SIAM J. SCI. COMPUT.
Vol. 37, No. 1, pp. A488–A514

ORTHOGONAL RANK-ONE MATRIX PURSUIT FOR LOW RANK MATRIX COMPLETION[*]

ZHENG WANG[†], MING-JUN LAI[‡], ZHAOSONG LU[§], WEI FAN[¶], HASAN DAVULCU[∥], AND JIEPING YE[#]

Abstract. In this paper, we propose an efficient and scalable low rank matrix completion algorithm. The key idea is to extend the orthogonal matching pursuit method from the vector case to the matrix case. We further propose an economic version of our algorithm by introducing a novel weight updating rule to reduce the time and storage complexity. Both versions are computationally inexpensive for each matrix pursuit iteration and find satisfactory results in a few iterations. Another advantage of our proposed algorithm is that it has only one tunable parameter, which is the rank. It is easy to understand and to use by the user. This becomes especially important in large-scale learning problems. In addition, we rigorously show that both versions achieve a linear convergence rate, which is significantly better than the previous known results. We also empirically compare the proposed algorithms with several state-of-the-art matrix completion algorithms on many real-world datasets, including the large-scale recommendation dataset Netflix as well as the MovieLens datasets. Numerical results show that our proposed algorithm is more efficient than competing algorithms while achieving similar or better prediction performance.

Key words. low rank, singular value decomposition, rank minimization, matrix completion, matching pursuit

AMS subject classifications. 15A83, 68W40, 90C06

DOI. 10.1137/130934271

1. Introduction.

Recently, low rank matrix learning has attracted significant attention in machine learning and data mining due to its wide range of applications, such as collaborative filtering, dimensionality reduction, compressed sensing, multiclass learning, and multitask learning. See [1, 2, 3, 7, 9, 23, 34, 40, 37] and the references therein. In this paper, we consider the general form of low rank matrix completion: given a partially observed real-valued matrix $\mathbf{Y} \in \Re^{n\times m}$, the low rank matrix completion problem is to find a matrix $\mathbf{X} \in \Re^{n\times m}$ with minimum rank that best approximates the matrix $\mathbf{Y}$ on the observed elements. The mathematical formulation is given by

$$\begin{array}{ll} \min\limits_{\mathbf{X}\in\Re^{n\times m}} & \operatorname{rank}(\mathbf{X}) \\ \text{s.t.} & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{Y}), \end{array} \tag{1.1}$$

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 26, 2013; accepted for publication (in revised form) November 18, 2014; published electronically February 19, 2015. This research was partially supported by NSF (IIS-0953662, CCF-1025177, IIS-1421057), NIH (LM010730), China 973 Fundamental R&D Program (2014CB340304), NSERC Discovery Grant and a collaboration grant from the Simons Foundation.

http://www.siam.org/journals/sisc/37-1/93427.html

†Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109 (zhengwang@umich.edu).

‡Department of Mathematics, University of Georgia, Athens, GA 30602 (mjlai@math.uga.edu).

§Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 156, Canada (zhaosong@sfu.ca).

¶Huawei Noah's Ark Lab, Units 520–530 Core Building 2, Hong Kong Science Park, Hong Kong (wei.fan@gmail.com).

∥School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 (hasandavulcu@asu.edu).

#Department of Computational Medicine and Bioinformatics, and Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 (jpye@umich.edu).

where Ω is the set of all index pairs (i, j) of observed entries, and P_Ω is the orthogonal projector onto the span of matrices vanishing outside of Ω.

1.1. Related works. As it is intractable to minimize the matrix rank exactly in the general case, many approximate solutions have been proposed to attack the problem (1.1) (cf., e.g., [7, 24, 28]). A widely used convex relaxation of matrix rank is the trace norm or nuclear norm [7]. The matrix trace norm is defined by the Schatten p-norm with $p = 1$. For matrix $\mathbf{X}$ with rank r, its Schatten p-norm is defined by $(\sum_{i=1}^r \sigma_i^p)^{1/p}$, where $\{\sigma_i\}$ are the singular values of $\mathbf{X}$ and without loss of generality we assume they are sorted in descending order. Thus, the trace norm of $\mathbf{X}$ is the ℓ_1 norm of the matrix spectrum as $||\mathbf{X}||_* = \sum_{i=1}^r |\sigma_i|$. Then the convex relaxation for problem (1.1) is given by

$$\begin{array}{cl} \min\limits_{\mathbf{X}\in\mathcal{R}^{n\times m}} & ||\mathbf{X}||_* \\ \text{s.t.} & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{Y}). \end{array} \tag{1.2}$$

Cai, Candès, and Shen [6] propose an algorithm to solve (1.2) based on soft singular value thresholding (SVT). Keshavan and Oh [21] and Jain, Meka, and Dhillon [18] develop more efficient algorithms by using the top-k singular pairs.

Many other algorithms have been developed to solve the trace norm penalized problem:

$$\min_{\mathbf{X}\in\mathcal{R}^{n\times m}} ||P_\Omega(\mathbf{X}) - P_\Omega(\mathbf{Y})||_F^2 + \lambda||\mathbf{X}||_*. \tag{1.3}$$

Ji and Ye [20], Liu, Sun, and Toh [27], and Toh and Yun [44] independently propose to employ the proximal gradient algorithm to improve the algorithm of [6] by significantly reducing the number of iterations. They obtain an ϵ-accurate solution in $O(1/\sqrt{\epsilon})$ steps. More efficient soft singular vector thresholding algorithms are proposed in [29, 30] by investigating the factorization property of the estimated matrix. Each step of the algorithms requires the computation of a partial singular value decomposition (SVD) for a dense matrix. In addition, several methods approximate the trace norm using its variational characterizations [32, 40, 46, 37] and proceed by alternating optimization. However, these methods lack global convergence guarantees.

Solving these low rank or trace norm problems is computationally expensive for large matrices, as it involves computing SVD. Most of the methods above involve the computation of SVD or truncated SVD iteratively, which is not scalable to large-scale problems. How to solve these problems efficiently and accurately for large-scale problems has attracted much attention in recent years.

Recently, the coordinate gradient descent method has been demonstrated to be efficient in solving sparse learning problems in the vector case [11, 39, 47, 48]. The key idea is to solve a very simple one-dimensional problem (for one coordinate) in each iteration. One natural question is whether and how such a method can be applied to solve the matrix completion problem. Some progress has been made recently in this direction. Dudík, Harchaoui, and Malick [9] propose a coordinate gradient descent solution for the trace norm penalized problem. They recast the nonsmooth objective in problem (1.3) as a smooth one in an infinite dimensional rank-one matrix space, then apply the coordinate gradient algorithm on the collection of rank-one matrices. Zhang, Yu, and Schuurmann [49] further improve the efficiency using the boosting method, and the improved algorithm guarantees an ϵ-accuracy within $O(1/\epsilon)$ iterations. Although these algorithms need slightly more iterations than the proximal methods, they are more scalable as they only need to compute the top singular vector pair in each iteration. Note that the top singular vector pair can be computed efficiently by the power method or Lanczos iterations [13]. Jaggi and Sulovský [17] propose an algorithm which achieves the same iteration complexity as the algorithm in [49] by directly applying Hazan's algorithm [15]. Tewari, Ravikumar, and Dhillon [42] solve a more general problem based on a greedy algorithm. Shalev-Shwartz, Gonen, and Shamir [38] further reduce the number of iterations based on heuristics without theoretical guarantees.

Most methods based on the top singular vector pair include two main steps in each iteration. The first step involves computing the top singular vector pair, and the second step refines the weights of the rank-one matrices formed by all top singular vector pairs obtained up to the current iteration. The main differences among these algorithms lie in how they refine the weights. Jaggi's algorithm (JS) [17] directly applies Hazan's algorithm [15], which relies on the Frank–Wolfe algorithm [10]. It updates the weights with a small step size and does not consider further refinement. It does not choose the optimal weights in each step, which leads to a slow convergence rate. Similar to JS, Tewari, Ravikumar, and Dhillon [42] use a small update step size for a general structure constrained problem. The greedy efficient component optimization (GECO) [38] optimizes the weights by solving another time-consuming optimization problem. It involves a smaller number of iterations than the JS algorithm. However, the sophisticated weight refinement leads to a higher total computational cost. The lifted coordinate gradient descent algorithm [9] updates the weights with a constant step size in each iteration and conducts a LASSO-type algorithm [43] to fully correct the weights. The weights for the basis update are difficult to tune as a large value leads to divergence and a small value makes the algorithm slow [49]. The matrix norm boosting approach (Boost) [49] learns the update weights and designs a local refinement step by a nonconvex optimization problem which is solved by alternating optimization. It has a sublinear convergence rate.

We summarize their common drawbacks as follows:

- Some weight refinement steps are inefficient, resulting in a slow convergence rate. The current best convergence rate is $O(1/\epsilon)$. Some refinement steps themselves contain computationally expensive iterations [9, 49], which do not scale to large-scale data.
- They have heuristic-based tunable parameters which are not easy to use. However, these parameters severely affect their convergence speed and the approximation result. In some algorithms, an improper parameter even makes the algorithm diverge [6, 9].

In this paper, we present a simple and efficient algorithm to solve the low rank matrix completion problem. The key idea is to extend the orthogonal matching pursuit (OMP) procedure [35] from the vector case to the matrix case. In each iteration, a rank-one basis matrix is generated by the left and right top singular vectors of the current approximation residual. In the standard version of the proposed algorithm, we fully update the weights for all rank-one matrices in the current basis set at the end of each iteration; this is achieved by performing an orthogonal projection of the observation matrix onto the spanning subspace of those rank-one matrices. The most time-consuming step of the proposed algorithm is to calculate the top singular vector pair of a sparse matrix, which involves $O(|\Omega|)$ operations in each iteration. An appealing feature of the proposed algorithm is that it has a linear convergence rate. This is different from traditional OMP or weak orthogonal greedy algorithms, whose convergence rate for sparse vector recovery is sublinear, as shown in [26]. See also

[8], [41], [45] for an extensive study on various greedy algorithms. With this rate of convergence, we only need $O(\log(1/\epsilon))$ iterations for achieving an ϵ-accuracy solution.

One drawback of the standard algorithm is that it needs to store all rank-one matrices in the current basis set for full weight updating, which contains $r|\Omega|$ elements in the rth iteration. This makes the storage complexity of the algorithm dependent on the number of iterations, which restricts the approximation rank especially for large-scale matrices. To tackle this problem, we propose an economic weight updating rule for this algorithm. In this economic version of the proposed algorithm, we only track two matrices in each iteration. One is the current estimated matrix and the other is the pursued rank-one matrix. When restricted to the observations in Ω, each has $|\Omega|$ nonzero elements. Thus the storage requirement, i.e., $2|\Omega|$, remains the same in different iterations, which is the same as the greedy algorithms [17, 42]. Interestingly, we show that using this economic updating rule we still retain the linear convergence rate. Besides the convergence property, we also analyze the recovery guarantee of our proposed algorithm. Specifically, we extend our proposed algorithm to a more general matrix sensing problem and show the recovery guarantee of the proposed algorithm under the rank-restricted isometry property [25]. We verify the efficiency of our algorithm empirically on large-scale matrix completion problems, such as MovieLens [31] and Netflix [4, 5].

The main contributions of our paper are as follows:

- We propose a computationally efficient and scalable algorithm for matrix completion, which extends OMP from the vector case to the matrix case.
- We theoretically prove the linear convergence rate of our algorithm. As a result, we only need $O(\log(1/\epsilon))$ iterations to obtain an ϵ-accuracy solution, and in each iteration we only need to compute the top singular vector pair, which can be computed efficiently.
- We further reduce the storage complexity of our algorithm based on an economic weight updating rule while retaining the linear convergence rate. This version of our algorithm has a constant storage complexity which is independent of the approximation rank and is more practical for large-scale matrices.
- We extend our proposed algorithm to a more general matrix sensing problem and show the recovery guarantee of the proposed algorithm under the rank-restricted isometry property.
- Both versions of our algorithm have only one free parameter, i.e., the rank of the estimated matrix. The proposed algorithm is guaranteed to converge, i.e., no risk of divergence.

1.2. Notation and organization. Let $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_m) \in \Re^{n\times m}$ be an $n \times m$ real matrix, and let $\Omega \subset \{1, \dots, n\} \times \{1, \dots, m\}$ denote the indices of the observed entries of $\mathbf{Y}$. P_Ω is the projection operator onto the space spanned by the matrices vanishing outside of Ω so that the (i, j)th component of $P_\Omega(\mathbf{Y})$ equals to $\mathbf{Y}_{i,j}$ for $(i, j) \in \Omega$ and zero otherwise. The Frobenius norm of $\mathbf{Y}$ is defined as $||\mathbf{Y}||_F = \sqrt{\sum_{i,j} \mathbf{Y}_{i,j}^2}$. Let $vec(\mathbf{Y}) = (\mathbf{y}_1^T, \dots, \mathbf{y}_m^T)^T$ denote a vector reshaped from matrix $\mathbf{Y}$ by concatenating all its column vectors. Let $\dot{\mathbf{y}} = vec_\Omega(\mathbf{Y}) = \{(y_{\boldsymbol{\omega}_1}, \dots, y_{\boldsymbol{\omega}_{|\Omega|}})^T \quad \forall\ \boldsymbol{\omega}_i \in \Omega\}$ denote a vector generated by concatenating all observed elements of $\mathbf{Y}$ indexed by Ω. The Frobenius inner product of two matrices $\mathbf{X}$ and $\mathbf{Y}$ is defined as $\langle \mathbf{X}, \mathbf{Y}\rangle = trace(\mathbf{X}^T\mathbf{Y})$, which also equals the componentwise inner product of the corresponding vectors as $\langle vec(\mathbf{X}), vec(\mathbf{Y})\rangle$. Given a matrix $\mathbf{A} \in \Re^{n\times m}$, we denote $P_\Omega(\mathbf{A})$ by $\mathbf{A}_\Omega$. For any two matrices $\mathbf{A}, \mathbf{B} \in \Re^{n\times m}$, we define $\langle \mathbf{A}, \mathbf{B}\rangle_\Omega = \langle \mathbf{A}_\Omega, \mathbf{B}_\Omega\rangle$ and

$\|\mathbf{A}\|_\Omega = \sqrt{\langle \mathbf{A}, \mathbf{A}\rangle_\Omega}$. Without further declaration, the matrix norm refers to the Frobenius norm, which is also written as $\|\mathbf{A}\| = \sqrt{\langle \mathbf{A}, \mathbf{A}\rangle}$.

The rest of the paper is organized as follows. We present the standard version of our algorithm in section 2. Section 3 analyzes the convergence rate of the standard version of our algorithm; we further propose an economic version of our algorithm and prove its linear convergence rate in Section 4. Section 5 extends the proposed algorithm to a more general matrix sensing case and presents its guarantee of finding the optimal solution under the rank-restricted isometry property condition. In section 6 we analyze the stability of both versions of our algorithm; empirical evaluations are presented in section 7 to verify the efficiency and effectiveness of our algorithm. We finally conclude our paper in section 8.

2. Orthogonal rank-one matrix pursuit.

It is well-known that any matrix $\mathbf{X} \in \Re^{n\times m}$ can be written as a linear combination of rank-one matrices, that is,

$$\mathbf{X} = \mathbf{M}(\boldsymbol{\theta}) = \sum_{i\in\mathcal{I}} \theta_i \mathbf{M}_i, \tag{2.1}$$

where $\{\mathbf{M}_i : i \in I\}$ is the set of all $n \times m$ rank-one matrices with unit Frobenius norm. Clearly, there are infinitely many choices of $\mathbf{M}_i$'s. Such a representation can be obtained via the standard SVD of $\mathbf{X}$.

The original low rank matrix approximation problem aims to minimize the zero-norm of θ subject to the constraint

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & ||\boldsymbol{\theta}||_0 \\ \text{s.t.} \quad & P_\Omega(\mathbf{M}(\boldsymbol{\theta})) = P_\Omega(\mathbf{Y}), \end{aligned} \tag{2.2}$$

where $||\boldsymbol{\theta}||_0$ denotes the number of nonzero elements of the vector $\boldsymbol{\theta}$.

If we reformulate the problem as

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & ||P_\Omega(\mathbf{M}(\boldsymbol{\theta})) - P_\Omega(\mathbf{Y})||_F^2 \\ \text{s.t.} \quad & ||\boldsymbol{\theta}||_0 \le r, \end{aligned} \tag{2.3}$$

we could solve it by an OMP type algorithm using rank-one matrices as the basis. In particular, we are to find a suitable subset of overcomplete rank-one matrix coordinates and learn the weight for each selected coordinate. This is achieved by executing two steps alternatively: one is to pursue the basis, and the other is to learn the weight of the basis.

Suppose that after the $(k-1)$th iteration, the rank-one basis matrices $\mathbf{M}_1, \dots,$ $\mathbf{M}_{k-1}$ and their current weight vector $\boldsymbol{\theta}^{k-1}$ are already computed. In the kth iteration, we are to pursue a new rank-one basis matrix $\mathbf{M}_k$ with unit Frobenius norm, which is mostly correlated with the current observed regression residual $\mathbf{R_k} = P_\Omega(\mathbf{Y}) - \mathbf{X}_{k-1}$, where

$$\mathbf{X}_{k-1} \ = \ (\mathbf{M}(\boldsymbol{\theta}^{k-1}))_\Omega \ = \ \sum_{i=1}^{k-1} \theta_i^{k-1} (\mathbf{M}_i)_\Omega.$$

Therefore, $\mathbf{M}_k$ can be chosen to be an optimal solution of the following problem:

$$\max_{\mathbf{M}} \left\{\langle \mathbf{M}, \mathbf{R}_k\rangle : \ \operatorname{rank}(\mathbf{M}) = 1, \ \|\mathbf{M}\|_F = 1\right\}. \tag{2.4}$$

Notice that each rank-one matrix $\mathbf{M}$ with unit Frobenius norm can be written as the product of two unit vectors, namely, $\mathbf{M} = \mathbf{u}\mathbf{v}^T$ for some $\mathbf{u} \in \Re^n$ and $\mathbf{v} \in \Re^m$ with $\|\mathbf{u}\| = \|\mathbf{v}\| = \mathbf{1}$. We then see that problem (2.4) can be equivalently reformulated as

$$\max_{\mathbf{u},\mathbf{v}} \{\mathbf{u}^T \mathbf{R}_k \mathbf{v} : \ \|\mathbf{u}\| = \|\mathbf{v}\| = \mathbf{1}\}. \tag{2.5}$$

Clearly, the optimal solution $(\mathbf{u}_*, \mathbf{v}_*)$ of problem (2.5) is a pair of top left and right singular vectors of $\mathbf{R}_k$. It can be efficiently computed by the power method [17, 9]. The new rank-one basis matrix $\mathbf{M}_k$ is then readily available by setting $\mathbf{M}_k = \mathbf{u}_* \mathbf{v}_*^T$.

After finding the new rank-one basis matrix $\mathbf{M}_k$, we update the weights $\boldsymbol{\theta}^k$ for all currently available basis matrices $\{\mathbf{M}_1, \dots, \mathbf{M}_k\}$ by solving the following least squares regression problem:

$$\min_{\boldsymbol{\theta} \in \Re^k} || \sum_{i=1}^{k} \theta_i \mathbf{M}_i - \mathbf{Y}||_\Omega^2. \tag{2.6}$$

By reshaping the matrices $(\mathbf{Y})_\Omega$ and $(\mathbf{M}_i)_\Omega$ into vectors $\dot{\mathbf{y}}$ and $\dot{\mathbf{m}}_i$, we can easily see that the optimal solution $\boldsymbol{\theta}^k$ of (2.6) is given by

$$\boldsymbol{\theta}^k = (\bar{\mathbf{M}}_\mathbf{k}^T \bar{\mathbf{M}}_\mathbf{k})^{-1} \bar{\mathbf{M}}_\mathbf{k}^T \dot{\mathbf{y}}, \tag{2.7}$$

where $\bar{\mathbf{M}}_\mathbf{k} = [\dot{\mathbf{m}}_1, \dots, \dot{\mathbf{m}}_k]$ is the matrix formed by all reshaped basis vectors. The row size of matrix $\bar{\mathbf{M}}_k$ is the total number of observed entries. It is computationally expensive to directly calculate the matrix multiplication. We simplify this step by an incremental process and give the implementation details in the appendix.

We run the above two steps iteratively until some desired stopping condition is satisfied. We can terminate the method based on the rank of the estimated matrix or the approximation residual. In particular, one can choose a preferred rank of the solution matrix. Alternatively, one can stop the method once the residual $\|\mathbf{R}_k\|$ is less than a tolerance parameter ε. The main steps of orthogonal rank-one matrix pursuit (OR1MP) are given in Algorithm 1.

ALGORITHM 1. OR1MP.

```
Input: $\mathbf{Y}_\Omega$ and stopping criterion.
Initialize: Set $\mathbf{X}_0 = 0$, $\boldsymbol{\theta}^0 = 0$ and $k = 1$.
repeat
    Step 1: Find a pair of top left and right singular vectors $(\mathbf{u}_k, \mathbf{v}_k)$ of the observed
    residual matrix $\mathbf{R}_k = \mathbf{Y}_\Omega - \mathbf{X}_{k-1}$ and set $\mathbf{M}_k = \mathbf{u}_k \mathbf{v}_k^T$.
    Step 2: Compute the weight vector $\boldsymbol{\theta}^k$ using the closed form least squares solu-
    tion $\boldsymbol{\theta}^k = (\bar{\mathbf{M}}_\mathbf{k}^T \bar{\mathbf{M}}_\mathbf{k})^{-1} \bar{\mathbf{M}}_\mathbf{k}^T \dot{\mathbf{y}}$.
    Step 3: Set $\mathbf{X}_k = \sum_{i=1}^k \theta_i^k (\mathbf{M}_i)_\Omega$ and $k \leftarrow k + 1$.
until stopping criterion is satisfied
Output: Constructed matrix $\hat{\mathbf{Y}} = \sum_{i=1}^k \theta_i^k \mathbf{M}_i$.
```

Remark 2.1. In our algorithm, we adapt OMP on the observed part of the matrix. This is similar to the GECO algorithm. However, GECO constructs the estimated matrix by projecting the observation matrix onto a much larger subspace, which is a product of two subspaces spanned by all left singular vectors and all right singular vectors obtained up to the current iteration. So it has a much higher computational complexity. Lee and Bresler [25] recently proposed the ADMiRA algorithm, which is also a greedy approach. In each step it first chooses $2r$ components by top-$2r$ truncated SVD and then uses another top-r truncated SVD to obtain a rank-r estimated matrix. Thus, the ADMiRA algorithm is computationally more expensive than the proposed algorithm. The difference between the proposed algorithm and ADMiRA is somewhat similar to the difference between OMP [35] for learning sparse vectors and CoSaMP [33]. In addition, the performance guarantees (including recovery guarantee and convergence property) of ADMiRA rely on strong assumptions, i.e., the matrix involved in the loss function satisfies a rank-restricted isometry property [25].

3. Convergence analysis of Algorithm 1.

In this section, we will show that Algorithm 1 is convergent and achieves a linear convergence rate. This result is given in the following theorem.

THEOREM 3.1. *OR1MP satisfies*

$$||\mathbf{R}_k|| \leq \left(\sqrt{1-\frac{1}{\min(m,n)}}\right)^{k-1} \|Y\|_\Omega \quad \forall k \geq 1.$$

Before proving Theorem 3.1, we need to establish some useful and preparatory properties of Algorithm 1. The first property says that $\mathbf{R}_{k+1}$ is perpendicular to all previously generated $\mathbf{M}_i$ for $i = 1, \ldots, k$.

PROPERTY 3.2. $\langle \mathbf{R}_{k+1}, \mathbf{M}_i \rangle = 0$ *for* $i = 1, \ldots, k$.

Proof. Recall that $\boldsymbol{\theta}^k$ is the optimal solution of problem (2.6). By the first-order optimality condition, one has

$$\left\langle \mathbf{Y} - \sum_{i=1}^{k} \theta_i^k \mathbf{M}_i, \mathbf{M}_i \right\rangle_\Omega = 0 \text{ for } i = 1, \ldots, k,$$

which together with $\mathbf{R}_k = \mathbf{Y}_\Omega - \mathbf{X}_{k-1}$ and $\mathbf{X}_k = \sum_{i=1}^{k} \theta_i^k (\mathbf{M}_i)_\Omega$ implies that $\langle \mathbf{R}_{k+1}, \mathbf{M}_i \rangle = 0$ for $i = 1, \ldots, k$. ∎

The following property shows that as the number of rank-one basis matrices $\mathbf{M}_i$ increases during our learning process, the residual $\|\mathbf{R}_k\|$ does not increase.

PROPERTY 3.3. $\|\mathbf{R}_{k+1}\| \leq \|\mathbf{R}_k\|$ *for all* $k \geq 1$.

Proof. We observe that for all $k \geq 1$,

$$\begin{aligned} \|\mathbf{R}_{k+1}\|^2 &= \min_{\boldsymbol{\theta} \in \Re^k} \{\|\mathbf{Y} - \sum_{i=1}^{k} \theta_i \mathbf{M}_i\|_\Omega^2\} \\ &\leq \min_{\boldsymbol{\theta} \in \Re^{k-1}} \{\|\mathbf{Y} - \sum_{i=1}^{k-1} \theta_i \mathbf{M}_i\|_\Omega^2\} \\ &= \|\mathbf{R}_k\|^2, \end{aligned}$$

and hence the conclusion holds. ∎

We next establish that $\{(\mathbf{M}_i)_\Omega\}_{i=1}^k$ is linearly independent unless $\|\mathbf{R}_k\| = 0$. It follows that formula (2.7) is well-defined and hence $\boldsymbol{\theta}^k$ is uniquely defined before the algorithm stops.

PROPERTY 3.4. *Suppose that* $\mathbf{R}_k \neq 0$ *for some* $k \geq 1$. *Then,* $\bar{\mathbf{M}}_i$ *has a full column rank for all* $i \leq k$.

Proof. Using Property 3.3 and the assumption $\mathbf{R}_k \neq 0$ for some $k \geq 1$, we see that $\mathbf{R}_i \neq 0$ for all $i \leq k$. We now prove the statement of this lemma by induction on i. Indeed, since $\mathbf{R}_1 \neq 0$, we clearly have $\bar{\mathbf{M}}_1 \neq 0$. Hence the conclusion holds for

$i = 1$. We now assume that it holds for $i-1 < k$ and need to show that it also holds for $i \le k$. By the induction hypothesis, $\bar{\mathbf{M}}_{i-1}$ has a full column rank. Suppose for contradiction that $\bar{\mathbf{M}}_i$ does not have a full column rank. Then, there exists $\boldsymbol{\alpha} \in \Re^{i-1}$ such that

$$(\mathbf{M}_i)_\Omega = \sum_{j=1}^{i-1} \alpha_j (\mathbf{M}_j)_\Omega,$$

which together with Property 3.2 implies that $\langle \mathbf{R}_i, \mathbf{M}_i \rangle = 0$. It follows that

$$\sigma_1(\mathbf{R}_i) = \mathbf{u}_i^T \mathbf{R}_i \mathbf{v}_i = \langle \mathbf{R}_i, \mathbf{M}_i \rangle = 0,$$

and hence $\mathbf{R}_i = 0$, which contradicts the fact that $\mathbf{R}_j \neq 0$ for all $j \le i$. Therefore, $\bar{\mathbf{M}}_i$ has a full column rank and the conclusion holds for general i. ▯

We next build a relationship between two consecutive residuals $\|\mathbf{R}_{k+1}\|$ and $\|\mathbf{R}_k\|$. For convenience, define $\theta_k^{k-1} = 0$ and let

$$\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} + \boldsymbol{\eta}^k.$$

In view of (2.6), one can observe that

$$\boldsymbol{\eta}^k = \arg\min_{\boldsymbol{\eta}} || \sum_{i=1}^{k} \eta_i \mathbf{M}_i - \mathbf{R}_k ||_\Omega^2. \tag{3.1}$$

Let

$$\mathbf{L}_k = \sum_{i=1}^{k} \eta_i^k (\mathbf{M}_i)_\Omega. \tag{3.2}$$

By the definition of $\mathbf{X}_k$, one can also observe that

$$\mathbf{X}_k = \mathbf{X}_{k-1} + \mathbf{L}_k,$$
$$\mathbf{R}_{k+1} = \mathbf{R}_k - \mathbf{L}_k.$$

PROPERTY 3.5. *$||\mathbf{R}_{k+1}||^2 = ||\mathbf{R}_k||^2 - ||\mathbf{L}_k||^2$ and $||\mathbf{L}_k||^2 \ge \langle \mathbf{M}_k, \mathbf{R}_k \rangle^2$, where $\mathbf{L}_k$ is defined in* (3.2).

Proof. Since $\mathbf{L}_k = \sum_{i \le k} \boldsymbol{\eta}_i^k (\mathbf{M}_i)_\Omega$, it follows from Property 3.2 that $\langle \mathbf{R}_{k+1}, \mathbf{L}_k \rangle = 0$. We then have

$$\begin{aligned} ||\mathbf{R}_{k+1}||^2 &= ||\mathbf{R}_k - \mathbf{L}_k||^2 \\ &= ||\mathbf{R}_k||^2 - 2\langle \mathbf{R}_k, \mathbf{L}_k \rangle + ||\mathbf{L}_k||^2 \\ &= ||\mathbf{R}_k||^2 - 2\langle \mathbf{R}_{k+1} + \mathbf{L}_k, \mathbf{L}_k \rangle + ||\mathbf{L}_k||^2 \\ &= ||\mathbf{R}_k||^2 - 2\langle \mathbf{L}_k, \mathbf{L}_k \rangle + ||\mathbf{L}_k||^2 \\ &= ||\mathbf{R}_k||^2 - ||\mathbf{L}_k||^2. \end{aligned}$$

We next bound $\|\mathbf{L}_k\|^2$ from below. If $\mathbf{R}_k = 0$, $||\mathbf{L}_k||^2 \ge \langle \mathbf{M}_k, \mathbf{R}_k \rangle^2$ clearly holds. We now suppose throughout the remaining proof that $\mathbf{R}_k \neq 0$. It then follows from Property 3.4 that $\bar{\mathbf{M}}_k$ has a full column rank. Using this fact and (3.1), we have

$$\boldsymbol{\eta}^k = \left(\bar{\mathbf{M}}_k^T \bar{\mathbf{M}}_k\right)^{-1} \bar{\mathbf{M}}_k^T \dot{\mathbf{r}}_k,$$

where $\dot{\mathbf{r}}_k$ is the reshaped residual vector of $\mathbf{R}_k$. Invoking that $\mathbf{L}_k = \sum_{i\leq k} \eta_i^k(\mathbf{M}_i)_\Omega$, we then obtain

$$||\mathbf{L}_k||^2 = \dot{\mathbf{r}}_k^T \bar{\mathbf{M}}_k(\bar{\mathbf{M}}_k^T\bar{\mathbf{M}}_k)^{-1}\bar{\mathbf{M}}_k^T\dot{\mathbf{r}}_k. \tag{3.3}$$

Let $\bar{\mathbf{M}}_k = \mathbf{QU}$ be the QR factorization of $\bar{\mathbf{M}}_k$, where $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and $\mathbf{U}$ is a $k \times k$ nonsingular upper triangular matrix. One can observe that $(\bar{\mathbf{M}}_k)_k = \dot{\mathbf{m}}_k$, where $(\bar{\mathbf{M}}_k)_k$ denotes the kth column of the matrix $\bar{\mathbf{M}}_k$ and $\dot{\mathbf{m}}_k$ is the reshaped vector of $(\mathbf{M}_k)_\Omega$. Recall that $\|\mathbf{M}_k\| = \|\mathbf{u}_k\mathbf{v}_k^T\| = 1$. Hence, $\|(\bar{\mathbf{M}}_k)_k\| \leq 1$. Due to $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$, $\bar{\mathbf{M}}_k = \mathbf{QU}$, and the definition of $\mathbf{U}$, we have

$$0 \;<\; |\mathbf{U}_{kk}| \;\leq\; \|\mathbf{U}_k\| \;=\; \|(\bar{\mathbf{M}}_k)_k\| \;\leq\; 1.$$

In addition, by Property 3.2, we have

$$\bar{\mathbf{M}}_k^T\dot{\mathbf{r}}_k = [0,\ldots,0,\langle\mathbf{M}_k,\mathbf{R}_k\rangle]^T. \tag{3.4}$$

Substituting $\bar{\mathbf{M}}_k = \mathbf{QU}$ into (3.3), and using $\mathbf{Q^TQ} = \mathbf{I}$ and (3.4), we obtain that

$$\begin{aligned}\|\mathbf{L}_k\|^2 &= \dot{\mathbf{r}}_k^T\bar{\mathbf{M}}_k(\mathbf{U}^T\mathbf{U})^{-1}\bar{\mathbf{M}}_k^T\dot{\mathbf{r}}_k\\ &= [0,\ldots,0,\langle\mathbf{M}_k,\mathbf{R}_k\rangle]\,\mathbf{U}^{-1}\mathbf{U}^{-T}\,[0,\ldots,0,\langle\mathbf{M}_k,\mathbf{R}_k\rangle]^T\\ &= \langle\mathbf{M}_k,\mathbf{R}_k\rangle^2/(\mathbf{U}_{kk})^2 \;\geq\; \langle\mathbf{M}_k,\mathbf{R}_k\rangle^2,\end{aligned}$$

where the last equality follows since $\mathbf{U}$ is upper triangular and the last inequality is due to $|\mathbf{U}_{kk}| \leq 1$. ◻

We are now ready to prove Theorem 3.1.

Proof of Theorem 3.1. Using the definition of $\mathbf{M}_k$, we have

$$\begin{aligned}\langle\mathbf{M}_k,\mathbf{R}_k\rangle &= \langle\mathbf{u}_k\mathbf{v}_k^T,\mathbf{R}_k\rangle \;=\; \sigma_1(\mathbf{R}_k)\\ &\geq \sqrt{\frac{\sum_i\sigma_i^2(\mathbf{R}_k)}{\operatorname{rank}(\mathbf{R}_k)}} \;=\; \sqrt{\frac{\|\mathbf{R}_k\|^2}{\operatorname{rank}(\mathbf{R}_k)}} \;\geq\; \sqrt{\frac{\|\mathbf{R}_k\|^2}{\min(m,n)}}.\end{aligned}$$

Using this inequality and Property 3.5, we obtain that

$$\begin{aligned}||\mathbf{R}_{k+1}||^2 &= ||\mathbf{R}_k||^2 - ||\mathbf{L}_k||^2 \;\leq\; ||\mathbf{R}_k||^2 - \langle\mathbf{M}_k,\mathbf{R}_k\rangle^2\\ &\leq (1-\tfrac{1}{\min(m,n)})||\mathbf{R}_k||^2.\end{aligned}$$

In view of this relation and the fact that $\|\mathbf{R}_1\| = \|\mathbf{Y}\|^2_\mathbf{\Omega}$, we easily conclude that

$$||\mathbf{R}_k|| \;\leq\; \left(\sqrt{1-\frac{1}{\min(m,n)}}\right)^{k-1}\|\mathbf{Y}\|_\mathbf{\Omega}.$$

This completes the proof. ◻

Remark 3.6. If Ω is the entire set of all indices of $\{(i,j), i = 1,\ldots,n, j = 1,\ldots,m\}$, our OR1MP algorithm equals the standard SVD using the power method. In particular, when Ω is the set of all indices while the given entries are noisy values of an exact matrix, our OR1MP algorithm can help remove the noise.

Remark 3.7. In a standard study of the convergence rate of OMP or the orthogonal greedy algorithm, one can only get $|\langle\mathbf{M}_k,\mathbf{R}_k\rangle| \geq \|\mathbf{R}_k\|^2$, which leads to a sublinear convergence. Our $\mathbf{M}_k$ is a data dependent construction which is based on the top left and right singular vectors of the residual matrix $\mathbf{R}_k$. It thus has a better estimate which gives us the linear convergence.

ALGORITHM 2. EOR1MP.

```
Input: $\mathbf{Y}_\Omega$ and stopping criterion.
Initialize: Set $\mathbf{X}_0 = 0$, $\boldsymbol{\theta}^0 = 0$ and $k = 1$.
repeat
    Step 1: Find a pair of top left and right singular vectors $(\mathbf{u}_k, \mathbf{v}_k)$ of the observed residual matrix $\mathbf{R}_k = \mathbf{Y}_\Omega - \mathbf{X}_{k-1}$ and set $\mathbf{M}_k = \mathbf{u}_k\mathbf{v}_k^T$.
    Step 2: Compute the optimal weights $\boldsymbol{\alpha}^k$ for $\mathbf{X}_{k-1}$ and $\mathbf{M}_k$ by solving $\min_{\boldsymbol{\alpha}} ||\alpha_1\mathbf{X}_{k-1} + \alpha_2(\mathbf{M}_k)_\Omega - \mathbf{Y}_\Omega||^2$.
    Step 3: Set $\mathbf{X}_k = \alpha_1^k\mathbf{X}_{k-1} + \alpha_2^k(\mathbf{M}_k)_\Omega$; $\theta_k^k = \alpha_2^k$ and $\theta_i^k = \theta_i^{k-1}\alpha_1^k$ for $i < k$; $k \leftarrow k + 1$.
until stopping criterion is satisfied
Output: Constructed matrix $\hat{\mathbf{Y}} = \sum_{i=1}^k \theta_i^k\mathbf{M}_i$.
```

4. An economic OR1MP algorithm. The proposed OR1MP algorithm has to track all pursued bases and save them in the memory. It demands $O(r|\Omega|)$ storage complexity to obtain a rank-r estimated matrix. For large-scale problems, such storage requirement is not negligible and restricts the rank of the matrix to be estimated. To adapt our algorithm to large-scale problems with a large approximation rank, we simplify the orthogonal projection step by only tracking the estimated matrix $\mathbf{X}_{k-1}$ and the rank-one update matrix $\mathbf{M}_k$. In this case, we only need to estimate the weights for these two matrices by solving the following least squares problem:

$$\boldsymbol{\alpha}^k = \arg\min_{\boldsymbol{\alpha}=\{\alpha_1,\alpha_2\}} ||\alpha_1\mathbf{X}_{k-1} + \alpha_2\mathbf{M}_k - \mathbf{Y}||_\Omega^2. \tag{4.1}$$

This still fully corrects all weights of the existed bases, though the correction is suboptimal. If we write the estimated matrix as a linear combination of the bases, we have $\mathbf{X}_k = \sum_{i=1}^k \theta_i^k(\mathbf{M}_i)_\Omega$ with $\theta_k^k = \alpha_2^k$ and $\theta_i^k = \theta_i^{k-1}\alpha_1^k$, for $i < k$. The detailed procedure of this simplified method is given in Algorithm 2.

The proposed economic orthogonal rank-one matrix pursuit algorithm (EOR1MP) uses the same amount of storage as the greedy algorithms [17, 42], which is significantly smaller than that required by our OR1MP algorithm, i.e., Algorithm 1. Interestingly, we can show that the EOR1MP algorithm is still convergent and retains the linear convergence rate. The main result is given in the following theorem.

THEOREM 4.1. *Algorithm 2, the EOR1MP algorithm, satisfies*

$$||\mathbf{R}_k|| \;\leq\; \left(\sqrt{1 - \frac{1}{\min(m,n)}}\right)^{k-1} \|\mathbf{Y}\|_\Omega \quad \forall k \geq 1.$$

Before proving Theorem 4.1, we present several useful properties of our Algorithm 2. The first property says that $\mathbf{R}_{k+1}$ is perpendicular to matrix $\mathbf{X}_{k-1}$ and matrix $\mathbf{M}_k$.

PROPERTY 4.2. $\langle\mathbf{R}_{k+1}, \mathbf{X}_{k-1}\rangle = 0$ *and* $\langle\mathbf{R}_{k+1}, \mathbf{M}_k\rangle = 0$.

Proof. Recall that $\boldsymbol{\alpha}^k$ is the optimal solution of problem (4.1). By the first-order optimality condition according to $\mathbf{X}_{k-1}$ and $\mathbf{M}_k$, one has

$$\langle\mathbf{Y} - \alpha_1^k\mathbf{X}_{k-1} - \alpha_2^k\mathbf{M}_k, \mathbf{X}_{k-1}\rangle_\Omega = 0$$

and

$$\langle \mathbf{Y} - \alpha_1^k \mathbf{X}_{k-1} - \alpha_2^k \mathbf{M}_k, \mathbf{M}_k \rangle_\Omega = 0,$$

which together with $\mathbf{R}_k = \mathbf{Y}_\Omega - \mathbf{X}_{k-1}$ imply that $\langle \mathbf{R}_{k+1}, \mathbf{X}_{k-1} \rangle = 0$ and $\langle \mathbf{R}_{k+1}, \mathbf{M}_k \rangle = 0$. ∎

PROPERTY 4.3. *$\|\mathbf{R}_{k+1}\|^2 = \|\mathbf{Y}_\Omega\|^2 - \|\mathbf{X}_k\|^2$ for all $k \geq 1$.*

Proof. We observe that for all $k \geq 1$,

$$\begin{aligned}\|\mathbf{Y}_\Omega\|^2 &= \|\mathbf{R}_{k+1} + \mathbf{X}_k\|^2 \\ &= \|\mathbf{R}_{k+1}\|^2 + \|\mathbf{X}_k\|^2 + 2\langle \mathbf{R}_{k+1}, \mathbf{X}_k \rangle \\ &= \|\mathbf{R}_{k+1}\|^2 + \|\mathbf{X}_k\|^2\end{aligned}$$

as $\langle \mathbf{R}_{k+1}, \mathbf{X}_k \rangle = \alpha_1^k \langle \mathbf{R}_{k+1}, \mathbf{X}_{k-1} \rangle + \alpha_2^k \langle \mathbf{R}_{k+1}, \mathbf{M}_k \rangle = 0$, and hence the conclusion holds. ∎

The following property shows that as the number of rank-one basis matrices $\mathbf{M}_i$ increases during our iterative process, the residual $\|\mathbf{R}_k\|$ decreases.

PROPERTY 4.4. *$\|\mathbf{R}_{k+1}\| \leq \|\mathbf{R}_k\|$ for all $k \geq 1$.*

Proof. We observe that for all $k \geq 1$,

$$\begin{aligned}\|\mathbf{R}_k\|^2 &= \min_{\boldsymbol{\alpha} \in \Re^2} \|\mathbf{Y} - \alpha_1 \mathbf{X}_{k-2} - \alpha_2 \mathbf{M}_{k-1}\|_\Omega^2 \\ &= \|\mathbf{Y} - (\alpha_1^{k-1} \mathbf{X}_{k-2} + \alpha_2^{k-1} \mathbf{M}_{k-1})\|_\Omega^2 \\ &\geq \min_{\boldsymbol{\alpha} \in \Re^2} \|\mathbf{Y} - \alpha_1(\alpha_1^{k-1} \mathbf{X}_{k-2} + \alpha_2^{k-1} \mathbf{M}_{k-1}) - \alpha_2 \mathbf{M}_k\|_\Omega^2 \\ &= \min_{\boldsymbol{\alpha} \in \Re^2} \|\mathbf{Y} - \alpha_1 \mathbf{X}_{k-1} - \alpha_2 \mathbf{M}_k\|_\Omega^2 \\ &= \|\mathbf{R}_{k+1}\|^2,\end{aligned}$$

and hence the conclusion holds. ∎

Let

$$\mathbf{A}_k = \mathbf{B}_k^T \mathbf{B}_k = \begin{bmatrix} \langle \mathbf{X}_{k-1}, \mathbf{X}_{k-1} \rangle & \langle \mathbf{X}_{k-1}, \mathbf{M}_k \rangle \\ \langle \mathbf{M}_k, \mathbf{X}_{k-1} \rangle & \langle \mathbf{M}_k, \mathbf{M}_k \rangle_\Omega \end{bmatrix}$$

and $\mathbf{B}_k = [vec(\mathbf{X}_{k-1}), vec((\mathbf{M}_k)_\Omega)]$. The solution of problem (4.1) is $\boldsymbol{\alpha}^k = \mathbf{A}_k^{-1} \mathbf{B}_k^T vec(\mathbf{Y}_\Omega)$. We next establish that $vec(\mathbf{X}_{k-1})$ and $vec((\mathbf{M}_k)_\Omega)$ are linearly independent unless $\|\mathbf{R}_k\| = 0$. It follows that $\mathbf{A}_k$ is invertible and hence $\boldsymbol{\alpha}^k$ is uniquely defined before the algorithm stops.

PROPERTY 4.5. *If $\mathbf{X}_{k-1} = \beta(\mathbf{M}_k)_\Omega$ for some $\beta \neq 0$, then $\|\mathbf{R}_{k+1}\| = \|\mathbf{R}_k\|$.*

Proof. If $\mathbf{X}_{k-1} = \beta(\mathbf{M}_k)_\Omega$ with nonzero β, we get

$$\begin{aligned}\|\mathbf{R}_{k+1}\|^2 &= \min_{\boldsymbol{\alpha} \in \Re^2} \|\mathbf{Y} - \alpha_1 \mathbf{X}_{k-1} - \alpha_2 \mathbf{M}_k\|_\Omega^2 \\ &= \min_{\boldsymbol{\alpha} \in \Re^2} \|\mathbf{Y} - (\alpha_1 + \alpha_2/\beta) \mathbf{X}_{k-1}\|_\Omega^2 \\ &= \min_{\gamma \in \Re} \|\mathbf{Y} - \gamma \mathbf{X}_{k-1}\|_\Omega^2 \\ &= \min_{\gamma \in \Re} \|\mathbf{Y} - \gamma \alpha_1^{k-1} \mathbf{X}_{k-2} - \gamma \alpha_2^{k-1} \mathbf{M}_{k-1}\|_\Omega^2 \\ &\geq \min_{\boldsymbol{\gamma} \in \Re^2} \|\mathbf{Y} - \gamma_1 \mathbf{X}_{k-2} - \gamma_2 \mathbf{M}_{k-1}\|_\Omega^2 \\ &= \|\mathbf{Y} - \mathbf{X}_{k-1}\|_\Omega^2 \\ &= \|\mathbf{R}_k\|^2,\end{aligned}$$

and hence the conclusion holds with $\|\mathbf{R}_k\|^2 \geq \|\mathbf{R}_{k+1}\|^2$ given in Property 4.4. ∎

PROPERTY 4.6. *Let $\sigma_1(\mathbf{R}_k)$ be the maximum singular value of $\mathbf{R}_k$. $\langle \mathbf{M}_k, \mathbf{R}_k\rangle = \sigma_1(\mathbf{R}_k) \geq \frac{\|\mathbf{R}_k\|}{\sqrt{\min(m,n)}}$ for all $k \geq 1$.*

Proof. The optimum $\mathbf{M}_k$ in our algorithm satisfies

$$\langle \mathbf{M}_k, \mathbf{R}_k\rangle = \max_{\text{rank}(\mathbf{M})=1} \langle \mathbf{M}, \mathbf{R}_k\rangle = \sigma_1(\mathbf{R}_k).$$

Using the fact that $\sqrt{\text{rank}(\mathbf{R}_k)}\sigma_1(\mathbf{R}_k) \geq \|\mathbf{R}_k\|$ and $\text{rank}(\mathbf{R}_k) \leq \min(m,n)$, we get the conclusion. ∎

PROPERTY 4.7. *Suppose that $\mathbf{R}_k \neq 0$ for some $k \geq 1$. Then, $\mathbf{X}_{k-1} \neq \beta(\mathbf{M}_k)_\Omega$ for all $\beta \neq 0$.*

Proof. If $\mathbf{X}_{k-1} = \beta(\mathbf{M}_k)_\Omega$ with $\beta \neq 0$, we have

$$\begin{aligned}
\|\mathbf{R}_{k+1}\|^2 &= \|\mathbf{Y} - \mathbf{X}_k\|_\Omega^2 \\
&= \min_{\boldsymbol{\alpha}\in\Re^2} \|\mathbf{Y} - \alpha_1\mathbf{X}_{k-1} - \alpha_2\mathbf{M}_k\|_\Omega^2 \\
&= \min_{\boldsymbol{\alpha}\in\Re^2} \|\mathbf{Y} - (\alpha_1 + \alpha_2/\beta)\mathbf{X}_{k-1}\|_\Omega^2 \\
&= \min_{\gamma\in\Re} \|\mathbf{Y} - \gamma\mathbf{X}_{k-1}\|_\Omega^2 \\
&= \|\mathbf{Y} - \gamma^k\mathbf{X}_{k-1}\|_\Omega^2 \\
&= \|\mathbf{R}_k\|^2 \\
&= \|\mathbf{Y} - \mathbf{X}_{k-1}\|_\Omega^2.
\end{aligned}$$

As $\mathbf{R}_k \neq 0$, we have $(\mathbf{M}_k)_\Omega \neq 0$ and $\mathbf{X}_{k-1} \neq 0$. Then from the above equality, we conclude that $\gamma^k = 1$ is the unique optimal solution of the minimization in terms of γ, and thus we obtain its first-order optimality condition: $\langle \mathbf{X}_{k-1}, \mathbf{R}_k\rangle = 0$. However, this contradicts

$$\langle \mathbf{X}_{k-1}, \mathbf{R}_k\rangle = \beta\langle \mathbf{M}_k, \mathbf{R}_k\rangle = \beta\sigma_1(\mathbf{R}_k) \neq 0.$$

This completes the proof. ∎

We next build a relationship between two consecutive residuals $\|\mathbf{R}_{k+1}\|$ and $\|\mathbf{R}_k\|$.

PROPERTY 4.8. $\|\mathbf{R}_{k+1}\|^2 \leq \|\mathbf{R}_k\|^2 - \frac{\sigma_1^2(\mathbf{R}_k)}{\langle \mathbf{M}_k, \mathbf{M}_k\rangle_\Omega}$.

Proof.

$$\begin{aligned}
\|\mathbf{R}_{k+1}\|^2 &= \min_{\boldsymbol{\alpha}\in\Re^2} \|\mathbf{Y} - \alpha_1\mathbf{X}_{k-1} - \alpha_2\mathbf{M}_k\|_\Omega^2 \\
&\leq \min_{\alpha_2\in\Re} \|\mathbf{Y} - \mathbf{X}_{k-1} - \alpha_2\mathbf{M}_k\|_\Omega^2 \\
&= \min_{\alpha_2\in\Re} \|\mathbf{R}_k - \alpha_2\mathbf{M}_k\|_\Omega^2.
\end{aligned}$$

This has a closed form solution as $\alpha_2^* = \frac{\langle \mathbf{R}_k, \mathbf{M}_k\rangle}{\langle \mathbf{M}_k, \mathbf{M}_k\rangle_\Omega}$. Plugging this optimum α_2^* back into the formulation, we get

$$\begin{aligned}
\|\mathbf{R}_{k+1}\|^2 &\leq \|\mathbf{R}_k - \frac{\langle \mathbf{R}_k, \mathbf{M}_k\rangle}{\langle \mathbf{M}_k, \mathbf{M}_k\rangle_\Omega}\mathbf{M}_k\|_\Omega^2 \\
&= \|\mathbf{R}_k\|^2 - \frac{\langle \mathbf{R}_k, \mathbf{M}_k\rangle^2}{\langle \mathbf{M}_k, \mathbf{M}_k\rangle_\Omega} \\
&= \|\mathbf{R}_k\|^2 - \frac{\sigma_1^2(\mathbf{R}_k)}{\langle \mathbf{M}_k, \mathbf{M}_k\rangle_\Omega}.
\end{aligned}$$

This completes the proof. ∎

We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1. Using the definition of $\mathbf{M}_k$ with its normalization property $\langle \mathbf{M}_k, \mathbf{M}_k \rangle_\Omega \le 1$, Property 4.8, and Property 4.6, we obtain that

$$\begin{aligned} ||\mathbf{R}_{k+1}||^2 &\le ||\mathbf{R}_k||^2 - \tfrac{\sigma_1^2(\mathbf{R}_k)}{\langle \mathbf{M}_k, \mathbf{M}_k \rangle_\Omega} \le ||\mathbf{R}_k||^2 - \sigma_1^2(\mathbf{R}_k) \\ &\le \left(1 - \tfrac{1}{\min(m,n)}\right) ||\mathbf{R}_k||^2. \end{aligned}$$

In view of this relation and the fact that $\|\mathbf{R}_1\| = \|\mathbf{Y}\|_{\mathbf{\Omega}}^{\mathbf{2}}$, we easily conclude that

$$||\mathbf{R}_k|| \le \left(\sqrt{1 - \frac{1}{\min(m,n)}}\right)^{k-1} \|\mathbf{Y}\|_{\mathbf{\Omega}}.$$

This completes the proof. ◻

5. An extension to the matrix sensing problem and its convergence analysis. In this section, we extend our algorithm to deal with the following matrix sensing problem (cf. [36, 25, 18, 19]):

$$\min_{\mathbf{X} \in \Re^{n \times m}} \operatorname{rank}(\mathbf{X}) : \mathcal{A}(\mathbf{X}) = \mathcal{A}(\mathbf{Y}), \tag{5.1}$$

where $\mathbf{Y}$ is a target low rank matrix and $\mathcal{A}$ is a linear operator, e.g., $\mathcal{A}$ consists of a set of measurements $\langle \mathbf{A}_i, \mathbf{X} \rangle = \langle \mathbf{A}_i, \mathbf{Y} \rangle$ for a sequence of matrices $\{\mathbf{A}_i\}$. $\mathcal{A}(\mathbf{X})$ could be written in a compact form as

$$\mathcal{A}(\mathbf{X}) = \begin{bmatrix} vec(\mathbf{A}_1)^T \\ \vdots \\ vec(\mathbf{A}_d)^T \end{bmatrix} vec(\mathbf{X})$$

for d measurements. Clearly, the matrix completion studied in the previous sections is a special case of the above problem by setting the linear operator $\mathcal{A}$ to be the observation operator P_Ω.

We first explain how to use our algorithm to solve this matrix sensing problem (5.1). Recall a linear operator vec which maps a matrix $\mathbf{X}$ of size $n \times m$ to a vector $vec(\mathbf{X})$ of size $mn \times 1$. We now define an inverse operator mat_{nm} which converts a vector $\mathbf{v}$ of size $mn \times 1$ to a matrix $\mathbf{V} = mat_{nm}(\mathbf{v})$ of size $n \times m$. Note that when $\mathbf{X}$ is vectorized into $vec(\mathbf{X})$, the linear operator $\mathcal{A}$ can be expressed in terms of matrix $\mathbf{A} = [vec(\mathbf{A}_1), \ldots, vec(\mathbf{A}_d)]^T$. That is, $\mathcal{A}(\mathbf{X}) = \mathcal{A}(\mathbf{Y})$ can be rewritten as $\mathbf{A}vec(\mathbf{X}) = \mathbf{A}vec(\mathbf{Y})$. For convenience, we can write $\mathcal{A} = \mathbf{A}vec$. It is clear that $\mathbf{A}$ is a matrix of size $d \times mn$. Certainly, one can find its pseudoinverse $\mathbf{A}^\dagger$ which is $\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}$ as we have assumed that $\mathbf{A}$ is of full row rank. We note that since $d << mn$, $\mathbf{A}\mathbf{A}^\dagger = \mathbf{I}_d$ while $\mathbf{A}^\dagger\mathbf{A} \neq \mathbf{I}_{mn}$, where $\mathbf{I}_d$ and $\mathbf{I}_{mn}$ are the identity matrices of size $d \times d$ and $mn \times mn$, respectively. For convenience, we let $\mathcal{A}^{-1}$ denote $mat_{nm} \circ \mathbf{A}^\dagger$, where $\circ$ is the Hadamard product. The linear operators satisfy

$$\mathcal{A}\mathcal{A}^{-1}\mathbf{b} = \mathbf{b}$$

for any vector $\mathbf{b}$ of size $d \times 1$, while $\mathcal{A}^{-1}\mathcal{A}$ is not an identity operator. We are now ready to tackle the matrix sensing problem (5.1) as follows: let $\mathbf{b} = \mathcal{A}(\mathbf{Y}) = \mathbf{A}vec(\mathbf{Y})$ and $\mathbf{R}_0 = \mathcal{A}^{-1}(\mathbf{b})$ be the given matrix. We apply Algorithm 3 to obtain $\mathbf{M}(\boldsymbol{\theta}^k)$ in $k \ge r$ steps.

Algorithm 3. Rank-one matrix pursuit for matrix sensing.

```
Input: $\mathbf{R}_0$ and stopping criterion.
Initialize: Set $\mathbf{X}_0 = 0$ and $k = 1$.
repeat
    Step 1: Find a pair of top left and right singular vectors $(\mathbf{u}_k, \mathbf{v}_k)$ of the residual matrix $\mathbf{R}_k$ by using the power method and set $\mathbf{M}_k = \mathbf{u}_k \mathbf{v}_k^T$.
    Step 2: Compute the weight vector $\boldsymbol{\theta}^k$ using the closed form least squares approximation of $\mathbf{R}_0$ by the best rank-one matrices $\mathbf{M}_i$, $i = 1, \ldots, k$:
        $\boldsymbol{\theta}^k = \arg\min_{\theta_1,\ldots,\theta_k} \|\mathbf{R}_0 - \sum_{i=1}^k \theta_i \mathcal{A}^{-1}\mathcal{A}(\mathbf{M}_i)\|_F^2.$
    Step 3: Set $\mathbf{M}(\boldsymbol{\theta}^k) = \sum_{i=1}^k \theta_i^k \mathbf{M}_i$, $\mathbf{R}_{k+1} = \mathbf{R}_0 - \mathcal{A}^{-1}\mathcal{A}(\mathbf{M}(\boldsymbol{\theta}^k))$ and set $k \leftarrow k+1$.
until stopping criterion is satisfied
Output: the constructed matrix $\hat{\mathbf{Y}} = \mathbf{M}(\boldsymbol{\theta}^k)$.
```

We shall show that $\mathbf{M}(\boldsymbol{\theta}^k)$ converges to the exact rank-r matrix $\mathbf{Y}$. First of all, Algorithm 3 can also be proved to be linearly convergent using the same procedure as in the proof of Theorem 3.1. We thus have the following theorem without presenting a detailed proof.

Theorem 5.1. *Each step in Algorithm 3 satisfies*

$$||\mathbf{R}_k|| \leq \left(\sqrt{1 - \frac{1}{\min(m,n)}}\right)^{k-1} \|\mathcal{A}^{-1}(\mathbf{b})\| \quad \forall k \geq 1.$$

This holds for all matrices $\mathbf{Y}$ of rank at most r.

We now show $\mathbf{M}(\boldsymbol{\theta}^k)$ approximates the exact matrix $\mathbf{Y}$ for a large k. In the setting of matrix sensing, we are able to use the rank-RIP condition. Let us recall the following.

Definition 5.2. *Let $\mathcal{A}$ be a linear map on linear space of matrices of size $n \times m$ with $n \leq m$. For every integer r with $1 \leq r \leq n$, let the rank-r restricted isometry constant be the smallest number $\delta_r(\mathcal{A})$ such that*

$$(1 - \delta_r(\mathcal{A}))\|\mathbf{X}\|_F^2 \leq \|\mathcal{A}(\mathbf{X})\|_2^2 \leq (1 + \delta_r(\mathcal{A}))\|\mathbf{X}\|_F^2$$

holds for all matrices $\mathbf{X}$ of rank at most r.

It is known that for some random matrices $\mathbf{A}$, $\mathcal{A} = \mathbf{A}vec$ satisfies the rank-RIP condition with high probability [36]. Armed with the rank-RIP condition, we are able to establish the following result.

Theorem 5.3. *Let $\mathbf{Y}$ be a matrix of rank r. Suppose the measurement mapping $\mathcal{A}(\mathbf{X})$ satisfies rank-RIP for rank-r_0 with $\delta_{r_0} = \delta_{r_0}(\mathcal{A}) < 1$ with $r_0 \geq 2r$. The output matrix $\mathbf{M}(\boldsymbol{\theta}^k)$ from Algorithm 3 approximates the exact matrix $\mathbf{Y}$ in the following sense: there is a positive constant $\tau < 1$ such that*

$$\|\mathbf{M}(\boldsymbol{\theta}^k) - \mathbf{Y}\|_F \leq \frac{C}{\sqrt{1 - \delta_{r_0}}} \tau^k$$

for all $k = 1, \ldots, r_0 - r$, where $C > 0$ is a constant dependent on $\mathcal{A}$.

Proof. Using the definition of δ_{r_0}, for $k + r \leq r_0$, we have

$$\begin{aligned}(1-\delta_{r_0})\|\mathbf{M}(\boldsymbol{\theta}^k) - \mathbf{Y}\|_F^2 &\leq \|\mathcal{A}(\mathbf{M}(\boldsymbol{\theta}^k)) - \mathcal{A}(\mathbf{Y})\|_2^2 \\ &= \|\mathcal{A}(\mathbf{R}_k)\|_2^2 = \|\mathbf{A}vec(\mathbf{R}_k)\|_2^2 \\ &\leq \|\mathbf{A}\|_2^2\|vec(\mathbf{R}_k)\|_2^2 = \|\mathbf{A}\|_2^2\|\mathbf{R}_k\|_F^2 \\ &\leq \|\mathbf{A}\|_2^2\tau^{2k}\|\mathcal{A}^{-1}(\mathbf{b})\|_F^2,\end{aligned}$$

where the last inequality follows from Theorem 5.1 with $\tau = \sqrt{1 - \frac{1}{\min\{m,n\}}}$. It follows that

$$\|\mathbf{M}(\boldsymbol{\theta}^k) - \mathbf{Y}\|_F^2 \leq \frac{\|\mathbf{A}\|_2^2\tau^{2k}}{1-\delta_{r_0}}\|\mathcal{A}^{-1}(\mathbf{b})\|_F^2.$$

Therefore, we have the desired result. □

Similarly we can extend our economic algorithm to the setting of matrix sensing. We leave it to the interested reader. In the above convergence analysis, we require $k \leq r_0 - r$, which guarantees the matrix-RIP condition for all estimated matrices during the learning process. It will be interesting to explore if a similar result can be obtained for any $k > 0$.

6. Effect of inexact top singular vectors.

In our rank-one matrix pursuit algorithms, we need to calculate the top singular vector pair of the residual matrix in each iteration. We rewrite it here as

$$\max_{\mathbf{u},\mathbf{v}} \{\mathbf{u}^T\mathbf{R}_k\mathbf{v} : \ \|\mathbf{u}\| = \|\mathbf{v}\| = \mathbf{1}\}. \tag{6.1}$$

We solve this problem efficiently by the power method, which is an iterative method. In practice, we obtain a solution with approximation error less than a small tolerance $\delta_k \geq 0$, that is,

$$\tilde{\mathbf{u}}^T\mathbf{R}_k\tilde{\mathbf{v}} \geq (1-\delta_k)\max_{\|\mathbf{u}\|=\|\mathbf{v}\|=\mathbf{1}}\{\mathbf{u}^T\mathbf{R}_k\mathbf{v}\}. \tag{6.2}$$

We show that the proposed algorithms still retain the linear convergence rate when the top singular pair computed at each iteration satisfies (6.2) for $0 \leq \delta_k < 1$. This result is given in the following theorem.

THEOREM 6.1. *Assume that there is a tolerance parameter* $0 \leq \delta < 1$ *such that* $\delta_k \leq \delta$ *for all* k. *Then the orthogonal rank-one matrix pursuit algorithms achieve a linear convergence rate*

$$||\mathbf{R}_k|| \ \leq \ \left(\sqrt{1 - \frac{q^2}{\min(m,n)}}\right)^{k-1}\|\mathbf{Y}\|_{\mathbf{\Omega}},$$

where $q = 1 - \delta$ *satisfies* $0 < q \leq 1$.

Proof. In Step 1 of our algorithms, we iteratively solve the problem (6.1) using the power method. In this method, we stop the iteration such that

$$\tilde{\mathbf{u}}_k^T\mathbf{R}_k\tilde{\mathbf{v}}_k \geq (1-\delta_k)\max_{\|\mathbf{u}\|=\mathbf{1},\|\mathbf{v}\|=\mathbf{1}}\{\mathbf{u}^T\mathbf{R}_k\mathbf{v}\} \geq 0$$

with $0 \leq \delta_k \leq \delta < 1$. Denote $\tilde{\mathbf{M}}_k = \tilde{\mathbf{u}}_k \tilde{\mathbf{v}}_k^T$ as the generated basis. Next, we show that the following holds for both OR1MP and EOR1MP:

$$\|\mathbf{R}_{k+1}\|^2 \leq \|\mathbf{R}_k\|^2 - \langle \tilde{\mathbf{M}}_k, \mathbf{R}_k \rangle^2.$$

For the OR1MP algorithm, we have

$$\begin{aligned}\|\mathbf{R}_{k+1}\|^2 &= \min_{\boldsymbol{\theta} \in \Re^k} \|\mathbf{Y} - \sum_{i=1}^{k} \theta_i \tilde{\mathbf{M}}_i\|_\Omega^2 \\ &\leq \min_{\theta_k \in \Re} \|\mathbf{Y} - \mathbf{X}_{k-1} - \theta_k \tilde{\mathbf{M}}_k\|_\Omega^2 \\ &= \min_{\theta_k \in \Re} \|\mathbf{R}_k - \theta_k \tilde{\mathbf{M}}_k\|_\Omega^2.\end{aligned}$$

For the EOR1MP algorithm, we have

$$\begin{aligned}\|\mathbf{R}_{k+1}\|^2 &= \min_{\boldsymbol{\alpha} \in \Re^2} \|\mathbf{Y} - \alpha_1 \mathbf{X}_{k-1} - \alpha_2 \tilde{\mathbf{M}}_k\|_\Omega^2 \\ &\leq \min_{\alpha_2 \in \Re} \|\mathbf{Y} - \mathbf{X}_{k-1} - \alpha_2 \tilde{\mathbf{M}}_k\|_\Omega^2 \\ &= \min_{\alpha_2 \in \Re} \|\mathbf{R}_k - \alpha_2 \tilde{\mathbf{M}}_k\|_\Omega^2.\end{aligned}$$

In both cases, we obtain closed form solutions as $\frac{\langle \mathbf{R}_k, \tilde{\mathbf{M}}_k \rangle}{\langle \tilde{\mathbf{M}}_k, \tilde{\mathbf{M}}_k \rangle_\Omega}$. Plugging the optimum solution into the corresponding formulations, we get

$$\begin{aligned}\|\mathbf{R}_{k+1}\|^2 &\leq \|\mathbf{R}_k - \frac{\langle \mathbf{R}_k, \tilde{\mathbf{M}}_k \rangle}{\langle \tilde{\mathbf{M}}_k, \tilde{\mathbf{M}}_k \rangle_\Omega} \tilde{\mathbf{M}}_k\|_\Omega^2 \\ &= \|\mathbf{R}_k\|^2 - \frac{\langle \mathbf{R}_k, \tilde{\mathbf{M}}_k \rangle^2}{\langle \tilde{\mathbf{M}}_k, \tilde{\mathbf{M}}_k \rangle_\Omega^2} \langle \tilde{\mathbf{M}}_k, \tilde{\mathbf{M}}_k \rangle_\Omega \\ &\leq \|\mathbf{R}_k\|^2 - \langle \mathbf{R}_k, \tilde{\mathbf{M}}_k \rangle^2,\end{aligned}$$

as $\langle \tilde{\mathbf{M}}_k, \tilde{\mathbf{M}}_k \rangle_\Omega \leq 1$. It follows from Properties 4.5 and 4.6 that

$$\langle \mathbf{R}_k, \tilde{\mathbf{M}}_k \rangle \geq (1 - \delta_k) \sigma_1(\mathbf{R}_k) \geq (1 - \delta_k) \frac{\|\mathbf{R}_k\|}{\sqrt{\operatorname{rank}(\mathbf{R}_k)}}.$$

Combining the above two results, we get

$$\|\mathbf{R}_{k+1}\|^2 \leq \left(1 - \frac{(1 - \delta_k)^2}{\min(m, n)}\right) \|\mathbf{R}_k\|^2.$$

In view of this relation and the fact that $\|\mathbf{R}_1\| = \|\mathbf{Y}\|_{\boldsymbol{\Omega}}^{\mathbf{2}}$, we conclude that

$$||\mathbf{R}_k|| \ \leq \ \left(\sqrt{1 - \frac{q^2}{\min(m, n)}}\right)^{k-1} \|\mathbf{Y}\|_{\boldsymbol{\Omega}},$$

where $q = 1 - \delta \leq \inf(1 - \delta_k) = 1 - \sup \delta_k$ and is a constant between $(0, 1]$. This completes the proof. $\square$

7. Experiments.

In this section, we compare the two versions of our algorithm, e.g., OR1MP and EOR1MP, with several state-of-the-art matrix completion methods in the literature. The competing algorithms include SVP [18], SVT [7], Jaggi's fast algorithm for trace norm constraint (JS) [17], the spectral regularization algorithm (SoftImpute) [30], low rank matrix fitting (LMaFit) [46], a boosting-type accelerated matrix-norm penalized solver (Boost) [49], atomic decomposition for minimum rank approximation (ADMiRA) [25], and GECO [38]. The first three solve trace norm constrained problems; the next three solve trace norm penalized problems; the last two directly solve the low rank constrained problem. The general greedy method [42] is not included in our comparison, as it includes JS and GECO (included in our comparison) as special cases for matrix completion. The lifted coordinate descent method [9] is not included in our comparison as it is sensitive to the parameters and is less efficient than Boost proposed in [49].

The codes for most of these methods are available online:

- SVP, http://www.cs.utexas.edu/~pjain/svp;
- SVT, http://svt.stanford.edu;
- SoftImpute, http://www-stat.stanford.edu/~rahulm/software.html;
- LMaFit, http://lmafit.blogs.rice.edu;
- Boost, http://webdocs.cs.ualberta.ca/~xinhua2/boosting.zip;
- GECO, http://www.cs.huji.ac.il/~shais/code/geco.zip.

We compare these algorithms in two applications: image recovery and collaborative filtering or recommendation problem. The data size for image recovery is relatively small, and the recommendation problem is large-scale. All the competing methods are implemented in MATLAB[1] and call some external packages for fast computation of SVD[2] and sparse matrix computations. The experiments are run on a PC with the windows 7 system, Intel 4 core 3.4-GHz CPU, and 8G RAM.

In the following experiments, we follow the recommended settings of the parameters for the competing algorithms. If no recommended parameter value is available, we choose the best one from a candidate set using cross validation. For our OR1MP and EOR1MP algorithms, we only need a stopping criterion. For simplicity, we stop our algorithms after r iterations. In this way, we approximate the ground truth using a rank-r matrix. We present the experimental results using two metrics, *peak signal-to-noise ratio* (PSNR) [16] and *root-mean-square error* (RMSE) [22]. PSNR is a test metric specific for images. A higher value in PSNR generally indicates better quality [16]. RMSE is a general metric for prediction. It measures the approximation error of the corresponding result.

7.1. Convergence and efficiency.

Before we present the numerical results from these comparison experiments, we shall include another algorithm called the forward rank-one matrix pursuit algorithm (FR1MP), which extends the matching pursuit method from the vector case to the matrix case. The detailed procedure of this method is given in Algorithm 4.

In FR1MP, we add the pursued rank-one matrix with an optimal weight in each iteration, which is similar to the forward selection rule [14]. This is a standard algorithm to find SVD of any matrix $\mathbf{Y}$ if all its entries are given. In this case, the FR1MP algorithm is more efficient in finding SVD of the matrix than our two proposed al-

[1]GECO is written in C++ and we call its executable file in MATLAB.

[2]PROPACK is used in SVP, SVT, SoftImpute and Boost. It is an efficient SVD package, which is implemented in C and Fortran. It can be downloaded from http://soi.stanford.edu/~rmunk/PROPACK.

ALGORITHM 4. FR1MP.

```
Input: $\mathbf{Y}_\Omega$ and stopping criterion.
Initialize: Set $\mathbf{X}_0 = 0$, $\boldsymbol{\theta}^0 = 0$ and $k = 1$.
repeat
  Step 1: Find a pair of top left and right singular vectors $(\mathbf{u}_k, \mathbf{v}_k)$ of the observed residual matrix $\mathbf{R}_k = \mathbf{Y}_\Omega - \mathbf{X}_{k-1}$ and set $\mathbf{M}_k = \mathbf{u}_k \mathbf{v}_k^T$.
  Step 2: Set $\theta_k^k = (\mathbf{u}_k^T \mathbf{R}_k \mathbf{v}_k)/\|\mathbf{M}_k\|_\Omega$, and $\theta_i^k = \theta_i^{k-1}$ for $i \leq k-1$.
  Step 3: Set $\mathbf{X}_k = \mathbf{X}_{k-1} + \theta_k^k (\mathbf{M}_k)_\Omega$; $k \leftarrow k+1$.
until stopping criterion is satisfied
Output: Constructed matrix $\hat{\mathbf{Y}} = \sum_{i=1}^k \theta_i^k \mathbf{M}_i$.
```

gorithms. However, when only partial entries are known, the FR1MP algorithm will not be able to find the best low rank solution. The computational step to find $\boldsymbol{\theta}^k$ in our proposed algorithms is necessary.

The empirical results for convergence efficiency of our proposed algorithms are reported in Figures 1 and 2. They are based on an image recovery experiment as well as an experiment of a movie recommendation dataset, Netflix [22, 4, 5]. The Netflix dataset has 10^8 ratings of 17,770 movies by 480,189 Netflix[3] customers. This is a large-scale dataset, and most of the competing methods are not applicable for this dataset. In Figure 1, we present the convergence characteristics of the proposed OR1MP algorithm. As the memory demand is increasing w.r.t. the iterations, we can run it for only about 40 iterations on the Netflix dataset. The EOR1MP algorithm has no such limitation. The results in Figure 2 show that our EOR1MP algorithm rapidly reduces the approximation error. We also present the same residual curves in logarithmic scale with a relatively large number of iterations in Figure 3, which verify the linear convergence property of our algorithms. These results are consistent with our theoretical analysis.

In the convergence analysis, we derive the upper bound for the convergence speed of our proposed algorithms. From Theorems 3.1 and 4.1, the convergence speed is controlled by the value of $\|\mathbf{R}_k\|_F^2/\sigma_{k,*}^2$, where $\sigma_{k,*}$ is the maximum singular value of the residual matrix $\mathbf{R}_k$ in the kth iteration. A smaller value indicates a faster convergence of our algorithms. Though it has a worst-case upper bound of $\|\mathbf{R}_k\|_F^2/\sigma_{k,*}^2 \leq \operatorname{rank}(\mathbf{R}_k) \leq \min(m, n)$, in the following experiments, we empirically verify that its value is much smaller than the theoretical worst case. Thus the convergence speed of our algorithms is much faster than the theoretical worst case. We present the values of $\|\mathbf{R}_k\|_F^2/\sigma_{k,*}^2$ at different iterations on the Lenna image and the MovieLens1M dataset for both of our algorithms in Figure 4. The results show that the quantity $\|\mathbf{R}_k\|_F^2/\sigma_{k,*}^2$ is much smaller than $\min(m, n)$.

In the following experiments, we plot the residual curves over iterations for different rank-one matrix pursuit algorithms, including our OR1MP algorithm, our EOR1MP algorithm, and the FR1MP algorithm. The evaluations are conducted on the Lenna image and the MovieLens1M dataset, which are given in Figure 5. The results show that among the three algorithms, EOR1MP and OR1MP perform better than the forward pursuit algorithm. It is interesting to note that EOR1MP achieves a similar performance as OR1MP, while it demands much less computational cost.

[3]http://www.netflixprize.com.

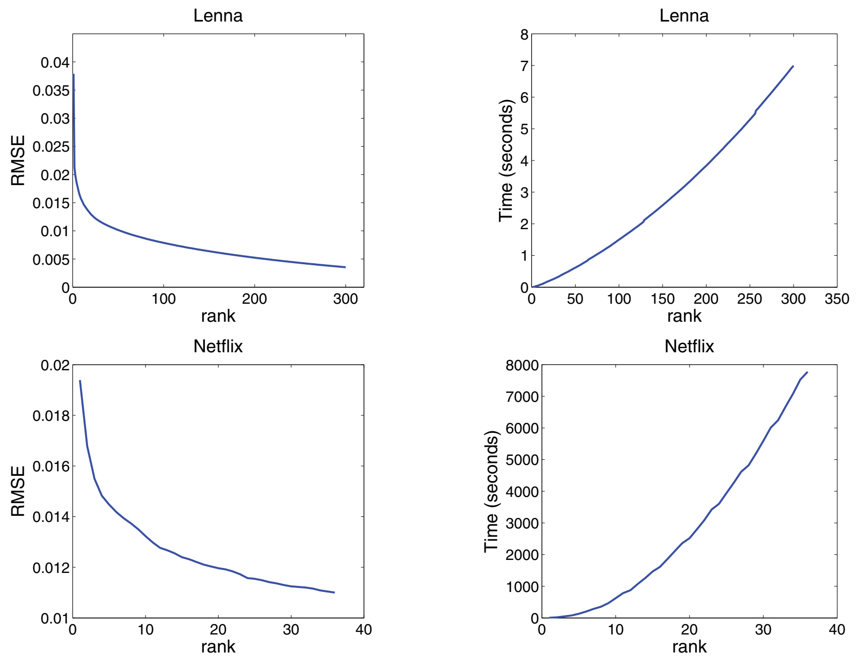

FIG. 1. *Illustration of convergence of the proposed OR1MP algorithm on the Lenna image and the Netflix dataset: the x-axis is the rank, the y-axis is the RMSE (left column), and the running time is measured in seconds (right column).*

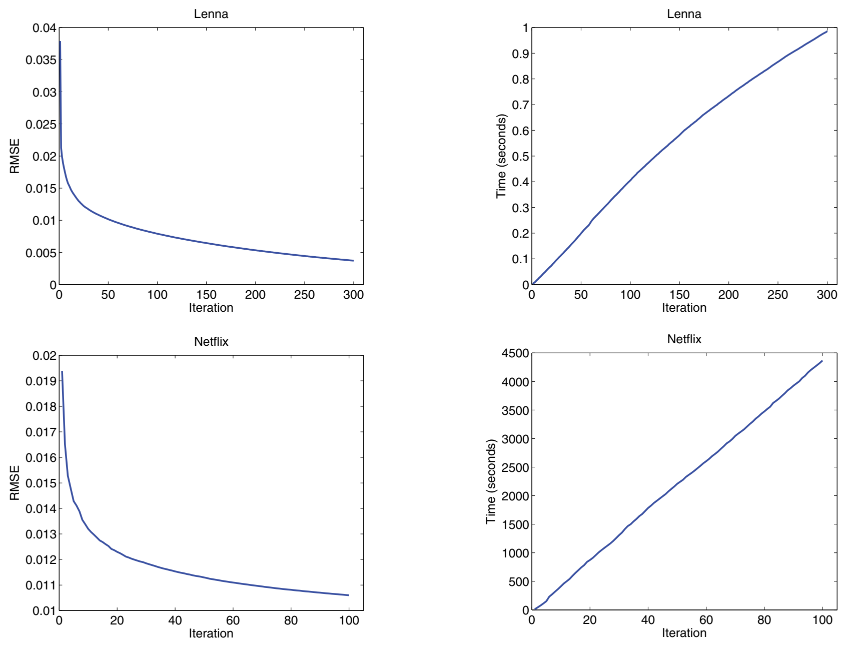

FIG. 2. *Illustration of convergence of the proposed EOR1MP algorithm on the Lenna image and the Netflix dataset: the x-axis is the rank, the y-axis is the RMSE (left column), and the running time is measured in seconds (right column).*

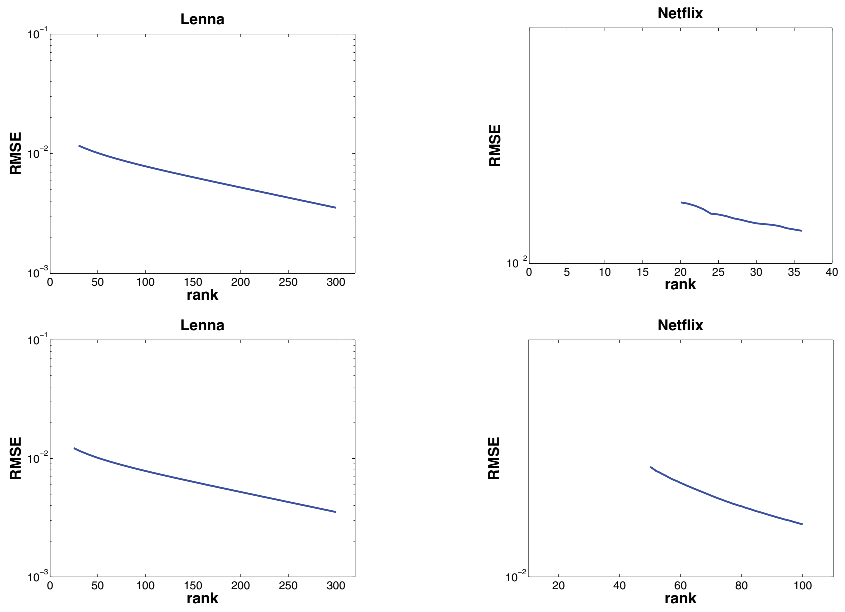

FIG. 3. *Illustration of the linear convergence of different rank-one matrix pursuit algorithms on the Lenna image and the Netflix dataset: the x-axis is the iteration, and the y-axis is the RMSE in log scale. The curves in the first row are the results for OR1MP and the curves in the second row are the results for EOR1MP.*

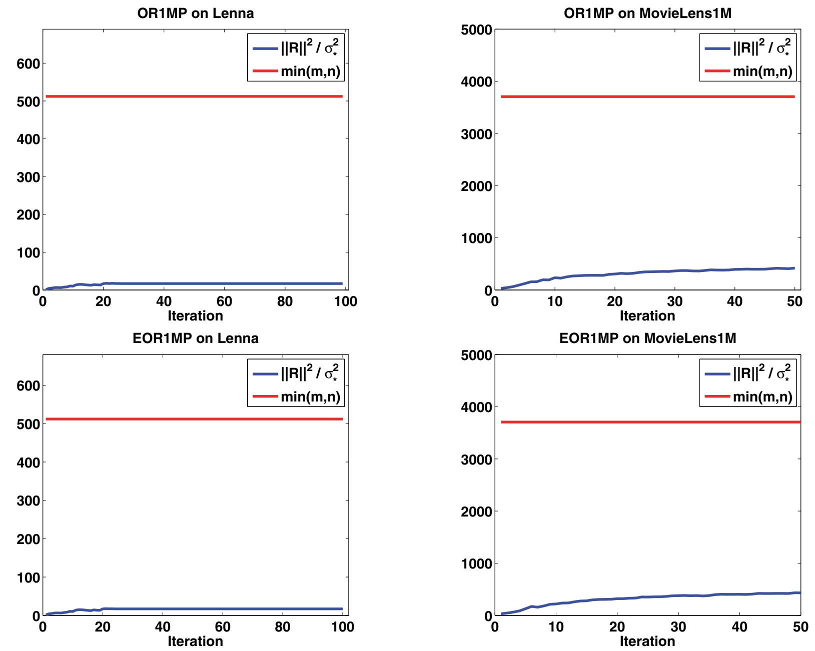

FIG. 4. *Illustration of the values of* $\|\mathbf{R}\|^2/\sigma_*^2$ *at different iterations and the value of* $\min(m,n)$ *on the Lenna image and MovieLens1M for both R1MP and ER1MP algorithms: the x-axis is the iteration number; the y-axis is the value.*

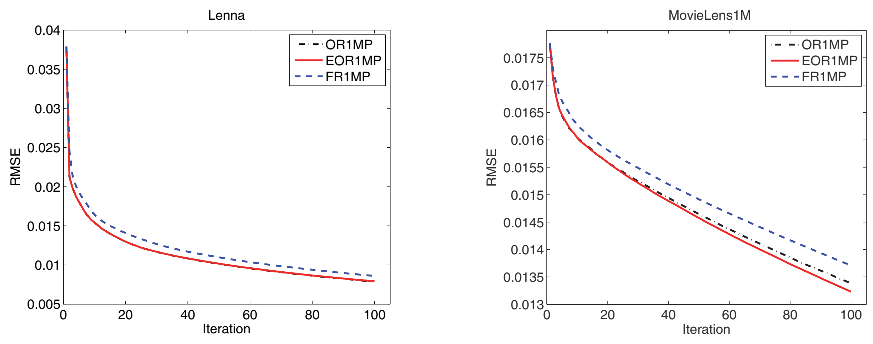

FIG. 5. *Illustration of convergence speed of different rank-one matrix pursuit algorithms on the Lenna image and the MovieLens1M dataset: the x-axis is the iteration; the y-axis is the RMSE.*

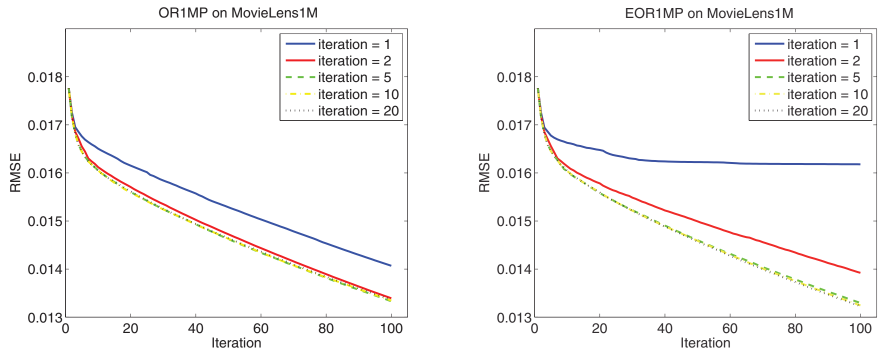

FIG. 6. *Illustration of convergence property of the proposed algorithms with different iteration numbers in the power method on the MovieLens1M dataset: the x-axis is the outer iteration number; the y-axis is the RMSE.*

7.2. Inexact top singular vectors. We empirically analyze the performance of our algorithms with inexact singular vector computation. In the experiments, we control the total number of iterations in the power method for computing the top singular vector pair. The numbers of iterations are set as $\{1, 2, 5, 10, 20\}$. We plot the learning curves for the OR1MP and EOR1MP algorithms on the MovieLens1M dataset in Figure 6. The results show that the linear convergence speed is preserved for different iteration numbers. However, the results under the same outer iterations depend on the accuracy of the power methods. This verifies our theoretical results. Our empirical results also suggest that in practice we need to run more than 5 iterations in the power method, as the learning curves for 5, 10, and 20 power method iterations are close to each other but are far away from the other two curves, especially for the EOR1MP algorithm.

7.3. Recovery on synthetic data. In this experiment, we use synthetic data to evaluate the recovery performance of different matrix completion algorithms. We generate a square $n \times n$ matrix $\mathbf{Y}$ of rank r as the ground truth. We construct $\mathbf{Y}$ by first generating a random matrix with i.i.d. entries drawn from the standard normal distribution and then setting its ith singular value to 2^{r-i+1}. Given this matrix $\mathbf{Y}$, we

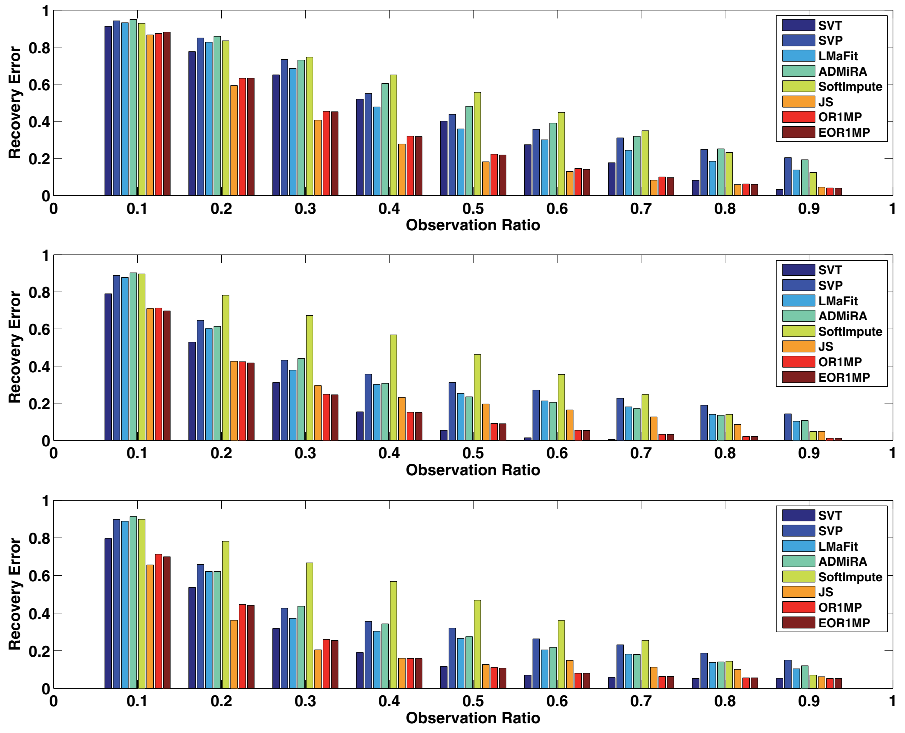

Fig. 7. *Comparison of recovery performance of different matrix completion algorithms with different percentages of observations: the three figures correspond to the results on three rank*-10 *random matrices of size* 50×50 *without noise (top figure), size* 100×100 *without noise (middle figure), and size* 100×100 *with Gaussion noise (bottom figure); the x-axis is the percentage of observations; the y-axis is the recovery error.*

sample a subset Ω of l entries uniformly at random as the observations. We run the experiment in two different settings—noise-free matrix completion and noisy matrix completion. We fix the rank of the ground truth matrices as $r = 10$ in all experiments. In the noise-free case, we use two different matrix sizes in the experiment: $n = 50$ and $n = 100$. In the noisy case, we use $n = 100$ with 5% Gaussion noise. The entries of the noise matrix are drawn from the standard normal distribution and are normalized to make the matrix Frobenius norm equal to $0.05\|\mathbf{Y}\|_F$. We evaluate the recovery performance of the algorithms based on the relative reconstruction error calculated as $\frac{\|\mathbf{Y}-\hat{\mathbf{Y}}\|_F}{\|\mathbf{Y}\|_F}$, with $\hat{\mathbf{Y}}$ as the reconstructed matrix. In the experiment, we fix the number of iterations to 200 for the JS algorithm. For OR1MP and EOR1MP, we stop the algorithm after 50 iterations. For other algorithms, we use the true rank $r = 10$ for the estimated matrix.

For each algorithm, we present its average result with 50 runs at different percentages of observations in Figure 7. We can observe from the figure that for most algorithms the recovery error decreases with an increasing number of observations. The proposed algorithms are very competitive in most cases, particularly when the observations are scarce.

Table 1
Image recovery results measured in terms of the PSNR.

Dataset	SVT	SVP	SoftImpute	LMaFit	ADMiRA	JS	OR1MP	EOR1MP
Barbara	**26.9635**	25.2598	25.6073	25.9589	23.3528	23.5322	26.5314	26.4413
Cameraman	25.6273	25.9444	26.7183	24.8956	26.7645	24.6238	**27.8565**	27.8283
Clown	**28.5644**	19.0919	26.9788	27.2748	25.7019	25.2690	28.1963	28.2052
Couple	23.1765	23.7974	26.1033	25.8252	25.6260	24.4100	**27.0707**	27.0310
Crowd	**26.9644**	22.2959	25.4135	26.0662	24.0555	18.6562	26.0535	26.0510
Girl	29.4688	27.5461	27.7180	27.4164	27.3640	26.1557	**30.0878**	30.0565
Goldhill	28.3097	16.1256	27.1516	22.4485	26.5647	25.9706	**28.5646**	28.5101
Lenna	**28.1832**	25.4586	26.7022	23.2003	26.2371	24.5056	28.0115	27.9643
Man	**27.0223**	25.3246	25.7912	25.7417	24.5223	23.3060	26.5829	26.5049
Peppers	25.7202	26.0223	26.8475	27.3663	25.8934	24.0979	**28.0781**	28.0723

7.4. Image recovery. In the image recovery experiments, we use the following benchmark test images: Barbara, Cameraman, Clown, Couple, Crowd, Girl, Goldhill, Lenna, Man, and Peppers.[4] The size of each image is 512×512. We randomly exclude 50% of the pixels in the image, and the remaining ones are used as the observations. As the image matrix is not guaranteed to be low rank, we use rank 50 for the estimated matrix for each experiment. In our OR1MP and EOR1MP algorithms, we stop the algorithms after 150 iterations. The JS algorithm does not explicitly control the rank, thus we fix its number of iterations to 2000. The numerical results in terms of the PSNR are listed in Table 1. We also present the images recovered by different algorithms for Lenna in Figure 8. The results show SVT, our OR1MP, and EOR1MP achieve the best numerical performance. However, our algorithm is much better than SVT for Cameraman, Couple, Peppers but only slightly worse than SVT for Lenna, Barbara, and Clown. Besides, our algorithm is much faster and more stable than SVT (SVT may diverge). For each image, EOR1MP uses around 3.5 seconds, but SVT consumes around 400 seconds. Image recovery needs a relatively higher approximation rank; both GECO and Boost fail to find a good recovery in most cases, so we do not include them in the result tables.

7.5. Recommendation. In the following experiments, we compare different matrix completion algorithms using large recommendation datasets: Jester [12] and MovieLens [31]. We use six datasets: Jester1, Jester2, Jester3, MovieLens100K, MovieLens1M, and MovieLens10M. The statistics of these datasets are given in Table 2. The Jester datasets were collected from a joke recommendation system. They contain anonymous ratings of 100 jokes from the users. The ratings are real values ranging from -10.00 to $+10.00$. The MovieLens datasets were collected from the MovieLens website.[5] They contain anonymous ratings of the movies on this web made by its users. For MovieLens100K and MovieLens1M, there are 5 rating scores (1–5), and for MovieLens10M there are 10 levels of scores with a step size 0.5 in the range of 0.5 to 5. In the following experiments, we randomly split the ratings into training and test sets. Each set contains 50% of the ratings. We compare the running time and the prediction result from different methods. In the experiments, we use 100 iterations for the JS algorithm, and for other algorithms we use the same rank for the estimated matrices; the values of the rank are $\{10, 10, 5, 10, 10, 20\}$ for the six corresponding datasets. We first show the running time of different methods in

[4]Images are downloaded from http://www.utdallas.edu/~cxc123730/mh_bcs_spl.html.
[5]http://movielens.umn.edu.

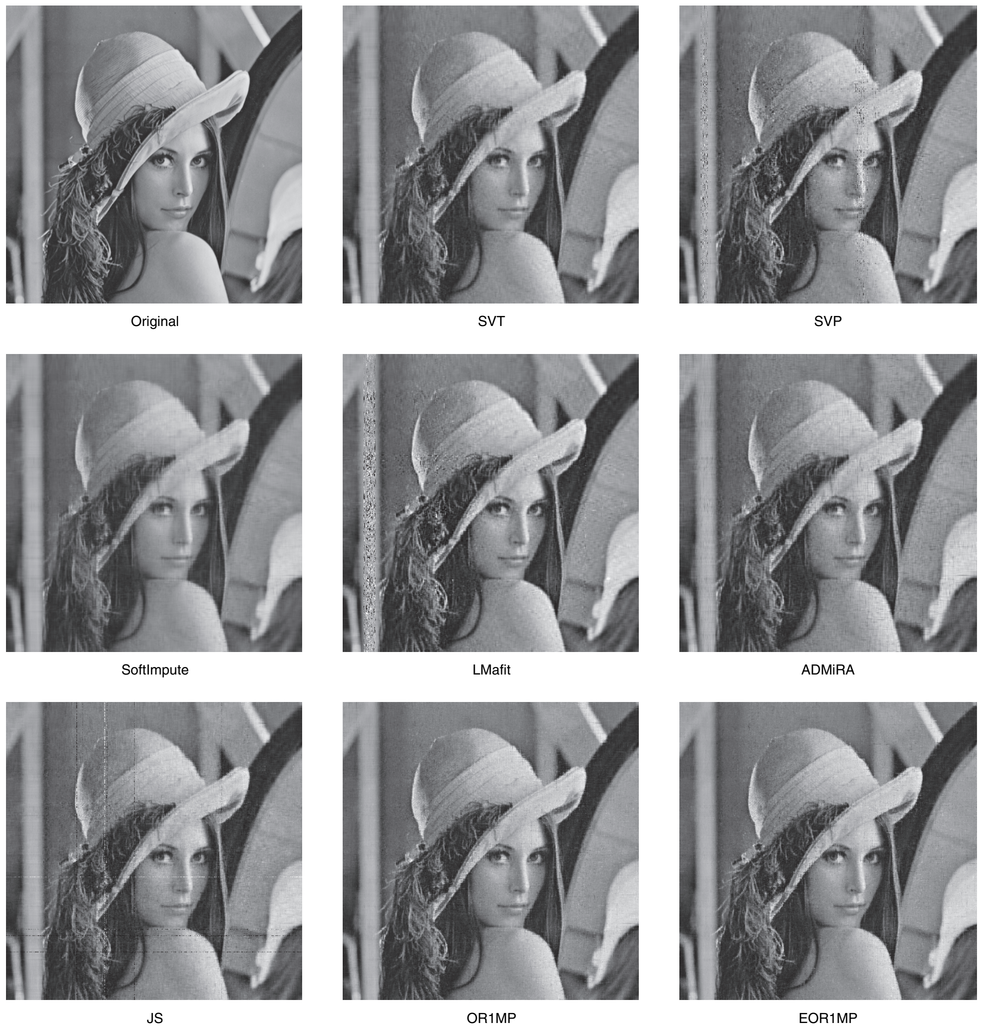

Fig. 8. *The original image and images recovered by different methods used on the Lenna image.*

Table 2
Characteristics of the recommendation datasets.

Dataset	# row	# column	# rating
Jester1	24983	100	10^6
Jester2	23500	100	10^6
Jester3	24983	100	6×10^5
MovieLens100k	943	1682	10^5
MovieLens1M	6040	3706	10^6
MovieLens10M	69878	10677	10^7

Table 3. The reconstruction results in terms of the RMSE are given in Table 4. We can observe from the above experiments that our EOR1MP algorithm is the fastest among all competing methods to obtain satisfactory results.

Table 3
The running time (measured in seconds). Boost fails on MovieLens10M.

Dataset	SVP	SoftImpute	LMaFit	Boost	JS	GECO	OR1MP	EOR1MP
Jester1	18.35	161.49	3.68	93.91	29.68	$> 10^4$	1.83	0.99
Jester2	16.85	152.96	2.42	261.70	28.52	$> 10^4$	1.68	0.91
Jester3	16.58	10.55	8.45	245.79	12.94	$> 10^3$	0.93	0.34
MovieLens100K	1.32	128.07	2.76	2.87	2.86	10.83	0.04	0.04
MovieLens1M	18.90	59.56	30.55	93.91	13.10	$> 10^4$	0.87	0.54
MovieLens10M	$> 10^3$	$> 10^3$	154.38	–	130.13	$> 10^5$	23.05	13.79

Table 4
Recommendation results measured in terms of the RMSE.

Dataset	SVP	SoftImpute	LMaFit	Boost	JS	GECO	OR1MP	EOR1MP
Jester1	4.7311	5.1113	4.7623	5.1746	4.4713	4.3680	4.3418	4.3384
Jester2	4.7608	5.1646	4.7500	5.2319	4.5102	4.3967	4.3649	4.3546
Jester3	8.6958	5.4348	9.4275	5.3982	4.6866	5.1790	4.9783	5.0145
MovieLens100K	0.9683	1.0354	1.2308	1.1244	1.0146	1.0243	1.0168	1.0261
MovieLens1M	0.9085	0.8989	0.9232	1.0850	1.0439	0.9290	0.9595	0.9462
MovieLens10M	0.8611	0.8534	0.8625	–	0.8728	0.8668	0.8621	0.8692

8. Conclusion.

In this paper, we propose an efficient and scalable low rank matrix completion algorithm. The key idea is to extend the OMP method from the vector case to the matrix case. We also propose a novel weight updating rule under this framework to reduce the storage complexity and make it independent of the approximation rank. Our algorithms are computationally inexpensive for each matrix pursuit iteration and find satisfactory results in a few iterations. Another advantage of our proposed algorithms is they have only one tunable parameter, which is the rank. It is easy to understand and to use by the user. This becomes especially important in large-scale learning problems. In addition, we rigorously show that both algorithms achieve a linear convergence rate, which is significantly better than the previous known results (a sublinear convergence rate). We also extend our proposed algorithm to a more general matrix sensing case and analyze its recovery guarantee under rank-restricted isometry property. We empirically compare the proposed algorithms with state-of-the-art matrix completion algorithms, and our results show that the proposed algorithms are more efficient than competing algorithms while achieving similar or better prediction performance. We plan to generalize our theoretical and empirical analysis to other loss functions in the future.

Appendix A. Inverse matrix update.

In our OR1MP algorithm, we use the least squares solution to update the weights for the rank-one basis matrices. In this step, we need to calculate $(\bar{\mathbf{M}}_k^T\bar{\mathbf{M}}_k)^{-1}$. To directly compute this inverse is computationally expensive, as the matrix $\bar{\mathbf{M}}_k$ has a large row size. We implement this efficiently using an incremental method. As

$$\bar{\mathbf{M}}_k^T\bar{\mathbf{M}}_k = [\bar{\mathbf{M}}_{k-1}, \dot{\mathbf{m}}_k]^T[\bar{\mathbf{M}}_{k-1}, \dot{\mathbf{m}}_k],$$

its inverse can be written in block matrix form:

$$(\bar{\mathbf{M}}_k^T\bar{\mathbf{M}}_k)^{-1} = \begin{bmatrix} \bar{\mathbf{M}}_{k-1}^T\bar{\mathbf{M}}_{k-1} & \bar{\mathbf{M}}_{k-1}^T\dot{\mathbf{m}}_k \\ \dot{\mathbf{m}}_k^T\bar{\mathbf{M}}_{k-1}^T & \dot{\mathbf{m}}_k^T\dot{\mathbf{m}}_k \end{bmatrix}^{-1}.$$

Then it is calculated by blockwise inverse as

$$\begin{bmatrix} \mathbf{A} + d\mathbf{A}\mathbf{b}\mathbf{b}^T\mathbf{A} & -d\mathbf{A}\mathbf{b} \\ -d\mathbf{b}^T\mathbf{A} & d \end{bmatrix},$$

where $\mathbf{A} = (\bar{\mathbf{M}}_{k-1}^T\bar{\mathbf{M}}_{k-1})^{-1}$ is the corresponding inverse matrix in the last step, $\mathbf{b} = \bar{\mathbf{M}}_{k-1}^T\dot{\mathbf{m}}_k$ is a vector with $|\Omega|$ elements, and $d = (\mathbf{b}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{b})^{-1} = 1/(\mathbf{b}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{b})$ is a scalar. $\bar{\mathbf{M}}_k^T\dot{\mathbf{y}}$ is also calculated incrementally by $[\bar{\mathbf{M}}_{k-1}^T\dot{\mathbf{y}}, \dot{\mathbf{m}}_k^T\dot{\mathbf{y}}]$, as $\dot{\mathbf{y}}$ is fixed.

REFERENCES

[1] A. Argyriou, T. Evgeniou, and M. Pontil, *Convex multi-task feature learning*, Mach. Learn., 73 (2008), pp. 243–272.

[2] F. Bach, *Consistency of trace norm minimization*, J. Mach. Learn. Res., 9 (2008), pp. 1019–1048.

[3] L. Balzano, R. Nowak, and B. Recht, *Online identification and tracking of subspaces from highly incomplete information*, in Proceedings of the Allerton Conference on Communication, Control and Computing, 2010.

[4] R. Bell and Y. Koren, *Lessons from the netflix prize challenge*, ACM SIGKDD Explorations, 9 (2007), pp. 75–79.

[5] J. Bennett and S. Lanning, *The netflix prize*, in Proceedings of KDD Cup and Workshop, 2007.

[6] J.-F. Cai, E. J. Candès, and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM J. Optim., 20 (2010), pp. 1956–1982.

[7] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2009), pp. 717–772.

[8] R. A. DeVore and V. N. Temlyakov, *Some remarks on greedy algorithms*, Adv. Comput. Math., 5 (1996), pp. 173–187.

[9] M. Dudík, Z. Harchaoui, and J. Malick, *Lifted coordinate descent for learning with trace-norm regularization*, in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS), 2012.

[10] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Res. Logist. Quart., 3 (1956), pp. 95–110.

[11] J. H. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, J. Statist. Software, 33 (2010), pp. 1–22.

[12] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, *Eigentaste: A constant time collaborative filtering algorithm*, Inform. Retrieval, 4 (2001), pp. 133–151.

[13] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[14] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, Springer-Verlag, New York, 2009.

[15] E. Hazan, *Sparse approximate solutions to semidefinite programs*, in Proceedings of the 8th Latin American Conference on Theoretical Informatics, 2008.

[16] Q. Huynh-Thu and M. Ghanbari, *Scope of validity of psnr in image/video quality assessment*, Electron. Lett., 44 (2008), pp. 800–801.

[17] M. Jaggi and M. Sulovský, *A simple algorithm for nuclear norm regularized problems*, in Proceedings of the 27th International Conference on Machine Learning (ICML), 2010, pp. 471–478.

[18] P. Jain, R. Meka, and I. S. Dhillon, *Guaranteed rank minimization via singular value projection*, Adv. Neural Inf. Process. Syste. 22 (2010), pp. 937–945.

[19] P. Jain, P. Netrapalli, and S. Sanghavi, *Low-rank matrix completion using alternating minimization*, in Proceedings of the 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC), 2013, pp. 665–674.

[20] S. Ji and J. Ye, *An accelerated gradient method for trace norm minimization*, in Proceedings of the 26th International Conference on Machine Learning (ICML), 2009, pp. 457–464.

[21] R. Keshavan and S. Oh, *Optspace: A Gradient Descent Algorithm on the Grassmann Manifold for Matrix Completion*, http://arxiv.org/abs/0910.5260 (2009).

[22] Y. Koren, *Factorization meets the neighborhood: A multifaceted collaborative filtering model*, in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2008.

[23] Y. Koren, R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems*, Computer, 42 (2009), pp. 30–37.

[24] M.-J. Lai, Y. Xu, and W. Yin, *Improved iteratively reweighted least squares for unconstrained smoothed ℓ_q minimization*, SIAM J. Numer. Anal., 51 (2013), pp. 927–957.

[25] K. Lee and Y. Bresler, *Admira: atomic decomposition for minimum rank approximation*, IEEE Trans. Inform. Theory, 56 (2010), pp. 4402–4416.

[26] E. Liu and T. N. Temlyakov, *The orthogonal super greedy algorithm and applications in compressed sensing*, IEEE Trans. Inform. Theory, 58 (2012), pp. 2040–2047.

[27] Y.-J. Liu, D. Sun, and K.-C. Toh, *An implementable proximal point algorithmic framework for nuclear norm minimization*, Math. Program., 133 (2012), pp. 399–436.

[28] Z. Lu and Y. Zhang, *Penalty Decomposition Methods for Rank Minimization*, http://arxiv.org/abs/1008.5373 (2010).

[29] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and bregman iterative methods for matrix rank minimization*, Math. Program., 128 (2011), pp. 321–353.

[30] R. Mazumder, T. Hastie, and R. Tibshirani, *Spectral regularization algorithms for learning large incomplete matrices*, J. Mach. Learn. Res., 99 (2010), pp. 2287–2322.

[31] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, *MovieLens unplugged: Experiences with an occasionally connected recommender system*, in Proceedings of the 8th International Conference on Intelligent User Interfaces, 2003, pp. 263–266.

[32] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre, *Low-rank optimization with trace norm penalty*, SIAM J. Optim., 23 (2013), pp. 2124–2149.

[33] D. Needell and J. A. Tropp, *Cosamp: Iterative signal recovery from incomplete and inaccurate samples*, Comm. ACM, 53 (2010), pp. 93–100.

[34] S. Negahban and M. Wainwright, *Estimation of (near) low-rank matrices with noise and high-dimensional scaling*, in Proceedings of the 27th International Conference on Machine Learning (ICML), 2010.

[35] Y. C. Pati, R. Rezaiifar, Y. C. P. R. Rezaiifar, and P. S. Krishnaprasad, *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition*, in Proceedings of the 27th Annual Asilomar Conference on Signals, Systems, and Computers, 1993, pp. 40–44.

[36] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501.

[37] B. Recht and C. Ré, *Parallel stochastic gradient algorithms for large-scale matrix completion*, Math. Program. Comput., 5 (2013), pp. 201–226.

[38] S. Shalev-Shwartz, A. Gonen, and O. Shamir, *Large-scale convex minimization with a low-rank constraint*, in Proceedings of the 28th International Conference on Machine Learning (ICML), 2011, pp. 329–336.

[39] S. Shalev-Shwartz and A. Tewari, *Stochastic methods for l1 regularized loss minimization*, in Proceedings of the 26th International Conference on Machine Learning (ICML), 2009, pp. 929–936.

[40] N. Srebro, J. Rennie, and T. Jaakkola, *Maximum-margin matrix factorizations*, Adv. Neural Inf. Process. Syst., 17 (2004), pp. 1329–1336.

[41] V. N. Temlyakov, *Greedy approximation*, Acta Numer., 17 (2008), pp. 235–409.

[42] A. Tewari, P. Ravikumar, and I. S. Dhillon, *Greedy algorithms for structurally constrained high dimensional problems*, Adv. Neural Inf. Process. Syst., 24 (2011), pp. 882–890.

[43] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. R. Stat. Soc. Ser. B, 58 (1994), pp. 267–288.

[44] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optim., 6 (2010), pp. 615–640.

[45] J. A. Tropp, *Greed is good: Algorithmic results for sparse approximation*, IEEE Trans. Inform. Theory, 50 (2004), pp. 2231–2242.

[46] Z. Wen, W. Yin, and Y. Zhang, *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*, Math. Program. Comput., 4 (2012), pp. 333–361.

[47] T. T. Wu and K. Lange, *Coordinate descent algorithms for lasso penalized regression*, Ann. Appl. Stat., 2 (2008), pp. 224–244.

[48] S. Yun and K.-C. Toh, *A coordinate gradient descent method for l1-regularized convex minimization*, Comput. Optim. Appl., 48 (2011), pp. 273–307.

[49] X. Zhang, Y. Yu, and D. Schuurmans, *Accelerated training for matrix-norm regularization: A boosting approach*, Adv. Neural Inf. Process. Syst., 25 (2012), pp. 2906–2914.