2015 IEEE International Conference on Data Mining

Fast Low-Rank Matrix Learning with Nonconvex Regularization

Quanming Yao James T. Kwok Wenliang Zhong
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
$\{$*qyaoaa, jamesk, wzhong*$\}$*@cse.ust.hk*

***Abstract*—Low-rank modeling has a lot of important applications in machine learning, computer vision and social network analysis. While the matrix rank is often approximated by the convex nuclear norm, the use of nonconvex low-rank regularizers has demonstrated better recovery performance. However, the resultant optimization problem is much more challenging. A very recent state-of-the-art is based on the proximal gradient algorithm. However, it requires an expensive full SVD in each proximal step. In this paper, we show that for many commonly-used nonconvex low-rank regularizers, a cutoff can be derived to automatically threshold the singular values obtained from the proximal operator. This allows the use of power method to approximate the SVD efficiently. Besides, the proximal operator can be reduced to that of a much smaller matrix projected onto this leading subspace. Convergence, with a rate of $O(1/T)$ where T is the number of iterations, can be guaranteed. Extensive experiments are performed on matrix completion and robust principal component analysis. The proposed method achieves significant speedup over the state-of-the-art. Moreover, the matrix solution obtained is more accurate and has a lower rank than that of the traditional nuclear norm regularizer.**

***Keywords*-Low-rank matrix, Nonconvex optimization, Proximal gradient, Matrix completion, Robust PCA**

I. INTRODUCTION

The learning of low-rank matrices is a central issue in many machine learning problems. For example, matrix completion [1], which is one of the most successful approaches in collaborative filtering, assumes that the target ratings matrix is low-rank. Besides collaborative filtering, matrix completion has also been used on tasks such as sensor networks [2], social network analysis [3], and image processing [4, 5].

Another important use of low-rank matrix learning is robust principal component analysis (RPCA) [6], which assumes the target matrix is low-rank and also corrupted by sparse data noise. It is now popularly used in various computer vision applications, such as shadow removal of aligned faces and background modeling of surveillance videos [6, 7]. Besides, low-rank minimization has also been used in tasks such as multilabel learning [8] and multitask learning [9].

However, rank minimization is NP-hard. To alleviate this problem, a common approach is to use instead a convex surrogate such as the nuclear norm (which is the sum of singular values of the matrix). It is known that the nuclear norm is the tightest convex lower bound of the rank. Besides, there are theoretical guarantees that the incomplete matrix can be recovered with nuclear norm regularization [1, 6]. Moreover, though the nuclear norm is non-smooth, the resultant optimization problem can often be solved efficiently using modern tools such as accelerated proximal gradient descent [10], Soft-Impute [11], and active subspace selection methods [12].

Despite its success, recently there have been numerous attempts that use nonconvex surrogates to better approximate the rank function. The key idea is that the larger, and thus more informative, singular values should be less penalized. Example nonconvex low-rank regularizers include the capped-ℓ_1 penalty [13], log-sum penalty (LSP) [14], truncated nuclear norm (TNN) [15], smoothly clipped absolute deviation (SCAD) [16], and minimax concave penalty (MCP) [17]. Empirically, these nonconvex regularizers achieve better recovery performance than the convex nuclear norm regularizer.

However, the resultant nonconvex optimization problem is much more challenging. One approach is to use the concave-convex procedure [18], which decomposes the nonconvex regularizer into a difference of convex functions [13, 15]. However, a sequence of relaxed problems have to be solved, and can be computationally expensive [19]. A more efficient method is the recently proposed iteratively reweighted nuclear norm (IRNN) algorithm [20]. It is based on the observation that existing nonconvex regularizers are all concave and their super-gradients are non-increasing. Though IRNN still has to iterate, each of its iterations only involves computing the super-gradient of the regularizer and a singular value decomposition (SVD). However, performing SVD on a $m \times n$ matrix (where $m \geq n$) still takes $O(mn^2)$ time, and can be expensive when the matrix is large.

Recently, the proximal gradient algorithm has also been used on this nonconvex low-rank minimization problem [7, 15, 20, 21]. However, it requires performing the full SVD in each proximal step, which is expensive for large-scale applications. To alleviate this problem, we first observe that for the commonly-used nonconvex low-rank regularizers, the singular values obtained from the corresponding proximal operator can be automatically thresholded. One then only

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.9

needs to find the leading singular values/vectors in order to generate the next iterate. By using the power method [22], a fast and accurate approximation of such a subspace can be obtained. Moreover, instead of computing the proximal operator on a large matrix, one only needs to compute that on its projection onto this leading subspace. The size of the matrix is significantly reduced and the proximal operator can be made much more efficient. In the context of matrix completion problems, further speedup is possible by exploiting a special "sparse plus low-rank" structure of the matrix iterate.

The rest of the paper is organized as follows. Section II reviews the related work. The proposed algorithm is presented in Section III; Experimental results on matrix completion and RPCA are shown in Section IV, and the last section gives some concluding remarks.

In the sequel, the transpose of vector/matrix is denoted by the superscript $(\cdot)^\top$. For a $m\times n$ matrix X, $\text{tr}(X)$ is its trace, $\|X\|_F = \text{tr}(X^\top X)$ is the Frobenius norm, and $\|X\|_* = \sum_{i=1}^m \sigma_i$ is the nuclear norm. Given $x = [x_i] \in \mathbb{R}^m$, $\text{Diag}(x)$ constructs a $m \times m$ diagonal matrix whose ith diagonal element is x_i. Moreover, I denotes the identity matrix. For a differentiable function f, we use ∇f for its gradient. For a nonsmooth function, we use ∂f for its subdifferential.

II. Background

A. Proximal Gradient Algorithms

In this paper, we consider composite optimization problems of the form

$$\min_x F(x) \equiv f(x) + \lambda r(x), \tag{1}$$

where f is smooth and r is nonsmooth. In many machine learning problems, f is the loss and r a low-rank regularizer. In particular, we make the following assumptions on f.

A1. f, not necessarily convex, is differentiable with ρ-Lipschitz continuous gradient, i.e., $\|\nabla f(X_1) - \nabla f(X_2)\|_F \le \rho\|X_1 - X_2\|_F$. Without loss of generality, we assume that $\rho \le 1$.

A2. f is bounded below, i.e., $\inf f(X) > -\infty$.

In recent years, proximal gradient algorithms [23] have been widely used for solving (1). At each iteration t, a quadratic function is used to upper-bound the smooth f at the current iterate x^t, while leaving the nonsmooth r intact. For a given stepsize τ, the next iterate x^{t+1} is obtained as

$$\begin{aligned}
&\arg\min_x \nabla f(x^t)^\top (x - x^t) + \frac{\tau}{2}\|x - x^t\|^2 + \lambda r(x) \\
&= \quad \arg\min_x \frac{1}{2}\|x - z^t\|^2 + \frac{\lambda}{\tau} r(x) \equiv \text{prox}_{\frac{\lambda}{\tau}r}(z^t),
\end{aligned}$$

where $z^t = x^t - \frac{1}{\tau}\nabla f(x^t)$, and $\text{prox}_{\frac{\lambda}{\tau}r}(\cdot)$ is the proximal operator [23]. Proximal gradient algorithms can be further accelerated, by replacing z^t with a proper linear combination of x^t and x^{t-1}. In the sequel, as our focus is on learning low-rank matrices, x in (1) becomes a $m \times n$ matrix X.[1]

B. Convex and Nonconvex Low-Rank Regularizers

An important factor for the success of proximal gradient algorithms is that its proximal operator $\text{prox}_{\mu r}(\cdot)$ can be efficiently computed. For example, for the nuclear norm $\|X\|_*$, the following Proposition shows that its proximal operator has a closed-form solution.

Proposition II.1. *[24] $\text{prox}_{\mu\|\cdot\|_*}(X) = U(\Sigma - \mu I)_+ V^\top$, where $U\Sigma V^\top$ is the SVD of X, and $(Z)_+ = [\max(Z_{ij}, 0)]$.*

While the convex nuclear norm makes the low-rank optimization problem easier, it may not be a good approximation of the matrix rank [7, 15, 20, 21]. As mentioned in Section I, a number of nonconvex surrogates for the rank have been recently proposed. In this paper, we make the following assumption on the low-rank regularizer r in (1).

A3. r is possibly non-smooth and nonconvex, and of the form $r(X) = \sum_{i=1}^m \hat{r}(\sigma_i)$, where $\sigma_1 \ge \cdots \ge \sigma_m \ge 0$ are singular values of X, and $\hat{r}(\sigma)$ is a concave and non-decreasing function of $\sigma \ge 0$ with $\hat{r}(0) = 0$.

All nonconvex low-rank regularizers introduced in Section I satisfy this assumption. Their corresponding $\hat{r}$'s are shown in Table I.

Table I
$\hat{r}$'S FOR SOME POPULAR NONCONVEX LOW-RANK REGULARIZERS. FOR THE TNN REGULARIZER, $\theta \in \{1, \dots, n\}$ IS THE NUMBER OF LEADING SINGULAR VALUES THAT ARE NOT PENALIZED.

	$\mu\hat{r}(\sigma_i)$
capped-ℓ_1	$\mu\min(\sigma_i, \theta),\ \theta > 0$
LSP	$\mu\log\left(\frac{\sigma_i}{\theta} + 1\right),\ \theta > 0$
TNN	$\begin{cases} \mu\sigma_i & i > \theta \\ 0 & i \le \theta \end{cases}$
SCAD	$\begin{cases} \mu\sigma_i & \sigma_i \le \mu \\ \frac{-\sigma_i^2 + 2\theta\mu\sigma_i - \mu^2}{2(\theta-1)} & \mu < \sigma_i \le \theta\mu,\ \theta > 2 \\ \frac{(\theta+1)\mu^2}{2} & \sigma_i > \theta\mu \end{cases}$
MCP	$\begin{cases} \mu\sigma_i - \frac{\sigma_i^2}{2\theta} & \sigma_i \le \theta\mu \\ \frac{\theta\mu^2}{2} & \sigma_i > \theta\mu \end{cases},\theta > 0$

The Iteratively Reweighted Nuclear Norm (IRNN) algorithm [20] is a state-of-the-art solver for nonconvex low-rank minimization. It is based on upper-bounding the nonconvex r, and approximates the matrix rank by a weighted version of the nuclear norm $\|X\|_w = \sum_{i=1}^m w_i\sigma_i$, where $0 \le w_1 \le \cdots \le w_m$, Intuitively, $\|X\|_w$ imposes a smaller penalty on the larger (and more informative) singular values. Other solvers that are designed only for specific nonconvex low-rank regularizers include [7] (for the capped-ℓ_1), [15] (for the TNN), and [25] (for the MCP). All these (including IRNN) need SVD in each iteration. It takes $O(m^2 n)$ time, and thus can be slow.

[1]In the following, we assume $m \le n$.

While proximal gradient algorithms have mostly been used on convex problems, recently they are also applied to nonconvex ones [7, 15, 20, 21]. In particular, in the very recent generalized proximal gradient (GPG) algorithm [21], it is shown that for any nonconvex r satisfying assumption A3, its proximal operator can be computed by the following generalized singular value thresholding (GSVT) operator.

Proposition II.2. *[21] $\text{prox}_{\mu r}(X) = U\text{Diag}(y^*)V^\top$, where $U\Sigma V^\top$ is the SVD of X, and $y^* = [y_i^*]$ with*

$$y_i^* \in \arg\min_{y_i \geq 0} \frac{1}{2}\left(y_i - \sigma_i\right)^2 + \mu \hat{r}(y_i). \tag{2}$$

In GPG, problem (2) is solved by a fixed-point iteration algorithm. Indeed, closed-form solutions exist for the regularizers in Table I [19]. While the obtained proximal operator can then be immediately plugged into a proximal gradient algorithm, Proposition II.2 still involves SVD.

III. Proposed Algorithm

In this section, we show that the GSVT operator $\text{prox}_{\mu r}(\cdot)$ can be computed more efficiently. It is based on two ideas. First, the singular values in $\text{prox}_{\mu r}(\cdot)$ are automatically thresholded. Second, $\text{prox}_{\mu r}(\cdot)$ can be obtained from the proximal operator on a smaller matrix.

A. Automatic Thresholding of Singular Values

The following Proposition shows that y_i^* in (2) becomes zero when σ_i is smaller than a regularizer-specific threshold. Because of the lack of space, proofs will be reported in a longer version of this paper.

Proposition III.1. *For any $\hat{r}$ satisfying Assumption A3, there exists a threshold $\gamma > 0$ such that $y_i^* = 0$ when $\sigma_i \leq \gamma$.*

By examining the optimality conditions of (2), simple closed-form solutions can be obtained for the nonconvex regularizers in Table I.

Corollary III.2. *For the nonconvex regularizers in Table I, their γ values are equal to*

- *capped-ℓ_1: $\gamma = \min\left(\mu, \theta + \frac{\mu}{2}\right)$;*
- *LSP: $\gamma = \min\left(\frac{\mu}{\theta}, \theta\right)$;*
- *TNN: $\gamma = \max\left(\mu, \sigma_{\theta+1}\right)$;*
- *SCAD: $\gamma = \mu$;*
- *MCP: $\gamma = \sqrt{\theta}\mu$ if $0 < \theta < 1$, and μ otherwise.*

Proposition III.1 suggests that in each proximal iteration t, we only need to compute the leading singular values/vectors of the matrix iterate Z^t. The power method (Algorithm 1) [22] is a fast and accurate algorithm for obtaining an approximation of such a subspace. Besides the power method, algorithms such as PROPACK [26] have also been used [27]. However, the power method is more efficient than PROPACK [22]. It also allows warm-start, which is particularly useful because of the iterative nature of the proximal gradient algorithm.

Algorithm 1 Power method to obtain an approximate left subspace of Z.

```
Require: matrix $Z \in \mathbb{R}^{m \times n}$, $R \in \mathbb{R}^{n \times k}$.
1: $Y^1 \leftarrow ZR$;
2: for $t = 1, 2, \dots, T_{pm}$ do
3:     $Q^{t+1} = \text{QR}(Y^t)$;    // QR decomposition
4:     $Y^{t+1} = Z(Z^\top Q^{t+1})$;
5: end for
6: return $Q^{T_{pm}+1}$.
```

B. Proximal Operator on a Smaller Matrix

Assume that Z^t has $\hat{k} \leq n$ singular values larger than γ, and its rank-$\hat{k}$ SVD is $U_{\hat{k}}\Sigma_{\hat{k}}V_{\hat{k}}^\top$. The following Proposition shows that $\text{prox}_{\mu r}(Z^t)$ can be obtained from the proximal operator on a smaller matrix.

Proposition III.3. *Assume that $Q \in \mathbb{R}^{m \times k}$, where $k \geq \hat{k}$, is orthogonal and span$(U_{\hat{k}}) \subseteq$ span(Q). Then, $\text{prox}_{\mu r}(Z^t) = Q \cdot \text{prox}_{\mu r}(Q^\top Z^t)$.*

Though SVD is still needed to obtain $\text{prox}_{\mu r}(Q^\top Z^t)$, $Q^\top Z^t$ is much smaller than Z^t ($k \times n$ vs $m \times n$). This smaller SVD takes $O(nk^2)$ time, and the other matrix multiplication steps take $O(mnk)$ time. Thus, the time complexity for this SVD step is reduced from $O(m^2 n)$ to $O((m+k)nk)$.

C. Complete Procedure

The complete procedure (Algorithm 2) will be called FaNCL (Fast NonConvex Lowrank). The core steps are 9–16. We first use the power method to efficiently obtain an approximate Q, whose singular values are then thresholded according to Corollary III.2. With $k \geq \hat{k}$, the rank of $\tilde{X}^p$ will be equal to that of $\text{prox}_{\mu r}(Z^t)$. In each iteration, we ensure a sufficient decrease of the objective:

$$F(X^{t+1}) \leq F(X^t) - c_1 \|X^{t+1} - X^t\|_F^2, \tag{3}$$

where $c_1 = \frac{\tau - \rho}{4}$; otherwise, the power method is restarted. Moreover, similar to [12, 27], steps 6-7 use the column spaces of the previous iterates (V^t and V^{t-1}) to warm-start the power method. For further speedup, we employ a continuation strategy as in [11, 20, 27]. Specifically, λ^t is initialized to a large value and then decreases gradually.

Algorithm 2 can also be used with the nuclear norm. It can be shown that the threshold γ is equal to λ/τ, and y_i^* in step 15 has the closed-form solution $\max(\sigma_i - \lambda^t/\tau, 0)$. However, since our focus is on nonconvex regularizers, using Algorithm 2 for nuclear norm minimization will not be further pursued in the sequel.

The power method has also been recently used to approximate the SVT in nuclear norm minimization [12]. However, [12] is based on active subspace selection (which uses SVT to update the active row and column subspaces of the current solution), and is thus very different from the proposed

Algorithm 2 FaNCL (Fast NonConvex Low-rank).

```
1: choose $\tau > \rho$, $c_1 = \frac{\tau-\rho}{4}$, $\lambda^0 > \lambda$ and $\nu \in (0,1)$;
2: initialize $V_0, V_1 \in \mathbb{R}^{n\times k}$ as random Gaussian matrices,
   and $X^1 = 0$;
3: for $t = 1, 2, \dots T$ do
4:   $\lambda^t \leftarrow (\lambda^{t-1} - \lambda)\nu + \lambda$;
5:   $Z^t \leftarrow X^t - \frac{1}{\tau}\nabla f(X^t)$;
6:   $V^{t-1} \leftarrow V^{t-1} - V^t({V^t}^\top V^{t-1})$, and
     remove any zero columns;
7:   $R^1 \leftarrow$ QR($[V^t, V^{t-1}]$);
8:   for $p = 1, 2, \dots$ do
9:     $Q \leftarrow$ PowerMethod($Z^t, R^p$);
10:    $[U_A^p, \Sigma_A^p, V_A^p] \leftarrow$ SVD($Q^\top Z^t$);
11:    $\hat{k} \leftarrow$ number of $\sigma_A$'s are $> \gamma$ in Corollary III.2;
12:    $\tilde{U}^p \leftarrow \hat{k}$ leading columns of $U_A^p$;
13:    $\tilde{V}^p \leftarrow \hat{k}$ leading columns of $V_A^p$;
14:    for $i = 1, 2, \dots, \hat{k}$ do
15:      obtain $y_i^*$ from (2) with $\mu = 1/\tau$ and $\lambda^t$;
16:    end for
17:    $\tilde{X}^p \leftarrow (Q\tilde{U}^p)\text{Diag}(y_1^*, \dots, y_{\hat{k}}^*)(\tilde{V}^p)^\top$;
18:    if $F(\tilde{X}^p) \le F(X^t) - c_1\|\tilde{X}^p - X^t\|_F^2$ then
19:      $X^{t+1} \leftarrow \tilde{X}^p$,   $V^{t+1} \leftarrow \tilde{V}^p$;
20:      break;
21:    else
22:      $R^{p+1} = V_A^p$;
23:    end if
24:  end for
25: end for
26: return $X^{T+1}$.
```

algorithm (which is a proximal gradient algorithm). In Section IV, it will be shown that the proposed method has better empirical performance. Moreover, [12] is only designed for nuclear norm minimization, and cannot be extended for the nonconvex regularizers considered here.

A breakdown of the time complexity of Algorithm 2 is as follows. For simplicity, assume that X^t's always have rank k. Step 5 takes $O(mn)$ time; step 6 and 7 take $O(nk^2)$ time; step 9 and 10 take $O(mnkT_{pm})$ time; step 17 takes $O(mnk)$ time; and step 18 takes $O(mn)$ time. Thus, the per-iteration time complexity is $O(mnkpT_{pm})$. In the experiment, we set $T_{pm} = 3$ and $p = 1$. Empirically, this setting is enough to guarantee (3). In contrast, SVDs in GPG and IRNN take $O(m^2n)$ time, and are thus much slower as $k \ll m$.

D. Convergence Analysis

The following Proposition shows that $\{X^t\}$ from Algorithm 2 converges to a limit point $X^\infty = \lim_{t\to\infty} X^t$.

Proposition III.4. $\sum_{t=1}^{\infty} \|X^{t+1} - X^t\|_F^2 < \infty$.

The following shows that it is also a critical point.[2]

Theorem III.5. *$\{X^t\}$ converges to a critical point X^* of problem (1) in a finite number of iterations.*

By combining with Proposition III.4, the following shows that $\|X^{t+1} - X^t\|_F^2$ converges to zero at a rate of $O(1/T)$.

Corollary III.6. $\min_{t=1,\dots,T} \|X^{t+1} - X^t\|_F^2 \le \frac{1}{c_1 T}\left[F(X^1) - F(X^*)\right]$.

E. Further Speedup for Matrix Completion

In matrix completion, one attempts to recover a low-rank matrix $O \in \mathbb{R}^{m\times n}$ by observing only some of its elements. Let the observed positions be indicated by $\Omega \in \{0,1\}^{m\times n}$, such that $\Omega_{ij} = 1$ if O_{ij} is observed, and 0 otherwise. It can be formulated as an optimization problem in (1), with $f(X) = \frac{1}{2}\|\mathcal{P}_\Omega(X - O)\|_F^2$, where $[\mathcal{P}_\Omega(A)]_{ij} = A_{ij}$ if $\Omega_{ij} = 1$ and 0 otherwise, and r is a low-rank regularizer.

It can be easily seen that step 5 in Algorithm 2 becomes $Z^t = X^t - \frac{1}{\tau}\mathcal{P}_\Omega(X^t - O)$. By observing that X^t is low-rank and $\frac{1}{\tau}\mathcal{P}_\Omega(X^t - O)$ is sparse, Mazumder et al. [11] showed that this "sparse plus low-rank" structure allows matrix multiplications of the form ZA and $Z^\top B$ to be efficiently computed. Here, this trick can also be directly used to speed up the computation of $Z(Z^\top Q^{t+1})$ in Algorithm 1. Since $\|\Omega\|_1$ is very sparse, this step takes $O(kT_{pm}\|\Omega\|_1)$ time instead of $O(mnkT_{pm})$, thus is much faster.

The following Proposition shows that $\|\tilde{X}^p - X^t\|_F^2$ in step 18 of Algorithm 2 can also be easily computed.

Proposition III.7. *Let the reduced SVD of X be $U\Sigma V^\top$, and P, Q be orthogonal matrices such that span(U) $\subseteq$ span(P) and span(V) $\subseteq$ span(Q). Then $\|X\|_F = \|P^\top XQ\|_F$.*

Let the reduced SVDs of $\tilde{X}^p$ and X^t be $\tilde{U}\tilde{\Sigma}\tilde{V}^\top$ and $U^t\Sigma^t{V^t}^\top$, respectively. Let $P =$ QR($[\tilde{U}, U^t]$) and $Q =$ QR($[\tilde{V}, V^t]$). Using Proposition III.7, $\|\tilde{X}^p - X^t\|_F = \|P^\top(\tilde{X}^p - X^t)Q\|_F = \|(P^\top\tilde{U})\tilde{\Sigma}(\tilde{V}^\top Q) - (P^\top U^t)\Sigma^t({V^t}^\top Q)\|_F$. This takes $O(nk^2)$ instead of $O(mn)$ time. The per-iteration time complexity is reduced from $O(mnkT_{pm})$ to $O((nk + T_{pm}|\Omega|_1)k)$ and is much faster. Table II compares the per-iteration time complexities and convergence rates for the various low-rank matrix completion solvers used in the experiments (Section IV-A).

F. Handling Multiple Matrix Variables

The proposed algorithm can be extended for optimization problems involving N matrices $X_1, \dots, X_N$:

$$\min F(X_1, \dots, X_N) \equiv f(X_1, \dots, X_N) + \sum_{i=1}^{N} \lambda_i r_i(X_i). \quad (4)$$

[2]Since r is nonconvex and its subdifferential for points in its domain may be empty, we define X^* as a critical point by extending the definition in [19], namely that $0 \in \nabla f(X^*) + \lambda\partial r_1(X^*) - \lambda\partial r_2(X^*)$, where $r(X) = r_1(X) - r_2(X)$, and r_1 and r_2 are convex.

Table II
COMPARISON OF THE PER-ITERATION TIME COMPLEXITIES AND CONVERGENCE RATES OF VARIOUS MATRIX COMPLETION SOLVERS. HERE, $\nu \in (0,1)$ IS A CONSTANT.

regularizer	method	complexity	rate
(convex) nuclear norm	APG [10, 27]	$O(mnk)$	$O(1/T^2)$
	Soft-Impute [11]	$O(k\|\Omega\|_1)$	$O(1/T)$
	active ALT [12]	$O(kT_{in}\|\Omega\|_1)$	$O(\nu^T)$
fixed-rank factorization	LMaFit [28]	$O(k\|\Omega\|_1)$	—
	R1MP [29]	$O(\|\Omega\|_1)$	$O(\nu^T)$
nonconvex	IRNN [20]	$O(m^2 n)$	—
	GPG [21]	$O(m^2 n)$	—
	FaNCL	$O(k\|\Omega\|_1)$	$O(1/T)$

Assumptions A1-A3 are analogously extended. In particular, A1 now assumes that $\|\nabla f_i(X) - \nabla f_i(Y)\|_F \leq \rho_i \|X - Y\|_F$ for some ρ_i, where $f_i(X)$ is the function obtained by keeping all the X_j's (where $i \neq j$) in f fixed.

Many machine learning problems can be cast into this form. One example that will be considered in Section IV is robust principal component analysis (RPCA) [6]. Given a noisy data matrix O, RPCA assumes that O can be approximated by the sum of a low-rank matrix X plus sparse data noise Y. Mathematically, we have

$$\min_{X,Y} F(X,Y) \equiv f(X,Y) + \lambda r(X) + \beta\|Y\|_1, \quad (5)$$

where $f(X,Y) = \frac{1}{2}\|X + Y - O\|_F^2$, r is a low-rank regularizer on X, and $\|Y\|_1$ encourages Y to be sparse. Since both r and the ℓ_1 regularizer $\|\cdot\|_1$ are nonsmooth, (5) does not fit into formulation (1). Besides RPCA, problems such as subspace clustering [30], multilabel learning [8] and multitask learning [9] can also be cast as (4).

For simplicity, we focus on the case with two parameter blocks. Extension to multiple blocks is straightforward. To solve the two-block problem in (5), we perform alternating proximal steps on X and Y at each iteration t:

$$X^{t+1} = \arg\min_X \tfrac{1}{2}\|X - Z_X^t\|_F^2 + \tfrac{\lambda}{\tau} r(X) = \text{prox}_{\frac{\lambda}{\tau}r}(Z_X^t),$$
$$Y^{t+1} = \arg\min_Y \tfrac{1}{2}\|Y - Z_Y^t\|_F^2 + \tfrac{\beta}{\tau}\|Y\|_1 = \text{prox}_{\frac{\beta}{\tau}\|\cdot\|_1}(Z_Y^t),$$

where $Z_X^t = X^t - \frac{1}{\tau}\nabla f(X^t, Y^t)$, and $Z_Y^t = Y^t - \frac{1}{\tau}\nabla f(X^{t+1}, Y^t)$. Y^{t+1} can be easily obtained as $Y_{ij}^{t+1} = \text{sign}\left([Z_Y^t]_{ij}\right)\left(|[Z_Y^t]_{ij}| - \frac{\beta}{\tau}\right)_+$, where $\text{sign}(x)$ denotes the sign of x. Similar to (3), we ensure a sufficient decrease of the objective in each iteration:

$$\begin{aligned} F_{Y^t}(X^{t+1}) &\leq F_{Y^t}(X^t) - c_1\|X^{t+1} - X^t\|_F^2, \\ F_{X^{t+1}}(Y^{t+1}) &\leq F_{X^{t+1}}(Y^t) - c_1\|Y^{t+1} - Y^t\|_F^2, \end{aligned}$$

where $F_Y(X) = f(X,Y) + \lambda r(X)$, and $F_X(Y) = f(X,Y) + \beta\|Y\|_1$. The resultant algorithm is similar to Algorithm 2.

When F is convex, convergence of this alternating minimization scheme has been well studied [31]. However, here F is nonconvex. We extend the convergence results in Section III-D to the following.

Theorem III.8. *With N parameter blocks and $\{(X_1^t, \dots, X_N^t)\}$ generated by the algorithm, we have*

1) *$\sum_{t=1}^{\infty}\sum_{i=1}^{N}\|X_i^{t+1} - X_i^t\|_F^2 < \infty$;*
2) *$\{(X_1^t, \dots, X_N^t)\}$ converges to a critical point $(X_1^*, \dots, X_N^*)$ of (4) in a finite number of iterations;*
3) *$\min_{t=1,\dots,T}\sum_{i=1}^{N}\|X_i^{t+1} - X_i^t\|_F^2 \leq \frac{1}{c_1 T}[F(X_1^1, \dots, X_N^1) - F(X_1^*, \dots, X_N^*)]$.*

IV. EXPERIMENTS

A. Matrix Completion

We compare a number of low-rank matrix completion solvers, including models based on (i) the commonly used (convex) nuclear norm regularizer; (ii) fixed-rank factorization models [28, 29], which decompose the observed matrix O into a product of rank-k matrices U and V. Its optimization problem can be written as: $\min_{U,V} \frac{1}{2}\|\mathcal{P}_\Omega(UV - O)\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2)$; and (iii) nonconvex regularizers, including the capped-ℓ_1 (with θ in Table I set to 2λ), LSP (with $\theta = \sqrt{\lambda}$), and TNN (with $\theta = 3$).

The nuclear norm minimization algorithms to be compared include:

1) Accelerated proximal gradient (APG)[3] algorithm [10, 27], with the partial SVD by PROPACK [26];
2) Soft-Impute[4] [11], which iteratively replaces the missing elements with those obtained from SVT. The "sparse plus low-rank" structure of the matrix iterate is utilized to speed up computation (Section III-E);
3) Active alternating minimization[5] (denoted "active ALT") [12], which adds/removes rank-one subspaces from the active set in each iteration. The nuclear norm optimization problem is then reduced to a smaller problem defined only on this active set.

We do not compare with the Frank-Wolfe algorithm [32] and stochastic gradient descent [33], as they have been shown to be less efficient [12]. For the fixed-rank factorization models (where the rank is tuned by the validation set), we compare with the two state-of-the-art algorithms:

1) Low-rank matrix fitting (LMaFit) algorithm[6] [28]; and
2) Rank-one matrix pursuit (R1MP) [29], which pursues a rank-one basis in each iteration.

We do not compare with the concave-convex procedure [13, 15], since it has been shown to be inferior to IRNN [19].

For models with nonconvex low-rank regularizers, we compare the following solvers:

1) Iterative reweighted nuclear norm (IRNN)[7] [20];

[3]http://perception.csl.illinois.edu/matrix-rank/Files/apg_partial.zip
[4]http://cran.r-project.org/web/packages/softImpute/index.html
[5]http://www.cs.utexas.edu/~cjhsieh/nuclear_active_1.1.zip
[6]http://www.caam.rice.edu/~optimization/L1/LMaFit/download.html
[7]https://sites.google.com/site/canyilu/file/2014-CVPR-IRNN.zip?attredirects=0&d=1

Table III
Matrix completion performance on the synthetic data. Here, NMSE is scaled by $\times 10^{-2}$, and CPU time is in seconds.

		$m = 500$			$m = 1000$			$m = 1500$			$m = 2000$		
		(observed: 12.43%)			(observed: 6.91%)			(observed: 4.88%)			(observed: 3.80%)		
		NMSE	rank	time	NMSE	rank	time	NMSE	rank	time	NMSE	rank	time
nuclear norm	APG	3.95 ± 0.16	49	4.8	3.90 ± 0.05	59	59.5	3.74 ± 0.02	71	469.3	3.69 ± 0.04	85	1383.3
	Soft-Impute	3.95 ± 0.16	49	64.9	3.90 ± 0.05	59	176.0	3.74 ± 0.02	71	464.4	3.69 ± 0.04	85	1090.2
	active ALT	3.95 ± 0.16	49	17.1	3.90 ± 0.05	59	81.9	3.74 ± 0.02	71	343.8	3.69 ± 0.04	85	860.1
fixed rank	LMaFit	2.63 ± 0.10	5	0.6	2.85 ± 0.10	5	1.7	2.54 ± 0.09	5	4.5	2.40 ± 0.09	5	7.1
	R1MP	22.72 ± 0.63	39	0.3	20.89 ± 0.66	54	0.8	20.04 ± 0.66	62	1.4	19.53 ± 0.61	63	3.4
capped ℓ_1	IRNN	$\mathbf{1.98} \pm 0.07$	5	8.5	$\mathbf{1.89} \pm 0.04$	5	75.5	$\mathbf{1.81} \pm 0.02$	5	510.8	$\mathbf{1.80} \pm 0.02$	5	1112.3
	GPG	$\mathbf{1.98} \pm 0.07$	5	8.5	$\mathbf{1.89} \pm 0.04$	5	72.4	$\mathbf{1.81} \pm 0.02$	5	497.0	$\mathbf{1.80} \pm 0.02$	5	1105.8
	FaNCL	$\mathbf{1.98} \pm 0.07$	5	**0.3**	$\mathbf{1.89} \pm 0.04$	5	**0.9**	$\mathbf{1.81} \pm 0.02$	5	**2.6**	$\mathbf{1.80} \pm 0.02$	5	**4.1**
LSP	IRNN	$\mathbf{1.98} \pm 0.07$	5	21.8	$\mathbf{1.89} \pm 0.04$	5	223.9	$\mathbf{1.81} \pm 0.02$	5	720.9	$\mathbf{1.80} \pm 0.02$	5	2635.0
	GPG	$\mathbf{1.98} \pm 0.07$	5	21.2	$\mathbf{1.89} \pm 0.04$	5	235.3	$\mathbf{1.81} \pm 0.02$	5	687.4	$\mathbf{1.80} \pm 0.02$	5	2612.0
	FaNCL	$\mathbf{1.98} \pm 0.07$	5	**0.5**	$\mathbf{1.89} \pm 0.04$	5	**2.2**	$\mathbf{1.81} \pm 0.02$	5	**3.3**	$\mathbf{1.80} \pm 0.02$	5	**7.6**
TNN	IRNN	$\mathbf{1.98} \pm 0.07$	5	8.5	$\mathbf{1.89} \pm 0.04$	5	72.6	$\mathbf{1.81} \pm 0.02$	5	650.7	$\mathbf{1.80} \pm 0.02$	5	1104.1
	GPG	$\mathbf{1.98} \pm 0.07$	5	8.3	$\mathbf{1.89} \pm 0.04$	5	71.7	$\mathbf{1.81} \pm 0.02$	5	655.3	$\mathbf{1.80} \pm 0.02$	5	1098.2
	FaNCL	$\mathbf{1.98} \pm 0.07$	5	**0.3**	$\mathbf{1.89} \pm 0.04$	5	**0.8**	$\mathbf{1.81} \pm 0.02$	5	**2.7**	$\mathbf{1.80} \pm 0.02$	5	**4.2**

2) Generalized proximal gradient (GPG) algorithm [21], with the underlying problem (2) solved more efficiently using the closed-form solutions in [19];
3) The proposed FaNCL algorithm ($T_{pm} = 3$, $p = 1$).

All algorithms are implemented in Matlab. The same stopping criterion is used, namely that the algorithm stops when the difference in objective values between consecutive iterations is smaller than a given threshold. Experiments are run on a PC with i7 4GHz CPU and 24GB memory.

1) Synthetic Data: The observed $m \times m$ matrix is generated as $O = UV + G$, where the elements of $U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{k \times m}$ (with $k = 5$) are sampled i.i.d. from the normal distribution $\mathcal{N}(0, 1)$, and elements of G sampled from $\mathcal{N}(0, 0.1)$. A total of $\|\Omega\|_1 = 2mk \log(m)$ random elements in O are observed. Half of them are used for training, and the rest as validation set for parameter tuning. Testing is performed on the non-observed (missing) elements.

For performance evaluation, we use (i) the normalized mean squared error NMSE $= \sqrt{\sum_{(i,j) \notin \Omega} (X_{ij} - [UV]_{ij})^2} / \sqrt{\sum_{(i,j) \notin \Omega} [UV]_{ij}^2}$, where X is the recovered matrix; (ii) rank of X; and (iii) training CPU time. We vary m in the range $\{500, 1000, 1500, 2000\}$. Each experiment is repeated five times.

Results are shown in Table III. As can be seen, the nonconvex regularizers (capped-ℓ_1, LSP and TNN) lead to much lower NMSE's than the convex nuclear norm regularizer and fixed-rank factorization. Moreover, as is also observed in [33], the nuclear norm needs to use a much higher rank than the nonconvex ones. In terms of speed, FaNCL is the fastest among the nonconvex low-rank solvers. Figure 1 shows its speedup over GPG (which in turn is faster than IRNN). As can be seen, the larger the matrix, the higher is the speedup.

Recall that the efficiency of the proposed algorithm comes from (i) automatic singular value thresholding; (ii) computing the proximal operator on a smaller matrix; and (iii) exploiting the "sparse plus low-rank" structure in matrix completion. Their individual contributions are examined in Table IV. The baseline is GPG, which uses none of these; while the proposed FaNCL uses all. As all the variants produce the same solution, the obtained NMSE and rank values are not shown. As can be seen, tricks (i), (ii) and (iii) lead to average speedups of about 6, 4, and 3, respectively; and are particularly useful on the large data sets.

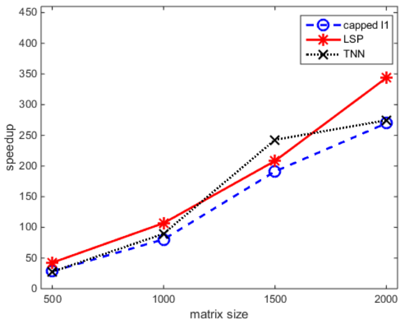


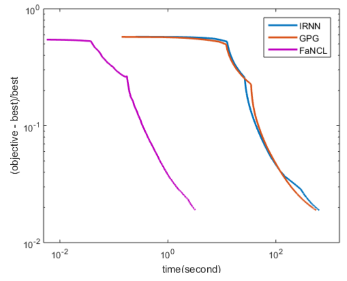
Figure 1. Speedup of FaNCL over GPG at different matrix sizes.

Table IV
Effects of the three tricks on CPU time (in seconds) using the synthetic data. (i) automatic singular value thresholding; (ii) computing the proximal operator on a smaller matrix; and (iii) "sparse plus low-rank" structure.

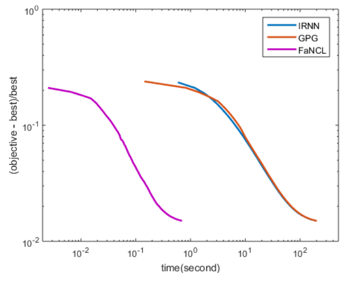
	solver	500	1000	1500	2000
capped ℓ_1	baseline (GPG)	8.5	72.4	497.0	1105.8
	✓ i	5.4	37.6	114.8	203.7
	✓ i, ii	0.6	3.2	11.4	25.6
	✓ i, ii, iii (FaNCL)	**0.3**	**0.9**	**2.6**	**6.8**
LSP	baseline (GPG)	21.2	235.3	687.4	2612.0
	✓ i	4.9	44.0	70.0	154.9
	✓ i, ii	1.0	9.7	14.8	31.1
	✓ i, ii, iii (FaNCL)	**0.5**	**2.2**	**3.3**	**8.2**
TNN	baseline (GPG)	8.3	71.7	655.3	1098.2
	✓ i	5.4	32.5	122.3	194.1
	✓ i, ii	0.6	2.8	10.3	15.8
	✓ i, ii, iii (FaNCL)	**0.3**	**0.8**	**2.7**	**3.3**

2) MovieLens: Experiment is performed on the popular MovieLens[8] data set (Table V), which contain ratings of different users on movies. We follow the setup in [29],

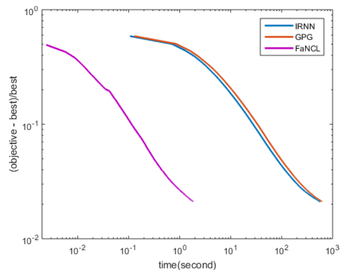
[8]http://grouplens.org/datasets/movielens/



(a) capped-ℓ_1.



(b) LSP.



(c) TNN.

Figure 2. Objective value vs CPU time for the various nonconvex low-rank regularizers on the MovieLens-100K data set.

and use 50% of the observed ratings for training, 25% for validation and the rest for testing. For performance evaluation, we use the root mean squared error on the test set Ω: RMSE $= \sqrt{\|\mathcal{P}_\Omega(X-O)\|_F^2/\|\Omega\|_1}$, where X is the recovered matrix. The experiment is repeated five times.

Table V
RECOMMENDATION DATA SETS USED IN THE EXPERIMENTS.

		#users	#movies	#ratings
MovieLens	100K	943	1,682	100,000
	1M	6,040	3,449	999,714
	10M	69,878	10,677	10,000,054
netflix		480,189	17,770	100,480,507
yahoo		249,012	296,111	62,551,438

Results are shown in Table VI. Again, nonconvex regularizers lead to the lowest RMSE's. Moreover, FaNCL is also the fastest among the nonconvex low-rank solvers, even faster than the state-of-the-art. In particular, it is the only solver (among those compared) that can be run on the MovieLens-10M data. Table VII examines the usefulness of the three tricks. The behavior is similar to that as observed in Table IV. Figures 2 and 3 compare the objective and RMSE vs CPU time for the various methods on the MovieLens-100K data set. As can be seen, FaNCL decreases the objective and RMSE much faster than the others.

Table VII
EFFECTS OF THE THREE TRICKS ON CPU TIME (IN SECONDS) ON THE MOVIELENS DATA.

	solver	100K	1M	10M
capped ℓ_1	baseline (GPG)	523.6	$> 10^4$	$> 10^5$
	✓ i	212.2	1920.5	$> 10^5$
	✓ i, ii	29.2	288.8	$> 10^5$
	✓ i, ii, iii (FaNCL)	**3.2**	**29.4**	**634.6**
LSP	baseline (GPG)	192.8	$> 10^4$	$> 10^5$
	✓ i	35.8	2353.8	$> 10^5$
	✓ i, ii	5.6	212.4	$> 10^5$
	✓ i, ii, iii (FaNCL)	**0.7**	**25.6**	**616.3**
TNN	baseline (GPG)	572.7	$> 10^4$	$> 10^5$
	✓ i	116.9	1944.8	$> 10^5$
	✓ i, ii	15.4	256.1	$> 10^5$
	✓ i, ii, iii (FaNCL)	**1.9**	**25.8**	**710.7**

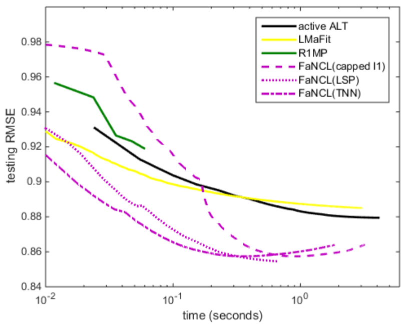
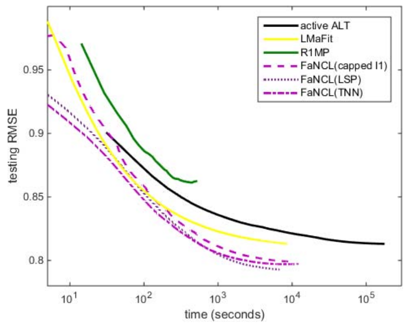
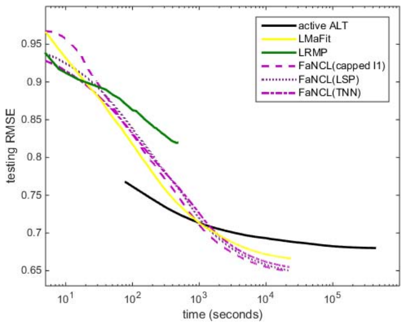


Figure 3. RMSE vs CPU time on the MovieLens-100K data set.



(a) netflix.



(b) yahoo.

Figure 4. RMSE vs CPU time on the netflix and yahoo data sets.

3) Netflix and Yahoo: Next, we perform experiments on two very large recommendation data sets, Netflix[9] and

[9]http://archive.ics.uci.edu/ml/datasets/Netflix+Prize

Table VI
MATRIX COMPLETION RESULTS ON THE MOVIELENS DATA SETS (TIME IS IN SECONDS).

		MovieLens-100K			MovieLens-1M			MovieLens-10M		
		RMSE	rank	time	RMSE	rank	time	RMSE	rank	time
nuclear norm	APG	0.879 ± 0.001	36	18.9	0.818 ± 0.001	67	735.8	—	—	$> 10^5$
	Soft-Impute	0.879 ± 0.001	36	13.8	0.818 ± 0.001	67	311.8	—	—	$> 10^5$
	active ALT	0.879 ± 0.001	36	4.1	0.818 ± 0.001	67	133.4	0.813 ± 0.001	119	3675.2
fixed rank	LMaFit	0.884 ± 0.001	2	3.0	0.818 ± 0.001	6	39.2	0.795 ± 0.001	9	650.1
	R1MP	0.924 ± 0.003	5	0.1	0.862 ± 0.004	19	2.9	0.850 ± 0.008	29	37.3
capped-ℓ_1	IRNN	0.863 ± 0.003	3	558.9	—	—	$> 10^4$	—	—	$> 10^5$
	GPG	0.863 ± 0.003	3	523.6	—	—	$> 10^4$	—	—	$> 10^5$
	FaNCL	0.863 ± 0.003	3	**3.2**	0.797 ± 0.001	5	**29.4**	0.783 ± 0.002	8	**634.6**
LSP	IRNN	$\mathbf{0.855} \pm 0.002$	2	195.9	—	—	$> 10^4$	—	—	$> 10^5$
	GPG	$\mathbf{0.855} \pm 0.002$	2	192.8	—	—	$> 10^4$	—	—	$> 10^5$
	FaNCL	$\mathbf{0.855} \pm 0.002$	2	**0.7**	$\mathbf{0.786} \pm 0.001$	5	**25.6**	$\mathbf{0.777} \pm 0.001$	9	**616.3**
TNN	IRNN	0.862 ± 0.003	3	621.9	—	—	$> 10^4$	—	—	$> 10^5$
	GPG	0.862 ± 0.003	3	572.7	—	—	$> 10^4$	—	—	$> 10^5$
	FaNCL	0.862 ± 0.003	3	**1.9**	0.797 ± 0.004	5	**25.8**	0.783 ± 0.002	8	**710.7**

Yahoo[10] (Table V). We randomly use 50% of the observed ratings for training, 25% for validation and the rest for testing. Each experiment is repeated five times.

Results are shown in Table VIII. APG, Soft-Impute, GPG and IRNN cannot be run as the data set is large. Figure 4 shows the objective and RMSE vs time for the compared methods.[11] Again, the nonconvex regularizers converge faster, yield lower RMSE's and solutions of much lower ranks. Moreover, FaNCL is fast.

B. Robust Principal Component Analysis

1) Synthetic Data: In this section, we first perform experiments on a synthetic data set. The observed $m \times m$ matrix is generated as $O = UV + \tilde{Y} + G$, where elements of $U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{k \times m}$ (with $k = 0.01m$) are sampled i.i.d. from $\mathcal{N}(0, 1)$, and elements of G are sampled from $\mathcal{N}(0, 0.1)$. Matrix $\tilde{Y}$ is sparse, with 1% of its elements randomly set to $5\|UV\|_\infty$ or $-5\|UV\|_\infty$ with equal probabilities. The sparsity regularizer is the standard ℓ_1, while different convex/nonconvex low-rank regularizers are used.

For performance evaluation, we use (i) NMSE $= \|(X + Y) - (UV + \tilde{Y})\|_F / \|UV + \tilde{Y}\|_F$, where X and Y are the recovered low-rank and sparse components, respectively in (5); (ii) accuracy on locating the sparse support of $\tilde{Y}$ (i.e., percentage of entries that both $\tilde{Y}_{ij}$ and Y_{ij} are nonzero or zero together); and (iii) the recovered rank. We vary m in $\{500, 1000, 1500, 2000\}$. Each experiment is repeated five times.

Note that IRNN and the active subspace selection method cannot be used here. Their objectives are of the form "smooth function plus low-rank regularizer", while RPCA has a nonsmooth ℓ_1 regularizer besides its low-rank regularizer. Similarly, Soft-Impute is for matrix completion only.

[10]http://webscope.sandbox.yahoo.com/catalog.php?datatype=c

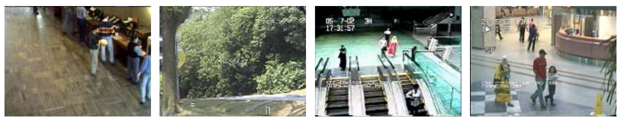
[11]On these two data sets, R1MP easily overfits as the rank increases. Hence, the validation set selects a rank which is small (relative to that obtained by the nuclear norm) and R1MP stops earlier. However, as can be seen, its RMSE is much worse.

Results are shown in Table IX. The accuracy on locating the sparse support are always 100% for all methods, and thus are not shown. As can be seen, while both convex and nonconvex regularizers can perfectly recover the matrix rank and sparse locations, the nonconvex regularizers have lower NMSE's. Moreover, as in matrix completion, FaNCL is again much faster. The larger the matrix, the higher is the speedup.

2) Background Removal on Videos: In this section, we use RPCA to perform video denoising on background removal of corrupted videos. Four benchmark videos[12] in [6, 7] are used (Table X), and example image frames are shown in Figure 5. As discussed in [6], the stable image background can be treated as low-rank, while the foreground moving objects contribute to the sparse component.

Table X
VIDEOS USED IN THE EXPERIMENT.

	bootstrap	*campus*	*escalator*	*hall*
#pixels / frame	19,200	20,480	20,800	25,344
total #frames	9,165	4,317	10,251	10,752



(a) *bootstrap*. (b) *campus*. (c) *escalator*. (d) *hall*.

Figure 5. Example image frames in the videos.

Each image frame is reshaped as a column vector, and all frames are then stacked together to form a matrix. The pixel values are normalized to $[0, 1]$, and Gaussian noise from $\mathcal{N}(0, 0.15)$ is added. The experiment is repeated five times.

For performance evaluation, we use the commonly used peak signal-to-noise ratio [34]: PSNR $= -10 \log_{10}(\text{MSE})$, where MSE $= \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} - O_{ij})^2$, $X \in \mathbb{R}^{m \times n}$ is the recovered video, and $O \in \mathbb{R}^{m \times n}$ is the ground-truth.

[12]http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html

Table VIII
RESULTS ON THE NETFLIX AND YAHOO DATA SETS (CPU TIME IS IN HOURS).

		netflix			yahoo		
		RMSE	rank	time	RMSE	rank	time
nuclear norm	active ALT	0.814 ± 0.001	399	47.6	0.680 ± 0.001	221	118.9
fixed rank	LMaFit	0.813 ± 0.003	16	2.4	0.667 ± 0.002	10	6.6
	R1MP	0.861 ± 0.006	31	0.2	0.810 ± 0.005	92	0.3
capped-ℓ_1	FaNCL	0.799 ± 0.001	15	2.5	$\mathbf{0.650} \pm 0.001$	8	5.9
LSP	FaNCL	$\mathbf{0.793} \pm 0.002$	13	1.9	$\mathbf{0.650} \pm 0.001$	9	6.1
TNN	FaNCL	0.798 ± 0.001	17	3.3	0.655 ± 0.002	8	6.2

Table IX
RPCA PERFORMANCE OF THE VARIOUS METHODS ON SYNTHETIC DATA. THE STANDARD DEVIATIONS OF NMSE ARE ALL SMALLER THAN 0.0002 AND SO NOT REPORTED. CPU TIME IS IN SECONDS.

		$m = 500$			$m = 1000$			$m = 1500$			$m = 2000$		
		NMSE	rank	time	NMSE	rank	time	NMSE	rank	time	NMSE	rank	time
nuclear norm	APG	0.46	5	1.5	0.30	10	9.7	0.25	15	33.9	0.18	20	94.7
capped-ℓ_1	GPG	**0.36**	5	0.9	**0.25**	10	6.7	**0.21**	15	18.7	**0.15**	20	60.4
	FaNCL	**0.36**	5	**0.2**	**0.25**	10	**1.4**	**0.21**	15	**2.7**	**0.15**	20	**6.5**
LSP	GPG	**0.36**	5	2.7	**0.25**	10	18.5	**0.21**	15	111.2	**0.15**	20	250.2
	FaNCL	**0.36**	5	**0.4**	**0.25**	10	**1.8**	**0.21**	15	**3.9**	**0.15**	20	**7.1**
TNN	GPG	**0.36**	5	0.8	**0.25**	10	6.0	**0.21**	15	23.1	**0.15**	20	51.4
	FaNCL	**0.36**	5	**0.2**	**0.25**	10	**1.2**	**0.21**	15	**2.9**	**0.15**	20	**5.8**

(a) original. (b) nuclear norm. (c) capped-ℓ_1. (d) LSP. (e) TNN.

Figure 6. Example foreground images in *bootstrap*, as recovered by using various low-rank regularizers.

Results are shown in Table XI. As can be seen, the nonconvex regularizers lead to better PSNR's than the convex nuclear norm. Moreover, FaNCL is more than 10 times faster than GPG. Figure 6 shows an example of the recovered foreground in the *bootstrap* video. As can been seen, the nonconvex regularizers can better separate foreground from background. Figure 7 shows the PSNR vs time on *bootstrap*. Again, FaNCL converges much faster than others.

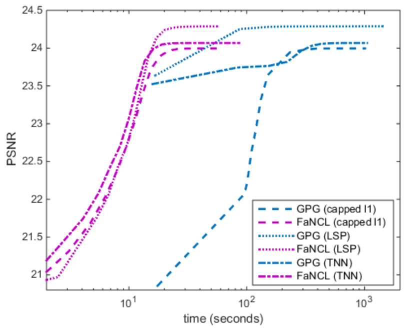


Figure 7. PSNR vs CPU time on the *bootstrap* data set.

V. CONCLUSION

In this paper, we considered the challenging problem of nonconvex low-rank matrix optimization. The key observations are that for the popular low-rank regularizers, the singular values obtained from the proximal operator can be automatically thresholded, and also that the proximal operator can be computed on a smaller matrix. For matrix completion, extra speedup can be achieved by exploiting the "sparse plus low-rank" structure of the matrix estimate in each iteration. The resultant algorithm is guaranteed to converge to a critical point of the nonconvex optimization problem. Extensive experiments on matrix completion and RPCA show that the proposed algorithm is much faster than the state-of-art convex and nonconvex low-rank solvers. It also demonstrates that nonconvex low-rank regularizers outperform the convex nuclear norm regularizer in terms of recovery accuracy and the rank obtained.

ACKNOWLEDGMENT

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 614513).

REFERENCES

[1] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[2] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Transactions on Sensor Networks*, vol. 2, no. 2, pp. 188–220, 2006.

Table XI
PSNR (in dB) and CPU time (in seconds) on the video background removal experiment. For comparison, the PSNRs for all the input videos are 16.47dB.

		bootstrap		*campus*		*escalator*		*hall*	
		PSNR	time	PSNR	time	PSNR	time	PSNR	time
nuclear norm	APG	23.01±0.03	688.4	22.90±0.02	102.6	23.56±0.01	942.5	23.62±0.01	437.7
capped-ℓ_1	GPG	24.00±0.03	1009.3	23.14±0.02	90.6	**24.33**±0.02	1571.2	24.95±0.02	620.0
	FaNCL	24.00±0.03	**60.4**	23.14±0.02	**12.4**	**24.33**±0.02	**68.3**	24.95±0.02	**34.7**
LSP	GPG	**24.29**±0.03	1420.2	**23.96**±0.02	88.9	24.13±0.01	1523.1	**25.08**±0.01	683.9
	FaNCL	**24.29**±0.03	**56.0**	**23.96**±0.02	**17.4**	24.13±0.01	**54.5**	**25.08**±0.01	**35.8**
TNN	GPG	24.06±0.03	1047.5	23.11±0.02	130.3	24.29±0.01	1857.7	24.98±0.02	626.2
	FaNCL	24.06±0.03	**86.3**	23.11±0.02	**12.6**	24.29±0.01	**69.6**	24.98±0.02	**37.4**

[3] M. Kim and J. Leskovec, “The network completion problem: Inferring missing nodes and edges in networks,” in *Proceedings of the 11th International Conference on Data Mining*, 2011, pp. 47–58.

[4] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[5] Q. Yao and J. Kwok, “Colorization by patch-based local low-rank matrix completion,” 2015.

[6] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?” *Journal of the ACM*, vol. 58, no. 3, p. 11, 2011.

[7] Q. Sun, S. Xiang, and J. Ye, “Robust principal component analysis via capped norms,” in *Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 311–319.

[8] G. Zhu, S. Yan, and Y. Ma, “Image tag refinement towards low-rank, content-tag prior and error sparsity,” in *Proceedings of the International Conference on Multimedia*, 2010, pp. 461–470.

[9] P. Gong, J. Ye, and C. Zhang, “Robust multi-task feature learning,” in *Proceedings of the 18th international Conference on Knowledge Discovery and Data Mining*, 2012, pp. 895–903.

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

[11] R. Mazumder, T. Hastie, and R. Tibshirani, “Spectral regularization algorithms for learning large incomplete matrices,” *Journal of Machine Learning Research*, vol. 11, pp. 2287–2322, 2010.

[12] C.-J. Hsieh and P. Olsen, “Nuclear norm minimization via active subspace selection,” in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 575–583.

[13] T. Zhang, “Analysis of multi-stage convex relaxation for sparse regularization,” *Journal of Machine Learning Research*, vol. 11, pp. 1081–1107, 2010.

[14] E. J. Candès, M. B. Wakin, and S. P. Boyd, “Enhancing sparsity by reweighted ℓ_1 minimization,” *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[15] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He, “Fast and accurate matrix completion via truncated nuclear norm regularization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2117–2130, 2013.

[16] J. Fan and R. Li, “Variable selection via nonconcave penalized likelihood and its oracle properties,” *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[17] C.-H. Zhang, “Nearly unbiased variable selection under minimax concave penalty,” *Annals of Statistics*, vol. 38, no. 2, pp. 894–942, 2010.

[18] A. L. Yuille and A. Rangarajan, “The concave-convex procedure,” *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.

[19] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye, “A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems,” in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 37–45.

[20] C. Lu, J. Tang, S. Yan, and Z. Lin, “Generalized nonconvex nonsmooth low-rank minimization,” in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2014, pp. 4130–4137.

[21] C. Lu, C. Zhu, C. Xu, S. Yan, and Z. Lin, “Generalized singular value thresholding,” in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[22] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[23] N. Parikh and S. Boyd, “Proximal algorithms,” *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[24] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[25] S. Wang, D. Liu, and Z. Zhang, “Nonconvex relaxation approaches to robust matrix recovery,” in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013, pp. 1764–1770.

[26] R. M. Larsen, “Lanczos bidiagonalization with partial reorthogonalization,” Department of Computer Science, Aarhus University, DAIMI PB-357, 1998.

[27] K.-C. Toh and S. Yun, “An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems,” *Pacific Journal of Optimization*, vol. 6, no. 615-640, p. 15, 2010.

[28] Z. Wen, W. Yin, and Y. Zhang, “Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm,” *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[29] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye, “Rank-one matrix pursuit for matrix completion,” in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 91–99.

[30] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, “Robust recovery of subspace structures by low-rank representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

[31] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[32] X. Zhang, D. Schuurmans, and Y.-l. Yu, “Accelerated training for matrix-norm regularization: A boosting approach,” in *Advances in Neural Information Processing Systems*, 2012, pp. 2906–2914.

[33] H. Avron, S. Kale, V. Sindhwani, and S. P. Kasiviswanathan, “Efficient and practical stochastic subgradient descent for nuclear norm regularization,” in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 1231–1238.

[34] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-D transform-domain collaborative filtering,” *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.