Nearly Optimal Robust Matrix Completion

Yeshwanth Cherapanamjeri 1 Kartik Gupta 1 Prateek Jain 1

Abstract
In this paper, we consider the problem of Robust Matrix Completion (RMC) where the goal is to recover a low-rank matrix by observing a small number of its entries out of which a few can be arbitrarily corrupted. We propose a simple projected gradient descent-based method to estimate the low-rank matrix that alternately performs a projected gradient descent step and cleans up a few of the corrupted entries using hard-thresholding. Our algorithm solves RMC using nearly optimal number of observations while tolerating a nearly optimal number of corruptions. Our result also implies significant improvement over the existing time complexity bounds for the low-rank matrix completion problem. Finally, an application of our result to the robust PCA problem (low-rank+sparsely corrupted low rank matrix into its low rank and global positioning. Note that it is a special case of the RMC problem where the matrix $S^* = 0$. State-of-the-art result for MC uses nuclear norm minimization and requires $|\Omega| \geq \mu^2 m^3 \log^2 n$ under standard $\mu$-incoherence assumption (see Section 3). But it requires $O(m^2 n)$ time in general. The best sample complexity result for a non-convex iterative method (with at most logarithmic dependence on the condition number of $L^*$) achieve exact recovery when $|\Omega| \geq \mu^6 m^5 \log^2 n$ and needs $O(|\Omega| r)$ computational steps.

Robust PCA (RPCA): RPCA aims to decompose a sparsely corrupted low rank matrix into its low rank and sparse components. It corresponds to another special case of RMC where the whole matrix is observed. State-of-the-art results for RPCA shows exact recovery of a rank-$r$, $\mu$-incoherent $L^*$ (see Assumption 1, Section 3) if at most $\rho = O \left( \frac{1}{\mu r^2} \right)$ fraction of the entries in each row/column of $S^*$ are corrupted (Hsu et al., 2011; Netrapalli et al., 2014). Moreover, St-NcRPCA algorithm (Netrapalli et al., 2014) solves the problem in time $O(m n^2 r^2)$.

Therefore, an efficient solution to the RMC problem implies efficient solutions to both the MC and RPCA problems. In this work, we attempt to answer the following open question (assuming $m \leq n$):

Can RMC be solved exactly by using $|\Omega| = O(r n \log n)$ observations out of which $O(\frac{1}{\mu r^2})$ fraction of the observed

1Microsoft Research India. Correspondence to: Prateek Jain <prajain@microsoft.com>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).
entries in each row/column are corrupted?

Note that both $|\Omega|$ (for uniformly random $\Omega$) and $\rho$ values mentioned in the question above denote the information theoretic limits. Hence, the goal is to solve RMC for nearly-optimal number of samples and nearly-optimal fraction of corruptions.

The existing state-of-the-art results for RMC with optimal $\rho = \frac{1}{\sqrt{\kappa}r^2}$ fraction of corrupted entries, either require at least a constant fraction of the entries of $L^*$ to be observed (Chen et al., 2011; Candès et al., 2011) or require restrictive assumptions like the support of corruptions being uniformly random (Li, 2013). Klopp et al. (2014) also considers RMC problem but studies the noisy setting and does not provide exact recovery bounds. Moreover, most of the existing methods for RMC use convex relaxation for both low-rank and sparse components, and in general exhibit large time complexity ($O(m^2n)$).

Under standard assumptions on $L^*$, $S^*$, $\Omega$ and for $n = O(m)$, we answer the above question in affirmative albeit with $|\Omega|$ which is $O(r)$ (ignoring log factors) larger than the optimal sample complexity (see Theorem 1). In particular, we propose a simple projected gradient (PGD) style method for RMC that alternately cleans up corrupted entries by hard-thresholding and performs a projected gradient descent update onto the space of low rank matrices; our method’s computational complexity is also nearly optimal ($O(|\Omega|r + (m + n)r^2 + r^3)$). Our algorithm is based on projected gradient descent for estimating $L^*$ and alternating projection on the set of sparse matrices for estimating $S^*$. Note that projection is onto non-convex sets of low-rank matrices (for $L^*$) and sparse matrices (for $S^*$), hence standard convex analysis techniques cannot be used for our algorithm.

Several recent results (Jain & Netrapalli, 2015; Netrapalli et al., 2014; Jain et al., 2014; Hardt & Wooters, 2014; Blumensath, 2011) show that under certain assumptions, projection onto non-convex sets indeed lead to provable algorithms with fast convergence to the global optimum. However, as explained in Section 3, RMC presents a unique set of challenges as we have to perform error analysis with the errors arising due to missing entries as well as sparse corruptions, both of which interact among themselves as well. In contrast, MC and RPCA only handle noise arising from one of the two sources. To overcome this challenge, we perform error analysis by segregating the effects due to random sampling and sparse corruptions and leverage their unique structure to obtain our result (See proof of Lemma 14). Another consequence of our careful error analysis is improved results for the MC as well as the RPCA problem.

Our empirical results on synthetic data demonstrates effectiveness of our method. We also apply our method to the foreground background separation problem and find that our method is an order of magnitude faster than the state-of-the-art method (St-NcRPCA) while achieving similar accuracy.

In a concurrent and independent work, Yi et al. (2016) also studied the RMC problem and obtained similar results. They study an alternating gradient descent style algorithm while our algorithm is based on low-rank projected gradient descent. Our sample complexity, corruption tolerance as well as time complexity differ along certain critical parameters: a) Sample complexity: Our sample complexity bound is dependent only logarithmically on $\kappa$, the condition number of the matrix $L^*$ (see Table 1). On the other hand, result of (Yi et al., 2016) depends quadratically on $\kappa$, which can be significantly large. Another difference is that our sample complexity bound depends logarithmically on the final error $\epsilon$ (defined as $\epsilon = \|L - L^*\|_2$); which for typical finite precision computation only introduces an extra constant factor. b) Corruption tolerance: Our result allows the fraction of corrupted entries to be information theoretic optimal (up to a constant) $O(\frac{1}{\mu^2r})$, while the result of (Yi et al., 2016) allows only $O\left(\min\left(\frac{1}{\mu r^2}, \frac{1}{\mu^2 r^2}\right)\right)$ fraction of corrupted entries. c) Time Complexity: As a consequence of the sample complexity bounds, running time of the method by (Yi et al., 2016) depends quintically on $\kappa$ whereas our algorithm only has a polylogarithmic dependence.

In summary, this paper’s main contributions are:

(a) RMC: We propose a nearly linear time method that solves RMC with $|\Omega| = O(nr^2\log^2 n \log^2 \|M\|_2/\epsilon)$ random entries and with optimal fraction of corruptions up to constant factors ($\rho = O(\frac{1}{\mu r^2})$).

(b) Matrix Completion: Our result improves upon the existing linear time algorithm’s sample complexity by an $O(1/r^2)$ factor, and time complexity by $O(1/r^2)$ factor, although with an extra $O(\log \|L^*\|_2/\epsilon)$ factor in both time and sample complexity.

(c) RPCA: We present a nearly linear time ($O(nr^3)$) algorithm for RPCA under optimal fraction of corruptions, improving upon $O(mnr^2)$ time complexity of the existing methods.

Notations: We assume that $M = L^* + S^*$ and $P_{\Omega}(M) = P_{\Omega}(L^*) + S^*$, i.e., $S^* = P_{\Omega}(S^*)$. $\|v\|_p$ norm of a vector $v$: $\|v\|$ denotes $\ell_2$ norm of $v$: $\|A\|_2$, $\|A\|_F$, $\|A\|$ denotes the operator, Frobenius, and nuclear norm of $A$, respectively; by default $\|A\| = \|A\|_2$. Operator $P_{\Omega}$ is given by $P_{\Omega}(A)$. $\sigma_i(A)$ denotes the $i$-th singular value of $A$ and $\sigma_i^*$ the $i$-th singular value of $L^*$. 

Nearly Optimal Robust Matrix Completion
Paper Organization: We present our main algorithm in Section 2 and our main results in Section 3. We also present an overview of the proof in Section 5. Section 4 presents our empirical results. Due to lack of space, we present most of the proofs and useful lemmas in the appendix.

2. Algorithm

In this section we present our algorithm for solving the RMC (Robust Matrix Completion) problem: given $\Omega$ and $P_\Omega(M)$ where $M = L^* + S^* \in \mathbb{R}^{m \times n}$, $rank(L^*) \leq r$, $\|S^*\|_0 \leq s$ and $S^* = P_\Omega(S^*)$, the goal is to recover $L^*$. To this end, we focus on solving the following non-convex optimization problem:

$$(L^*, S^*) = \arg\min_{L, S} \|P_\Omega(M) - P_\Omega(L) - S\|_F^2,$$

s.t., $rank(L) \leq r, P_\Omega(S) = S, \|S\|_0 \leq s$.  

(3)

For the above problem, we propose a simple iterative algorithm that combines projected gradient descent (for $L$) with alternating projections (for $S$). In particular, we maintain iterates $L^{(t)}$ and $S^{(t)}$, where $L^{(t)}$ is the current low-rank approximation of $L^*$ and $S^{(t)}$ is the current sparse approximation of $S^*$. $L^{(t+1)}$ is computed using gradient descent step for objective (3) and then projecting back onto the set of rank-$k$ matrices. That is,

$$L^{(t+1)} = P_k(L^{(t)} - \frac{1}{p}P_\Omega(M - L^{(t)} - S^{(t)})),$$

(4)

where $P_k(A)$ denotes projection of $A$ onto the set of rank-$k$ matrices and can be computed efficiently using SVD of $A$, $p = \frac{|\Omega|}{mn}$. $S^{(t+1)}$ is computed by projecting the residual $P_\Omega(M - L^{(t+1)})$ onto set of sparse matrices using a hard-thresholding operator, i.e.,

$$S^{(t+1)} = \mathcal{H}_{\zeta}(M - L^{(t+1)}),$$

(5)

where $\mathcal{H}_{\zeta}: \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ is the hard thresholding operator defined as: $(\mathcal{H}_{\zeta}(A))_{ij} = A_{ij}$ if $|A_{ij}| \geq \zeta$ and 0 otherwise. Intuitively, a better estimate of the sparse corruptions for each iteration will reduce the noise of the projected gradient descent step and a better estimate of the low rank matrix will enable better estimation of the sparse corruptions. Hence, under correct set of assumptions, the algorithm should recover $L^*, S^*$ exactly.

Unfortunately, just the above two simple iterations cannot handle problems where $L^*$ has poor condition number, as the intermediate errors can be significantly larger than the smallest singular values of $L^*$, making recovery of the corresponding singular vectors challenging. To alleviate this issue, we propose an algorithm that proceeds in stages. In the $q$-th stage, we project $L^{(t)}$ onto set of rank-$k_q$ matrices. Rank $k_q$ is monotonic w.r.t. $q$. Under standard assumptions, we show that we can increase $k_q$ in a manner such that after each stage $\|L^{(t)} - L^*\|_\infty$ decreases by at least a constant factor. Hence, the number of stages is only logarithmic in the condition number of $L^*$. See Algorithm 1 (PG-RMC) for a pseudo-code of the algorithm.

We require an upper bound of the first singular value for our algorithm to work. Specifically, we require $\sigma = O(\sigma^1)$. Alternatively, we can also obtain an estimate of $\sigma^1$ by using the thresholding technique from [Yi et al., 2016] although this requires an estimate of the number of corruptions in each row and column. We also use a simplified version of Algorithm 5 from [Hardt & Wootters, 2014] to form independent sets of samples for each iteration which is required for our theoretical analysis. Our algorithm has an “outer loop” (see Line 6) which sets rank $k_q$ of iterates $L^{(t)}$ appropriately (see Line 7). We then update $L^{(t)}$ and $S^{(t)}$ in the “inner loop” using (4, 5). We set threshold for the hard-thresholding operator using singular values of current gradient descent update (see Line 12). Note that, we divide $\Omega$ uniformly into $Q \cdot T$ sets, where $Q$ is an upper bound on the number of outer iterations and $T$ is the number of inner iterations. This division ensures independence across iterates that is critical to application of standard concentration bounds; such division is a standard technique in the matrix completion related literature (Jain & Netrapalli, 2013; Hardt & Wootters, 2014; Recht, 2011). Also, $q$ is a tunable parameter which should be less than one and is smaller for “easier” problems.

Note that updating $S^{(t)}$ requires $O(|\Omega| \cdot r + (m + n) \cdot r)$ computational steps. Computation of $L^{(t+1)}$ requires computing SVD for projection $P_r$, which can be computed in $O(|\Omega| \cdot r + (m + n) \cdot r^2 + r^3)$ time (ignoring log factors); see [Jain et al., 2010] for more details. Hence, the computational complexity of each step of the algorithm is linear in $|\Omega| \cdot r$ (assuming $|\Omega| \geq r \cdot (m + n)$). As we show in the next section, the algorithm exhibits geometric convergence rate under standard assumptions and hence the overall complexity is still nearly linear in $|\Omega|$ (assuming $r$ is just a constant).

Rank based Stagewise algorithm: We also provide a rank-based stagewise algorithm (R-RMC) where the outer loop increments $k_q$ by one at each stage, i.e., the rank is $q$ in the $q$-th stage. Our analysis extends for this algorithm as well, however, its time and sample complexity trades off a factor of $O(\log(\sigma_1/\epsilon))$ from the complexity of PG-RMC with a factor of $r$ (rank of $L^*$). We provide the detailed algorithm in Appendix B.3 due to lack of space (see Algorithm 3).
Algorithm 1 $\hat{L} = \text{PG-RMC} (\Omega, \mathcal{P}_{\Omega}(M), c, r, \mu, \eta, \sigma)$

1: Input: Observed entries $\Omega$, Matrix $\mathcal{P}_{\Omega}(M) \in \mathbb{R}^{m \times n}$, convergence criterion $c$, target rank $r$, incoherence parameter $\mu$, thresholding parameter $\eta$, estimate of first singular value $\sigma$
2: $T \leftarrow 10 \log \frac{2 m^2 n \sigma}{c}$, $Q \leftarrow T$
3: Partition $\Omega$ into $Q \cdot T + 1$ subsets $\Omega_0 \cup \{\Omega_{t,q} : q \in [Q], t \in [T]\}$ with $p$ set to $\frac{\Omega}{QT + 1}$ using algorithm 2
4: $L^{(0)} = 0$, $\zeta^{(0)} \leftarrow \eta \sigma$
5: $M^{(0)} = \max_{\Omega_{t,q}} \mathcal{P}_{\Omega_{t,q}}(M - \mathcal{H}_{\zeta^{(0)}}(M))$
6: $k_0 \leftarrow 0$, $q \leftarrow 0$
7: while $\sigma_{k+1}(M^{(0)}) > \frac{c}{\eta m}$ do
8: \hspace{1em} $q \leftarrow q + 1$
9: \hspace{1em} $k_q \leftarrow \left\{i : \sigma_i(M^{(0)}) \geq \frac{\sigma_{k+1}(M^{(0)})}{2}\right\}$
10: \hspace{1em} for Iteration $t$ = 0 to $T$ do
11: \hspace{2em} $S^{(t)} = \mathcal{H}_{\zeta^{(t)}}(\mathcal{P}_{\Omega_{t,q}}(M - L^{(t)}))$
12: \hspace{2em} $M^{(t)} = L^{(t)} - \frac{\sigma_{k_q}(M^{(t)})}{\eta} \mathcal{P}_{\Omega_{t,q}}(\mathcal{H}_{\zeta^{(t)}}(M^{(t)}))$
13: \hspace{2em} $L^{(t+1)} = \mathcal{P}_{\Omega_{t,q}}(M^{(t)})$
14: \hspace{2em} $\zeta^{(t+1)} \leftarrow \eta \left(\sigma_{k_q}(M^{(t)}) + \frac{1}{2} \sigma_{k_q}(M^{(t)})^{1/2}\right)
15: \hspace{1em} end for
16: $S^{(0)} = S^{(T)}$, $L^{(0)} = L^{(T+1)}$, $M^{(0)} = M^{(T)}$
17: $\zeta^{(0)} = \zeta^{(T+1)}$
18: Return: $L^{(T)}$

Algorithm 2 $\{\Omega_1, \ldots, \Omega_T\} = \text{SplitSamples}(\Omega, p, T)$

1: Input: Random samples with probability $T \cdot \Omega$, Required Sampling Probability $p$, Number of Sets $T$
2: Output: $T$ independent sets of entries $\{\Omega_1, \ldots, \Omega_T\}$ sampled with sampling probability $p$
3: $p' \leftarrow 1 - (1 - p)^T$
4: $\Omega'$ be sampled from $\Omega$ with each entry being included independently with probability $p'/p$
5: for $r = 1$ to $r = T$ do
6: \hspace{1em} $q_r \leftarrow \left\{i \in \Omega' : q_i \right\}$ for $i \in \{1, \ldots, T\}$
7: \hspace{1em} for Sample $s \in \Omega'$ do
8: \hspace{2em} for $t \in \{1, \ldots, T\}$ with probability $q_r$
9: \hspace{3em} Draw $r \in \{1, \ldots, T\}$ with probability $q_r$
10: \hspace{3em} Draw a random subset $S$ of size $r$ from $\{1, \ldots, T\}$
11: \hspace{3em} Add $s$ to $\Omega_t$ for $i \in S$
12: end for
13: end for

3. Analysis

We now present our analysis for both of our algorithms PG-RMC (Algorithm 1) and R-RMC (Algorithm 3). In general the problem of Robust PCA with Missing Entries is harder than the standard Matrix Completion problem and hence is NP-hard (Hardt et al., 2014). Hence, we need to impose certain (by now standard) assumptions on $L^*$, $\tilde{S}^*$ and $\Omega$ to ensure tractability of the problem:

**Assumption 1.** Rank and incoherence of $L^*$: $L^* \in \mathbb{R}^{m \times n}$ is a rank-$r$ incoherent matrix, i.e., $\|e_i^* U^*\|_2 \leq \mu \sqrt{\frac{r}{m}}$, $\|e_i^* V^*\|_2 \leq \mu \sqrt{\frac{r}{n}}$, $\forall i \in [m]$, $\forall j \in [n]$, where $L^* = U^* \Sigma^* (V^*)^\top$ is the SVD of $L^*$.

**Assumption 2.** Sampling ($\Omega$): $\Omega$ is obtained by sampling each entry with probability $p = \frac{\Omega}{m \Omega}$.

**Assumption 3.** Sparsity of $\tilde{S}^*$, $S^*$: We assume that at most $\rho \leq \frac{\rho}{\mu} \sigma$ fraction of the elements in each row and column of $S^*$ are non-zero for a small enough constant $c$. Moreover, we assume that $\Omega$ is independent of $\tilde{S}^*$. Hence, $S^* = \mathcal{P}_\Omega(\tilde{S}^*)$ also has $p \cdot \rho$ fraction of the entries in expectation.

Assumptions 1, 2 are standard assumptions in the provable matrix completion literature (Candes & Recht, 2009; Recht, 2011; Jain & Netrapalli, 2015), while Assumptions 1, 3 are standard assumptions in the robust PCA (low-rank + sparse matrix recovery) literature (Chandrasekaran et al., 2011; Candès et al., 2011; Hsu et al., 2011). Hence, our setting is a generalization of both the standard and popular problems and as we show later in the section, our result can be used to meaningfully improve the state-of-the-art for both of these problems.

We first present our main result for Algorithm 1 under the assumptions given above.

**Theorem 1.** Let Assumptions 1, 2 and 3 on $L^*$, $\tilde{S}^*$ and $\Omega$ hold respectively. Let $\mu \leq n$, $n = O(m)$, and let the number of samples $|\Omega|$ satisfy:

$$\mathbb{E}[|\Omega|] \geq C \alpha^2 \mu^4 r^2 n \log^2 (n) \log^2 \left(\frac{\mu^2 \sigma_1}{\epsilon}\right),$$

where $C$ is a global constant. Then, with probability at least $1 - n^{-2} \log \frac{2 \mu^2 \sigma_1}{\epsilon}$, Algorithm 1 with $\eta = \frac{4 \mu^2 \sigma_1}{\epsilon}$, at most $O(\log (\|M\|_2 / \epsilon))$ outer iterations and $O(\log (\frac{\mu^2 \sigma_1}{\epsilon}))$ inner iterations, outputs a matrix $\hat{L}$ such that:

$$\|\hat{L} - L^*\|_F \leq \epsilon.$$

Note that the number of samples matches information theoretic bound up to $O(r \log n \log^2 \sigma_1 / \epsilon)$ factor. Also, the number of allowed corruptions in $\tilde{S}^*$ also matches the known lower bounds (up to a constant factor) and cannot be improved upon information theoretically.

We now present our result for the rank based stagewise algorithm (Algorithm 3).

**Theorem 2.** Under Assumptions 1, 2 and 3 on $L^*$, $\tilde{S}^*$ and $\Omega$ respectively and $\Omega$ satisfying:

$$\mathbb{E}[|\Omega|] \geq C \alpha^2 \mu^4 r^3 n \log^2 (n) \log \left(\frac{\mu^2 r \sigma_1}{\epsilon}\right),$$
for a large enough constant $C$, then Algorithm 3 with $\eta$ set to $4\mu^r n$ outputs a matrix $\hat{L}$ such that: $\|\hat{L} - L^*\|_F \leq \epsilon$, w.p. 
$1 - n^{-\log 2} \cdot \frac{1}{n}$. Notice that the sample complexity of Algorithm 3 has an additional multiplicative factor of $O(\epsilon)$ when compared to that of Algorithm 1, but shaves off a factor of $O(\log(\kappa))$. Similarly, computational complexity of Algorithm 3 also trades off an $O(\log \kappa)$ factor from the computational complexity of Algorithm 1.

**Result for Matrix Completion:** Note that for $\tilde{S} = 0$, the RMC problem with Assumptions 1, 2 is exactly the same as the standard matrix completion problem and hence, we get the following result as a corollary of Theorem 1.

**Corollary 1 (Matrix Completion).** Suppose we observe $\Omega$ and $\mathcal{P}_\Omega(L^*)$ where Assumptions 1, 2 hold for $L^*$ and $\Omega$. Also, let $E[|\Omega|] \geq CN^2 \mu^r \sqrt{n} \log^2 n \log^2 \sigma_1 / \epsilon$ and $m \leq n$. Then, w.p. $1 - n^{-\log 2} \cdot \frac{1}{n}$, Algorithm 1 outputs $\hat{L}$ s.t. $\|\hat{L} - L^*\|_2 \leq \epsilon$.

Table 1 compares our sample and time complexity bounds for low-rank MC. Note that our sample complexity is nearly the same as that of nuclear-norm methods while the running time of our algorithm is significantly better than the existing results that have at most logarithmic dependence on the condition number of $L^*$.

**Result for Robust PCA:** Consider the standard Robust PCA problem (RPCA), where the goal is to recover $L^*$ from $M = L^* + \tilde{S}$. For RPCA as well, we can randomly sample $|\Omega|$ entries from $M$, where $\Omega$ satisfies the assumption required by Theorem 1. This leads us to the following corollary:

**Corollary 2 (Robust PCA).** Suppose we observe $M = L^* + S^*$, where Assumptions 1, 3 hold for $L^*$ and $S^*$. Generate $\Omega \in [m] \times [n]$ by sampling each entry independently with probability $p$, s.t., $E[|\Omega|] \geq C \mu^r \sqrt{n} \log^2 n \log^2 \sigma_1 / \epsilon$. Let $m \leq n$. Then, w.p. $1 - n^{-\log 2} \cdot \frac{1}{n}$, Algorithm 1 outputs $\hat{L}$ s.t. $\|\hat{L} - L^*\|_2 \leq \epsilon$.

Hence, using Theorem 1, we will still be able to recover $L^*$ but using only the sampled entries. Moreover, the running time of the algorithm is only $O(\mu^r \sqrt{n} \log^2 n \log^2 (\sigma_1 / \epsilon))$, i.e., we are able to solve RPCA problem in time almost linear in $n$. To the best of our knowledge, the existing state-of-the-art methods for RPCA require at least $O(n^2 \epsilon)$ time to perform the same task (Netrapalli et al., 2014; Gu et al., 2016). Similarly, we don’t need to load the entire data matrix in memory, but we can just sample the matrix and work with the obtained sparse matrix with at most linear number of entries. Hence, our method significantly reduces both run-time and space complexities, and as demonstrated empirically in Section 4 can help scale our algorithm to very large data sets without losing accuracy.

### 3.1. Proof Outline for Theorem 1

We now provide an outline of our proof for Theorem 1 and motivate some of our proof techniques; the proof of Theorem 2 follows similarly. Recall that we assume that $M = L^* + S^*$ and define $S^* = \mathcal{P}_\Omega(S^*)$. Similarly, we define $S(t) = \mathcal{H}_t(M - L(t))$. Critically, $S(t) = \mathcal{P}_\Omega(S(t))$ (see Line 9 of Algorithm 1), i.e., $S(t)$ is the set of iterates that we “could” obtain if entire $M$ was observed. Note that we cannot compute $\hat{S}(t)$, it is introduced only to simplify our analysis.

We first re-write the projected gradient descent step for $L(t+1)$ as described in (4):

$$L(t+1) = \mathcal{P}_{k_q} \left( L^* + \left( \tilde{S}^* - \tilde{S}(t) \right) + E_1 \right)$$

$$\left( I - \frac{\mathcal{P}_{\Omega_{k_q}}}{p} \right) \left( (L(t) - L^*) + (\tilde{S}(t) - \tilde{S}^*) \right)$$

That is, $L(t+1)$ is obtained by rank-$k_q$ SVD of a perturbed version of $L^*$: $L^* + E_1 + E_3$. As we perform entrywise thresholding to reduce $\|\tilde{S}^* - \tilde{S}(t)\|_\infty$, we need to bound $\|L(t+1) - L^*\|_\infty$. To this end, we use techniques from Jain & Netrapalli (2015), Netrapalli et al. (2014) that explicitly model singular vectors of $L(t+1)$ and argue about the infinity norm error using a Taylor series expansion. However, in our case, such an error analysis requires analyzing the following key quantities ($H = E_1 + E_3$):

$$A_j := \max_{q \in [m]} \|e_q^\top (H^\top H)^{\frac{1}{2}} V^*_j\|_2 \quad \forall 1 \leq j, \text{ s.t., } j \text{ even}$$

$$B_j := \max_{q \in [m]} \|e_q^\top (H H^\top)^{\frac{1}{2}} U^*_j\|_2$$

$$C_j := \max_{q \in [m]} \|e_q^\top H (H H^\top)^{\frac{1}{2}} V^*_j\|_2$$

$$D_j := \max_{q \in [m]} \|e_q^\top H (H^\top H)^{\frac{1}{2}} V^*_j\|_2 \quad \forall 1 \leq j, \text{ s.t., } j \text{ odd}$$

Note that $E_1 = 0$ in the case of standard RPCA which was analyzed in Netrapalli et al. (2014), while $E_3 = 0$ in the case of standard MC which was considered in Jain & Netrapalli (2015). In contrast, in our case both $E_2$ and $E_3$ are non-zero. Moreover, $E_3$ is dependent on random variable $\Omega$. Hence, for $j \geq 2$, we will get cross terms between $E_2$ and $E_1$ that will also have dependent random variables which precludes application of standard Bernstein-style tail bounds. To overcome this issue, we first carefully segregate the errors arising due to the randomness in the sampling process and the deterministic sparse corruptions in $S^*$. We do this by introducing $S(t)$ which is the sparse iterate we
would have obtained had the whole matrix been observed. This allows us to decompose the error term into the sum of $E_1$ and $E_3$ where $E_1$ represents the error due to the sparse corruptions and $E_3$ represents the error arising from the randomness in the sampling procedure. We then incorporate this decomposition into a careful combinatorial-style argument similar to that of Erdos et al. (2013) and Jain & Netrapalli (2015) to bound the above given quantity. That is, we can provide the following key lemma:

**Lemma 1.** Let $L^*$, $\Omega$, and $\tilde{S}^*$ satisfy Assumptions 1, 2 and 3 respectively. Let $L^* = U^*\Sigma^*(V^*)^\top$ be the singular value decomposition of $L^*$. Furthermore, suppose that the $t$th iteration of the $q$th stage of $\tilde{S}^{(t)}$ defined as $HT\zeta(M - L^{(t)})$, satisfies $\text{Supp}(\tilde{S}^{(t)}) \subseteq \text{Supp}(\tilde{S}^*)$, then we have:

$$
\max\{A, B, C, D\} \leq \mu \sqrt{\frac{r}{m}} \left( \rho n \|E_1\|_\infty \right)
$$

$$+ c \sqrt{n} \left( \|E_1 - E_2\|_\infty \log n \right)^a,$$

\forall c > 0, \forall 0 \leq j \leq \log n \text{ w.p. } 1 - n^{-2\log^4}, \text{ where } E_1, E_2 \text{ and } E_3 \text{ are defined in } \Box. \text{ } A, B, C, D \text{ are defined in } \Box.

**Remark:** We would like to note that even for the standard MC setting, i.e., when $E_1 = 0$, we obtain better bound than that of Jain & Netrapalli (2015) as we can bound $\max_i \|e_i^T(E_3)^{\top}U\|_2$ directly rather than the weaker $\sqrt{r} \max_i \|e_i^T(E_3)^{\top}u_j\|$ bound that Jain & Netrapalli (2015) uses.

Now, using Lemmas 1 and 2 and by using a hard-thresholding argument we can bound $\|L^{(t-1)} - L^*\|_\infty \leq 2\mu r \sigma_{k_{q+1}} + (\frac{1}{2}) \sigma_k$ (see Lemma 2) in the $q$th stage. Hence, after $O(\log(\sigma/\epsilon))$ “inner” iterations, we can guarantee in the $q$-th stage:

$$
\|L^{(t)} - L^*\|_\infty \leq \frac{4\mu r \sigma_{k_{q+1}}}{m}
$$

$$
\|E_1\|_\infty + \|E_2\|_\infty \leq \frac{20\mu r \sigma_{k_{q+1}}}{m}.
$$

Moreover, by using sparsity of $\tilde{S}^*$ and the special structure of $E_3$ (See Lemma 7), we have: $\|E_1 + E_2\|_2 \leq c \cdot \sigma_{k_{q+1}}$, where $c$ is a small constant.

Now, the outer iteration sets the next stage’s rank $k_{q+1}$ as: $k_{q+1} = \|i : \sigma_i(L^* + E_1 + E_2) \geq 0.5 \cdot \sigma_{k_{q+1}}(L^* + E_1 + E_2)\|$. Using the bound on $\|E_1 + E_2\|_2$ and Weyl’s eigenvalue perturbation bound (Lemma 2), we have: $\sigma_{k_{q+1}} \leq 0.6 \sigma_{k_{q+1}}$. Hence, after $Q = O(\log(\sigma/\epsilon))$ “outer” iterations, Algorithm 1 converges to an $\epsilon$-approximate solution to $L^*$.

4. Experiments

In this section we discuss the performance of Algorithm 1 on synthetic data and its use in foreground background separation. The goal of the section is two-fold: a) to demonstrate practicability and effectiveness of Algorithm 1 for the RMC problem, b) to show that Algorithm 1 indeed solves RPCA problem in significantly smaller time than that required by the existing state-of-the-art algorithm (St-NcRPCA (Netrapalli et al., 2014)). To this end, we use synthetic data as well as video datasets where the goal is to perform foreground-background separation (Candes et al., 2011).

We implemented our algorithm in MATLAB and the results for the synthetic data set were obtained by averaging over 20 runs. We obtained a matlab implementation of St-NcRPCA (Netrapalli et al., 2014) from the authors of (Netrapalli et al., 2014). Note that if the sampling probability is $p = 1$, then our method is similar to St-NcRPCA; the key difference being how the rank is selected in each stage.

### Table 1: Comparison of PG-RMC and R-RMC with Other Matrix Completion Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Sample Complexity</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuclear norm (Recht, 2011)</td>
<td>$O(\mu^2 r n \log^2 n)$</td>
<td>$O(n^3 \log \frac{1}{\epsilon})$</td>
</tr>
<tr>
<td>SVP (Jain &amp; Netrapalli, 2015)</td>
<td>$O(\mu^4 r^3 n \log^3 n)$</td>
<td>$O(\mu^4 r^3 n \log^3 n \log^2 \frac{1}{\epsilon})$</td>
</tr>
<tr>
<td>Alt. Min. (Hardt &amp; Wootters, 2014)</td>
<td>$O(n \mu^4 r^9 \log^3 (\kappa) \log^2 n)$</td>
<td>$O(n \mu^4 r^{13} \log^3 (\kappa) \log^2 n)$</td>
</tr>
<tr>
<td>Alt. Grad. Desc. (Sun &amp; Luo, 2015)</td>
<td>$O(n r \kappa^2 \max(\mu^2 \log n, \mu^4 r \kappa^4))$</td>
<td>$O(\mu^2 r \kappa^4 \log^4 \frac{1}{\epsilon})$</td>
</tr>
<tr>
<td><strong>R-RMC (This Paper)</strong></td>
<td>$O(\mu^4 r^3 n \log^2 (n) \log^2 \frac{1}{\epsilon}^*)$</td>
<td>$O(\mu^4 r^4 n \log^2 (n) \log^2 \frac{1}{\epsilon}^*)$</td>
</tr>
<tr>
<td><strong>PG-RMC (This Paper)</strong></td>
<td>$O(\mu^4 r^2 n \log^2 (n) \log^2 \frac{1}{\epsilon})$</td>
<td>$O(\mu^4 r^3 n \log^2 (n) \log^2 \frac{1}{\epsilon})$</td>
</tr>
</tbody>
</table>
We also implemented the Alternating Minimization based algorithm from (Gu et al. 2016). However, we found it to be an order of magnitude slower than Algorithm 1 on the foreground-background separation task. For example, on the escalator video, the algorithm did not converge in less than 150 seconds despite discounting for the expensive sorting operation in the truncation step. On the other hand, our algorithm finds the foreground in about 8 seconds.

**Parameters.** The algorithm has three main parameters: 1) threshold $\zeta$, 2) incoherence $\mu$ and 3) sampling probability $p$. We observed that keeping $\zeta \approx \mu \frac{\|M - S(t)\|_2}{\sqrt{n}}$ speeds up the recovery while for background extraction keeping $\zeta \approx \mu \frac{\|M - S(t)\|_2}{n}$ gives a better quality output. The value of $\mu$ for real world data sets was figured out using cross validation while for the synthetic data the same value was used as used in data generation. The sampling probability for the synthetic data could be kept as low as $(\theta =)2r \log^2(n)/n$ while for the real world data set we get good results for $p = 0.05$. We define effective sample size as the ratio between the sampling probability and $\theta$. Also, rather than splitting samples, we use the entire set of observed entries to perform our updates (see Algorithm 1).

**Synthetic data.** We generate $M = L^* + \tilde{S}^*$ of two sizes, where $L^* = UV^T \in \mathbb{R}^{2000 \times 2000}$ (and $\mathbb{R}^{5000 \times 5000}$) is a random rank-5 (and rank-10 respectively) matrix with incoherence $\approx 1$. $\tilde{S}^*$ is generated by considering a uniformly random subset of size $\frac{\|S^*\|_0}{m} \approx \frac{\sqrt{m} \|S^*\|_0}{n}$ from $[m] \times [n]$ where every entry is i.i.d. from the uniform distribution in $[\frac{1}{2\sqrt{mn}}, \frac{1}{2\sqrt{mn}}]$. This is the same setup as used in (Candes et al. 2011).

Figure 1 (a) plots recovery error ($\|L - L^*\|_F$) vs computational time for our PG-RMC method (with different sampling probabilities) as well as the St-NcRPCA algorithm. Note that even for very small values of sampling $p$, we can achieve the same recovery error as when using significantly small values. For example, our method with $p = 0.1$ achieve 0.01 error ($\|L - L^*\|_F$ in $\approx 2.5s$ while St-NcRPCA method requires $\approx 10s$ to achieve the same accuracy. Note that we do not compare against the convex relaxation based methods like IALM from (Candes et al. 2011), as (Netrapalli et al. 2014) shows that St-NcRPCA is significantly faster than IALM and several other convex relaxation solvers.

Figure 1 (b) plots time required to achieve different recovery errors ($\|L - L^*\|_F$) as the sampling probability $p$ increases. As expected, we observe a linear increase in the run-time with $p$. Interestingly, for very small values of $p$, we observe an increase in running time. In this regime, $\|P_\Omega(\tilde{M})\|_2$ becomes very large (as $p$ doesn’t satisfy the sampling requirements). Hence, the increase in the number of iterations ($T \approx \log \frac{\|P_\Omega(\tilde{M})\|_2}{p\varepsilon}$) dominates the decrease in per iteration time complexity.

Figure 1 (c), (d) plot computation time required by our method (PG-RMC), Algorithm 1 versus rank and incoherence, respectively. As expected, as these two problem parameters increase, our method requires more time. Note that our run-time dependence on rank seems to be linear, while our results require $O(r^3)$ time. This hints at the possibility of further improving the computational complexity analysis of our algorithm.

Figure 1 (e) plots the effective sample size against success probability. We see that the probability of recovering the underlying low rank matrix undergoes a rapid phase
Nearly Optimal Robust Matrix Completion

![Figure 1](image1.png)

**Figure 2:** PG-RMC on Shopping video. (a) a video frame (b) a video frame (c) an extracted background frame (d) an extracted background frame (e) time vs error for different sampling probabilities; PG-RMC takes 38.7s while St-NcRPAC takes 204.4s. PG-RMC on Restaurant video. (b) a video frame (d) an extracted background frame (f) time vs error for different sampling probabilities; PG-RMC takes 7.9s while St-NcPCA takes 27.8s

Figure 2 shows one frame each from two such videos (a shopping center video, a restaurant video). Figure 2(a), (d) show the extracted background from the two videos by using our method (PG-RMC, Algorithm 1) with probability of sampling \( p = 0.05 \). Figure 2(c), (f) compares objective function value for different \( p \) values. Clearly, PG-RMC can recover the true background with \( p \) as small as 0.05. We also observe an order of magnitude speedup \((\approx 5x)\) over St-NcRPCA (Netrapalli et al., 2014). We present results on the video Escalator in Appendix

**Conclusion.** In this work, we studied the Robust Matrix Completion problem. For this problem, we provide exact recovery of the low-rank matrix \( L^* \) using nearly optimal number of observations as well as nearly optimal fraction of corruptions in the observed entries. Our RMC result is based on a simple and efficient PGD algorithm that has nearly linear time complexity as well. Our result improves state-of-the-art sample and run-time complexities for the related Matrix Completion as well as Robust PCA problem. For Robust PCA, we provide first nearly linear time algorithm under standard assumptions.

Our sample complexity depends on \( \epsilon \), the desired accuracy in \( L^* \). Removing this factor will be an interesting future work. Moreover, improving dependence of sample complexity on \( r \) (from \( r^2 \) to \( r \)) also represents an important direction. Finally, similar to foreground background separation, we would like to explore more applications of RMC/RPCA.

**References**


Nearly Optimal Robust Matrix Completion


