Low-Rank and Sparse Structure Pursuit via Alternating Minimization

Quanquan Gu
Department of Systems
and Information Engineering
University of Virginia

Zhaoran Wang
Dept. of Operations Research
and Financial Engineering
Princeton University

Han Liu
Dept. of Operations Research
and Financial Engineering
Princeton University

Abstract

In this paper, we present a nonconvex alternating minimization optimization algorithm for low-rank and sparse structure pursuit. Compared with convex relaxation based methods, the proposed algorithm is computationally more efficient for large scale problems. In our study, we define a notion of bounded difference of gradients, based on which we rigorously prove that with suitable initialization, the proposed nonconvex optimization algorithm enjoys linear convergence to the global optima and exactly recovers the underlying low rank and sparse matrices under standard conditions such as incoherence and sparsity conditions. For a wide range of statistical models such as multi-task learning and robust principal component analysis (RPCA), our algorithm provides a principled approach to learning the low rank and sparse structures with provable guarantee. Thorough experiments on both synthetic and real datasets backup our theory.

1 Introduction

In many machine learning problems, the data are generated from models which have inherent low rank and sparse structures. One example is multi-task learning [11], which exploits the relationships among multiple related tasks to improve the generalization performance. A common assumption in multi-task learning is that all tasks should share some common structures in model parameters. Typical common structures include low rank subspace sharing [26], (group) sparse feature set sharing [3, 25, 24, 19], as well as both low-rank and sparse parameter sharing [13, 1]. Another example is finding low-dimensional structure in high-dimensional data. For instance, it has been observed that principal component analysis (PCA) is sensitive to outliers. To overcome this problem, Candès et al. [9] proposed a robust principal component analysis (RPCA). The goal of robust PCA is to remove sparse corruptions from the input matrix and obtain a low-rank approximation of the input matrix.

In this paper, motivated by the above applications, we consider the following general model, with its parameter matrix being the superposition of a low rank matrix and sparse matrix[1]:

$$\mathbf{Y} = \mathbf{X}(\mathbf{L}^* + \mathbf{S}^*), \tag{1.1}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d_1}$ is a general design matrix, $\mathbf{L}^* \in \mathbb{R}^{d_1\times d_2}$ is an unknown low rank matrix with $\text{rank}(\mathbf{L}^*) = r$, and $\mathbf{S}^* \in \mathbb{R}^{d_1\times d_2}$ is an unknown sparse matrix with $\|\mathbf{S}^*\|_{0,0} = s^*$ nonzero elements, $\mathbf{Y} \in \mathbb{R}^{n\times d_2}$ is the response matrix. Similar models have been considered in [12, 1, 33] with or without noise. In order to recover the low rank component $\mathbf{L}^*$ and the sparse component $\mathbf{S}^*$ together based on $\mathbf{X}$ and $\mathbf{Y}$, the most widely used method is based on convex relaxation, solving the following convex problem:

$$\operatorname*{argmin}_{\mathbf{L},\mathbf{S}} \frac{1}{2}\|\mathbf{Y} - \mathbf{X}(\mathbf{L}+\mathbf{S})\|_F^2 + \lambda\|\mathbf{L}\|_* + \mu\|\mathbf{S}\|_{1,1}. \tag{1.2}$$

Here $\|\mathbf{L}\|_*$ is the nuclear norm of $\mathbf{L}$, the tightest convex relaxation of $\text{rank}(\mathbf{L})$. Meanwhile, $\|\mathbf{S}\|_{1,1}$ is the elementwise ℓ_1 norm of $\mathbf{S}$, the tightest convex relaxation of $\|\mathbf{S}\|_{0,0}$. Besides, $\lambda, \mu > 0$ are the regularization parameters. The relaxed problem in (1.2) can be solved by

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

[1]For the ease of presentation, we consider the noiseless case, while our algorithm is directly applied to the case with additive noise, and our theory can be generalized straightforwardly as well.

convex optimization algorithms such as proximal gradient descent [36] and alternating direction method of multipliers (ADMM) [23]. While these convex optimization algorithms enjoy good convergence guarantees, due to the nuclear norm regularization, they involve a singular value decomposition at each iteration, which is computationally very expensive for high dimensional problems with large d_1 and d_2.

In our study, to overcome computational limitations of existing optimization algorithms for low-rank and sparse matrix learning, we reparameterize the unknown low-rank matrix $\mathbf{L}$ as the product of two much smaller matrices, i.e., $\mathbf{L} = \mathbf{U}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{d_1 \times r}$ and $\mathbf{V} \in \mathbb{R}^{d_2 \times r}$, such that the low-rank constraint is automatically satisfied. This gives rise to the following constrained problem

$$\operatorname*{argmin}_{\mathbf{U},\mathbf{V},\mathbf{S}} \frac{1}{2}\|\mathbf{Y} - \mathbf{X}(\mathbf{U}\mathbf{V}^\top + \mathbf{S})\|_F^2 \quad \text{subject to } \|\mathbf{S}\|_{0,0} \le s, \tag{1.3}$$

where $s > 0$ is a tuning parameter. The resulting optimization problem can be solved very efficiently through alternating minimization, i.e., solving $\mathbf{U}$ (resp. $\mathbf{V}$ and $\mathbf{S}$) while the other two variables are fixed until convergence. Despite the great empirical success of this reparameterization [20, 29], the theoretical understanding of the algorithms for the factorization based formulation is fairly limited because the theoretical properties of nonconvex optimization algorithms are more difficult to analyze. In this paper, we establish the linear convergence rate of the proposed algorithm. At the core of our theory is the notion of *bounded difference of gradients*, which provides a sufficient condition for the convergence of nonconvex optimization. Based on this notion, we show that under certain conditions, the alternating minimization algorithm converges exponentially fast to the global optima of the non-convex optimization problem in (1.3), and thus recovers the true low-rank and the sparse matrices. Extensive experiments on synthetic and real datasets corroborate our theory.

Notation and Organization of This Paper

Let $\mathbf{A} = [A_{ij}] \in \mathbb{R}^{d\times d}$ be a $d \times d$ matrix and $\mathbf{x} = [x_1, \ldots, x_d]^\top \in \mathbb{R}^d$ be a d-dimensional vector. For $0 < q < \infty$, we define the ℓ_0, ℓ_q and ℓ_∞ vector norms as $\|\mathbf{x}\|_0 = \sum_{i=1}^d \mathbb{I}(x_i \neq 0), \|\mathbf{x}\|_q = (\sum_{i=1}^d |x_i|^q)^{\frac{1}{q}}$, and $\|\mathbf{x}\|_\infty = \max_{1\le i\le d}|x_i|$, where $\mathbb{I}(\cdot)$ represents the indicator function. We use the following notation for the matrix ℓ_q, $\ell_{\max}$ and ℓ_F norms: $\|\mathbf{A}\|_q = \max_{\|\mathbf{x}\|=1}\|\mathbf{A}\mathbf{x}\|_q, \|\mathbf{A}\|_{\max} = \max_{ij}|A_{ij}|, \|\mathbf{A}\|_1 = \sum_{i,j=1}^d |A_{ij}|$, and $\|\mathbf{A}\|_F = (\sum_{ij}|A_{ij}|^2)^{\frac{1}{2}}$. Moreover, we define the nuclear norm of $\mathbf{A}$, i.e., $\|\mathbf{A}\|_*$ to be the sum of all singular values of $\mathbf{A}$. We use the notation $\odot$ to denote the element-wise product.

The remainder of this paper is organized as follows: Section 2 is devoted to two application examples and the related work. We present the nonconvex optimization algorithm in Section 3. The main theoretical results are provided in Section 4 and the proof is sketched in Section 5. The numerical experiments are reported in Section 6. Section 7 concludes the paper.

2 Examples and Related Work

In this section, we introduce two examples of the model introduced in (1.1), followed which we review a sequence of related work.

2.1 Examples

As we mentioned in the introduction, there are two widely studied problems, which fit in our model in (1.1).

Example 2.1 (Multi-Task Learning)**.** In multi-task learning, we are given a collection of d_2 regression problems in $\mathbb{R}^{d_1}$, each of which takes the form $\mathbf{y}_j = \mathbf{X}\boldsymbol{\beta}_j^*$ for $j = 1, 2, \ldots, d_2$. Here each $\boldsymbol{\beta}_j^* \in \mathbb{R}^{d1}$ is an unknown regression vector, and $\mathbf{X} \in \mathbb{R}^{n\times d_1}$ is the design matrix. We refer to each regression problem as a task. This family of models can be written in a convenient matrix form as $\mathbf{Y} = \mathbf{X}\mathbf{B}^*$, where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{d_2}] \in \mathbb{R}^{n\times d2}$ and $\mathbf{B}^* = [\boldsymbol{\beta}_1^*, \ldots, \boldsymbol{\beta}_{d_2}^*] \in \mathbb{R}^{d_1\times d_2}$ is a matrix of regression vectors. Each row of $\mathbf{B}^*$ corresponds to a feature. We assume that some subset of features are shared across tasks, and some other subset of features vary substantially across tasks. This kind of structure can be captured by assuming that the unknown regression matrix $\mathbf{B}^*$ has a low-rank plus sparse decomposition [1, 13], i.e., $\mathbf{B}^* = \mathbf{L}^* + \mathbf{S}^*$, where $\mathbf{L}^*$ is low-rank and $\mathbf{S}^*$ is sparse, with a relatively small number of non-zero entries, corresponding to feature/task pairs that differ significantly across tasks.

Example 2.2 (Robust Principal Component Analysis)**.** In Robust PCA [9], the observation matrix $\mathbf{Y} \in \mathbb{R}^{d_1\times d_2}$ is assumed to be the superposition of a low rank matrix $\mathbf{L}^*$ and an error matrix $\mathbf{S}^*$, where the error matrix $\mathbf{S}^*$ can be arbitrary in magnitude, but is assumed to be sparsely supported, affecting only a fraction of the entries in $\mathbf{L}^*$, i.e., $\mathbf{Y} = \mathbf{L}^* + \mathbf{S}^*$. It is easy to see that it is a special case of the model in (1.1), with the design matrix $\mathbf{X}$ chosen as the identity matrix $\mathbf{I} \in \mathbb{R}^{n\times n}$, and $n = d_1$.

2.2 Related Work

Despite the superior empirical performance of the nonconvex alternating minimization algorithm, its theoretical analysis is relatively lagging behind. Only until recently has there been significant process on its theory. In particular, alternating optimization has been recently analyzed for matrix completion [18, 16, 6, 35], dictionary learning [2], sparse coding [4], phase retrieval [27], and expectation maximization (EM) algorithm [5, 30]. However, the proof techniques in [18, 16, 6, 2, 27] are built up on perturbed power methods, and cannot be applied to our algorithm due to the existence of the sparse component.

Waters et al. [31] proposed a greedy algorithm for recovering low-rank and sparse matrices from compressive measurements. Kyrillidis and Cevher [21] proposed a projected gradient method over non-convex sets for recovering a sparse plus low-rank decomposition of a matrix given noisy linear measurements. Yan et al. [33] studied a similar model as ours but with additive noise. They present a nonconvex alternating direction algorithm to estimate the low-rank and sparse components alternatively. However, they did not factorize $\mathbf{L}$ as $\mathbf{L} = \mathbf{U}\mathbf{V}^\top$, and therefore, their algorithm involves a singular value decomposition in each iteration, which is time consuming for high dimensional problems. Moreover, they only achieve sublinear convergence rate, e.g., $O(1/\sqrt{t})$, which is significantly slower than our algorithm. For RPCA, there at least exist two related works, which present nonconvex optimization algorithms. Feng et al. [15] proposed an online algorithm for RPCA based on the reparameterization $\mathbf{L} = \mathbf{U}\mathbf{V}^\top$ and ℓ_1 regularization on $\mathbf{S}$. However, they can only prove that their algorithm converges to the optimal solution asymptotically. They do not provide any convergence rate guarantee. Netrapalli et al. [28] proposed a nonconvex optimization algorithm for RPCA with provable guarantee. Their algorithm involves alternating between projecting the appropriate residuals onto the set of low-rank matrices and the set of sparse matrices. Although the algorithm also enjoys linear convergence rate, it is specially designed for solving RPCA and it is unclear whether it can be extended to solve the general low-rank and sparse structure pursuit in (1.3). As we will show, our algorithm enjoys a lower per iteration computational complexity. Chen and Wainwright [14] proposed a projected gradient descent algorithm for fast low-rank matrix estimation with provable guarantee, which includes RPCA as a special example. However, they focus on symmetric positive semidefinite low rank matrix problems. In contrast, our proposed algorithm and theory are for general low rank matrix.

3 The Proposed Algorithm

In this section, we outline the nonconvex optimization algorithm for solving (1.3). Let $\mathcal{L}(\mathbf{U}, \mathbf{V}, \mathbf{S}) = 1/2\|\mathbf{Y} - \mathbf{X}(\mathbf{U}\mathbf{V}^\top + \mathbf{S})\|_F^2$ be the shorthand for our loss function, the algorithm is displayed in Algorithm 1. To handle the sparsity constraint in (1.3), in line 9 of Algorithm 1, we perform a truncation step to enforce the sparsity of the iterate $\mathbf{S}$. In detail, we define the Truncate$(\cdot, s)$ function in line 9 as

$$\big[\text{Truncate}(\mathbf{S}, s)\big]_{jk} = \begin{cases} S_{jk}, & \text{if } S_{jk} \text{ is among the top } s \text{ large magnitudes,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Note that $\mathbf{S}^{(t+1)}$ is the output of line 8 at the t-th iteration of the algorithm. To obtain $\bar{\mathbf{S}}^{(t+1)}$, the truncation step (line 9) preserves the entries of $\mathbf{S}^{(t+1)}$ with the top s large magnitudes and sets the rest to zero. Here s is a tuning parameter that controls the sparsity level (line 1). By iteratively performing the alternating minimization and truncation step, the proposed algorithm converges to the global optimal solutions. Here T is the total number of iterations. The truncation step enforces the sparse structure along the solution path. It is worth noting that, the truncation strategy has been previously adopted in power method [34]. Such a truncation operation poses significant challenges for our analysis. In addition, we apply QR decomposition to factorize $\mathbf{U}^{(t+1)}$ into $\overline{\mathbf{U}}^{(t+1)}$ and $\mathbf{R}_1^{(t+1)}$, where $\overline{\mathbf{U}}^{(t+1)}$ is the orthonormal basis of $\mathbf{U}^{(t+1)}$, and $\mathbf{R}_1^{(t+1)}$ is the corresponding upper triangular matrix. Note that the QR decomposition in Algorithm 1 is not necessary in practice, but only for the purpose of theoretical analysis. The QR decomposition has also been employed in [18, 16] for the theoretical analysis. In fact, they strictly proved that QR decomposition does not affect the solution. In other words, in our implementation, we do not need QR. Also note that although the reparameterization $\mathbf{L} = \mathbf{U}\mathbf{V}^\top$ makes the optimization problem less easier to analyze, it significantly boosts the computational efficiency. In fact, such kind of reparameterization has been widely used in many literatures in practical systems [20, 29].

Algorithm 1 Alternating Minimization for Low-rank and Sparse Structure Pursuit

```
1: Inputs: Number of iterations $T$, rank parameter $k$, sparsity parameter $s$
2: Initilize: $\overline{\mathbf{V}}^{(0)} \in \mathbb{R}^{d_2 \times k}$ and $\bar{\mathbf{S}}^{(0)} \in \mathbb{R}^{d_1 \times d_2}$
3: for $t = 0$ to $T - 1$ do
4:     $\mathbf{U}^{(t+1)} = \text{argmin}_{\mathbf{U}} \mathcal{L}(\mathbf{U}, \overline{\mathbf{V}}^{(t)}, \bar{\mathbf{S}}^{(t)})$.
5:     $\{\overline{\mathbf{U}}^{(t+1)}, \mathbf{R}_1^{(t+1)}\} = \text{Thin\_QR}(\mathbf{U}^{(t+1)})$.
6:     $\mathbf{V}^{(t+1)} = \text{argmin}_{\mathbf{V}} \mathcal{L}(\overline{\mathbf{U}}^{(t+1)}, \mathbf{V}, \bar{\mathbf{S}}^{(t)})$.
7:     $\{\overline{\mathbf{V}}^{(t+1)}, \mathbf{R}_2^{(t+1)}\} = \text{Thin\_QR}(\mathbf{V}^{(t+1)})$.
8:     $\mathbf{S}^{(t+1)} = \text{argmin}_{\mathbf{S}} \mathcal{L}(\overline{\mathbf{U}}^{(t+1)}, \mathbf{V}^{(t+1)}, \mathbf{S})$.
9:     $\bar{\mathbf{S}}^{(t+1)} = \text{Truncate}(\mathbf{S}^{(t+1)}, s)$.
10: end for
```

In details, Table 1 compares the per-iteration complexity our algorithm with existing methods for multi-task regression and robust PCA, e.g., proximal gradient descent and ADMM. Here without loss of generality, we assume that $d_1 \geq d_2$. It is obvious that our algorithm enjoys much smaller per-iteration complexity than convex relaxation methods. In next section, we will prove that our algorithm attains linear convergence rate. In contrast, proximal gradient descent only attains linear convergence rate when the loss function is strongly convex, which does not hold in the high dimensional regime [32]. The argument applies to ADMM [17]. Therefore, in terms of both convergence rate and per-iteration complexity, our algorithm is more efficient than the convex relaxation based methods. Furthermore, the per-iteration complexity of nonconvex RPCA algorithm [28] is also much smaller than the convex relaxation methods, but slightly worse than our algorithm, especially when the rank r is big. More importantly, it is limited to RPCA and it is unclear whether it can be applied to multi-task learning and other cases.

4 Main Theory

In this section, we present the main theory for the nonconvex optimization algorithm in Algorithm 1.

Let the singular value decomposition of $\mathbf{L}^* \in \mathbb{R}^{d_1\times d_2}$ be $\mathbf{L}^* = \overline{\mathbf{U}}^*\mathbf{\Sigma}^*\overline{\mathbf{V}}^{*\top}$, where $\overline{\mathbf{U}}^* \in \mathbb{R}^{d_1\times r}$ and $\overline{\mathbf{V}}^* \in \mathbb{R}^{d_2\times r}$ are the left and right singular matrices, and $\mathbf{\Sigma}^* = \mathrm{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$ is the diagonal singular value matrix. Then we have $\mathbf{Y} = \mathbf{X}(\overline{\mathbf{U}}^*\mathbf{\Sigma}^*\overline{\mathbf{V}}^{*\top} + \mathbf{S}^*)$. We denote the condition number of $\mathbf{L}^*$ by $\kappa = \sigma_{\max}(\mathbf{L}^*)/\sigma_{\min}(\mathbf{L}^*) = \sigma_1/\sigma_r$. Before we lay out the main theory, we first make several assumptions, which are necessary to our theoretical analysis.

The first assumption is known as incoherence condition [7, 9].

Assumption 4.1. We assume $\mathrm{rank}(\mathbf{L}^*) \leq r$. $\mathbf{L}^*$ is μ-incoherent, i.e., suppose the singular value decomposition of $\mathbf{L}^*$ is $\mathbf{L}^* = \overline{\mathbf{U}}^*\mathbf{\Sigma}^*\overline{\mathbf{V}}^{*\top}$, it holds that

$$\max_{1\leq i\leq d_1} \|\mathbf{e}_i^\top \overline{\mathbf{U}}^*\|_2 \leq \mu\sqrt{\frac{r}{d_1}},$$
$$\max_{1\leq i\leq d_2} \|\mathbf{e}_i^\top \overline{\mathbf{V}}^*\|_2 \leq \mu\sqrt{\frac{r}{d_2}}.$$

The incoherence assumption guarantees that $\mathbf{L}^*$ is not too spiky. In other words, it precludes those matrices which are sparse and have very few large entries. This assumption has been widely made in the literature of matrix completion [7] and decomposition [9].

The second assumption is known as the restricted isometry property (RIP) [8].

Assumption 4.2. For any $\mathbf{L} \in \mathbb{R}^{d_1\times d_2}$ such that $\mathrm{rank}(\mathbf{L}) \leq r$, the following holds

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

And for any $\mathbf{S} \in \mathbb{R}^{d_1\times d_2}$ such that $\|\mathbf{S}\|_{0,0} \leq s$, the following holds

$$(1-\zeta_s)\|\mathbf{S}\|_F^2 \leq \|\mathbf{X}(\mathbf{S})\|_F^2 \leq (1+\zeta_s)\|\mathbf{S}\|_F^2$$

Assumption 4.2 is a common assumption. Many types of design matrix $\mathbf{X}$ satisfy the RIP condition. For example, several random matrix ensembles including Gaussian ensemble satisfy RIP with high probability. In addition, the identity design matrix satisfies the RIP with $\delta_r = 0$ and $\zeta_s = 0$.

Now we are ready to present the main theorem. The following theorem establishes the linear convergence rate of the nonconvex optimization algorithm in Algorithm 1. Before we lay out the theorem, we lay out several technical conditions that will greatly simplify our notation. Let

$$\nu = \max\big\{\sqrt{(1+\delta_{2r})(1+\zeta_{2s})}, (1+\delta_1)/(1-\delta_1)\big\}.$$

Condition 4.3. The dimension $d = \min(d_1, d_2)$ is sufficiently large such that

$$d \geq \max\big\{120\sqrt{2r}\nu\mu\kappa\sigma_1, 16\sqrt{r}/(7\sqrt{2}\kappa\sigma_1/16-1),$$
$$8\mu\sqrt{r}/(3\sqrt{2}\kappa\sigma_1 - 3\delta_1\sqrt{r}-1)\big\}^{2/3}s^*.$$

Meanwhile, δ_1 is sufficiently small such that $\delta_1 \leq 8\mu(s^*/d)^{3/2}$.

For notational simplicity, we define

$$\gamma = \frac{3\sqrt{2}\kappa}{1-24\sqrt{2}\nu\mu\sqrt{r}(s^*/d)^{3/2}\kappa\sigma_1}, \tag{4.1}$$

and

$$\rho = \frac{96\sqrt{2}\nu\mu\sqrt{r}(s^*/d)^{3/2}\kappa\sigma_1}{1-24\sqrt{2}\nu\mu\sqrt{r}(s^*/d)^{3/2}\kappa\sigma_1}. \tag{4.2}$$

Theorem 4.4. Suppose the sparsity parameter is chosen as $s = 4s^*$. Assume the initializations $\overline{\mathbf{U}}^{(0)}$, $\overline{\mathbf{V}}^{(0)}$ and $\overline{\mathbf{S}}^{(0)}$ satisfy $\big\|\overline{\mathbf{S}}^{(0)} - \mathbf{S}^*\big\|_2 \leq \sigma_1$,

$$\big\|\overline{\mathbf{V}}^{(0)} - \overline{\mathbf{V}}^*\big\|_F \leq 2\mu\sqrt{\frac{(1+\zeta_{2s})r}{(1+\delta_{2r})}}\left(\frac{s}{d}\right)^{3/2},$$
$$\big\|\overline{\mathbf{U}}^{(0)} - \overline{\mathbf{U}}^*\big\|_F \leq \left(3\delta_1\sqrt{r} + \mu\sqrt{\frac{r}{d_1}}\frac{s^{3/2}}{d_2}\right).$$

Then under Condition 4.3 we have that for all t, it holds that

$$\big\|\overline{\mathbf{U}}^{(t+1)} - \overline{\mathbf{U}}^*\big\|_F + \gamma\big\|\mathbf{V}^{(t+1)} - \mathbf{V}^*\big\|_F$$
$$\leq \rho\big(\big\|\overline{\mathbf{U}}^{(t)} - \overline{\mathbf{U}}^*\big\|_F + \gamma\big\|\mathbf{V}^{(t)} - \mathbf{V}^*\big\|_F\big),$$

where γ and ρ are defined in (4.1) and (4.2). In addition, we have

$$\big\|\mathbf{L}^{(t+1)} - \mathbf{L}^*\big\|_F \leq \sigma_1\rho^t\big(\big\|\overline{\mathbf{U}}^{(0)} - \overline{\mathbf{U}}^*\big\|_F$$
$$+ \gamma\big\|\mathbf{V}^{(0)} - \mathbf{V}^*\big\|_F\big).$$

Table 1: Comparison of our algorithm with convex relaxation based method in terms of per-iteration complexity. N/A means the algorithm does not apply to the problem.

Methods	Multi-task Learning	RPCA
Convex Relaxation	$O(nd_1d_2 + d_1d_2^2)$	$O(d_1d_2 + d_1d_2^2)$
Noncovex RPCA [28]	N/A	$O(r^2d_1d_2)$
Ours	$O(nd_1r)$	$O(rd_1d_2)$

Remark 4.5. In Theorem 4.4, we require the initial iterates are close enough to $\mathbf{U}^*$, $\mathbf{V}^*$ and $\mathbf{S}^*$, from where it will converge. This requirement is easy to attain. For example, we can always scale the design matrix $\mathbf{X}$ and the response matrix $\mathbf{Y}$ such that σ_1 is sufficiently large. Therefore, $\|\overline{\mathbf{S}}^{(0)} - \mathbf{S}^*\|_2$ can be satisfied. In addition, the condition that the initial solutions $\overline{\mathbf{U}}^{(0)}$ and $\overline{\mathbf{V}}^{(0)}$ are sufficiently close to the true solution is standard and prevailing in existing literature [18, 16] on alternating minimization. In practice, to satisfy this condition, one often tries multiple random initializations, and chooses the one that minimizes the objective function value. In fact, this heuristic has been widely used in many machine learning methods such as EM algorithm and matrix factorization.

Remark 4.6. The result in Theorem 4.4 can be translated to the case of RPCA. In particular, in the case of RPCA, since the design matrix is identity, we have $\delta_{2r} = \delta_1 = 0$, and therefore $\nu = 1$. The sample complexity of our algorithm for RPCA is $O(\mu^{4/3}\kappa^{4/3}r^{2/3}s^*)$. By comparing with [9], it is not optimal because its dependence on μ and r is not optimal. In fact, the sample complexity of the nonconvex optimization algorithm is often suboptimal compared with convex relaxation based methods [18, 16, 27]. The detailed derivation can be found in the supplemental material.

5 Proof Sketch of the Main Theory

In this section, we present the proof sketch. The complete detailed proofs can be found in the supplementary material. We first provide some intuitions that explain why alternating minimization algorithm in Algorithm 1 is guaranteed to converge to the global optimal solution.

The first key property that enables our algorithm to converge is: for fixed $\mathbf{V}$ and $\mathbf{S}$, the loss function is strongly convex with respect to $\mathbf{U}$ under suitable conditions. The same property is satisfied by $\mathbf{V}$ and $\mathbf{S}$ as well. For the ease of presentation, we summarize this property as the following lemma.

Lemma 5.1 (Strong Convexity)**.** The function $\mathcal{L}(\mathbf{U}, \mathbf{V}^*, \mathbf{S}^*)$ is λ_1-strongly convex, the function $\mathcal{L}(\mathbf{U}^*, \mathbf{V}, \mathbf{S}^*)$ is λ_2-strongly convex, and the function $\mathcal{L}(\mathbf{U}^*, \mathbf{V}^*, \mathbf{S})$ is λ_3-strongly convex. That is, for any $\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2, \mathbf{S}_1, \mathbf{S}_2$, it holds that

$$\begin{aligned}
&\mathcal{L}(\mathbf{U}_1, \overline{\mathbf{V}}^*, \mathbf{S}^*) - \mathcal{L}(\mathbf{U}_2, \overline{\mathbf{V}}^*, \mathbf{S}^*) \\
&\quad - \langle \nabla_{\mathbf{U}}\mathcal{L}(\mathbf{U}_2, \overline{\mathbf{V}}^*, \mathbf{S}^*), \mathbf{U}_1 - \mathbf{U}_2\rangle \geq \lambda_1/2 \cdot \|\mathbf{U}_1 - \mathbf{U}_2\|_F^2, \\
&\mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}_1, \mathbf{S}^*) - \mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}_2, \mathbf{S}^*) \\
&\quad - \langle \nabla_{\mathbf{V}}\mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}_2, \mathbf{S}^*), \mathbf{V}_1 - \mathbf{V}_2\rangle \geq \lambda_2/2 \cdot \|\mathbf{V}_1 - \mathbf{V}_2\|_F^2, \\
&\mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}^*, \mathbf{S}_1) - \mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}^*, \mathbf{S}_2) \\
&\quad - \langle \nabla_{\mathbf{S}}\mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}^*, \mathbf{S}_2), \mathbf{S}_1 - \mathbf{S}_2\rangle \geq \lambda_3/2 \cdot \|\mathbf{S}_1 - \mathbf{S}_2\|_F^2.
\end{aligned}$$

It is easy to verify that Lemma 5.1 holds for both multi-task learning and RPCA. For multi-task learning, it holds with $\lambda_1 = \lambda_2 = 1 - \delta_{2r}$ and $\lambda_3 = 1 - \zeta_{2s}$. For RPCA, it holds with $\lambda_1 = \lambda_2 = \lambda_3 = 1$. We emphasize that for many other design matrices, the loss function satisfies the strong convexity.

The second key gradient that ensures our algorithm to converge to the global optimal solution is referred to as *bounded difference of gradients*.

Definition 5.2 (Bounded Difference of Gradients)**.** Let $\mathbf{U}^{(t+1)}$ be the solution of ∇. Then there exists $L_1, L_1', L_2, L_2', L_3, L_3' > 0$ such that

$$\begin{aligned}
&\big\|\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^{(t)}, \overline{\mathbf{S}}^{(t)}\big) - \nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^*, \mathbf{S}^*\big)\big\|_F \\
&\quad \leq L_1\big\|\overline{\mathbf{V}}^{(t)} - \overline{\mathbf{V}}^*\big\|_F + L_1'\big\|\overline{\mathbf{S}}^{(t)} - \overline{\mathbf{S}}^*\big\|_F, \\
&\big\|\nabla_{\mathbf{V}}\mathcal{L}(\overline{\mathbf{U}}^{(t+1)}, \mathbf{V}^{(t+1)}, \overline{\mathbf{S}}^{(t)}) - \nabla_{\mathbf{V}}\mathcal{L}(\overline{\mathbf{U}}^*, \mathbf{V}^{(t+1)}, \mathbf{S}^*)\big\|_F \\
&\quad \leq L_2\big\|\overline{\mathbf{U}}^{(t+1)} - \overline{\mathbf{U}}^*\big\|_F + L_2'\big\|\overline{\mathbf{S}}^{(t)} - \overline{\mathbf{S}}^*\big\|_F, \\
&\big\|\nabla_{\mathbf{S}}\mathcal{L}\big(\overline{\mathbf{U}}^{(t+1)}, \overline{\mathbf{V}}^{(t+1)}, \mathbf{S}^{(t+1)}\big) - \nabla_{\mathbf{S}}\mathcal{L}\big(\overline{\mathbf{U}}^*, \overline{\mathbf{V}}^*, \mathbf{S}^{(t+1)}\big)\big\|_F \\
&\quad \leq L_3\big\|\overline{\mathbf{U}}^{(t+1)} - \overline{\mathbf{U}}^*\big\|_F + L_3'\big\|\overline{\mathbf{V}}^{(t+1)} - \overline{\mathbf{V}}^*\big\|_F.
\end{aligned}$$

Note that the bounded difference of gradients property is similar to the Lipschitz property on gradient. A similar conditions has been proposed in [10] for the analysis of Wirtinger flow. Taking the first inequality in Definition 5.2 for example. Here the Lipschitz gradient property is defined only between the true parameters $\overline{\mathbf{V}}^*, \mathbf{S}^*$ and arbitrary $\overline{\mathbf{V}}^{(t)}, \overline{\mathbf{S}}^{(t)}$, rather than between arbitrary pair of parameters $\overline{\mathbf{V}}, \overline{\mathbf{S}}$. Intuitively speaking, if we set $\overline{\mathbf{V}}$ and $\overline{\mathbf{S}}$ to $\overline{\mathbf{V}}^*$ and $\overline{\mathbf{S}}^*$ respectively, and optimize $\mathbf{U}$ by solving

$$\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^*, \mathbf{S}^*\big) = 0,$$

we can immediately obtain $\mathbf{U}^*$ because of the strong convexity in Lemma 5.1. However, in practice, we do not know $\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^*, \mathbf{S}^*\big)$, and we use $\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^{(t)}, \overline{\mathbf{S}}^{(t)}\big)$ to approximate $\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^*, \mathbf{S}^*\big)$. In other words, $\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^{(t)}, \overline{\mathbf{S}}^{(t)}\big)$ can be seen as an approximation of its convex counterpart, i.e., $\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^*, \mathbf{S}^*\big)$. As long as this approximation is sufficiently accurate, we still can gradually approach $\mathbf{U}^*$ by solving $\nabla_{\mathbf{U}}\mathcal{L}\big(\mathbf{U}^{(t+1)}, \overline{\mathbf{V}}^{(t)}, \overline{\mathbf{S}}^{(t)}\big) = 0$. Similar arguments apply to $\mathbf{V}^{(t+1)}$ and $\mathbf{S}^{(t+1)}$ as well. In the supplementary material, we will show that the loss function satisfies the bounded difference of gradients property.

In Assumption 4.1, we assume that $\overline{\mathbf{U}}^*$ and $\overline{\mathbf{V}}^*$ are μ-incoherent. The following lemma shows that the iterates after orthonormalization in the process of alternating minimization are also μ-incoherent.

Lemma 5.3 (Incoherence)**.** Under the assumption that $\|\overline{\mathbf{S}}^{(0)} - \mathbf{S}^*\|_2 \le \sigma_1$. We have

$$\max_{1\le i\le d_1} \big\|\mathbf{e}_i^\top \overline{\mathbf{U}}^{(t+1)}\big\|_2 \le \frac{16\sigma_1\mu}{\sigma_r}\sqrt{\frac{r}{d_1}},$$
$$\max_{1\le i\le d_2} \big\|\mathbf{e}_i^\top \overline{\mathbf{V}}^{(t+1)}\big\|_2 \le \frac{16\sigma_1\mu}{\sigma_r}\sqrt{\frac{r}{d_2}}.$$

As we mentioned before, the truncation step in Algorithm 1 is essential to take into account the sparsity constraint. The following lemma characterizes the error incurred by the truncation step in Algorithm 1. More specifically, it ensure that the error after truncation step $\overline{\mathbf{S}}^{(t+1)} - \mathbf{S}^*$ can be upper bounded by the error before truncation step $\mathbf{S}^{(t+1)} - \mathbf{S}^*$ up to some problem dependent parameters.

Lemma 5.4 (Truncation)**.** Suppose that we have $\big\|\mathbf{S}^{(t+1)} - \mathbf{S}^*\big\|_F \le c\big\|\mathbf{S}^*\big\|_F$ for some $c \in (0.18, 5.8)$, and $s \ge \frac{4c}{(1-c)^2}s^*$, then it holds that

$$\big\|\overline{\mathbf{S}}^{(t+1)} - \mathbf{S}^*\big\|_F \le \big(1 + 2\sqrt{s^*/s}\big)\big\|\mathbf{S}^{(t+1)} - \mathbf{S}^*\big\|_F.$$

Lemma 5.4 indicates that in Algorithm 1, the parameter s for the truncation step should be set sufficiently larger than s^*, e.g., $s = 4s^*$.

The next lemma establishes the recurrence for the convergence of $\mathbf{U}^{(t+1)}, \overline{\mathbf{U}}^{(t+1)}, \mathbf{V}^{(t+1)}, \overline{\mathbf{V}}^{(t)}$ and $\mathbf{S}^{(t)}$.

Lemma 5.5. Under the same conditions of Theorem 4.4, we have

$$\begin{aligned}\big\|\mathbf{U}^{(t+1)} - \mathbf{U}^*\big\|_F &\le \nu\mu\sqrt{\frac{r}{d_2}}\frac{s^{3/2}}{d_1}\big\|\mathbf{S}^* - \overline{\mathbf{S}}^{(t)}\big\|_F \\ &\quad + \frac{\sigma_1}{2}(1+\delta_{2r})\big\|\overline{\mathbf{V}}^* - \overline{\mathbf{V}}^{(t)}\big\|_F^2, \\ \big\|\mathbf{V}^{(t+1)} - \mathbf{V}^*\big\|_F &\le \nu\bigg(3\delta_1\sqrt{r} + \mu\sqrt{\frac{r}{d_1}}\frac{s^{3/2}}{d_2}\bigg)\big\|\mathbf{S}^* - \overline{\mathbf{S}}^{(t)}\big\|_F \\ &\quad + \frac{\sqrt{1+\delta_{2r}}(1+\delta_1)^{3/2}}{1-\delta_1}\sigma_1\|\overline{\mathbf{U}}^{(t+1)} - \overline{\mathbf{U}}^*\|_F, \\ \big\|\overline{\mathbf{S}}^{(t+1)} - \mathbf{S}^*\big\|_F &\le (1+3\delta_1)(1+2\sqrt{s^*/s}) \\ &\quad \cdot\Big[\big(1 + \big\|\overline{\mathbf{U}}^* - \overline{\mathbf{U}}^{(t+1)}\big\|_F\big)\big\|\mathbf{V}^{(t+1)} - \mathbf{V}^*\big\|_F \\ &\quad + \sigma_1\big\|\overline{\mathbf{U}}^* - \overline{\mathbf{U}}^{(t+1)}\big\|_F\Big].\end{aligned}$$

The following lemma characterizes the effect the orthonormalization step using the QR decomposition. In other words, it bounds the distance between $\overline{\mathbf{U}}^{(t+1)}$ (resp. $\overline{\mathbf{V}}^{(t+1)}$) and $\mathbf{U}^*$ (resp. $\mathbf{V}^*$) by the distance between $\mathbf{U}^{(t+1)}$ (resp. $\mathbf{V}^{(t+1)}$) and $\mathbf{U}^*$ (resp. $\mathbf{V}^*$).

Lemma 5.6. We have

$$\begin{aligned}\big\|\overline{\mathbf{U}}^{(t+1)} - \overline{\mathbf{U}}^*\big\|_F &\le \sqrt{2}/\sigma_r\big(2\sigma_1 + \big\|\overline{\mathbf{S}}^{(t)} - \mathbf{S}^*\big\|_2\big) \\ &\quad \cdot\big\|\mathbf{U}^{(t+1)} - \mathbf{U}^*\big\|_F. \\ \big\|\overline{\mathbf{V}}^{(t+1)} - \overline{\mathbf{V}}^*\big\|_F &\le \sqrt{2}/\sigma_r\bigg(\frac{2+2\delta_1}{1-\delta_1}\sigma_1 \\ &\quad + \frac{1+3\delta_1}{1-\delta_1}\big\|\overline{\mathbf{S}}^{(t)} - \mathbf{S}^*\big\|_2\bigg)\big\|\mathbf{V}^{(t+1)} - \mathbf{V}^*\big\|_F.\end{aligned}$$

Proof of Theorem 4.4. Recall that we have

$$\begin{aligned}&\big\|\mathbf{L}^{(t+1)} - \mathbf{L}^*\big\|_F = \big\|\overline{\mathbf{U}}^{(t+1)}\mathbf{V}^{(t+1)\top} - \overline{\mathbf{U}}^*\mathbf{V}^{*\top}\big\|_F \\ &\le \big(1 + \big\|\overline{\mathbf{U}}^* - \overline{\mathbf{U}}^{(t+1)}\big\|_F\big)\big\|\mathbf{V}^{(t+1)} - \mathbf{V}^*\big\|_F \\ &\quad + \sigma_1\big\|\overline{\mathbf{U}}^* - \overline{\mathbf{U}}^{(t+1)}\big\|_F \\ &\le \sigma_1\bigg\{\big\|\overline{\mathbf{U}}^* - \overline{\mathbf{U}}^{(t+1)}\big\|_F + \bigg[1 + \frac{1}{\sqrt{(1+\delta_{2r})(1+\delta_1)}} \\ &\quad \cdot\bigg(3\delta_1\sqrt{r} + \mu\sqrt{\frac{r}{d_1}}\frac{s^{3/2}}{d_2}\bigg)\bigg]/\sigma_1\bigg\}\big\|\mathbf{V}^{(t+1)} - \mathbf{V}^*\big\|_F.\end{aligned} \tag{5.1}$$

Since we assume that

$$\begin{aligned}&\bigg[1 + \frac{1}{\sqrt{(1+\delta_{2r})(1+\delta_1)}}\bigg(3\delta_1\sqrt{r} + 8\mu\sqrt{r}\Big(\frac{s^*}{d}\Big)^{3/2}\bigg)\bigg] \\ &\cdot\bigg[1 - 24\sqrt{2}\nu\kappa\mu\sqrt{r}\Big(\frac{s^*}{d}\Big)^{3/2}\sigma_1\bigg] \le 3\sqrt{2}\kappa\sigma_1,\end{aligned}$$

Table 2: Comparison of our algorithm with convex relaxation based method for multi-task learning in terms of $\|\widehat{\mathbf{L}}-\mathbf{L}^*\|_F$ ($\times 10^{-2}$) and time (in second).

	$n=100$		$n=1000$		$n=10^4$	
Methods	Error	Time	Error	Time	Error	Time
Proximal Gradient [36]	1.57±0.07	0.47	4.36±0.04	626.86	N/A	N/A
Ours	0.00±0.00	0.07	0.00±0.00	4.45	0.00±0.00	293.24

Table 3: Comparison of optimizations algorithms for robust PCA in terms of $\|\widehat{\mathbf{L}}-\mathbf{L}^*\|_F$ ($\times 10^{-2}$) and time (in second). N/A means the algorithm did not output the solution in an hour.

	$d=100, r=5$		$d=500, r=20$		$d=5000, r=50$	
Methods	Error	Time	Error	Time	Error	Time
ADMM [23]	0.00±0.00	5.45	0.00±0.00	857.82	N/A	N/A
Nonconvex [28]	4.45±0.97	0.57	6.48±0.67	1.05	16.6±3.6	266.05
Ours	0.00±0.00	0.15	0.00±0.00	2.16	0.00±0.00	92.51

the right hand side of (5.1) can be further bounded by

$$\begin{aligned}&\big\|\mathbf{L}^{(t+1)}-\mathbf{L}^*\big\|_F\\&\leq \sigma_1\big\|\overline{\mathbf{U}}^{(t+1)}-\overline{\mathbf{U}}^*\big\|_F+3\sqrt{2}\kappa\big\|\mathbf{V}^{(t+1)}-\mathbf{V}^*\big\|_F\\&\leq \sigma_1\Big[\big\|\overline{\mathbf{U}}^{(t+1)}-\overline{\mathbf{U}}^*\big\|_F+\gamma\big\|\mathbf{V}^{(t+1)}-\mathbf{V}^*\big\|_F\Big]\\&\leq \sigma_1\rho\Big[\big\|\overline{\mathbf{U}}^{(t)}-\overline{\mathbf{U}}^*\big\|_F+\gamma\big\|\mathbf{V}^{(t)}-\mathbf{V}^*\big\|_F\Big]\\&\leq \sigma_1\rho^t\Big[\big\|\overline{\mathbf{U}}^{(0)}-\overline{\mathbf{U}}^*\big\|_F+\gamma\big\|\mathbf{V}^{(0)}-\mathbf{V}^*\big\|_F\Big].\end{aligned}$$

This completes the proof. □

6 Experiments

In this section, we will present numerical results on both synthetic and real datasets to verify the performance of the proposed algorithms, and compare it with convex relaxation based algorithm, i.e., ADMM. In particular, we investigate both RPCA and multi-task learning. For RPCA, besides ADMM [23], we also compare our algorithm with the very recent nonconvex optimization algorithm [28]. For multi-task learning, we compare with proximal gradient descent [36]. The implementations of the baseline algorithms are downloaded from the authors' website. We use two measure to evaluate the algorithms. The first measure is estimation error based on Frobenius norm, i.e., $\|\widehat{\mathbf{L}}-\mathbf{L}^*\|_F$. The second measure is the computational time in second. For our algorithm, we initialize $\overline{\mathbf{U}}^{(1)}$ as a semi-orthogonal matrix, and $\mathbf{S}^{(1)}$ as a zero matrix. We set r as the true rank, and s as $4s^*$.

6.1 Synthetic Data

RPCA In the first experiment, following [9, 28], the low rank matrix $\mathbf{L}^*=\mathbf{U}\mathbf{V}^\top$ is generated using normally distributed $\mathbf{U}\in\mathbb{R}^{d_1\times r}$ and $\mathbf{V}\in\mathbb{R}^{d_2\times r}$. In addition, supp($\mathbf{S}^*$) is generated by sampling a uniformly random subset $[d_1]\times[d_2]$ with size $\|\mathbf{S}^*\|_{0,0}=s$, and each nonzero element S^*_{ij} is drawn i.i.d. from the uniform distribution over $[r/(2\sqrt{d_1d_2}), r/\sqrt{d_1d_2}]$. In the simulation, we set $d_1=d_2=d$. We study different settings, i.e., (1) $d=100, r=5$, (2) $d=500, r=10$, and (3) $d=5000, r=50$. In all settings, we set s as $0.25d^2$. In other words, 25% of the entries are corrupted with noise. For each setting, we repeat the experiments for 20 times, and report the mean and standard deviation of the estimation error, as well as its average computational time in Table 3. Note that we did not show $\|\widehat{\mathbf{S}}-\mathbf{S}^*\|_F$ because it is identical to $\|\widehat{\mathbf{L}}-\mathbf{L}^*\|_F$. We can see that our proposed algorithm can exactly recover the low-rank and the sparse components. This is consistent with our theory. In addition, our algorithm is the fastest algorithm. ADMM [23] can also exactly recovers the low-rank and the sparse matrices for $d=100$ and $d=500$. However, it takes much longer time and it cannot be scaled up to $d=5000$. In addition, the nonconvex algorithm proposed in [28] is also very efficient but still slower than our algorithm. However, its estimation error is much worse and it cannot exactly recover the low rank and sparse matrices.

Multi-task learning In the second experiment, we compare our algorithm with proximal gradient descent implemented in [36] for multi-task learning. We generate $\mathbf{L}^*$ and $\mathbf{S}^*$ using the same method as adopted in RPCA. In addition, we generate each entry of $\mathbf{X}$ from standard normal distribution. Then we can generate $\mathbf{Y}$ from $\mathbf{X}, \mathbf{L}^*$ and $\mathbf{S}^*$. In this experiment, we also set $d_1=d_2=d$, and try three parameter settings: (1) $n=100, d=50, r=5$, (2) $n=1000, d=500, r=20$, and (3) $n=10^4, d=2000, r=50$. For each setting, we repeat the experiments for 20 times. Table 2 shows the experimental results. We can see that our proposed algorithm can exactly recover the low-rank and the sparse components. This again confirms our theory. In addition, our algorithm is much faster than the proximal gradient algorithm. In sharp contrast, the proximal

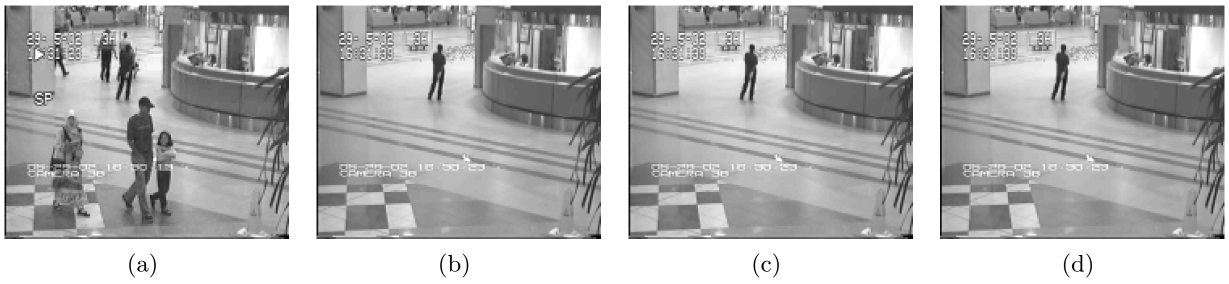

Figure 2: Background estimation for the "Hall of a business building" video. (a) One of the original image frame (b) Background frame estimated by ADMM algorithm [23] for RPCA. (c) Background frame estimated by Nonconvex method [28] for RPCA. (d) Background frame estimated by our proposed alternating minimization algorithm for RPCA.

gradient algorithm [23] fails to exactly recovers the low-rank and the sparse matrices. The main reason is that it has two regularization parameters in (1.2), which are difficult to tune. Moreover, it cannot be scaled up to $n = 10^4, d = 2000, r = 50$. Figure 1 illustrates the geo-

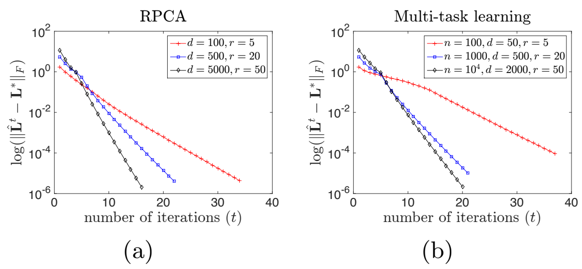

Figure 1: Evolution of the estimation error $\left\|\widehat{\mathbf{L}}^{(t)} - \mathbf{L}^*\right\|_F$ (in logarithmic scale). (a) The alternating minimization algorithm for RPCA. (b) The alternating minimization algorithm for multi-task learning.

metric decay of Algorithm 1 for RPCA and multi-task learning, as predicted by Theorem 4.4. In particular, we plot the logarithm of $\left\|\widehat{\mathbf{L}} - \mathbf{L}^*\right\|_F$ against the iteration number t. It is obvious that our proposed algorithm enjoys linear convergence rate.

6.2 Real Data

In this subsection, we evaluate the proposed RPCA for background modeling [22]. Video frames with a static background constitute a set of coherent images. It is often desirable to detect any activity in the foreground for surveillance purposes. A video sequence satisfies the low-rank plus sparse structure, because backgrounds of all the frames are related, while the variation and the moving objects are sparse and independent. Therefore, RPCA has been widely used to separate the foreground pixels from the background [9, 37, 28]. We apply RPCA to one surveillance video "Hall of a business building" from [22]. In detail, this video is composed of 200 frames with the resolution 144×176. We convert each frame to a vector and use the first 200 frames of that video to form a data matrix. Thus the resulting data matrix $\mathbf{X}$ is of size $25,344 \times 200$. We show the estimated background frames by different algorithms of RPCA (i.e., the low-rank matrix $\widehat{\mathbf{L}}$) in Figure2. The results of ADMM, Nonconvex method and our algorithm are comparable with each other. However, compared with ADMM (about 18 minutes to process the video sequence) and Nonconvex method (about 1 minute to process the video sequence), our algorithm only takes around 22 seconds to process the video sequence. Therefore, our algorithm is much faster. Since this is a real data, those assumptions/conditions in our theory may be violated on it. Nevertheless, our algorithm still outputs a good estimator as the state-of-art methods. This verifies the effectiveness of our algorithm on real data.

7 Conclusions

In this paper, we present a nonconvex optimization algorithm for low-rank plus sparse structure pursuit based on alternating minimization. We show that with certain initialization, the proposed nonconvex optimization algorithm enjoys linear convergence to the global optima and exactly recovers the underlying low rank and sparse matrices under standard conditions. We applied our algorithm to multi-task learning and RPCA on synthetic datasets, and validate our theory.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. Gu is supported by his startup funding at Department of Systems and Information Engineering, University of Virginia. Wang and Liu are partially supported by the grants NSF IIS1408910, NSF IIS1332109, NIH R01MH102339, NIH R01GM083084, and NIH R01HG06841.

References

[1] Alekh Agarwal, Sahand Negahban, and Martin J. Wainwright. Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions. *The Annals of Statistics*, 40(2):1171–1197, 04 2012. doi: 10.1214/12-AOS1000.

[2] Alekh Agarwal, Animashree Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon. Learning sparsely used overcomplete dictionaries via alternating minimization. *arXiv preprint arXiv:1310.7991*, 2013.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[4] Sanjeev Arora, Rong Ge, Tengyu Ma, and Ankur Moitra. Simple, efficient, and neural algorithms for sparse coding. *arXiv preprint arXiv:1503.00778*, 2015.

[5] Sivaraman Balakrishnan, Martin J Wainwright, and Bin Yu. Statistical guarantees for the em algorithm: From population to sample-based analysis. *arXiv preprint arXiv:1408.2156*, 2014.

[6] Srinadh Bhojanapalli, Prateek Jain, and Sujay Sanghavi. Tighter low-rank approximation via sampling the leveraged element. *arXiv preprint arXiv:1410.3886*, 2014.

[7] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, 2012.

[8] Emmanuel J Candés and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

[9] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11, 2011.

[10] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *Information Theory, IEEE Transactions on*, 61(4):1985–2007, 2015.

[11] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[12] Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

[13] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *KDD*, pages 42–50. ACM, 2011.

[14] Yudong Chen and Martin J Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

[15] Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust pca via stochastic optimization. In *NIPS*, pages 404–412, 2013.

[16] Moritz Hardt. Understanding alternating minimization for matrix completion. In *FOCS*, pages 651–660. IEEE, 2014.

[17] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*, 2012.

[18] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *STOC*, pages 665–674, 2013.

[19] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. A dirty model for multi-task learning. In *NIPS*, pages 964–972, 2010.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[21] Anastasios Kyrillidis and Volkan Cevher. Matrix alps: Accelerated low rank and sparse matrix reconstruction. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 185–188. IEEE, 2012.

[22] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *Image Processing, IEEE Transactions on*, 13(11):1459–1472, 2004.

[23] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[24] Karim Lounici, Massimiliano Pontil, Alexandre B Tsybakov, and Sara Van De Geer. Taking advantage of sparsity in multi-task learning. *arXiv preprint arXiv:0903.1468*, 2009.

[25] Sahand Negahban and Martin J Wainwright. Joint support recovery under high-dimensional scaling: Benefits and perils of $\ell_{1,\infty}$-regularization. *NIPS*, 21:1161–1168, 2008.

[26] Sahand Negahban and Martin J. Wainwright. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, 39(2):1069–1097, 04 2011.

[27] Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. Phase retrieval using alternating minimization. In *NIPS*, pages 2796–2804, 2013.

[28] Praneeth Netrapalli, UN Niranjan, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-

convex robust pca. In *NIPS*, pages 1107–1115, 2014.

[29] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[30] Zhaoran Wang, Quanquan Gu, Yang Ning, and Han Liu. High dimensional em algorithm: Statistical optimization and asymptotic normality. In *Advances in Neural Information Processing Systems*, pages 2512–2520, 2015.

[31] Andrew E Waters, Aswin C Sankaranarayanan, and Richard Baraniuk. Sparcs: Recovering low-rank and sparse matrices from compressive measurements. In *NIPS*, pages 1089–1097, 2011.

[32] Lin Xiao and Tong Zhang. A proximal-gradient homotopy method for the sparse least-squares problem. *arXiv preprint arXiv:1203.3002*, 2012.

[33] Qi Yan, Jieping Ye, and Xiaotong Shen. Simultaneous pursuit of sparseness and rank structures for matrix decomposition. *Journal of Machine Learning Research*, 16:47–75, 2015.

[34] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14(1):899–925, April 2013. ISSN 1532-4435.

[35] Tuo Zhao, Zhaoran Wang, and Han Liu. A nonconvex optimization framework for low rank matrix estimation. In *Advances in Neural Information Processing Systems*, pages 559–567, 2015.

[36] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011. URL `http://www.public.asu.edu/~jye02/Software/MALSAR`.

[37] Tianyi Zhou and Dacheng Tao. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. 2011.