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-$\ell_1$ 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 re-weighted 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 an $m \times n$ 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

\[ 1 \text{Code is at: https://github.com/quanmingyao/FaNCL} \]
operator can be automatically thresholded. One then only 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 \( (\cdot)^T \). 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 \( i \)th diagonal element is \( x_i \). Moreover, \( I \) denotes the identity matrix. For a differentiable function \( f \), we use \( \nabla f \) for its gradient. For a nonsmooth function, we use \( \partial f \) for its subdifferential.

II. BACKGROUND

A. Proximal Gradient Algorithms

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

\[
\min_x F(x) \equiv f(x) + \lambda r(x),
\]

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 \( \rho \)-Lipschitz continuous gradient, i.e., \( \|\nabla f(X_1) - \nabla f(X_2)\|_F \leq \rho \|X_1 - X_2\|_F \). Without loss of generality, we assume that \( \rho \leq 1 \).

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 \( \tau \), the next iterate \( x^{t+1} \) is obtained as

\[
\arg\min_x \nabla f(x^t)^\top (x - x^t) + \frac{\tau}{2} \|x - x^t\|^2 + \lambda r(x) = \arg\min_x \frac{1}{2} \|x - x^t\|^2 + \frac{\lambda}{\tau} r(x) \equiv \text{prox}_{\frac{\lambda}{\tau} r}(z^t),
\]

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 \( X \).

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] \( \text{prox}_{\mu r}(X) = U (\Sigma - \mu I)_+ V^\top \), 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 \( r(X) = \sum_{i=1}^m \tilde{r}(\sigma_i) \), where \( \sigma_1 \geq \cdots \geq \sigma_m \geq 0 \) are singular values of \( X \), and \( \tilde{r}(\sigma) \) is a concave and non-decreasing function of \( \sigma \geq 0 \) with \( \tilde{r}(0) = 0 \).

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

| \( \tilde{r} \)'S FOR SOME POPULAR NONCONVEX LOW-RANK REGULARIZERS, FOR THE TNN REGULARIZER, \( \theta \in \{1, \ldots, n\} \) IS THE NUMBER OF LEADING SINGULAR VALUES THAT ARE NOT PENALIZED. |
|-----------------|-----------------|-----------------|
| capped-\( \ell_1 \) | \( \mu \min(\sigma_i, \theta) \), \( \theta > 0 \) | \( \mu \log \left( \frac{\mu}{\theta + 1} \right) \), \( \theta > 0 \) |
| LSP | \( \mu \sigma_i \), \( \theta > 0 \) | \( \mu \sigma_i > \theta \mu \) |
| TNN | \( \mu \sigma_i > \theta \mu \) \( \sigma_i \leq \mu \) | \( \mu \leq \theta \sigma_i \) \( \sigma_i > \theta \mu \) |
| SCAD | \( \mu \sigma_i - \frac{\mu \sigma_i^2}{\theta + 1} \) \( \sigma_i \leq \theta \mu \) \( \theta > 0 \) | \( \sigma_i > \theta \mu \) |
| MCP | \( \mu \sigma_i - \frac{\mu \sigma_i^2}{\theta} \) \( \sigma_i > \theta \mu \) |

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 \( 0 \leq w_1 \leq \cdots \leq w_m \). 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-\( \ell_1 \)), [15] (for the TNN), and [25] (for the MCP). All these (including IRNN) need SVD in each iteration. It takes \( O(mn^2) \) time, and thus can be slow.

\(^2\)In the following, we assume \( m \leq n \).
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] \( \text{prox}_{\mu r}(X) = U \text{Diag}(y^*) V^T \), where \( U \Sigma V^T \) is the SVD of \( X \), and \( y^* = [y^*_1, \ldots, y^*_n] \) with

\[
y^*_i \in \arg\min_{\|y_i\|_2} \frac{1}{2} (y_i - \sigma_i)^2 + \mu r(y_i). \tag{2}
\]

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 \( \text{prox}_{\mu r}() \) can be computed more efficiently. It is based on two ideas. First, the singular values in \( \text{prox}_{\mu r}() \) are automatically thresholded. Second, \( \text{prox}_{\mu r}() \) 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 \( \sigma_i \) 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 \( i \) 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 \( \gamma \) values are equal to

- **capped-\ell_1: \( \gamma = \min (\mu, \theta + \frac{\mu}{2}) \);**
- **LSP: \( \gamma = \min \left( \frac{\gamma}{\theta}, \theta \right) \);**
- **TNN: \( \gamma = \max (\mu, \sigma_{\theta+1}) \);**
- **SCAD: \( \gamma = \mu; \)**
- **MCP: \( \gamma = \sqrt{\theta} \mu \) if \( 0 < \theta < 1 \), and \( \mu \) 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.

#### B. Proximal Operator on a Smaller Matrix

Assume that \( Z^t \) has \( \hat{k} \leq n \) singular values larger than \( \gamma \), and its rank-\( \hat{k} \) SVD is \( U_\hat{k} \Sigma_\hat{k} V_\hat{k}^T \). 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 \( (U_\hat{k} \Sigma_\hat{k}) \subseteq \text{span}(Q) \). Then, \( \text{prox}_{\mu r}(Z^t) = Q \cdot \text{prox}_{\mu r}(Q^T Z^t) \).

Though SVD is still needed to obtain \( \text{prox}_{\mu r}(Q^T Z^t) \), \( Q^T Z^t \) is much smaller than \( Z^t \) \((k \times n \text{ vs } m \times n)\). 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 \( O(m^2n) \) 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 \( X^p \) will be equal to that of \( \text{prox}_{\mu r}(Z^t) \). In each iteration, we ensure a sufficient decrease of the objective:

\[
F(X^{t+1}) \leq F(X^t) - c_1 \|X^{t+1} - X^t\|_2^2, \tag{3}
\]

where \( c_1 = \frac{\gamma - \theta}{4} \); otherwise, the power method is restarted. Moreover, similar to [12, 27], steps 6–7 use the column spaces of the previous iterates \( (V^t \text{ and } V^{t-1}) \) to warm-start the power method. For further speedup, we employ a continuation strategy as in [11, 20, 27]. Specifically, \( \lambda^t \) 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 \( \gamma \) is equal to \( \lambda^t / \tau \), 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 method.

**Algorithm 1** Power method to obtain an approximate left subspace of \( Z \).

1. \( Y^1 \leftarrow ZR; \)
2. for \( t = 1, 2, \ldots, T_{pm} \) do
3. \( Q^t+1 = QR(Y^t); \) // QR decomposition
4. \( Y^{t+1} = Z(Z^T Q^t+1); \)
5. end for
6. return \( Q^T_{pm+1} \).
Algorithm 2 FanCL (Fast NonConvex Low-rank).

1: choose $\tau > \rho$, $c_1 = \frac{\tau - \rho}{\rho}$, $\lambda^0 > \lambda$ 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 $t = 1, 2, \ldots, T$ do
4: $\lambda^t \leftarrow (\lambda^{t-1} - \lambda)\nu + \lambda$;
5: $Z^t \leftarrow X^t - \frac{1}{2} \nabla f(X^t)$;
6: $V^{t-1} \leftarrow \nabla f(X^t) V^{t-1}$, and remove any zero columns;
7: $R^t \leftarrow QR(V^t, V^{t-1})$;
8: for $p = 1, 2, \ldots, \ldots$ do
9: $Q \leftarrow \text{PowerMethod}(Z^t, R^p)$;
10: $[U^p, \Sigma^p, V^p] \leftarrow \text{SVD}(Q^T Z^t)$;
11: $k \leftarrow$ number of $\sigma^t$’s are $\gamma$ in Corollary III.2;
12: $\hat{U}^p \leftarrow k$ leading columns of $U^p$;
13: $\tilde{V}^p \leftarrow k$ leading columns of $V^p$;
14: for $i = 1, 2, \ldots, k$ do
15: obtain $y_i^p$ from (2) with $\mu = 1/\tau$ and $\lambda^t$;
16: end for
17: $\bar{X}^p \leftarrow (Q\hat{U}^p)\text{Diag}(y_1^p, \ldots, y_k^p)(\tilde{V}^p)^T$;
18: if $F(\bar{X}^p) < F(X^t) - c_i \|\bar{X}^p - X^t\|_F^2$ then
19: $X^{t+1} \leftarrow X^t$, $V^{t+1} \leftarrow \tilde{V}^p$;
20: break;
21: else
22: $R^{p+1} = V^p$;
23: end if
24: end for
25: end for
26: return $X^{T+1}$.

The following shows that it is also a critical point.\footnote{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 $0 \in \nabla f(X^*) + \lambda \nu r_1(X^*) - \lambda \nu r_2(X^*)$, where $r(X) = r_1(X) - r_2(X)$, and $r_1$ and $r_2$ are convex.}

**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.** \[ \min_{t=1, \ldots, T} \|X^{t+1} - X^t\|_F^2 \leq \frac{1}{T} \left[ F(X^1) - F(X^*) \right] \]

**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 $f(X) = \frac{1}{2} \|P_\Omega (X - O)\|_F^2$, where $P_\Omega (A)|_{\Omega_{ij}} = A_{ij}$ if $\Omega_{ij} = 1$ and 0 otherwise, and $\tau$ is a low-rank regularizer.

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

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

**Proposition III.7.** Let the reduced SVD of $X$ be $USV^T$, and $P, Q$ be orthonormal matrices such that span($P$) $\subseteq$ span($X$) and span($V$) $\subseteq$ span($Q$). Then $\|X\|_F = \|P^T X Q\|_F$.

Let the reduced SVDs of $\bar{X}^p$ and $X^t$ be $\hat{U} \hat{S} \hat{V}^\top$ and $U^t \Sigma^t V^{t\top}$, respectively. Let $P = QR(\hat{U}, U^t)$ and $Q = QR(\hat{V}, V^t)$. Using Proposition III.7, $\|\bar{X}^p - X^t\|_F = \|P^T (\bar{X}^p - X^t) Q\|_F = \|P^T \hat{U} \hat{S} \hat{V}^\top Q - P^T U^t \Sigma^t V^{t\top} Q\|_F$. This takes $O(nk^2)$ instead of $O(mntm)$ time. The per-iteration time complexity is reduced from $O(mntm)$ to $O((nk + T_{\Omega m}|\Omega|_1)k)$ 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 $X_1, \ldots, X_N$:

$$ \min \{F(X_1, \ldots, X_N) \equiv f(X_1, \ldots, X_N) + \sum_{i=1}^{N} \lambda_i r_i(X_i) \} \quad (4) $$
Assumptions 1A-1A 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 \( \rho_i \), where \( f_i(X) \) is the function obtained by keeping all the \( X_j \)'s (where \( i \neq j \)) if \( f \) 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 plus sparse data noise \( Y \). Mathematically, we have

\[
\min_{X,Y} F(X,Y) = f(X,Y) + \lambda r(X) + \beta \| Y \|_1, \tag{5}
\]

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 \( \ell_1 \) regularizer \( \| \cdot \|_1 \) are nonsmooth, (5) does not fit into formulation (1). Besides RPCA, problems such as subspace clustering [30], multitask 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 \( t \):

\[
X^{t+1} = \arg\min_X \frac{1}{2} \| X - Z^t_X \|_F^2 + \frac{\rho}{\lambda} r(X) = \prox_{\frac{\rho}{\lambda} r} (Z^t_X), \quad \tag{6}
\]

\[
Y^{t+1} = \arg\min_Y \frac{1}{2} \| Y - Z^t_Y \|_F^2 + \frac{\beta}{\lambda} \| Y \|_1 = \prox_{\frac{\beta}{\lambda} \| \cdot \|_1} (Z^t_Y), \quad \tag{7}
\]

where \( Z^t_X = X^t - \frac{1}{\tau} \nabla f(X^t,Y^t) \), and \( Z^t_Y = Y^t - \frac{1}{\tau} \nabla f(X^{t+1},Y^t) \). \( Y^{t+1} \) can be easily obtained as \( Y^{t+1} = \text{sign}(Z^t_{Yi}) \frac{([Z^t_{Yi}])}{\beta + \frac{2}{\tau} \| Y \|_1} \), where \( \text{sign}(x) \) denotes the sign of \( x \). Similar to (3), we ensure a sufficient decrease of the objective in each iteration:

\[
F_{Y^t}(X^{t+1}) \leq F_{Y^t}(X^t) - c_1 \| X^{t+1} - X^t \|_F^2,
\]

\[
F_{X^{t+1}}(Y^{t+1}) \leq F_{X^{t+1}}(Y^t) - c_1 \| Y^{t+1} - Y^t \|_F^2,
\]

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 \( \{X_1, \ldots, X_N\} \) generated by the algorithm, we have

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

**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: \( \min_{U,V} \frac{1}{2} \| P \|_F (U V - O) \|_F^2 + \frac{1}{2} (\| U \|_F^2 + \| V \|_F^2) \); and (iii) nonconvex regularizers, including the capped-\( \ell_1 \) (with \( \theta \) in Table I set to 2\( \lambda \)), LSP (with \( \theta = \sqrt{\lambda} \)), and TNN (with \( \theta = 3 \)).

The nuclear norm minimization algorithms to be compared include:

1) Accelerated proximal gradient (APG)\(^4\) algorithm [10, 27], with the partial SVD by PROPACK [26];
2) Soft-Impute\(^5\) [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\(^6\) (denoted “active AL-T”)[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\(^7\) [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)\(^8\) [20];

---

\(^4\)http://perception.csl.illinois.edu/matrix-rank/Files/apg_partial.zip
\(^5\)http://cran.r-project.org/web/packages/softImpute/index.html
\(^6\)http://www.cs.utexas.edu/~cjhsieh/nuclear_active_1.1.zip
\(^7\)http://www.caam.rice.edu/~optimization/L1/MLaFit/download.html
\(^8\)https://sites.google.com/site/canyihui/file/2014-CVPR-IRNN.zip?
attredirects=0&d=1
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 ($T_{prm} = 3$, $p = 1$).

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 $\mathbf{O} = \mathbf{UV} + \mathbf{G}$, where the elements of $\mathbf{U} \in \mathbb{R}^{m \times k}, \mathbf{V} \in \mathbb{R}^{k \times m}$ (with $k = 5$) are sampled i.i.d. from the normal distribution $\mathcal{N}(0, 1)$, and elements of $\mathbf{G}$ sampled from $\mathcal{N}(0, 0.1)$. A total of $||\Omega|| = 2mk \log(m)$ random elements in $\mathbf{O}$ 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 the normalized mean squared error $\text{NMSE} = \sqrt{\sum_{(i,j) \notin \Omega} (\hat{X}_{ij} - [\mathbf{U} \mathbf{V}]_{ij})^2 / \sum_{(i,j) \notin \Omega} [\mathbf{U} \mathbf{V}]_{ij}^2}$, where $\hat{X}$ is the recovered matrix; (ii) rank of $\hat{X}$; 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-$\ell_1$, LSP and TNN) lead to much lower NMSE’s than the convex 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.
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 $\Omega$: $\text{RMSE} = \sqrt{\|P_{\Omega}(X - O)\|_F/\|\Omega\|_1}$, where $X$ is the recovered matrix. The experiment is repeated five times.

Table V
RECOMMENDATION DATA SETS USED IN THE EXPERIMENTS.

<table>
<thead>
<tr>
<th></th>
<th>#users</th>
<th>#movies</th>
<th>#ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>100K</td>
<td>943</td>
<td>1,682</td>
</tr>
<tr>
<td></td>
<td>1M</td>
<td>6,040</td>
<td>3,449</td>
</tr>
<tr>
<td></td>
<td>10M</td>
<td>69,878</td>
<td>10,677</td>
</tr>
<tr>
<td>netflix</td>
<td>480,189</td>
<td>17,770</td>
<td>100,480,507</td>
</tr>
<tr>
<td>yahoo</td>
<td>249,012</td>
<td>296,111</td>
<td>62,551,438</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>solver</th>
<th>100K</th>
<th>1M</th>
<th>10M</th>
</tr>
</thead>
<tbody>
<tr>
<td>capped $\ell_1$</td>
<td>baseline (GPG)</td>
<td>523.6</td>
<td>$&gt;10^4$</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i</td>
<td>212.2</td>
<td>1920.5</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i, ii</td>
<td>29.2</td>
<td>288.8</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i, ii, iii (FaNCL)</td>
<td>5.2</td>
<td>29.4</td>
<td>654.6</td>
</tr>
<tr>
<td>LSP</td>
<td>baseline (GPG)</td>
<td>192.8</td>
<td>$&gt;10^4$</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i</td>
<td>35.8</td>
<td>2353.8</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i, ii</td>
<td>5.6</td>
<td>212.4</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i, ii, iii (FaNCL)</td>
<td>0.7</td>
<td>25.6</td>
<td>616.3</td>
</tr>
<tr>
<td>TNN</td>
<td>baseline (GPG)</td>
<td>572.7</td>
<td>$&gt;10^4$</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i</td>
<td>116.9</td>
<td>1944.8</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i, ii</td>
<td>15.4</td>
<td>256.1</td>
<td>$&gt;10^5$</td>
</tr>
<tr>
<td></td>
<td>✓ i, ii, iii (FaNCL)</td>
<td>1.9</td>
<td>25.8</td>
<td>710.7</td>
</tr>
</tbody>
</table>

3) Netflix and Yahoo: Next, we perform experiments on two very large recommendation data sets, Netflix$^{10}$ and

$^{10}$http://archive.ics.uci.edu/ml/datasets/Netflix+Prize
Table VI
MATRIX COMPLETION RESULTS ON THE MOVIELENS DATA SETS (TIME IS IN SECONDS).

<table>
<thead>
<tr>
<th>Matrix Completion</th>
<th>MovieLens-100K</th>
<th>MovieLens-1M</th>
<th>MovieLens-10M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>rank</td>
<td>time</td>
</tr>
<tr>
<td>Nuclear norm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APG</td>
<td>0.879±0.001</td>
<td>36</td>
<td>18.9</td>
</tr>
<tr>
<td>Soft-Impute active ALT</td>
<td>0.879±0.001</td>
<td>36</td>
<td>13.8</td>
</tr>
<tr>
<td>fixed rank</td>
<td>0.884±0.001</td>
<td>2</td>
<td>3.0</td>
</tr>
<tr>
<td>RIMP</td>
<td>0.924±0.003</td>
<td>7</td>
<td>0.1</td>
</tr>
<tr>
<td>capped-\ell_1</td>
<td>IRNN</td>
<td>0.863±0.003</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GPG</td>
<td>0.863±0.003</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>0.863±0.003</td>
<td>5</td>
</tr>
<tr>
<td>LSP</td>
<td>IRNN</td>
<td>0.855±0.002</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>GPG</td>
<td>0.855±0.002</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>0.856±0.002</td>
<td>2</td>
</tr>
<tr>
<td>TNN</td>
<td>IRNN</td>
<td>0.862±0.003</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>GPG</td>
<td>0.862±0.003</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>0.862±0.003</td>
<td>3</td>
</tr>
</tbody>
</table>

Yahoo\(^{11}\) (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.\(^{12}\) 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 \(Y = UV + \tilde{Y} + G\), where elements of \(U \in \mathbb{R}^{n \times k}\), \(V \in \mathbb{R}^{k \times m}\) (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 \(\ell_1\), 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 \(\hat{Y}\) (i.e., percentage of entries that both \(Y_{ij}\) and \(\hat{Y}_{ij}\) 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 \(\ell_1\) regularizer besides its low-rank regularizer. Similarly, Soft-Impute is for matrix completion only.

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 NME’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\(^{13}\) 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.

<table>
<thead>
<tr>
<th>Video</th>
<th>bootstrap</th>
<th>campus</th>
<th>escalator</th>
<th>hull</th>
</tr>
</thead>
<tbody>
<tr>
<td>#pixels / frame</td>
<td>19,200</td>
<td>20,340</td>
<td>20,800</td>
<td>25,344</td>
</tr>
<tr>
<td>total frames</td>
<td>9,165</td>
<td>4,317</td>
<td>10,251</td>
<td>10,752</td>
</tr>
</tbody>
</table>

(a) bootstrap. (b) campus. (c) escalator. (d) hull.

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.

\(^{11}\)http://webscope.sandbox.yahoo.com/catalog.php?datatype=e

\(^{12}\)On these two data sets, RIMP 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 RIMP stops earlier. However, as can be seen, its RMSE is much worse.

\(^{13}\)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).

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th>RMSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>netflix</td>
<td></td>
<td>yahoo</td>
<td></td>
</tr>
<tr>
<td>nuclear norm</td>
<td>active ALT</td>
<td></td>
<td>rank</td>
<td>time</td>
</tr>
<tr>
<td>fixed rank</td>
<td>LMaFit</td>
<td>0.814 ± 0.001</td>
<td>399</td>
<td>47.6</td>
</tr>
<tr>
<td></td>
<td>R1MP</td>
<td>0.861 ± 0.006</td>
<td>31</td>
<td>0.2</td>
</tr>
<tr>
<td>capped-ℓ₁</td>
<td>FaNCL</td>
<td>0.799 ± 0.001</td>
<td>15</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>LSP</td>
<td>0.793 ± 0.002</td>
<td>13</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>TNN</td>
<td>0.798 ± 0.001</td>
<td>17</td>
<td>3.3</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>m = 500</th>
<th>m = 1000</th>
<th>m = 1500</th>
<th>m = 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMSE</td>
<td>rank</td>
<td>time</td>
<td>NMSE</td>
</tr>
<tr>
<td>nuclear norm</td>
<td>APG</td>
<td>0.46</td>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td>capped-ℓ₁</td>
<td>GPG</td>
<td>0.36</td>
<td>5</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>0.36</td>
<td>5</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>LSP</td>
<td>0.36</td>
<td>5</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>0.36</td>
<td>5</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>TNN</td>
<td>0.36</td>
<td>5</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>FaNCL</td>
<td>0.36</td>
<td>5</td>
<td>0.2</td>
</tr>
</tbody>
</table>

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

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 obser-

vations 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


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 16.47 dB.

<table>
<thead>
<tr>
<th></th>
<th>bootstrap</th>
<th>campus</th>
<th>escavator</th>
<th>hall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>time</td>
<td>PSNR</td>
<td>time</td>
</tr>
<tr>
<td>nuclear norm</td>
<td>GPG</td>
<td>25.01±0.03</td>
<td>688.4</td>
<td>22.90±0.02</td>
</tr>
<tr>
<td></td>
<td>FanCL</td>
<td>24.90±0.03</td>
<td>60.4</td>
<td>23.14±0.02</td>
</tr>
<tr>
<td>capped-ℓ1</td>
<td>GPG</td>
<td>24.90±0.03</td>
<td>1009.3</td>
<td>23.14±0.02</td>
</tr>
<tr>
<td></td>
<td>FanCL</td>
<td>24.90±0.03</td>
<td>60.4</td>
<td>23.14±0.02</td>
</tr>
<tr>
<td>LSP</td>
<td>GPG</td>
<td>24.09±0.03</td>
<td>1420.2</td>
<td>23.96±0.02</td>
</tr>
<tr>
<td></td>
<td>FanCL</td>
<td>24.29±0.03</td>
<td>60.4</td>
<td>23.96±0.02</td>
</tr>
<tr>
<td>TNN</td>
<td>GPG</td>
<td>24.06±0.03</td>
<td>1047.5</td>
<td>23.11±0.02</td>
</tr>
<tr>
<td></td>
<td>FanCL</td>
<td>24.06±0.03</td>
<td>60.4</td>
<td>23.11±0.02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PSNR</th>
<th>time</th>
<th>PSNR</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>time</td>
<td>PSNR</td>
<td>time</td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>time</td>
<td>PSNR</td>
<td>time</td>
</tr>
</tbody>
</table>


