. Set $k \leftarrow k + 1$. Check the convergence condition: $\|E^{k+1} + P^{k+1}Q^{k+1} - X\|_F < \epsilon$

end While

4. Numerical experiments

In this section, we test the efficiency of our algorithm, the Hidden Matrix Factorized Augmented Lagrangian Method (HMFALM), in the section and compare it with other algorithms. All of the algorithms are run and timed on a PC with 2.6 GHz Intel Core i5, and 8 GB of memory running OS X El Capitan and Matlab version 8.6.0.

Table 1
Comparison between HMFALM and LADM on synthetic data.

(s, p, d, r)	Method	Time (s)	Iterations	Accuracy (%)
(10, 20, 200, 5)	HMFALM	0.222605	246.333	98.17
	LADM	0.652839	39.	98.5
(15, 20, 300, 5)	HMFALM	0.542285	227.667	98.89
	LADM	1.58327	46.	99.33
(20, 25, 500, 5)	HMFALM	1.37254	187.667	99.80
	LADM	9.11552	57.	100
(30, 30, 900, 5)	HMFALM	5.18707	166.	99.93
	LADM	70.5994	69.	100
(35, 40, 1400, 5)	HMFALM	13.9157	154.	99.86
	LADM	552.614	157.	100
(40, 50, 2000, 5)	HMFALM	29.3892	127.333	99.90
	LADM	3808.43	615.	98.60

4.1. Experiments on low-rank representation

In this section, we test the efficiency of our algorithm, HMFALM, for solving the LRR problem and compare it with the LADM method in [11] and the IRLS method in [15]. For our algorithm, we set $\epsilon = 10^{-5}$, $\beta_1 = 10^{-5}$, $\beta_{\max} = 10^5$, $\rho = 2$, and the search gap $d = 0.025n$ where n is the size of original matrix. For the LADM, we use the same parameters as in [11].

4.1.1. Synthetic data

In this part, we use some synthetic data of different sizes to rank and evaluate the efficiency of our algorithm for solving the LRR problem. The method used to generate the data is similar to the method used in [13], [23] and [11]. The procedure for constructing synthetic data is detailed below.

The data is parameterized as (s, p, d, r) , which is created by the same procedure as in [11]. Firstly, we construct a base, U_1 , which is a $d \times r$ random orthogonal matrix. Then, another independent bases $\{U_i\}_{i=2}^s$ are generated by $U_{i+1} = TU_i$, where T is a random rotation. These s bases are used to construct s independent subspaces $\{S_i\}_{i=1}^s$. Then, p vectors are sampled from each subspace by $X_i = U_i Q_i$, where Q_i is an $r \times p$ i. i. d zero-mean unit variance Gaussian matrix $\mathcal{N}(0, 1)$. Then, some data vectors are chosen randomly for corruption. For instance, if the vector x is chosen, then we add a Gaussian noise with zero mean and variance depending on x (such as $0.1\|x\|_2$). Finally, we obtain the matrix $X = [X_1, X_2, \dots, X_s] \in R^{d \times sp}$ with rank sr . We set several values for μ in problem (17), such as $\mu = 0.2, 0.5, 1$ in the experiment.

The results are listed in [Table 1](#). From the results, we can see that the clustering accuracy of the two methods is almost same, which is near 100%, and our algorithm runs remarkably faster than the LADM on different values for μ and different sizes of data. The time cost in each iteration of our algorithm is also much less than the LADM. When μ becomes larger, the LADM always needs more iterations to converge, but our algorithm remains stable when μ changes.

4.1.2. Hopkins155 database

In this part, we test the efficiency of our algorithm on the Hopkins155 Database. This database consists of 156 sequences, each of which has a few data vectors drawn from two or three motions (each motion corresponds to a subspace). Therefore, we have 156 subspace clustering problems in total. These 156 sequences can be divided into a few groups according to the number of motions. The performance of our algorithm and other algorithms is listed in the following [Table 2](#). From the results, we can see that our algorithm achieves even higher accuracy, with a time cost of one one-hundredth of the other algorithms.

Table 2

Comparison among HMFALM, LADM, and IRLS on Hopkins155 Database.

Two motions			
Method	Time (s)	Iterations	Accuracy (%)
HMFALM	0.0946	162	97.72
LADM	22.6129	2505	96.25
IRLS	33.6640	189	97.15
Three motions			
Method	Time (s)	Iterations	Accuracy (%)
HMFALM	0.1237	173	96.16
LADM	33.6896	2733	90.08
IRLS	72.3639	182	95.90
All Motions			
Method	Time (s)	Iterations	Accuracy (%)
HMFALM	0.1013	164	97.24
LADM	25.1691	2557	94.39
IRLS	42.5947	188	96.77

Table 3

Comparison between HMFADM and SVT on clean synthetics data.

Size	Sparsity	Method	Time (s)	Iterations	Error
900	0.2	HMFALM	10.5092	359.	0.1953
	0.4	SVT	2638	1840	0.2266
	0.4	HMFALM	10.2916	309.	7.292×10^{-6}
	0.6	SVT	46.4511	168.	1.59124×10^{-4}
	0.6	HMFALM	12.3818	314.	2.835×10^{-9}
	0.8	SVT	43.0153	72.	1.334×10^{-4}
	0.8	HMFALM	13.4855	295.	1.373×10^{-9}
	0.8	SVT	15.2427	39.	1.13793×10^{-4}
1400	0.2	HMFALM	25.3181	361.	0.1644
	0.4	SVT	> 3600	1796	0.2216
	0.4	HMFALM	29.8295	346.	2.626×10^{-3}
	0.6	SVT	116.701	161.	1.611×10^{-4}
	0.6	HMFALM	34.613	352.	2.074×10^{-5}
	0.8	SVT	70.4354	70.	1.286×10^{-4}
	0.8	HMFALM	36.5016	336.	9.827×10^{-5}
	0.8	SVT	39.5721	39.	1.067×10^{-4}
2000	0.2	HMFALM	54.8070	334.	0.1844
	0.4	SVT	> 3600	1813	0.2185
	0.4	HMFALM	66.2042	349.	2.451×10^{-3}
	0.6	SVT	269.503	160.	1.559×10^{-4}
	0.6	HMFALM	77.8548	358.	1.660×10^{-6}
	0.8	SVT	152.795	69.	1.304×10^{-4}
	0.8	HMFALM	80.7487	337.	7.313×10^{-8}
	0.8	SVT	96.1347	39.	9.849×10^{-5}

4.2. Experiments on matrix completion

In this section, we test our algorithm in the tasks of matrix completion with clean data and noisy data. Finally, an experiment about movie recommendations has been tested. In this experiment we set the search gap $d = 0.01n$ where n is the size of original matrix.

4.2.1. Clean synthetic data

We use clean synthetic data to test the efficiency of **Algorithm 3**. Firstly, we sample two $n \times r$ factors X_L, X_R independently, each having i.i.d Gaussian entries. The original matrix is the product of these two matrices $X = X_L X_R^T$ with rank r . The observed entries are denoted as Ω which is sampled uniformly. The sparsity of data means the ratio $\frac{|\Omega|}{n^2}$. In experiments, we firstly generate such a matrix X as above and determine a sparsity σ . Then we randomly choose σn^2 entries in X as the observed set Ω . Finally we use the value on Ω to recover the original matrix X .

We denote the output of our algorithm as Z . Relative error is defined as $\|X - Z\|_F / \|X\|_F$. The results comparing with Singular Value Thresholding (SVT [3]) are listed in **Table 3**.

According to the comparison, when the sparsity of the matrices is high, such as 0.2, the SVT will take much more time than

Table 4

Comparison between HMFALM and SVT on corrupted synthetic data.

Size	Sparsity	Method	Time (s)	Iterations	Error
900	0.2	HMFALM	4.0067	130.	0.1782
	0.4	SVT	932	753	0.2296
	0.4	HMFALM	4.48665	135.	2.090×10^{-3}
	0.6	SVT	55.0842	116.	1.456×10^{-3}
	0.6	HMFALM	5.11051	137.	1.051×10^{-4}
	0.8	SVT	28.9732	52.	1.016×10^{-3}
	0.8	HMFALM	5.68116	138.	6.225×10^{-5}
	0.8	SVT	18.4593	29.	8.069×10^{-4}
1400	0.2	HMFALM	11.4067	132.	0.1652
	0.4	SVT	> 3600	823	0.2219
	0.4	HMFALM	11.7082	139.	1.618×10^{-4}
	0.6	SVT	160.966	117.	1.173×10^{-3}
	0.6	HMFALM	13.1661	137.	5.567×10^{-5}
	0.8	SVT	73.3628	53.	8.167×10^{-4}
	0.8	HMFALM	14.7731	138.	4.849×10^{-5}
	0.8	SVT	47.9073	30.	6.415×10^{-4}
2000	0.2	HMFALM	32.3517	166.	0.2439
	0.4	SVT	> 3600	882	0.2180
	0.4	HMFALM	34.2038	172.	5.123×10^{-3}
	0.6	SVT	166.272	119.	9.780×10^{-4}
	0.6	HMFALM	37.068	172.	1.636×10^{-4}
	0.8	SVT	113.733	54.	6.915×10^{-4}
	0.8	HMFALM	42.1163	175.	8.405×10^{-5}
	0.8	SVT	73.7739	31.	5.436×10^{-4}

Table 5

The performance of HMFALM on recommender system.

Method	Test number	Time (s)	RMSE
HMFALM	1	9.78	0.97
	2	9.57	0.95
	3	9.58	0.96
	4	9.84	0.96
	5	9.63	0.95
Regularized SVD	1	7.68	1.13
	2	8.70	1.14
	3	7.97	1.09
	4	7.17	1.10
	5	7.95	1.12
Neighborhood based method	1	107.84	1.12
	2	94.70	1.20
	3	96.77	1.19
	4	96.4219	1.19
	5	95.33	1.21

the HMFALM. Furthermore, when the dimensions of the matrices are also high, such large SVT time costs are unacceptable. However, the HMFALM performs well in both situations of high dimensions and high sparsity.

4.2.2. Corrupted synthetic data

Our algorithm can also be used to remove noise. We corrupt the clean data X as follows:

$$X_n = X + C \quad (33)$$

where C is a zero-mean Gaussian white noise with a standard deviation of 0.01. The observed set Ω is generated same with the clean case. Then we use the value on Ω to recover the original matrix X . **Table 4** lists these results. Similar to the results from the clean synthetic data, our algorithm can handle data with high dimensions and high sparsity, while SVT performs much worse in high sparsity cases.

4.2.3. Recommender systems

We test our algorithm on the dataset collected by GroupLens Research from the MovieLens web site. In the dataset, there are 10,000 ratings from 1,000 users on 1,700 movies. Their ratings range from 1 to 5. In the experiment, we have not changed the

constraint of P in (27) and use $\|\cdot\|_1$ to ensure the sparsity of Q in (27). We also use 80% of the data to calculate PQ and use the remaining 20% to test our results. The best recommendation algorithms [21] have an RMSE of about 0.9, which is better than our algorithm. However, those algorithms are much more complicated than ours since they consider many other situations and combine the results of different algorithms to obtain better accuracy. In the experiment, we compared our algorithm with two simple other algorithms used in Recommendation system, the regularized SVD method and Neighbor based method which are stated in the [1]. The regularized SVD is similar with ours, but its RMSE is always larger showing that it is necessary to add constraint to factorized matrix to improve the recommendation result. All test results are listed in the following table.

5. Conclusion

In this paper, we have proposed a novel algorithm, namely the Hidden Matrix Factorized Augmented Lagrangian Method (HM-FALM), for solving the low-rank matrix factorization problem. Our algorithm uses a factorized matrix to reflect the rank of a matrix and avoids SVD decomposition in the nuclear norm. Compared to other methods, our method truly solves the problem of ranking matrices and improves the speed. For large-scale and high-sparsity matrix decomposition, our method continues to function normally and efficiently, while others require immense amounts of time or even fail to perform.

Acknowledgement

We would like to thank the anonymous reviewers for their comments and suggestions. The work is supported by the NSF of China (No. 11626253), and the Fundamental Research Funds for the Central Universities (WK 0010460003).

Appendix A

A.1. A Proof of Theorem 2.3.1

Proof. The proof is inspired by [6].

(i) We consider two separate cases to prove statement (a) in Theorem (2.3.1).

Case 1: β_k is bounded. Then, we can see that $\{\beta_k\}$ is updated by (15) only for finite times, so that $\exists k_0 \in \mathbb{N}, \forall k \geq k_0$, that is

$$\|c(x^k)\| = \|E^{k+1} + P^{k+1}Q^{k+1} - X\| \leq \eta \|E^k + P^kQ^k - X\|$$

It implies that

$$\lim_{k \rightarrow \infty} \|c(x^k)\| = 0 \quad (34)$$

Then, statement (a) holds.

Case 2: β_k is unbounded. Then, we can see that $\{\beta_k\}$ is updated by (15) only for infinite times. Set $\{\beta_{j_1}, \beta_{j_2}, \dots\} \subset \{\beta_k\}$ as the elements that are updated by (15) and $J = \{j_1, j_2, \dots\}$. Then, $\{\beta_{j_l}\} \rightarrow \infty$ as $l \rightarrow \infty$ and

$$\beta_i = \beta_{j_l}, \quad j_l \leq i < j_{l+1}, \quad l \geq 1, \quad (35)$$

$$\beta_{j_l} = \max\{\rho\beta_{j_{l-1}}, \|Y^{j_l}\|^{1+\tau}\}, \quad l \geq 1. \quad (36)$$

We let $\underline{j}(k) = \max\{j \in J | k \geq j\}$ for every $k \geq j_1$.

If we claim $\forall k \geq j_1$,

$$\frac{\|Y^k\|}{\beta_k} \leq \frac{\|Y^{\underline{j}(k)}\|}{\beta_{\underline{j}(k)}} + \frac{1}{1-\eta} \|c(x^{\underline{j}(k)+i})\|. \quad (37)$$

Hence, when $k = \underline{j}(k)$, then (37) holds. In that case, we suppose $k > \underline{j}(k)$, and find that $\beta_{\underline{j}(k)+i} = \beta(\underline{j}(k))$ for $0 < i \leq k - \underline{j}(k)$. Using this and (13), we can get:

$$\begin{aligned} \frac{\|Y^{\underline{j}(k)+i}\|}{\beta_{\underline{j}(k)+i}} &= \frac{\|Y^{\underline{j}(k)+i}\|}{\beta_{\underline{j}(k)+i-1}} = \left\| \frac{Y^{\underline{j}(k)+i-1}}{\beta_{\underline{j}(k)+i-1}} + c(x^{\underline{j}(k)+i-1}) \right\| \\ &\leq \frac{\|Y^{\underline{j}(k)+i-1}\|}{\beta_{\underline{j}(k)+i-1}} + \|c(x^{\underline{j}(k)+i-1})\|, \quad 0 < i \leq k - \underline{j}(k). \end{aligned}$$

By summing up the above inequalities for $i = 1, \dots, k - \underline{j}_1$, we get:

$$\frac{\|Y^k\|}{\beta_k} \leq \frac{\|Y^{\underline{j}(k)}\|}{\beta_{\underline{j}(k)}} + \|c(x^{\underline{j}(k)+i})\| + \sum_{i=1}^{k-\underline{j}(k)} \|c(x^{\underline{j}(k)+i-1})\|. \quad (38)$$

This is because of (13), which implies

$$\|c(x^{\underline{j}(k)+i})\| \leq \eta^i \|c(x^{\underline{j}(k)})\|, \quad 0 < i \leq k - \underline{j}(k). \quad (39)$$

If we combine (38) and (39),

$$\begin{aligned} \frac{\|Y^k\|}{\beta_k} &\leq \frac{\|Y^{\underline{j}(k)}\|}{\beta_{\underline{j}(k)}} + \|c(x^{\underline{j}(k)+i})\| + \left(\sum_{i=0}^{k-\underline{j}(k)-1} \eta^i \right) \|c(x^{\underline{j}(k)})\| \\ &\leq \frac{\|Y^{\underline{j}(k)}\|}{\beta_{\underline{j}(k)}} + \|c(x^{\underline{j}(k)+i})\| + \frac{1}{1-\eta} \|c(x^{\underline{j}(k)})\|. \end{aligned}$$

The above inequality derivation takes as a fact that $\sum_{l=0}^{\infty} \eta^l \leq \frac{1}{1-\eta}$ when $\eta \in (0, 1)$. Then, we arrive at claim (37).

Next, we consider (15), which implies that $\|Y^{\underline{j}(k)}\|^{1+\tau} \leq \beta_{\underline{j}(k)}$ and $\beta_k \rightarrow \infty$ as $k \rightarrow \infty$, as well as the following:

$$\frac{\|Y^{\underline{j}(k)}\|}{\beta_{\underline{j}(k)}} \leq (\beta_{\underline{j}(k)})^{-\frac{\tau}{1+\tau}} \rightarrow 0 \text{ when } k \geq j_1. \quad (40)$$

Further, by (12) and (4), we have

$$\|c(x^{\underline{j}(k)}) + \frac{Y^{\underline{j}(k)}}{\beta}\|_F^2 \leq \frac{2}{\beta_{\underline{j}(k)}} [T - f(x^{\underline{j}(k)})] + \frac{\|Y^{\underline{j}(k)}\|^2}{\beta_{\underline{j}(k)}^2}. \quad (41)$$

Using this, (40), the boundedness of $\{f(x^k)\}$, and $\beta_k \rightarrow \infty$, we can see that

$$\lim_{k \rightarrow \infty} \|c(x^{\underline{j}(k)})\| = 0 \quad (42)$$

which implies statement (a).

(ii) The continuity of $c(x)$ and statement (a) directly lead to statement (b) in Theorem (2.3.1).

(iii) Finally, we statement (c) in Theorem (2.3.1). x^* is an accumulation point of $\{x^k\}$, which means that there is a subsequence \mathcal{K} such that $\{x^k\}_{\mathcal{K}} \rightarrow x^*$. According to (12), it is true that \widehat{Y}^k such that:

$$\|\nabla f(x^k) + \sum_{i=1}^m \widehat{Y}_i^k c_i(x^k)\| \leq \epsilon_k \quad (43)$$

for all k .

We adjust \mathcal{I} so that $\{\nabla c_i(x^*) | i \in \mathcal{I}\}$ is a basis for $\text{span}\{\nabla c_i(x^*) | i = 1, \dots, m\}$. Then, $\{\nabla c_i(x^k) | i \in \mathcal{I}\}$ is linearly independent for any sufficiently large $k \in \mathcal{K}$ because of $\{x^k\}_{\mathcal{K}} \rightarrow x^*$. We add the condition, RCPLD at x^* , so that $\{\nabla c_i(x^k) | i \in \mathcal{I}\}$ is a basis for $\text{span}\{\nabla c_i(x^k) | i = 1, \dots, m\}$ for any sufficiently large $k \in \mathcal{K}$. Furthermore, it is true that $\{\widehat{Y}_i^k | i \in \mathcal{I}\}$ such that

$$\|\nabla f(x^k) + \sum_{i \in \mathcal{I}} \widehat{Y}_i^k c_i(x^k)\| \leq \epsilon_k. \quad (44)$$

If $\mathcal{I} = \emptyset$, in view of (44) and $\{\epsilon_k\}_{\mathcal{K}} \rightarrow 0$, we have $\nabla f(x^*) = 0$, which implies that x^* is a KKT point of problem (1).

If $\mathcal{I} \neq \emptyset$, let

$$t_k := \max\{|\bar{Y}_i^k|, i \in \mathcal{I}\}.$$

Here, we claim that $\{t_k\}_{\mathcal{K}}$ is bounded, but suppose on the contrary that $\{t_k\}_{\mathcal{K}}$ is unbounded. In view of an accumulation point of $\frac{\bar{Y}_i^k}{t_k}$, so that as $\mathcal{K} \supset \mathbb{N} \ni k \rightarrow \infty$, $t_k \rightarrow \infty$ and

$$\frac{\bar{Y}_i^k}{t_k} \rightarrow \bar{Y}_i^*, i \in \mathcal{I}.$$

If we divide (44) by t_k and let $\mathcal{K} \ni k \rightarrow \infty$, then

$$\left\| \sum_{i \in \mathcal{I}} \hat{Y}_i^* c_i(x^k) \right\| = 0.$$

This combined with the fact $\max\{|\bar{Y}_i^*|, i \in \mathcal{I}\} = 1$ and $RCPLD$, the fact means that for any x sufficiently close to x^* , $\{\nabla c_i(x)|i \in \mathcal{I}\}$ is linearly dependent, which contradicts $\{\nabla c_i(x^k)|i \in \mathcal{I}\}$ as a basis.

Thus, $\{t_k\}_{\mathcal{K}}$ is bounded and $\{|\bar{Y}_i^k|\}_{\mathcal{K}}$ is also bounded.

Then, we can suppose that as $\mathcal{K} \supset \mathbb{N} \ni k \rightarrow \infty$, $\bar{Y}_i^k \rightarrow \bar{Y}_i^*$, wherein \bar{Y}_i^* is an accumulation point of $\{\bar{Y}_i^k\}$. By taking limits on both sides of (44), we have

$$\|\nabla f(x^*) + \sum_{i \in \mathcal{I}} \hat{Y}_i^* c_i(x^*)\| = 0.$$

which implies that x^* is a KKT point of problem (1). Then, the proof is complete. \square

A2. A Proof of Theorem 3.1.1

Proof. In the ideal case that $\text{rank}(X) = \text{rank}(Z_{r^*}) = r^*$, we can find a matrix decomposition $X = AZ_{r^*}$ (such as SVD decomposition), which implies $E_{r^*} = X - AZ_{r^*} = 0$ and $F(r^*) = r^*$. When r is larger than the optimum rank r^* , because of the larger free dimensions, we can also easily find a matrix decomposition $X = AZ_r$ with $\text{rank}(Z_r) = r$, $E_r = X - AZ_r = 0$. Then, $F(r) = \text{rank}(Z_r) + \mu \|E_r\|_1 \geq \text{rank}(Z_{r^*}) + \mu \|E_{r^*}\|_1 = F(r^*)$ and the proof is complete. \square

References

- [1] X. Amatriain, J. Basilio, Netflix recommendations: beyond the 5 stars (part 1), *Netflix Tech Blog* 6 (2012).
- [2] J. Bennett, C. Elkan, B. Liu, P. Smyth, D. Tikk, Kdd cup and workshop 2007, ACM SIGKDD Explor. Newsl. 9 (2) (2007) 51–52.
- [3] J.-F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, *SIAM J. Optimiz.* 20 (4) (2010) 1956–1982.
- [4] E.J. Candès, Y. Plan, Matrix completion with noise, *Proc. IEEE* 98 (6) (2010) 925–936.
- [5] E.J. Candès, B. Recht, Exact matrix completion via convex optimization, *Found. Comput. Math.* 9 (6) (2009) 717–772.
- [6] X. Chen, L. Guo, Z. Lu, J.J. Ye, An augmented Lagrangian method for non-Lipschitz nonconvex programming, *SIAM J. Numer. Anal.* 55 (1) (2017) 168–193.
- [7] M. Fazel, Matrix rank minimization with applications (Ph.D. thesis), Stanford University, 2002.
- [8] X. Lan, A.J. Ma, P.C. Yuen, Multi-cue visual tracking using robust feature-level fusion based on joint sparse representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1194–1201.
- [9] X. Lan, A.J. Ma, P.C. Yuen, R. Chellappa, Joint sparse representation and robust feature-level fusion for multi-cue visual tracking, *IEEE Trans. Image Process.* 24 (12) (2015) 5826–5841.
- [10] X. Lan, S. Zhang, P.C. Yuen, Robust joint discriminative feature learning for visual tracking, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2016, pp. 3403–3410.

- [11] Z. Lin, R. Liu, Z. Su, Linearized alternating direction method with adaptive penalty for low-rank representation, in: Advances in Neural Information Processing Systems, 2011, pp. 612–620.
- [12] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, *IEEE Trans. Pattern Anal. Mach. Intell.* 35 (1) (2013) 171–184.
- [13] G. Liu, Z. Lin, Y. Yu, Robust subspace segmentation by low-rank representation, in: Proceedings of International Conference on Machine Learning, 2010.
- [14] X. Liu, G. Zhao, J. Yao, C. Qi, Background subtraction based on low-rank and structured sparse decomposition, *IEEE Trans. Image Process.* 24 (8) (2015) 2502–2514.
- [15] C. Lu, Z. Lin, S. Yan, Smoothed low rank and sparse matrix recovery by iteratively reweighted least squares minimization, *IEEE Trans. Image Process.* 24 (2) (2015) 646–654.
- [16] L. Lu, R. Vidal, Combined central and subspace clustering for computer vision applications, in: Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 593–600.
- [17] Z. Lu, Y. Zhang, An augmented lagrangian approach for sparse principal component analysis, *Math. Program.* 135 (1–2) (2012) 149–193.
- [18] I. Markovsky, Low Rank Approximation: Algorithms, Implementation, Applications, Springer Science & Business Media, 2011.
- [19] H. Nguyen, W. Yang, F. Shen, C. Sun, Kernel low-rank representation for face recognition, *Neurocomputing* 155 (2015) 32–42.
- [20] W. Siming, L. Zhouchen, Analysis and improvement of low rank representation for subspace segmentation, arXiv preprint arXiv:1107.1561(2011).
- [21] G. Takács, I. Pilászy, B. Németh, D. Tikk, Major components of the gravity recommendation system, *ACM SIGKDD Explor. Newsl.* 9 (2) (2007) 80–83.
- [22] Z. Wen, W. Yin, Y. Zhang, Solving a Low-Rank Factorization Model for Matrix Completion by a Nonlinear Successive Over-Relaxation Algorithm, Springer and Mathematical Optimization Society, 2012.
- [23] S. Xiao, W. Li, D. Xu, D. Tao, FALRR: a fast low rank representation solver, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
- [24] H. Zhang, Z. Lin, C. Zhang, J. Gao, Robust latent low rank representation for subspace clustering, *Neurocomputing* 145 (2014) 369–373.
- [25] N. Zhang, J. Yang, Low-rank representation based discriminative projection for robust feature extraction, *Neurocomputing* 111 (2013) 13–20.



Baiyu Chen Master in Applied Mathematics at University of Science and Technology of China, 09/2014 – 04/2017 B.S in Applied Mathematics at University of Science and Technology of China, 09/2010 – 07/2014.



Zi Yang Ph.D. in Mathematics (In Progress) at UC San Diego, 09/2016 Present B.S in Applied Mathematics at University of Science and Technology of China, 09/2012–07/2016.



Zhouwang Yang is a Professor in the School of Mathematical Sciences at University of Science and Technology of China (USTC). He received his Bachelor degree, Master degree and Ph.D. degree in Mathematics from USTC in 1997, 2000 and 2005, respectively. He has been working on optimization methods and their applications in sparse recovery. He also has a particular interest in the integration of optimization, machine learning and statistics for solving big data problems.