Active Subspace:  
Towards Scalable Low-Rank Learning  
Guangcan Liu‡† and Shuicheng Yan†  
†Electrical and Computer Engineering, National University of Singapore.  
‡Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.  
Keywords: Large-Scale Optimization, Nuclear Norm Regularization, Augmented Lagrange Multiplier, Robust PCA, Low-Rank Representation.

Abstract

We address the scalability issues in low-rank matrix learning problems. Usually, these problems resort to solving nuclear norm regularized optimization problems (NNROPs), which often suffer from high computational complexities if based on existing solvers, especially under large-scale settings. Based on the fact that the optimal solution matrix to an NNROP is often low-rank, we revisit the classic mechanism of low-rank matrix factorization, based on which we present an active subspace algorithm for efficiently solving NNROPs by transforming large-scale NNROPs into small-scale problems. The transformation is achieved by factorizing the large-size solution matrix into the product of a small-size orthonormal matrix (active subspace) and another small-size matrix. Although such a transformation generally leads to non-convex problems, we show that suboptimal solution can be found by the augmented Lagrange alternating direction method. For the robust PCA (RPCA) (Candès et al., 2009) problem, which is a typical example of NNROPs, theoretical results verify sub-optimality of the solution produced by our algorithm. For the general NNROPs, we empirically show that our algorithm significantly reduces the computational complexity without loss of optimality.
1 Introduction

The problem of low-rank matrix learning (Bach, 2008) has been receiving broad attention in areas such as lossy data compression (Candès et al., 2009; Candès & Plan, 2010; Liu et al., 2011), collaborative filtering (Weimer et al., 2007), image processing (Zhang et al., 2010), text analysis (Min et al., 2010) and multimedia analysis (Zhu et al., 2010). The technique of nuclear norm (Fazel, 2002) minimization is a popular choice for low-rank matrix learning, due to its strong guarantees in theory (Bach, 2008; Candès & Recht, 2009; Liu et al., 2011) and competitive performance in practice (Liu et al., 2010; Zhang et al., 2010). However, it is nontrivial to solve the nuclear norm regularized optimization problems (NNROPs), which can be generally formulated as follows 1:

\[
\min_{X, E} \|X\|_* + \lambda \|E\|_{\ell}, \quad \text{s.t.} \quad D = A(X) + E,
\]

where \(\|\cdot\|_*\) denotes the nuclear norm, also known as the trace norm or Ky Fan norm (sum of the singular values), \(X \in \mathcal{R}^{m \times n}\) and \(E \in \mathcal{R}^{m' \times n'}\) are unknown matrices to learn, \(D \in \mathcal{R}^{m' \times n'}\) is a given matrix, \(A : \mathcal{R}^{m \times n} \rightarrow \mathcal{R}^{m' \times n'}\) is a linear operator, \(\lambda > 0\) is a weighting parameter and \(\|\cdot\|_{\ell}\) generally denotes certain measurement for characterizing the loss term \(E = D - A(X)\). The above formulation involves a wide range of problems such as matrix completion (Candès & Recht, 2009) (\(A\) is a sampling operator, and \(\|\cdot\|_F^2\) is chosen for characterizing \(E\)), robust principal component analysis (RPCA) (Candès et al., 2009; Wright et al., 2009) (\(A\) is the identity operator, and \(\|\cdot\|_1\) is chosen for \(E\)) and low-rank representation (LRR) (Liu et al., 2010) (\(A(X)\) is defined by \(A(X) = AX\) with \(A\) being a given matrix, and \(\|\cdot\|_{2,1}\) is chosen for \(E\)).

Problem (1) is convex and can be solved by various algorithms such as semi-definite programming (SDP) (Chandrasekaran et al., 2009; Jaggi & Sulovský, 2010), accelerated proximal gradient (APG) (Tseng, 2008) and augmented Lagrange multipliers (ALM) (Lin et al., 2009; Zhang, 2010). However, it is not easy to solve NNROPs efficiently, especially for large-scale cases. For example, the standard SDP solver has a complexity of \(O(n^6)\) (assume \(m = n\)), which is too expensive even for small-size matrices (e.g., \(200 \times 200\)). To relieve this issue, Lin et al. (2009) introduce the exact

---

1More generally, NNROPs are expressed as \(\min_X \|X\|_* + \lambda f(x)\), where \(f(x)\) is a convex function. In this work, we are particularly interested in the form (1), which has covered a wide range of problems.
and inexact ALM methods that achieve promising performances in solving the RPCA problem, which is a typical example of NNROPs. In (Liu et al., 2011, 2010), it is further shown that ALM can solve the LRR problem, which is more general than RPCA.

Generally, ALM possesses a complexity of $O(n^3)$ when solving the problem (1). This complexity is, unfortunately, still too high to afford for learning the large-size matrices, e.g., the word frequency matrices in document analysis (Min et al., 2010) and the tag matrices in multimedia analysis (Zhu et al., 2010). In this paper we thus aim at solving the large-scale NNROPs efficiently. In particular, we are interested in the cases where the loss term $\|E\|_\ell$ is non-differentiable (e.g., the $\|\cdot\|_1$ in RPCA and $\|\cdot\|_{2,1}$ in LRR).

Such optimization problems are more challenging than the smoothing cases addressed in (Jaggi & Sulovský, 2010; Shalev-Shwartz et al., 2011; Tomioka et al., 2010; Yang & Yuan, 2010).

Usually, the high computational complexity of ALM (or APG) is caused by the computation of singular value thresholding (SVT) (Cai et al., 2010), which involves the singular value decomposition (SVD) of a matrix. For efficiency, Cai & Osher (2010) have established the so called fast-SVT, which is to compute SVT without SVD. Fast-SVT can be several times faster than SVT, however, its computational complexity is still $O(n^3)$ (assume $m = n$). So this method is not very competitive for large-scale learning. For some specific problems such as matrix completion, where the only time-consuming step is to calculate the SVD of a matrix, ALM algorithms may be made scalable by partial-SVD techniques (Lin et al., 2011; Halko et al., 2009; Williams & Seeger, 2000).

However, partial-SVD may not generally reduce the complexity of solving NNROPs, which essentially involve $O(n^2)$ unknowns.

For generally reducing the computational complexity, the long discussed low-rank factorization (Burer & Monteiro, 2005; Srebro et al., 2004; Weimer et al., 2007; Shen et al., 2011) mechanism provides us a straightforward schema; that is, factorizing a large-size matrix into the product of two small-size matrices. Let $X^o$ be an optimal solution to problem (1). Due to the effect of nuclear norm regularization, $X^o$ is generally of low-rank, i.e., $X^o$ can be factorized into the product of two small-size matrices:

$$X^o = Q^o J^o, \quad Q^o \in \mathcal{R}^{m \times r}, \quad J^o \in \mathcal{R}^{r \times n},$$

where $r \ll m$ and $r \ll n$. Based on this observation, one may transform the convex
problem (1) into the following non-convex problem by replacing $X$ with $QJ$:

$$\min_{Q,J,E} \|QJ\|_* + \lambda \|E\|_\ell, \quad \text{s.t.} \quad D = A(QJ) + E.$$  \hspace{1cm} (2)

Since $nr + nr \ll mn$, the number of variables has been reduced largely. Nevertheless, it is even harder to solve (2) than (1). More precisely, to solve problem (2) by making use of SVT, one may need to transform it back into the following problem:

$$\min_{X,Q,J,E} \|X\|_* + \lambda \|E\|_\ell, \quad \text{s.t.} \quad X = QJ, D = A(X) + E,$$

which is even more complex than the original problem (1). So, it is infeasible to reduce the complexity of NNROPs by directly applying the low-rank factorization mechanism.

To efficiently find an optimal solution to a NNROP, we propose the so-called active subspace algorithm, which is a revisit to the low-rank factorization mechanism discussed above. Notice that the difficulty of solving (2) is mainly caused by the term $\|QJ\|_*$, which makes the SVT operator inapplicable. Fortunately, this issue can be eliminated by enforcing $Q$ (or $J$) to be column-orthonormal, because $\|QJ\|_* = \|J\|_*$ for $Q^TQ = I$ ($I$ denotes the identity matrix), resulting in an optimization problem constrained on Stiefel manifold (Edelman et al., 1999):

$$\min_{Q,J,E} \|J\|_* + \lambda \|E\|_\ell, \quad \text{s.t.} \quad D = A(QJ) + E, Q^TQ = I.$$ \hspace{1cm} (3)

Suppose $(Q^*, J^*, E^*)$ is a solution to the above problem, we could use $(X = Q^*J^*, E = E^*)$ to recover a solution to the original problem (1). So, optimizing (3) is equal to optimizing (1) with the constraint of only invoking a low-rank subspace. Since any optimal solution $X^o$ (assume $\text{rank}(X^o) \leq r$) to (1) can be factorized into $X^o = Q^oJ^o$ with $(Q^o)^TQ^o = I$, an optimal solution to (1) may be found efficiently by optimizing (3), namely seeking the optimal subspace (indicated by $Q^o$) and the corresponding coefficient matrix (indicated by $J^o$) simultaneously. Generally, the underlying principle here is similar to the well-known active set (Nocedal & Wright, 2006). So we call our mechanism as active subspace. While in principle our active subspace approach appears to be very similar to the low-rank factorization mechanism, there is a key difference: the two matrices ($Q$ and $J$) are treated differently and $Q$ is enforced to be orthonormal. This technical detail, as will be seen, can greatly facilitate the optimization and theoretical analysis.
It is easy to see that a globally optimal solution to (3) exactly recovers an optimal solution to the original problem (1), provided that the rank of an optimal solution to (1) is not greater than $r$. So, it is critical to remedy the non-convex nature of problem (3). To this end, the inexact ALM method, which is usually called alternating direction method (ADM), should be a promising choice: ADM performs well in solving low-rank factorization problems, in terms of both theoretical and empirical evaluations (Burer & Monteiro, 2005; Shen et al., 2011). It is hence natural to anticipate that ADM can well solve our active subspace problem (3). For the RPCA problem, which is a typical example of NNROPs, we devise an ADM based algorithm that obtains a suboptimal solution to (3). For the general case (1), we also establish an ADM based algorithm to solve the transformed problem (3). Empirically we show that the algorithm can reduce the computational complexity of LRR without loss of optimality. In summary, the main contribution of this work is

- We establish an efficient algorithm for solving the NNROPs that contains $O(mn)$ unknown variables to solve. The computational complexity of our algorithm is only $O(mnr)$ ($r \ll m, r \ll n$), which is a linear complexity and an affordable cost in large-scale learning.

The rest of this paper is organized as follows. Section 2 summarizes the mathematical notations. Section 3 introduces the theories and algorithms on active subspace. Section 4 presents the experimental results and Section 5 concludes this paper.

## 2 Summary of Notations

Matrices and scalars are represented with capital symbols and Greek letters, respectively. In particular, $I$ is used to denote the identity matrix. A variety of norms on matrices will be used. The matrix $\ell_1$-norm and $\ell_2/\ell_1$-norm are denoted by $\|M\|_1 = \sum_{i,j} |(M)_{ij}|$ and $\|M\|_{2,1} = \sum_j \sqrt{\sum_i ((M)_{ij})^2}$, respectively, where $(M)_{ij}$ is the $(i,j)$-th entry of a matrix. The $\ell_{\infty}$-norm is denoted by $\|M\|_{\infty} = \max_{i,j} |(M)_{ij}|$. The Frobenious norm and nuclear norm are denoted by $\|M\|_F$ and $\|M\|_*$, respectively. In particular, $\|M\|$ denotes the spectral norm of a matrix, i.e., the largest singular value of a matrix. The Euclidean inner product between two matrices is $\langle M, N \rangle = \text{tr} (M^T N)$, where $M^T$ is the transpose of a matrix and $\text{tr} (\cdot)$ is the trace of a square matrix.
Further, some matrix calculation operators will be used in this paper. For a given matrix \( M \), we use the thin SVD \(^2\) to define \( \mathcal{P}[M] = UV^T \) (note that \( \mathcal{P}[M] \) may be not unique for a given \( M \)). When \( M \in \mathcal{R}^{m \times n} \) has thin columns (i.e., \( m \geq n \)), it can be verified that \( \mathcal{P}[M] \) is always column-orthonormal (i.e., \( (\mathcal{P}[M])^T \mathcal{P}[M] = I \)). For ease of presentation, we denote the singular value thresholding (SVT) (Cai et al., 2010) operator by \( S_\delta[M] = UH_\delta[\Sigma]V^T \), where \( M = U\Sigma V^T \) is the skinny SVD of \( M \) and \( H_\delta[\Sigma] = \max(0, \Sigma - \delta) + \min(0, \Sigma + \delta) \) is the shrinkage operator (Lin et al., 2009).

Finally, for a convex function \( f : \mathcal{R}^{m \times n} \rightarrow \mathcal{R} \), we say that \( Y \) is a subgradient of \( f \) at \( X \), denoted as \( Y \in \partial f(X) \), if \( f(X') \geq f(X) + \langle X' - X, Y \rangle, \forall X' \).

### 3 Algorithms and Analysis

In this section, we introduce our algorithm for solving problem (3). For the ease of exploration, we shall begin with the RPCA problem, which is a typical example of (1) with \( A \) being the identity operator.

#### 3.1 Solving RPCA by Active Subspace

Given a data matrix \( D \in \mathcal{R}^{m \times n} \), RPCA is to decompose \( D \) into a low-rank part and a sparse part by minimizing:

\[
\min_{X, E} \|X\|_* + \lambda \|E\|_1, \quad \text{s.t.} \quad D = X + E. \tag{4}
\]

RPCA is emerging as a powerful tool for various applications, such as document analysis (Min et al., 2010), multimedia analysis (Zhu et al., 2010) and image processing (Zhang et al., 2010). However, when both \( m \) and \( n \) are both large (this happens fre-

---

\(^2\)For an \( m \times n \) matrix \( M \) (without loss of generality, assuming \( m \leq n \)), its SVD is defined by \( M = U[\Sigma, 0]V^T \), where \( U \) and \( V \) are orthogonal matrices and \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_m) \) with \( \{\sigma_i\}_{i=1}^m \) being singular values. The SVD defined in this way is also called the full SVD. If we only calculate the \( m \) column vectors of \( V \), i.e., \( M = U\Sigma V^T \) with \( U \in \mathcal{R}^{m \times m}, \Sigma \in \mathcal{R}^{m \times m}, \) and \( V \in \mathcal{R}^{n \times m} \), the simplified form is called the thin SVD. If we only keep the positive singular values, the reduced form is called the skinny SVD. For a matrix \( M \) of rank \( r \), its skinny SVD is computed by \( M = U_r\Sigma_r V_r^T \), where \( \Sigma_r = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r) \) with \( \{\sigma_i\}_{i=1}^r \) being positive singular values. More precisely, \( U_r \) and \( V_r \) are formed by taking the first \( r \) columns of \( U \) and \( V \), respectively.
quently in several domains, e.g., document and multimedia), it is not easy to solve the optimization problem (4) efficiently, as there are $O(mn)$ unknowns to solve.

### 3.1.1 Our Algorithm

The RPCA problem (4) can be solved with a complexity of $O(mnr)$ by our active subspace algorithm, which suggests to solve the following non-convex problem:

$$
\begin{align*}
\min_{Q,J,E} & \|J\|_* + \lambda \|E\|_1, \\
s.t. & D = QJ + E, Q^T Q = I, \tag{5}
\end{align*}
$$

where the sizes of $Q$ (active subspace), $J$ and $E$ are $m \times r$, $r \times n$ and $m \times n$, respectively. Although the above problem is non-convex, it is possible to obtain an accurate solution by ADM (i.e., inexact ALM). The ADM based algorithm is outlined in Algorithm 1, which is to minimize the following augmented Lagrange function:

$$
\mathcal{L}(Q, J, E, Y, \mu) = \|J\|_* + \lambda \|E\|_1 + \langle Y, D - QJ - E \rangle + \frac{\mu}{2} \|D - QJ - E\|_F^2,
$$

with respect to $Q$ ($Q^T Q = I$), $J$ and $E$, respectively, by fixing the other variables, and then updating the Lagrange multiplier $Y$ and the penalty parameter $\mu$.

Suppose the algorithm needs $n_s$ iterations to converge, the entire complexity of Algorithm 1 is

$$
O((r^3 + r^2(m + n) + rmn)n_s).
$$

So, the complexity of Algorithm 1 with regard to the matrix sizes is only $O(mn)$ (assuming $r \ll m$ and $r \ll n$), which illustrates that our algorithm can efficiently handle large-size matrices. Notice that the overall complexity of RPCA can be also reduced to $O(mnr)$ by the partial-SVD techniques (Halko et al., 2009; Williams & Seeger, 2000). However, it is worthy noting that our algorithm can be still several times faster than partial-SVD. This is because the major computation of our algorithm is actually the product of an $m \times r$ matrix and a $r \times n$ matrix, which is much faster than calculating the rank-$r$ partial-SVD of an $m \times n$ matrix, although their complexities are both $O(mnr)$.

### 3.1.2 Analysis

In this subsection, we show the theoretical properties of Algorithm 1. Without any assumptions, we prove that the algorithm can stop within a finite number of iterations,
**Algorithm 1** Solving the PRCA problem by Active Subspace (i.e., solving problem (5) by ADM)

**Parameters:** \( \rho > 1, \varepsilon > 0. \)

**Initialize:** \( Q_0 = 0, J_0 = 0, E_0 = 0, Y_0 = 0, \mu_0 = 1/\|D\| \) and \( k = 0. \)

**while** not converged **do**

1. Update \( Q \) by

\[
Q_{k+1} = \arg\min_{Q \in S} \|D - QJ_k - E_k + \frac{Y_k}{\mu_k}\|_F^2
\]

\[
= P[(D - E_k + \frac{Y_k}{\mu_k})J_k^T],
\]

where the solution is given by (Higham, 1995).

2. Update \( J \) by

\[
J_{k+1} = \arg\min_J \|J\|_* + \frac{\mu_k}{2} \|D - Q_{k+1}J - E_k + \frac{Y_k}{\mu_k}\|_F^2
\]

\[
= S_{1/\mu_k}(Q_{k+1}^T(D - E_k + \frac{Y_k}{\mu_k})].
\]

3. Update \( E \) by

\[
E_{k+1} = \arg\min_E \lambda \|E\|_1 + \frac{\mu_k}{2} \|D - Q_{k+1}J_{k+1} - E + \frac{Y_k}{\mu_k}\|_F^2
\]

\[
= \mathcal{H}_{\lambda/\mu_k}(D - Q_{k+1}J_{k+1} + \frac{Y_k}{\mu_k}].
\]

4. Update the multipliers by

\[
\hat{Y}_{k+1} = Y_k + \mu_k(D - Q_{k+1}J_{k+1} - E_{k+1}),
\]

\[
\hat{Y}_{k+1} = Y_k + \mu_k(D - Q_{k+1}J_{k+1} - E_k),
\]

\[
\tilde{Y}_{k+1} = Y_k + \mu_k(D - Q_{k+1}J_k - E_k).
\]

5. Update the parameter \( \mu \) by \( \mu_{k+1} = \rho \mu_k. \)

6. Compute \( \mathcal{E} = \max(\|Y_{k+1} - Y_k\|_\infty, \|\hat{Y}_{k+1} - Y_{k+1}\|_\infty, \|\tilde{Y}_{k+1} - \hat{Y}_{k+1}\|_\infty) \), and check for convergence by: If \( \mathcal{E} < \varepsilon \), stop; else, \( k = k + 1. \)

**end while**

**Output:** \((Q_{k+1}, J_{k+1}, E_{k+1})\), referred to as \((Q^*, J^*, E^*)\) for highlight.
as shown in the following theorem (the detailed proof can be found in the appendix).

**Lemma 3.1** The sequences $\{J_k\}$, $\{E_k\}$ and $\{Q_kJ_k\}$ produced by Algorithm 1 are Cauchy sequences. The produced solution $(Q^*, J^*, E^*)$ is feasible to problem (5), in a sense that

$$\|D - (Q^*J^* + E^*)\|_\infty < \varepsilon.$$  

Provide that the parameter $\varepsilon$ is small enough, i.e., $\varepsilon \to 0$, then $D = Q^*J^* + E^*$. In our experiments, we set $\varepsilon = 10^{-8}$. Note here that $Q^*, J^*, E^*$ do not refer to the accumulation points of the sequences $\{Q_k\}, \{J_k\}, \{E_k\}$, but instead being defined by Algorithm 1:

$$Q^* = Q_{k^*+1}, \quad J^* = J_{k^*+1} \quad E^* = E_{k^*+1},$$  

where $k^*$ is the number of iterations needed by Algorithm 1 to stop.

The above lemma only ensures the feasibility of the produced solution. Actually, it could be possible to prove the local optimality of the produced solution. Shen et al. (2011) applied ADM to a low-rank factorization problem, which is similar to (5) but drops the nuclear norm regularization term. They show that the ADM based algorithm can achieve local optimality. In this paper, we shall investigate the *sub-optimality*, which concerns the gap between the true minimum and the minimal objective function value achieved by Algorithm 1.

Let $Y^*$ (resp. $\hat{Y}^*$) denote the Lagrange multiplier $Y_{k^*+1}$ (resp. $\hat{Y}_{k^*+1}$) associated with $(Q^*, J^*, E^*)$, i.e.,

$$Y^* = Y_{k^*+1}, \quad \hat{Y}^* = \hat{Y}_{k^*+1},$$  

where $k^*$ is the number of iterations needed by Algorithm 1 to stop. Then the following lemma can be proven (see the appendix for details).

**Lemma 3.2** For the solution $(Q^*, J^*, E^*)$ produced by Algorithm 1, we have that

$$\|J\|_* + \lambda \|E\|_1 \geq \|J^*\|_* + \lambda \|E^*\|_1 + \langle Y^* - \hat{Y}^*, E - E^* \rangle - mn\varepsilon$$

holds for any feasible solution $(Q, J, E)$ to (5).
Figure 1: **Influence of the parameter** $\rho$. (a) Illustrating that $Y^* - \hat{Y}^*$ can vanish by setting the parameter $\rho$ to be relatively small. Here, “gap” is defined by: $\text{gap} = \|Y^* - \hat{Y}^*\|_\infty$. (b) The number of iterations required for convergence vs the parameter $\rho$. These results are obtained from running Algorithm 1 on synthetic data, averaged over 50 runs.

To reach the global optimality of (5) based on the above lemma, it is needed to show that the term $\langle Y^* - \hat{Y}^*, E - E^* \rangle$ vanishes. In theory, the proof procedures of Theorem 3.1 and Lemma 3.2 (see the appendix) simply conclude that

$$\|Y^* - \hat{Y}^*\|_\infty \leq \|Y^*\|_\infty + \|\hat{Y}^*\|_\infty \leq 1 + \lambda,$$

which means that $Y^* - \hat{Y}^*$ is bounded. Actually, our extensive numerical experiments suggest that $Y^* - \hat{Y}^*$ can vanish by setting the parameter $\rho$ to be relatively small, as shown in Fig.1(a) \(^3\). Hence, it is appropriate to assume that $\|Y^* - \hat{Y}^*\|_\infty$ is small. Let

$$\varepsilon_1 = \|Y^* - \hat{Y}^*\|_\infty, \quad (8)$$

then we have the following theorem that shows the sub-optimality of the produced solution (the proof is presented in the appendix).

---

\(^3\)Nevertheless, as shown in Fig.1(b), the algorithm is less efficient while using smaller $\rho$. So ones could choose this parameter by trading off between efficiency and optimality. Here, we introduce a heuristic technique that modifies Step 5 of Algorithm 1 into:

$$\mu_{k+1} = \min(10^6, \rho \mu_k).$$

In this way, it will be safe to use relatively large $\rho$. 

---
**Theorem 3.1** Let $f^0$ be the globally optimal objective function value of (5), and $f^* = \|J^*\|_* + \lambda \|E^*\|_1$ be the objective function value produced by Algorithm 1. We have that

$$f^* \leq f^0 + c_1 \varepsilon_1 + mn \varepsilon,$$

where $c_1$ is a constant defined by

$$c_1 = \lambda \|D\|mn \left(\frac{\rho(1 + \rho)}{\rho - 1} + \frac{1}{2(\rho \varepsilon)} \right) + \|D\|_1.$$

Note here that the value of $\varepsilon$ can be set to be arbitrarily small. Empirically we have also shown that $\varepsilon_1$ is able to vanish (see Fig.1(a)), although it is not easy to prove this (it could be proven in theory that $\varepsilon_1 < 1 + \lambda$).

For the solution $(Q^*, J^*, E^*)$ produced by Algorithm 1, a solution to the original RPCA problem (4) can be recovered by constructing $X^* = Q^*J^*$. Following Theorem 3.1, the following theorem can be proven (the proof is shown in the appendix).

**Theorem 3.2** Suppose $(X^o, E^o)$ is an optimal solution to the original RPCA problem (4), rank $(X^o) = r_0$ and $f^o = \|X^o\|_* + \lambda \|E^o\|_1$. Let $f^* = \|Q^*J^*\|_* + \lambda \|E^*\|_1$ be the objective function value recovered by Algorithm 1 with parameter $r > 0$. We have that

$$f^0 \leq f^* \leq f^o + c_1 \varepsilon_1 + mn \varepsilon + (\lambda \sqrt{mn} - 1) \sigma_{r+1} \max(r_0 - r, 0),$$

where $\sigma_1 \geq \sigma_2 \geq \cdots$, are the singular values of $X^o$.

When the parameter $r$ is chosen to be higher than the rank of an optimal solution to the RPCA problem, the above theorem directly concludes that

$$f^o \leq f^* \leq f^o + c_1 \varepsilon_1 + mn \varepsilon,$$

which implies that our algorithm is able to improve the efficiency of RPCA without notable loss of optimality.

**3.1.3 Connections to Previous Results**

Burer & Monteiro (2005) addressed the scalability issues of SDP:

$$\min_X \text{tr} (CX), \ s.t. \ A(X) = 0, X \succeq 0$$

(9)
which is more general than RPCA. To speed up the optimization procedure, they utilized the low-rank matrix factorization to reduce the number of unknowns:

$$\min_R \text{tr} \left( CRR^T \right), \quad \text{s.t.} \quad A(RR^T) = 0. \quad (10)$$

They proved that the locally optimal solution to the above transformed problem is globally optimal to the original SDP, provided that the rank of a solution to the original problem (9) is smaller than \( r \). With some assumptions, they further proved that the locally optimal solution of (10) can be found by the ADM algorithm. This implies that their algorithm is able to reduce the computational costs while preserving the global optimality. By assuming \( \varepsilon = \varepsilon_1 = 0 \) and \( r \leq r_0 \), Theorem 3.2 implies that our algorithm can also achieve global optimality. Indeed, the assumption \( \varepsilon = \varepsilon_1 = 0 \) is similar to those made by Burer & Monteiro (2005). For example, their assumption of \( \lim_{k \to \infty} A(R_kR_k^T) = 0 \) has the same role as our \( \varepsilon = 0 \), where \( \{R_k\} \) is the solution sequence produced by their algorithm.

Note that it would be inefficient for the approach of Burer & Monteiro (2005) to solve the RPCA problem: for a RPCA problem with \( O(mn) \) unknowns, the number of unknowns in its corresponding SDP is \( O(m^2n^2) \).

### 3.2 Solving General NNROPs by Active Subspace

**Algorithm 2** Solving NNROPs by Active Subspace (i.e., solving problem (3) by ADM)

**Parameters:** \( \rho > 1 \) and \( \tau > 0 \).

**Initialize:** \( Q_0 = 0, J_0 = 0, E_0 = 0, Y_0 = 0, \mu_0 = 1/\|D\| \) and \( k = 0 \).

**while** not converged **do**

1. \( Q_{k+1} = P[(Q_kJ_k - \tau A^T(A(Q_kJ_k) - (D - E_k + Y_k/\mu_k))]J_k^T]. \)
2. \( J_{k+1} = S_{\tau/\mu_k}[J_k + \tau Q_{k+1}^T(A^T(A(Q_{k+1}J_k) - (D - E_k + Y_k/\mu_k))]. \)
3. \( E_{k+1} = \arg \min_E \lambda \|E\|_1 + \frac{\mu_k}{2} \|E - (D - A(Q_{k+1}J_{k+1}) + Y_k/\mu_k)\|_F^2. \)
4. \( Y_{k+1} = Y_k + \mu_k(D - A(Q_{k+1}J_{k+1}) - E_{k+1}). \)
5. \( \mu_{k+1} = \min(10^6, \rho\mu_k). \)
6. **If** \( \|D - A(Q_{k+1}J_{k+1}) - E_{k+1}\|_F < 10^{-8}\|D\|_F, \text{ stop; else, } k = k + 1. \)**

end **while**

For the general case (1), its transformed problem (3) can be solved in a similar way as RPCA. The update rules are presented in Algorithm 2, which is to minimize the
following augmented Lagrange function:
\[ L(Q, J, E, Y, \mu) = \|J\|_* + \lambda \|E\|_\ell + \langle Y, D - A(QJ) - E \rangle + \frac{\mu}{2} \|D - A(QJ) - E\|_F^2. \]

Notice that Step 1 and Step 2 are deduced by using a recently proposed linearization technique (Yang & Yuan, 2010), which resolves the general case that \(A\) is not the identity operator. The parameter \(\tau\) is involved in (Yang & Yuan, 2010) and can be chosen as \(\tau = 1/\|A^TA\|_2\). The convex subproblem in Step 3 usually has a closed-form solution: for \(\|E\|_\ell \doteq \|E\|_1\), the solution is given by the shrinkage operator (Lin et al., 2009); for \(\|E\|_\ell \doteq \|E\|_2, 1\), the solution is given by Lemma 3.2 of (Liu et al., 2010).

Since \(r \ll n\) and \(r \ll m\), the cost of computing SVD of \(m \times r\) and \(n \times r\) matrices is ignorable. So the computational complexity of Algorithm 2 is \(O(mnr)\), which is mainly spent for the multiplication of \(m \times r\) and \(r \times n\) matrices. It is not easy to prove the convergence of Algorithm 2 in theory. The difficulties here are two-fold. First, since the the loss function \(\|E\|_\ell\) is non-differentiable, the theories in (Yang & Yuan, 2010) may not be applied here, and thus it is not easy to bring back the optimality gap caused by the linearization technique. Second, as the linear operator \(A\) can be chosen arbitrarily, it is not easy to migrate our theories from RPCA to this general case. Nevertheless, the algorithm performs well in solving the LRR (Liu et al., 2011, 2010) problem that is a representative example of NNROPs, as will be shown in our experiments.

4 Experiments

To verify our active subspace based algorithms (Algorithm 1 and Algorithm 2), we respectively simulate a matrix recovery task for RPCA and a data clustering task for LRR.

4.1 For the RPCA Problem

Given an observation matrix corrupted by sparse noise, the goal of matrix recovery is to correct the noise and recover the original data. To simulate this task, first, we generate a \(4000 \times 4000\) matrix \(X_0\) of rank 100 as a product of two \(4000 \times 100\) matrices \(X_0 = MN^T\). The entries of \(M\) and \(N\) are independently sampled from \(\mathcal{N}(0, 1)\). Second, we generate a high-rank observation matrix \(D\) by \(D = X_0 + E_0\), where \(E_0\) is a sparse matrix 70%
Figure 2: Evaluation results for the RPCA problem (matrix recovery). (a) The computation time (seconds, rescaled by $\log_{10}$) vs the parameter $r$. (b) The optimality gap vs the parameter $r$. (c) The recover error (defined by $\|X_0 - X^*\|_{\infty}$, scaled by $\log_{10}$) vs the parameter $r$. (d) The computation time (of our algorithm) vs the matrix size ($n \times n$), while keeping $r = 100$. The evaluation values have been averaged over 20 runs.
entries of which are zeros. The nonzero entries are independent $\mathcal{N}(0, 4)$ values. Finally, we try recovering $D_0$ from $D$ by using various algorithms to solve the RPCA problem. We repeat these steps 20 times and evaluate various algorithms in terms of \textit{averaged} efficiency and recover accuracy.

To acquire a ground-truth baseline, we use the exact ALM algorithm (denoted as “exact”) proposed by (Lin et al., 2009) to solve problem (4). Based on the ground-truth objective function value $f_o$, we can evaluate the optimality of a derived solution by \textit{optimality gap}: $f^* - f_o$, where $f^*$ is the objective function value estimated by an algorithm. By setting $\lambda = 0.015$, RPCA accurately recovers the original data matrix $D_0$. So we test various algorithms with $\lambda = 0.015$. For comparison, we consider the inexact ALM (i.e., ADM) algorithm presented in (Lin et al., 2009). Also, we choose two partial-SVD methods to speed up the SVD step: one is the widely used approach (denoted as “ParSVD”) established by Williams & Seeger (2000), the other is an efficient randomness algorithm (denoted as “RndSVD”) (Halko et al., 2009). To compute a rank-$r$ SVD of a matrix with size $n \times n$, ParSVD needs a complexity of $O(n^2r)$, while RndSVD only needs $O(nr^2 + r^3)$. However, ParSVD is more accurate than RndSVD.

Fig.2 presents the comparison results. From Fig.2(a), we can see that both RndSVD and our algorithm are efficient, spending about 3 minutes at $r = 120$, which is over 100 times faster than the exact solver that needs about 10 hours. However, as shown in Fig.2(c) and Fig.2(d), RndSVD does not ensure the optimality of its derived solution. Whereas, by setting $r$ to be relatively large, our algorithm can find the exactly optimal solution so as to recover the original data accurately. These results are consistent with the theorems in Section 3.1.2. Fig.2(d) shows that our algorithm is scalable with respect to the size of the matrix $D$, provided that $r$ is relatively small. To recover a $10000 \times 10000$ matrix (with rank 100) by running on a PC (4×2.93GHz CPU, Matlab platform), our algorithm costs about 25 minutes.

\subsection{4.2 For the LRR Problem}

To test Algorithm 2, we consider the LRR problem (Liu et al., 2011, 2010), which is a representative example of NNROPs with $\mathcal{A}(X) = AX$ and $\|E\|_r = \|E\|_{2,1}$. We consider a data clustering task that is a typical application of LRR. To simulate this task,
Figure 3: Evaluation results for the LRR problem (clustering). (a) An example of the singular value spectrum of the optimal solution (with respect to the variable $X$) derived from the exact solver. (b) The computation time (seconds, scaled by $\log_{10}$) vs the parameter $r$. (c) The optimality gap vs the parameter $r$. (d) The clustering accuracy (%) vs the parameter $r$. The evaluation values (except (a)) have been averaged over 38 runs.
we use Extended Yale Database B (Lee et al., 2005), which consists of 2414 frontal face images of 38 classes. Each class contains about 64 images captured under extremely different lighting conditions. We resize the images into $42 \times 48$ pixels and use the raw pixel values to form data vectors of dimension 2016. By setting $A = D$, we can use Algorithm 2 of (Liu et al., 2010) to group the images into clusters and observe clustering accuracy for evaluating various algorithms. To obtain a convincing evaluation, we only use 37 (out of 38) classes for each experiment and run 38 times in total.

To establish a benchmark baseline, we implement an exact ALM algorithm (denote as “exact”) to solve the LRR problem. The implementation is based on the instructions in (Lin et al., 2009). The convergence of the exact ALM algorithm is simple to prove. So this algorithm can provide us a ground truth baseline. For comparison, we also implement an ADM algorithm to directly solve the LRR problem. For fairness of comparison, here, the ADM algorithm is also based on the linearized approximation technique proposed by (Yang & Yuan, 2010). Compared to the algorithms in (Liu et al., 2011), our implementation is more efficient. Similar to the RPCA case, we use ParSVD and RndSVD to speed up the SVD step of this ADM algorithm. We also consider the recently established linearized ADM (LADM) (Lin et al., 2011) algorithm for comparison. By setting $\lambda = 0.5$, LRR achieves the best clustering performance. So we test various optimization algorithms with $\lambda = 0.5$. The computational efficiency and clustering accuracy are observed for evaluation.

Fig.3(a) shows an example of the singular value spectrum of the optimal solution derived from the exact solver. It can be seen that this solution has a long-tailed distribution and has a rank of about 440. In terms of computational efficiency, as shown in Fig.3(b), our active subspace based algorithm largely outperforms both the exact solver and RndSVD. Taking $r = 200$ for example, our algorithm costs about 5 minutes, which is about 60 times faster than the exact solver and 5 times faster than RndSVD. For the LRR problem, as mentioned, RndSVD cannot reduce the computational complexity of the whole optimization procedure, because calculating the product of two $n \times n$ matrices is of $O(n^3)$ complexity. Comparing to the recently established LADM algorithm, our algorithm is still slightly faster (the running time of LADM is about 1.3 times as our active subspace algorithm). This is because LADM needs more computational operators than our active subspace algorithm, although they both achieve the complexity
of $O(mnr)$.

The results shown in Fig.3(c) illustrate that our algorithm can obtain the exactly optimal solution by setting $r = 500$ (notice that the rank of the desired solution is about 440). When $r$ is relatively smaller, as shown in Fig.3(c), the solution derived from our algorithm is close to be exactly optimal. This phenomenon shows that our Algorithm 2 may find the globally optimal solution to the LRR problem, which is a representative example of NNROPs. Fig.3(d) shows that our active subspace mechanism may even improve the performance of LRR in terms of clustering accuracy, when the optimal $r$ is selected. This is not strange, because in pattern recognition applications it is possible that suboptimal solutions work better than the exactly optimal solution for classification purpose.

5 Conclusions and Future Work

We revisited the classic low-rank factorization mechanism, based on which we proposed the active subspace algorithm for solving NNROPs, targeting on scalable low-rank matrix learning. Our basic idea is to transform a large-scale problem into a small-scale one by factorizing a large-size matrix into the product of a small-size orthonormal matrix (active subspace) and another small-size matrix. By utilizing the method of ADM, we show that such a transformation may not cause the loss of optimality. In particular, we devised an algorithm to solve the RPCA problem and prove sub-optimality of the solution provided by our algorithm. Also, we established an algorithm for the general case. Empirically we showed that our algorithm can reduce the computational complexity of LRR without loss of optimality. In summary, the active subspace algorithm shows to be hopeful as a powerful tool for scalable low-rank matrix learning.

In this paper, the rank (dimension) of the active subspace is fixed to be a parameter pre-specified. Actually, it is possible to remove this parameter by dynamically updating the rank of the active subspace. We leave this as our future work.

A Proof of Lemma 3.1

The proof is based on the following two lemmas.
Lemma A.1 The sequences \( \{Y_k\}, \{\hat{Y}_k\} \) and \( \{\tilde{Y}_k\} \) are all bounded.

Proof By the optimality of \( E_{k+1} \), the standard conclusion from convex optimization states that

\[
0 \in \partial \mathcal{L}_E(Q_{k+1}, J_{k+1}, E_{k+1}, Y_k, \mu_k),
\]

i.e.,

\[
Y_k + \mu_k (D - Q_{k+1} + J_{k+1} - E_{k+1}) \in \lambda \partial \|E_{k+1}\|_1,
\]

which directly leads to

\[
Y_{k+1} \in \lambda \partial \|E_{k+1}\|_1, \quad \text{and so } \|Y_{k+1}\|_\infty \leq \lambda.
\] (11)

Hence, the sequence \( \{Y_k\} \) is bounded.

By the optimality of \( Q_{k+1} \), it can be calculated that

\[
\|\hat{Y}_{k+1}\|_F \leq \|Y_k + \mu_k (D - Q_k + J_k - E_k)\|_F = \|Y_k + \rho \mu_{k-1} (D - Q_k + J_k - E_k)\|_F
\]

\[
= \|(1 + \rho)Y_k - \rho Y_{k-1}\|_F.
\]

So \( \{\hat{Y}_k\} \) is bounded due to the boundedness of \( \{Y_k\} \).

By the optimality of \( J_{k+1} \), the standard conclusion from convex optimization states that

\[
0 \in \partial \mathcal{L}_J(Q_{k+1}, J_{k+1}, E_k, Y_k, \mu_k),
\]

which leads to

\[
Q_{k+1}^T \hat{Y}_{k+1} \in \partial \|J_{k+1}\|_* \text{, and so } \|Q_{k+1}^T \hat{Y}_{k+1}\|_2 \leq 1.
\] (12)

At the same time, let \( Q_{k+1}^\perp \) be the orthogonal component of \( Q_{k+1} \), it can be calculated that

\[
(Q_{k+1}^\perp)^T \hat{Y}_{k+1} = (Q_{k+1}^\perp)^T (Y_k + \mu_k (D - E_k)) = (Q_{k+1}^\perp)^T \tilde{Y}_{k+1}.
\]

Hence,

\[
\|(Q_{k+1}^\perp)^T \hat{Y}_{k+1}\|_2 = \|(Q_{k+1}^\perp)^T \tilde{Y}_{k+1}\|_2 \leq \|\tilde{Y}_{k+1}\|_2.
\]

So both \( Q_{k+1}^T \hat{Y}_{k+1} \) and \( (Q_{k+1}^\perp)^T \tilde{Y}_{k+1} \) are bounded, which implies that \( \hat{Y}_{k+1} \) is bounded.

\[\blacksquare\]
Lemma A.2 The sequences \{J_k\}, \{E_k\} and \{Q_kJ_k\} are all bounded.

Proof From the iteration procedure of Algorithm 1, we have that

\[
\mathcal{L}(Q_{k+1}, J_{k+1}, E_{k+1}, Y_k, \mu_k) \leq \mathcal{L}(Q_{k+1}, J_{k+1}, E_k, Y_k, \mu_k) \\
\leq \mathcal{L}(Q_k, J_k, E_k, Y_k, \mu_k) \\
= \mathcal{L}(Q_k, J_k, E_k, Y_{k-1}, \mu_{k-1}) + \frac{\mu_{k-1} + \mu_k}{2}\|Y_k - Y_{k-1}\|_F^2.
\]

So \{\mathcal{L}(Q_{k+1}, J_{k+1}, E_{k+1}, Y_k, \mu_k)\} is upper bounded due to the boundedness of \{Y_k\} and

\[
\sum_{k=1}^{\infty} \frac{\mu_{k-1} + \mu_k}{2\mu_{k-1}} = \frac{\rho(1+\rho)}{2\mu_0} \sum_{k=1}^{\infty} \rho^{-k} = \frac{\rho(1+\rho)}{2\mu_0(\rho - 1)}.
\]

Hence,

\[
\|J_k\|_* + \lambda\|E_k\|_1 = \mathcal{L}(Q_k, J_k, E_k, Y_{k-1}, \mu_{k-1}) - \frac{1}{2\mu_{k-1}}(\|Y_k\|_F^2 - \|Y_{k-1}\|_F^2)
\]

is upper bounded, which means that \{J_k\} and \{E_k\} are bounded. Since \|Q_kJ_k\|_* = \|J_k\|_*, \{Q_kJ_k\} is also bounded. ■

Proof (of Lemma 3.1). By the boundedness of \(Y_k, \hat{Y}_k\) and \(\tilde{Y}_{k+1}\) and the fact that \(\lim_{k \to \infty} \mu_k = \infty\),

\[
\frac{Y_{k+1} - Y_k}{\mu_k} \to 0, \\
\frac{\hat{Y}_{k+1} - Y_{k+1}}{\mu_k} \to 0, \\
\frac{\tilde{Y}_{k+1} - \hat{Y}_{k+1}}{\mu_k} \to 0.
\]

According to the definitions of \(Y_k\) and \(\hat{Y}_k\), it can be also calculated that

\[
E_{k+1} - E_k = \frac{\hat{Y}_{k+1} - Y_{k+1}}{\mu_k}, \\
J_{k+1} - J_k = \frac{Q_{k+1}^T(\hat{Y}_{k+1} - \hat{Y}_{k+1})}{\mu_k}, \\
D - Q_{k+1}J_{k+1} - E_{k+1} = \frac{Y_{k+1} - Y_k}{\mu_k}, \\
Q_{k+1}J_{k+1} - Q_kJ_k = \frac{(1+\rho)Y_k - (\hat{Y}_{k+1} + \rho Y_{k-1})}{\mu_k}.
\]
Hence, the sequences \( \{J_k\}, \{E_k\} \) and \( \{Q_k J_k\} \) are Cauchy sequences, and Algorithm 1 can stop within a finite number of iterations.

By the convergence conditions of Algorithm 1, it can be calculated that
\[
\|D - Q^* J^* - E^*\|_\infty = \frac{|Y_{k^*+1} - Y_{k^*}|}{\mu_k} \leq \varepsilon,
\]
where \( k^* \) is defined in (6), and \( \varepsilon > 0 \) is the control parameter set in Algorithm 1.

**Note.** One may have noticed that \( \{Q_k\} \) may not converge. This is because the basis of a subspace is not unique. Nevertheless, it is actually insignificant whether or not \( \{Q_k\} \) converges, because it is the product of \( Q^* \) and \( J^* \), namely \((X = Q^* J^*, E = E^*)\) that recovers a solution to the original RPCA problem.

### B Proof of Lemma 3.2

We prove the following lemma at first.

**Lemma B.1** Let \( X, Y \) and \( Q \) are matrices of compatible dimensions. If \( Q \) obeys \( Q^T Q = I \) and \( Y \in \partial \|X\|_s \), then \( QY \in \partial \|QX\|_s \).

**Proof** Let the skinny SVD of \( X \) is \( U \Sigma V^T \). By \( Y \in \partial \|X\|_s \), we have
\[
Y = UV^T + W, \text{ with } U^T W = 0, W V = 0 \text{ and } \|W\| \leq 1.
\]

Since \( Q \) is column-orthonormal, we have
\[
\partial \|QX\|_s = \{QUV^T + W | U^T Q W = 0, W_1 V = 0 \text{ and } \|W_1\| \leq 1\}.
\]

With the above notations, it can be verified that \( QY \in \partial \|QX\|_s \).

**Proof** of (Lemma 3.2) Let the skinny SVD of \( D - E_k + Y_k / \mu_k \) be \( D - E_k + Y_k / \mu_k = U_k \Sigma_k V_k^T \), then it can be calculated that
\[
Q_{k+1} = \mathcal{P}[(D - E_k + Y_k / \mu_k) J_k^T] = \mathcal{P}[U_k \Sigma_k V_k^T J_k^T].
\]

Let the full SVD of \( \Sigma_k V_k^T J_k^T \) be \( \Sigma_k V_k^T J_k^T = U \Sigma V^T \) (note that \( U \) and \( V \) are orthogonal matrices), then it can be calculated that
\[
Q_{k+1} = \mathcal{P}[U_k \Sigma_k V_k^T J_k^T] = \mathcal{P}[U_k U \Sigma V^T] = U_k U V^T.
\]
which simply leads to
\[ Q_{k+1}Q^T_{k+1} = U_k U^T V U^T U^T_k = U_k U^T_k. \]

Hence,
\[
\hat{Y}_{k+1} - Q_{k+1}Q^T_{k+1}\hat{Y}_{k+1} = \mu_k ((D - E_k + \frac{Y_k}{\mu_k}) - Q_{k+1}Q^T_{k+1}(D - E_k + \frac{Y_k}{\mu_k}))
\]
\[
= \mu_k (U_k \Sigma_k V^T_k - Q_{k+1}Q^T_{k+1} U_k \Sigma_k V^T_k)
\]
\[
= \mu_k (U_k \Sigma_k V^T_k - U_k U^T_k \Sigma_k V^T_k)
\]
\[
= \mu_k (U_k \Sigma_k V^T_k - U_k \Sigma_k V^T_k) = 0,
\]
i.e.,
\[ \hat{Y}_{k+1} = Q_{k+1}Q^T_{k+1} \hat{Y}_{k+1}. \]

According to (12) and Lemma B.1, we have
\[ Q_{k+1}Q^T_{k+1}\hat{Y}_{k+1} \in \partial \|Q_{k+1}J_{k+1}\|_\star. \]

Hence,
\[ \hat{Y}_{k+1} \in \partial \|Q_{k+1}J_{k+1}\|_\star \quad \text{and} \quad Y_{k+1} \in \lambda \partial \|E_{k+1}\|_1, \forall k. \]

where the conclusion of \( Y_{k+1} \in \lambda \partial \|E_{k+1}\|_1 \) is quoted from (11). Since the above conclusion holds for any \( k \), it naturally holds at \( (Q^*, J^*, E^*) \):
\[ \hat{Y}^* = \hat{Y}_{k^*+1} \in \partial \|Q^* J^*\|_\star \quad \text{and} \quad Y^* = Y_{k^*+1} \in \lambda \partial \|E^*\|_1. \] (13)

Given any feasible solution \((Q, J, E)\) to problem (5), by the convexity of matrix norms and (13), it can be calculated that
\[
\|J\|_\star + \lambda \|E\|_1 = \|Q J\|_\star + \lambda \|E\|_1
\]
\[
\geq \|Q^* J^*\|_\star + \langle \hat{Y}^*, Q J - Q^* J^* \rangle + \lambda \|E^*\|_1 + \langle Y^*, E - E^* \rangle
\]
\[
= \|J^*\|_\star + \lambda \|E^*\|_1 + \langle \hat{Y}^*, Q J + E - Q^* J^* - E^* \rangle + \langle Y^* - \hat{Y}^*, E - E^* \rangle.
\]

By Lemma 3.1, we have that
\[ \|Q J + E - Q^* J^* - E^*\|_\infty \leq \|D - Q^* J^* - E^*\|_\infty < \varepsilon, \]
which leads to
\[
|\langle \hat{Y}_*, Q J + E - Q^* J^* - E^* \rangle| \leq \|\hat{Y}_*\|_\infty \|Q J + E - Q^* J^* - E^*\|_1
\]
\[
\leq \|\hat{Y}_*\| \|D - Q^* J^* - E^*\|_1
\]
\[
\leq m\|D - Q^* J^* - E^*\|_\infty < mn\varepsilon.
\]
where \( \hat{Y}_s \leq 1 \) is due to (13). Hence,
\[
\|J\|_* + \lambda \|E\|_1 \geq \|J^*\|_* + \lambda \|E^*\|_1 + \langle Y^* - \hat{Y}^*, E - E^* \rangle - mn\varepsilon.
\]

\[\square\]

C Proof of Theorem 3.1

**Proof** Notice that \((Q^*, J = 0, E = D)\) is feasible to (5). Let \((Q^g, J^g, E^g)\) be a globally optimal solution to (5), then we have
\[
\lambda \|E^g\|_1 \leq \|J^g\|_* + \lambda \|E^g\|_1 \leq \lambda \|D\|_1.
\]

By the proof procedure of Lemma A.2, we have that \(E^*\) is bounded by
\[
\lambda \|E^*\|_1 \leq \|J^*\|_* + \lambda \|E\|_1 \\
\leq \mathcal{L}(Q_{k+1}, J_{k+1}, E_{k+1}, Y_{k+1}, \mu_{k+1}) + \frac{\|Y_{k+1}\|^2_F}{2\mu_{k+1}} \\
\leq \frac{mn\lambda^2}{\mu_0} \left( \frac{\rho(1 + \rho)}{\rho - 1} + \frac{1}{2\rho \varepsilon} \right) \\
= mn\|D\| \lambda^2 \left( \frac{\rho(1 + \rho)}{\rho - 1} + \frac{1}{2\rho \varepsilon} \right).
\]

Hence,
\[
\|E^g - E^*\|_1 \leq \|E^g\|_1 + \|E^*\|_1 \leq c_1. \tag{14}
\]

Note that \(\|\langle M, N \rangle\| \leq \|M\|_\infty \|N\|_1\) holds for any matrices \(M\) and \(N\). By Lemma 3.2 and (14), we have
\[
f^g = \|J^g\|_* + \lambda \|E^g\|_1 \geq \|J^*\|_* + \lambda \|E^*\|_1 + \langle Y^* - \hat{Y}^*, E^g - E^* \rangle - mn\varepsilon \\
\geq f^* - \|Y^* - \hat{Y}^*\|_\infty \|E^g - E^*\|_1 - mn\varepsilon \\
= f^* - \varepsilon_1 \|E^g - E^*\|_1 - mn\varepsilon \\
\geq f^* - c_1 \varepsilon_1 - mn\varepsilon,
\]

which simply leads to the inequality stated in Theorem 3.1. \[\square\]
D Proof of Theorem 3.2

Proof Let \( X = Q^*J^* \) and \( E = E^* \), then \((X, E)\) is a feasible solution to the original RPCA problem. By the convexity of the PRCA problem and the optimality of \((X^o, E^o)\), it naturally follows that

\[ f^0 \leq f^*. \]

Let \( X^o = U^o\Sigma^o(V^o)^T \) be the skinny SVD of \( X^o \). Construct \( Q' = U^o, J' = \Sigma^o(V^o)^T \) and \( E' = E^o \). When \( r \geq r_0 \), we have

\[ D = X^o + E^o = U^o\Sigma^o(V^o)^T + E^o = Q'J' + E', \]

i.e., \((Q', J', E')\) is a feasible solution to problem (5). By Theorem 3.1, it can be concluded that

\[ f^* - c_1\varepsilon_1 - mn\varepsilon \leq \|J'\|_s + \lambda\|E'\|_s = \|\Sigma^o\|_s + \lambda\|E^o\|_1 = f^o. \]

For \( r < r_0 \), we decompose the skinny SVD of \( X^o \) as

\[ X^o = U_0\Sigma_0V_0^T + U_1\Sigma_1V_1^T, \]

where \( U_0, V_0 \) (resp. \( U_1, V_1 \)) are the singular vectors associated with the \( r \) largest singular values (resp. the rest singular values smaller than or equal to \( \sigma_r \)). With these notations, we have a feasible solution to problem (5) by constructing

\[ Q'' = U_0, J'' = \Sigma_0V_0^T \text{ and } E'' = D - U_0\Sigma_0V_0^T = E^o + U_1\Sigma_1V_1^T. \]

By Theorem 3.1, it can be calculated that

\[ f^* - c_1\varepsilon_1 - mn\varepsilon \leq f'' \leq \|J''\|_s + \lambda\|E''\|_1 \]
\[ = \|\Sigma_0V_0^T\|_s + \lambda\|E^o + U_1\Sigma_1V_1^T\|_1 = \|\Sigma_0\|_s + \lambda\|E^o + U_1\Sigma_1V_1^T\|_1 \]
\[ \leq \|X^o\|_s - \|\Sigma_1\|_s + \lambda\|E^o + U_1\Sigma_1V_1^T\|_1 \]
\[ \leq \|X^o\|_s - \|\Sigma_1\|_s + \lambda\|E^o\|_1 + \lambda\|U_1\Sigma_1V_1^T\|_1 \]
\[ = f^o - \|\Sigma_1\|_s + \lambda\|U_1\Sigma_1V_1^T\|_1 \]
\[ \leq f^o - \|\Sigma_1\|_s + \lambda\sqrt{mn}\|U_1\Sigma_1V_1^T\|_F \]
\[ \leq f^o - \|\Sigma_1\|_s + \lambda\sqrt{mn}\|U_1\Sigma_1V_1^T\|_s \]
\[ = f^o + (\lambda\sqrt{mn} - 1)\|\Sigma_1\|_s \]
\[ \leq f^o + (\lambda\sqrt{mn} - 1)\sigma_{r+1}(r_0 - r). \]
References


Liu, Guangcan, Lin, Zhouchen, Yan, Shuicheng, Sun, Ju, Yu, Yong, and Ma, Yi. Robust recovery of subspace structures by low-rank representation. *TPAMI*, 2011.


