Efficient Nonconvex Regularized Tensor Completion with Structure-aware Proximal Iterations

Quanming Yao\textsuperscript{1,2} James T. Kwok\textsuperscript{2} Bo Han\textsuperscript{3}

Abstract
Nonconvex regularizers have been successfully used in low-rank matrix learning. In this paper, we extend this to the more challenging problem of low-rank tensor completion. Based on the proximal average algorithm, we develop an efficient solver that avoids expensive tensor folding and unfolding. A special “sparse plus low-rank” structure, which is essential for fast computation of individual proximal steps, is maintained throughout the iterations. We also incorporate adaptive momentum to further speed up empirical convergence. Convergence results to critical points are provided under smoothness and Kurdyka-Lojasiewicz conditions. Experimental results on a number of synthetic and real-world data sets show that the proposed algorithm is more efficient in both time and space, and is also more accurate than existing approaches.

1. Introduction
Tensors, which can be seen as high-order matrices, can be used to describe multilinear relationships inside the data (Kolda & Bader, 2009; Song et al., 2017). They have been popularly used in areas such as computer vision (Vasilescu & Terzopoulos, 2002), recommender systems (Candès & Recht, 2009), and signal processing (Cichocki et al., 2015). Many of these are third-order tensors, which are the focus in this paper. Examples include color images (Liu et al., 2013) and hyperspectral images (Signoretto et al., 2011). In Youtube, users can follow each other and can belong to the same subscribed channels. By treating channels as the third dimension, the users’ co-subscription network can again be represented as a third-order tensor (Lei et al., 2009).

In many applications, only a few tensor entries are observed. For example, often each Youtube user only interacts with a few other users (Lei et al., 2009; Davis et al., 2011). Tensor completion, which aims at filling in this partially observed tensor, has attracted a lot of interest (Rendle & Schmidt-Thieme, 2010; Signoretto et al., 2011; Bahadori et al., 2014; Cichocki et al., 2015). In the related task of matrix completion, different rows/columns of the underlying full matrix often share similar characteristics, and the matrix is thus low-rank (Candès & Recht, 2009). The nuclear norm, which is the tightest convex envelope of the rank (Boyd & Vandenberghe, 2009), is popularly used as a surrogate for the matrix rank in low-rank matrix completion (Cai et al., 2010; Mazumder et al., 2010).

In tensor completion, the low-rank assumption can also capture relatedness in the different tensor dimensions (Tomioka et al., 2010; Acar et al., 2011; Song et al., 2017). However, tensors are more complicated than matrices. Indeed, even computation of the tensor rank is NP-hard (Hillar & Lim, 2013). In recent years, many convex relaxations based on the matrix nuclear norm have been proposed for tensors. Examples include the tensor trace norm (Cheng et al., 2016), overlapped nuclear norm (Tomioka et al., 2010; Gandy et al., 2011), and latent nuclear norm (Tomioka et al., 2010). In particular, the overlapped nuclear norm is the most popular, as it (i) can be computed exactly (Cheng et al., 2016), (ii) has better low-rank approximation (Tomioka et al., 2010), and (iii) can lead to exact recovery (Tomioka et al., 2011; Tomioka & Suzuki, 2013; Mu et al., 2014).

However, the (overlapped) nuclear norm equally penalizes all singular values. Intuitively, larger singular values are more informative and should be less penalized (Mazumder et al., 2010; Lu et al., 2016; Yao et al., 2018a). In matrix completion, various adaptive nonconvex regularizers have been recently introduced to alleviate this problem. Examples include the capped-$\ell_1$ norm (Zhang, 2010b), log-sum-penalty (LSP) (Candès et al., 2008), truncated nuclear norm (TNN) (Hu et al., 2013), smoothed-capped-absolute-deviation (SCAD) (Fan & Li, 2001) and minimax concave penalty (MCP) (Zhang, 2010a). All these assign smaller penalties to the larger singular values. This leads to better empirical performance (Lu et al., 2016; Gu et al., 2017; Yao...
Efficient Nonconvex Regularized Tensor Completion with Structure-aware Proximal Iterations

Motivating by the success of adaptive nonconvex regularizers in matrix completion, we propose to develop a nonconvex variant of the overlapped nuclear norm regularizer for tensor completion. Unlike the standard convex tensor completion problem, the resulting optimization problem is nonconvex and more difficult to solve.

Based on the proximal average algorithm (Bauschke et al., 2008), we develop in this paper an efficient solver with much better time and space complexities. The keys to its success are on (i) avoiding expensive tensor folding and unfolding, (ii) maintaining a “sparse plus low-rank” structure on the iterates, and (iii) incorporating the adaptive momentum (Li et al., 2017). Convergence guarantees to critical points are provided under smoothness and Kurdyka-Lojasiewicz (Attouch et al., 2013) conditions. Experiments on a number of synthetic and real-world data sets show that the proposed algorithm is efficient and has much better empirical performance than other low-rank tensor regularization and decomposition methods.

Notation. Vectors are denoted by lowercase boldface, matrices by uppercase boldface, and tensors by boldface Euler. For a matrix $A \in \mathbb{R}^{m \times n}$ (assume that $m \geq n$) with singular values $\sigma_i$’s, its nuclear norm is $\|A\|_* = \sum_i \sigma_i$. For tensors, we follow the notation in (Kolda & Bader, 2009). For a third-order tensor $X \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ (without loss of generality, we assume $I_1 \geq I_2 \geq I_3$), its $(i_1, i_2, i_3)_{th}$ entry is $X_{i_1i_2i_3}$. One can unfold $X$ along its $d$th mode to obtain the matrix $X(d) \in \mathbb{R}^{I_d \times (I_1/I_d)I_2}$, where $(i_d, j)$ entry (where $j=1+\sum_{l=1,l \neq d}(i_l-1)I_{m=1,m \neq d}I_m$) is $X_{i_1i_2i_3}$. One can also fold a matrix $X$ back to a tensor $X(\bar{d})$, with $X_{i_1i_2i_3} = X_{i_dj}$, and $j$ as defined above. The inner product of two third-order tensors $X$ and $Y$ is $(X, Y) = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} X_{i_1i_2i_3}Y_{i_1i_2i_3}$. The Frobenius norm of $X$ is $\|X\|_F = \sqrt{(X, X)}$. For a proper and lower-semi-continuous function $f$, $\partial f$ denotes its Frechet subdifferential (Attouch et al., 2013). If $f$ is $\rho$-Lipschitz smooth, then $\|\nabla f(X) - \nabla f(Y)\|_F^2 \leq \rho \|X-Y\|_F^2$ for any $X$ and $Y$.

2. Related Works

2.1. Low-Rank Matrix Learning

Low-rank matrix learning can be formulated as the following optimization problem:

$$\min_X f(X) + \lambda r(X),$$

where $r$ is a low-rank regularizer (a common choice is the nuclear norm), $\lambda \geq 0$ is a hyper-parameter, and $f$ is a $\rho$-Lipschitz smooth loss. Using the proximal algorithm (Parikh & Boyd, 2013), the iterate is given by $X_{t+1} = \text{prox}_{\frac{\lambda}{\tau} \cdot} (Z_t)$, where

$$Z_t = X_t - \frac{1}{\tau} \nabla f(X_t),$$

$\tau > \rho$ is the stepsize, and

$$\text{prox}_{\frac{\lambda}{\tau} \cdot} (Z) = \arg \min_X \frac{1}{2} \|X - Z\|_F^2 + \frac{\lambda}{\tau} \|X\|_*$$

is the proximal step. The following Lemma shows that it can be obtained from the SVD of $Z$. Note that shrinking of the singular values encourages $X_t$ to be low-rank.

**Lemma 2.1.** (Cai et al., 2010) $\text{prox}_{\lambda \cdot} (Z) = U(S - \lambda I) + V^\top$, where $U \Sigma V^\top$ is the SVD of $Z$, and $[(X)_+]_{ij} = \max(0, X_{ij})$.

2.1.1. Matrix Completion

A special class of low-rank matrix learning problems is matrix completion, which attempts to find a low-rank matrix that agrees with the observations in data $O$:

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \|P_O(X - O)\|_F^2 + \lambda \|X\|_*.$$  

Here, the positions of observed elements in $O$ are indicated by 1’s in the binary matrix $\Omega$. $P_\Omega(X)_{ij} = X_{ij}$ if $\Omega_{ij} = 1$ and 0 otherwise. Setting $f(X) = \frac{1}{2} \|P_O(X - O)\|_F^2$ in (1), $Z_t$ in (2) becomes:

$$Z_t = X_t - \frac{1}{\tau} P_\Omega(X_t - O).$$

This has a “sparse plus low-rank” structure, with a low-rank component $X_t$ and a sparse component $\frac{1}{\tau} P_\Omega(X_t - O)$. This allows the SVD computation in Lemma 2.1 to be much more efficient (Mazumder et al., 2010). Specifically, on using the power method to compute $Z_t$’s SVD, most effort is spent on multiplications of the form $Z_t b$ and $a^\top Z_t$ (where $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$). Let $X_t$ in (5) be low-rank factorized as $U_t V_t^\top$ with rank $k_t$. Computing

$$Z_t b = U_t (V_t^\top b) - \frac{1}{\tau} P_\Omega(Y_t - O)b$$

takes $O((m + n)k_t + \|\Omega\|_1)$ time. Usually, $k_t \ll n$ and $\|\Omega\|_1 \ll mn$. This is much faster than direct multiplying $Z_t$ and $b$, which takes $O(mn)$ time. The same holds for $a^\top Z_t$. Thus, the proximal step in (3) takes $O((m + n)k_t{k_t+1} + \|\Omega\|_1 {k_t+1})$ time, while a direct computation without utilizing the “sparse plus low-rank” structure takes $O(mn{k_t+1})$ time. Besides, as only $P_\Omega(X_t)$ and the factorized form of $X_t$ need to be kept, the space complexity is reduced from $O(mn)$ to $O((m + n)k_t + \|\Omega\|_1)$.

2.1.2. Nonconvex Low-Rank Regularizer

Instead of using a convex $r$ in (1), the following nonconvex regularizer is commonly used (Gui et al., 2016; Lu et al., 2018a) and statistical guarantee (Gui et al., 2016).
the nuclear norm, the tensor completion problem becomes
\[
\phi(X) = \sum_{i=1}^{n} \kappa(\sigma_i(X)),
\]
where \(\kappa\) is nonconvex and possibly nonsmooth. We assume the following on \(\kappa\).

**Assumption 1.** \(\kappa(\alpha)\) is a concave, nondecreasing and \(L\)-Lipschitz continuous function on \(\alpha \geq 0\) with \(\kappa(0) = 0\).

Examples of \(\kappa\) include the capped-\(\ell_1\) penalty: \(\kappa(\sigma) = \log(\frac{\sigma}{\beta} + 1)\) (Zhang, 2010b), and log-sum-penalty (LSP): \(\kappa(\sigma) = \min(\sigma, \theta)\) (where \(\theta > 0\) is a constant) (Candès et al., 2008). More can be found in Appendix A. They have similar statistical guarantees (Gui et al., 2016), and empirically perform better than the convex nuclear norm (Lu et al., 2016; Yao et al., 2018a).

The proximal algorithm can again be used, and converges to a critical point (Attouch et al., 2013). Analogous to Lemma 2.1, the underlying proximal step
\[
\text{prox}_{\tau \phi}(Z) = \arg \min_{X} \frac{1}{2} \|X - Z\|^2_F + \frac{\lambda}{\tau} \phi(X)
\]
can be obtained as follows.

**Lemma 2.2.** (Lu et al., 2016) \(\text{prox}_{\lambda \phi}(Z) = U \text{Diag}(y_1, \ldots, y_n) V^T\), where \(U \Sigma V^T\) is the SVD of \(Z\), and \(y_i = \arg \min_{y \geq 0} \frac{1}{2}(y - \sigma_i)(Z)^2 + \lambda \kappa(y)\).

### 2.2. Low-Rank Tensor Learning

Recently, the nuclear norm regularizer has been extended to tensors. Here, we focus on the overlapped nuclear norm (Tomioka et al., 2010; Gandy et al., 2011), and its nonconvex extension that will be introduced in Section 3.

**Definition 1.** For a \(M\)-order tensor \(X\), the overlapped nuclear norm is \(\|X\|_{\text{overlap}} = \sum_{m=1}^{M} \lambda_m \|X_{(m)}\|_s\), where \(\{\lambda_m \geq 0\}\) are hyperparameters.

Factorization methods, such as the Tucker/CP (Kolda & Bader, 2009) and tensor-train decompositions (Osledechts, 2011), have also been used for low-rank tensor learning. Compared to nuclear norm regularization, they may lead to worse approximations and inferior performance (Tomioka et al., 2011; Liu et al., 2013; Guo et al., 2017).

### 3. Nonconvex Low-Rank Tensor Completion

By integrating a nonconvex \(\phi\) in (1) with the overlapped nuclear norm, the tensor completion problem becomes
\[
\min_{X} F(X) \equiv \frac{1}{2} \|P_{I}(X - \Theta)\|^2_F + \sum_{d=1}^{D} \frac{\lambda_d}{D} \phi(X_{(d)}).
\]
Note that we only sum over \(D \leq M\) modes. This is useful as the third mode is sometimes already small (e.g., the number of bands in images), and so does not need to be low-rank regularized. When \(D = 1\), (9) reduces to the matrix completion problem \(\min_{X \in \mathbb{R}^{I_1 \times I_2}} \frac{1}{2} \|P_{I_1}(X - \Theta_{(1)})\|^2_F + \lambda_1 \phi(X)\), which can be solved by the proximal algorithm as in (Lu et al., 2016; Yao et al., 2018a). In the sequel, we only consider \(D \neq 1\).

When \(\kappa(\alpha) = |\alpha|\), (9) reduces to (convex) overlapped nuclear norm regularization. While \(D\) may not be equal to \(M\), it can be easily shown that optimization solvers such as alternating direction of multiple multipliers (ADMM) and fast low-rank tensor completion (FaLRTC) can still be used as in (Boyd et al., 2011; Liu et al., 2013). However, when \(\kappa\) is nonconvex, ADMM no longer guarantees convergence, and FaLRTC’s dual cannot be derived.

#### 3.1. Structure-aware Proximal Iterations

As in Section 2.1, solve (9) with the proximal algorithm. However, the proximal step for \(\sum_{d=1}^{D} \lambda_d \phi(X_{(d)})\) is not simple. To address this problem, we use the proximal average (PA) algorithm (Bauschke et al., 2008; Yu, 2013). Consider the following problem with \(K\) possibly nonsmooth regularizers, whose individual proximal steps are assumed to be easily computable.

\[
\min_{X \in \mathcal{H}} \frac{f}{2} + \sum_{i=1}^{K} \frac{1}{K} g_i(X),
\]
where \(f\) is convex and Lipschitz-smooth, while each \(g_i\) is convex but possibly nonsmooth. The PA algorithm generates the iterates \(X_t\)’s as
\[
X_t = \frac{1}{K} \sum_{i=1}^{K} Y_t^i,
\]
\[
Z_t = X_t - \frac{1}{\tau} \nabla f(X_t),
\]
\[
Y_{t+1}^i = \text{prox}_{\lambda_i g_i}(Z_t), \quad i = 1, \ldots, K.
\]
Recently, the PA algorithm is also extended to nonconvex \(f\) and \(g_i\)’s, where each \(g_i\) admits a difference-of-convex decomposition\(^1\) (Zhong & Kwok, 2014).

Note that (9) is of the form in (10), and \(\phi\) in (7) admits a difference-of-convex decomposition (Yao et al., 2018a), the PA algorithm can be used to generate the iterates as:
\[
X_t = \frac{1}{D} \sum_{i=1}^{D} Y_t^i,
\]
\[
Z_t = X_t - \frac{1}{\tau} \frac{P_D}{P_I} (X_t - \Theta).
\]
However, as the regularizer \(\phi\) is imposed on the unfolded matrix \(X_{(d)}\), not on \(X\) directly, the proximal steps need to be performed as follows.

\(^1\)In other words, each \(g_i\) can be decomposed as \(g_i = \tilde{g}_i - \hat{g}_i\), where \(\tilde{g}_i\) and \(\hat{g}_i\) are two convex functions.
Proposition 3.1. For problem (9), step (13) of the PA algorithm can be performed as
\[ y_{t+1}^{i} = \left[ \text{prox}_{\frac{1}{2}d} (Z_t^{i}) \right]^{(i)}. \]  
(16)

The individual proximal steps in (16) can be computed using Lemma 2.2 based on SVD. However, tensor folding and unfolding are required in (16). A direct implementation takes \( O(I_t) \) space and \( O(I_t I_x) \) time per iteration, where \( I_x = \prod_{i=1}^{p} d_i \) and \( I_+ = \sum_{i=1}^{p} d_i \), and is expensive. In the following, we show how the PA iterations can be computed efficiently by utilizing the “sparse plus low-rank” structures.

3.1.1. Keeping the Low-Rank Factorizations

In (16), let \( Y_{t+1}^{i} = \text{prox}_{\frac{1}{2}d}(Z_{t}^{i}) \), where \( Z_t^{i} = [Z_t]^{(i)} \).

Recall that \( Y_t^{i} \) is low-rank. Let its rank be \( k_t^{i} \). In each iteration, we avoid constructing the dense \( Y_t^{i} \) by storing \( Y_t^{i} \) as \( U_t^{i}(V_t^{i})^{\top} \), where \( U_t^{i} \in \mathbb{R}^{I_t \times k_t^{i}} \) and \( V_t^{i} \in \mathbb{R}^{I_x I_t / k_t^{i} \times k_t^{i}} \). We also avoid getting \( X_t \) in (14) by storing it implicitly as
\[ X_t = \frac{1}{D} \sum_{i=1}^{D} (U_t^{i}(V_t^{i})^{\top})^{(i)}. \]  
(17)

3.1.2. Maintaining “Sparse plus Low-Rank”

Using (17), \( Z_t \) in (15) can be rewritten as
\[ Z_t = \frac{1}{D} \sum_{i=1}^{D} (U_t^{i}(V_t^{i})^{\top})^{(i)} - \frac{1}{\tau} P_{\Omega} (X_t - \Omega). \]  
(18)

The sparse tensor \( P_{\Omega} (X_t - \Omega) \) can be computed efficiently by using the coordinate format \(^2\) (Bader & Kolda, 2007). As \( \sum_{j=1}^{D} (U_t^{i}(V_t^{i})^{\top})^{(j)} \) is a sum of tensor (folded from low-rank matrices) and \( P_{\Omega} (X_t - \Omega) \) is sparse, \( Z_t \) is also “sparse plus low-rank”.

Recall that the proximal step in (16) requires SVD, which involves matrix multiplications of the form \( [Z_t]^{(i)} b \) (where \( b \in \mathbb{R}^{I_x I_t} \)) and \( a^{\top} [Z_t]^{(i)} \) (where \( a \in \mathbb{R}^{I_t} \)). Using the “sparse plus low-rank” structure in (18),
\[ [Z_t]^{(i)} b = \frac{1}{D} U_t^{i}(V_t^{i})^{\top} b + \frac{1}{D} \sum_{j \neq i}^{D} [(U_t^{j}(V_t^{j})^{\top})^{(j)} b] - \frac{1}{\tau} P_{\Omega} (X_t - \Omega) b, \]  
(19)
\[ a^{\top} [Z_t]^{(i)} = \frac{1}{D} (a^{\top} U_t^{i}) (V_t^{i})^{\top} + \frac{1}{D} \sum_{j \neq i}^{D} a^{\top} [(U_t^{j}(V_t^{j})^{\top})^{(j)}] - \frac{1}{\tau} a^{\top} P_{\Omega} (X_t - \Omega) b. \]  
(20)

\(^2\)For a sparse third-order tensor, its \( p \)th nonzero element is represented in the coordinate format as \( (i_1^{p}, i_2^{p}, i_3^{p}, v_p) \), where \( i_1^{p}, i_2^{p}, i_3^{p} \) are indices on each mode and \( v_p \) is the value. Using (17), \( v_p \) of \( P_{\Omega} (X_t - \Omega) \) can be computed by finding the corresponding rows in \( U_t^{i} \) and \( V_t^{i} \), which takes \( O(D k_t) \) time.

The first term in both (19) and (20) can be easily computed in \( O((I_x / I_t + I_+) k_t) \) space and time. \( [P_{\Omega} (X_t - \Omega)] b \) and \( a^{\top} [P_{\Omega} (X_t - \Omega)] b \) are sparse. Using sparse tensor packages such as the Tensor Toolbox (Bader & Kolda, 2007), \( [P_{\Omega} (X_t - \Omega)] b \) and \( a^{\top} [P_{\Omega} (X_t - \Omega)] b \) can be computed in \( O(||\Omega||) \) space and time.

Computing \( a^{\top} [(U_t^{j}(V_t^{j})^{\top})^{(j)}] b \) in (19), (20) involves folding/unfolding and is expensive. By examining how elements are ordered by folding and unfolding, the following shows that \( a^{\top} [(U_t^{j}(V_t^{j})^{\top})^{(j)}] b \) can be reformulated without explicit folding/unfolding, and thus be computed more efficiently.

Proposition 3.2. Let \( U \in \mathbb{R}^{I_t \times k} \), \( V \in \mathbb{R}^{I_x / I_t \times k} \), and \( u_p \) (resp. \( v_p \)) be the \( p \)th column of \( U \) (resp. \( V \)). For any \( a \in \mathbb{R}^{I_t} \) and \( b \in \mathbb{R}^{I_x / I_t} \), we have
\[ (i) \ a^{\top} [(U V)^{\top}] b = \sum_{p=1}^{k} u_p^{T} \otimes [a^{\top} \text{mat}(v_p)]; \]
\[ (ii) \ [(U V)^{\top}] b = \sum_{p=1}^{k} \text{mat}(v_p) \text{mat}(b)^{T} u_p, \]
where \( \otimes \) is the Kronecker product, and \( \text{mat}(c) \) reshapes vector \( c \in \mathbb{R}^{I_x / I_t} \) into a \( I_t \times I_x \) matrix.

Remark 3.1. As a special case, take \( i = 1 \) and \( I_j = 1 \) where \( j \in \{2,3\} \), the \( I_1 \times I_2 \times I_3 \) tensor reduces to a matrix. Proposition 3.2 then becomes \( a^{\top} [(U V)^{\top}] b = \sum_{p=1}^{k} u_p^{T} (a^{\top} v_p) = (a^{\top} U) V^{\top}, [(U V)^{\top}] b = \sum_{p=1}^{k} v_p^{T} (b^{T} u_p) = U (V^{\top} b) \), and (19) reduces to (6).

Computation of \( a^{\top} [(U_t^{j}(V_t^{j})^{\top})^{(j)}] b \) takes a total of \( O((I_x / I_t + I_+) k_t) \) time and \( O((I_x / I_t + I_+) I_x) \) space. The same holds for computation of \( [(U_t^{j}(V_t^{j})^{\top})^{(j)}] b \). This is much less expensive than direct evaluation, which takes \( O(k_t I_x) \) time and \( O(I_x) \) space.

Combining the above, and noting that we have to keep the factorized form \( U_t^{j}(V_t^{j})^{\top} \) of \( Y_t^{i} \), computing proximal steps in (16) takes \( O(D \sum_{i=1}^{D} \sum_{j \neq i}^{D} (I_t / I_+ + I_x / I_+) k_t I_x + ||\Omega||) \) space and \( O(D \sum_{i=1}^{D} \sum_{j \neq i}^{D} (I_t / I_+ + I_x / I_+) k_t I_x + ||\Omega||) \) time for each iteration. As \( k_t, k_{t+1} \ll I_x \), these are much lower than those of a direct implementation (Table 1).

3.1.3. Complexities

In each PA iteration, \( D \) proximal steps are performed in (16). The whole PA algorithm thus takes a total of \( O(D \sum_{i=1}^{D} \sum_{j \neq i}^{D} (I_t / I_+ + I_x / I_+) k_t I_x + ||\Omega||) \) time and \( O(D \sum_{i=1}^{D} \sum_{j \neq i}^{D} (I_t / I_+ + I_x / I_+) k_t I_x + ||\Omega||) \) time for each iteration. As \( k_t, k_{t+1} \ll I_x \), these are much lower than those of a direct implementation (Zhong & Kwok, 2014).
3.2. Use of Adaptive Momentum

The PA algorithm uses only first-order information, and empirically can be slow to converge (Parikh & Boyd, 2013). To address this problem, we adopt adaptive momentum, which has been popularly used for stochastic gradient descent (Duchi et al., 2011; Kingma & Ba, 2015) and proximal algorithms (Li & Lin, 2015; Yao et al., 2017; Li et al., 2017). The idea is to use historical iterates to speed up convergence. Here, we adopt the adaptive scheme in (Li et al., 2017). The resultant algorithm, shown in Algorithm 1, will be called NOncvx Regularized Tensor (NORT). Note that even when step 6 is performed, \( \mathbb{Z}_t \) in step 10 still has the “sparse plus low-rank” structure on \( \mathbb{Z}_t \), since \( \mathbb{Z}_t = [\frac{1}{D} \sum_{i=1}^{D} (U_i (V_i^{(t)})^T)^{(i)} - \frac{\gamma_t}{D} \sum_{i=1}^{D} (U_i V_i^{(t)}) - \frac{\gamma_t}{D} (U_i - \mathbb{O})]. \) The rank of each \( X_i^{(t)} \) is implicitly determined by the proximal step with the nonconvex regularizer at step 12. The resultant time and space complexities are the same as those in Section 3.1.3.

**Algorithm 1** NOncvx Regularized Tensor (NORT).

1: initialize \( \mathbb{X}_0 = \mathbb{X}_i = 0, \tau > \rho + DL \) and \( \gamma_t, \rho \in (0, 1) \);
2: for \( t = 1, \ldots, T \) do
3: \( \mathbb{X}_t^{(i)} = [\frac{1}{D} \sum_{i=1}^{D} (U_i (V_i^{(t)})^T)^{(i)}]; \)
4: \( \mathbb{X}_t = \mathbb{X}_t + \gamma_t (\mathbb{X}_t - \mathbb{X}_t^{(i)}); \)
5: if \( F_t (\mathbb{X}_t) \leq F_t (\mathbb{X}_t^{(i)}) \) then
6: \( \mathbb{V}_t = \mathbb{X}_t, \gamma_{t+1} = \min (\frac{\gamma_t}{\rho}, 1); \)
7: else
8: \( \mathbb{V}_t = \mathbb{X}_t, \gamma_{t+1} = \gamma_t; \)
9: end if
10: \( \mathbb{Z}_t = \mathbb{V}_t - \frac{\gamma_t}{D} (U_i - \mathbb{O}); \)
// compute \( \frac{\gamma_t}{D} (U_i - \mathbb{O}) \) using sparse tensor format;
11: for \( i = 1, \ldots, D \) do
12: \( X_i^{(t+1)} = \text{prox}_{\lambda_i} (\mathbb{Z}_t); \) // keep as \( U_i (V_i^{(t)})^T; \)
13: end for
14: end for
output \( \mathbb{X}_{T+1}. \)

3.3. Convergence Analysis

Adaptive momentum has not been used with the PA algorithm. Besides, previous proofs of the PA algorithm do not involve folding/unfolding operations. Thus, previous proofs cannot be directly used. In the following, first note that the proximal step in (16) implicitly corresponds to a new regularizer.

**Proposition 3.3.** There exists a function \( \bar{g} \) such that \( \text{prox}_{\frac{1}{\tau} \bar{g}} (Z_t) = \frac{1}{D} \sum_{i=1}^{D} \text{prox}_{\lambda_i} (Z_t^{(i)}) \) for any \( \tau > 0. \) Let the objective with the new regularizer be \( F_t (X) = f(X) + \bar{g}(X) \). The following bounds the difference between the optimal values \( F_{\text{min}} \) and \( F_{\tau_{\text{min}}} \), respectively of the objectives \( F \) in (9) and \( F_{\tau}. \)

**Proposition 3.4.** \( 0 \leq F_{\text{min}} - F_{\tau_{\text{min}}} \leq \frac{E^2}{\tau^2 D} \sum_{d=1}^{D} \lambda_d^2. \)

3.3.1. With Smooth Assumption

As in Section 2.1, we assume that \( f \) is \( \ell \)-Lipschitz smooth. The following shows that Algorithm 1 converges to a critical point (Theorem 3.5) at the rate of \( O(1/T) \) (Corollary 3.6). Note that this is the best possible rate for first-order methods on general nonconvex problems (Nesterov, 2013; Ghadimi & Lan, 2016).

**Theorem 3.5.** The sequence \( \{X_t\} \) generated from Algorithm 1 has at least one limit point, and all limit points are critical points of \( F_{\tau}(X). \)

**Corollary 3.6.** (i) If \( X_{t+1} = V_t \), then \( X_{t+1} \) is a critical point of \( F_{\tau} ; \) (ii) Let \( \eta = \tau - \rho - DL \). Then \( \min_{t=1, \ldots, T} \frac{1}{2} \|X_t - V_t\|^2_{\ell} \leq \frac{1}{\eta^2} \|F_t(X_t) - F_{\tau_{\text{min}}}^T\|_{\ell} \).

**Remark 3.2.** A larger \( \tau \) leads to a better approximation to the original problem \( F \) (Proposition 3.4). However, it also leads to smaller steps (step 12 in Algorithm 1) and thus slower convergence (Corollary 3.6).

3.3.2. With Kurdyka-Lojasiewicz Condition

The Kurdyka-Lojasiewicz (KL) condition (Attouch et al., 2013; Bolte et al., 2014) has been popularly used in nonconvex optimization, particularly in gradient (Attouch et al., 2013) and proximal gradient descent algorithms (Bolte et al., 2014; Li & Lin, 2015; Li et al., 2017).

**Definition 2.** A function \( h : \mathbb{R}^n \to (-\infty, \infty] \) has the uniformized KL property if for every compact set \( S \subset \text{dom}(h) \) on which \( h \) is a constant, there exist \( c, \epsilon > 0 \) such that for all \( u \in S \) and all \( \bar{u} \in \{ u : \min_{v \in S} \|u - v\|_2 \leq \epsilon \} \cap \{ u : f(\bar{u}) < f(u) < f(\bar{u}) + c, \} \) has \( \psi'(f(u) - f(\bar{u})) \min_{v \in \text{S}} \|v\|_F > 1, \) where \( \psi(\alpha) = \frac{C}{\alpha^\beta} \) for some \( C > 0, \alpha \in [0, c] \) and \( \beta \in (0, 1]. \)

The following extends this to Algorithm 1.

**Theorem 3.7.** Let \( r_t = F_{\tau}(X_t) - F_{\text{min}} \). If \( F_{\tau} \) has the uniformized KL property, for a sufficiently large \( t \), we have

a) If \( \beta = 1, r_t \) reduces to zero in finite steps;

b) If \( \beta \in [\frac{1}{2}, 1) \), \( r_t \leq \left( \frac{d_2 C}{D} \right)^{t-\tau_0} r_0 \) where \( d_2 = \frac{2(\tau_0)^2}{\eta}; \)

c) If \( \beta \in (0, \frac{1}{2}) \), \( r_t \leq \left( \frac{d_2 C}{(1-\tau_0)^2(1-2\beta)} \right)^{1/(1-2\beta)} \) where \( d_2 = \min\{ \frac{2 \tau_0 C}{1-\beta}, \frac{C}{1-\beta^2} (2 - \tau_0) - r_0 \}. \)
4. Experiments

In this section, experiments are performed on both synthetic (Section 4.1) and real-world data sets (Section 4.2).

4.1. Synthetic Data

The setup is as in (Song et al., 2017). We first generate \( \mathcal{O} = \sum_{i=1}^{n} s_i (a_i \circ b_i \circ c_i) \), where \( a_i \in \mathbb{R}^{l_1} \), \( b_i \in \mathbb{R}^{l_2} \) and \( c_i \in \mathbb{R}^{l_3} \), and \( \circ \) denotes the outer product (i.e., \( [a \circ b \circ c]_{ijk} = a_ib_jc_k \)). All elements in \( a_i \)'s, \( b_i \)'s, \( c_i \)'s and \( s_i \)'s are sampled independently from the standard normal distribution. This is then corrupted by Gaussian noise from \( \mathcal{N}(0, 0.01) \) to form \( \mathcal{O} \). A total of \( \| \Omega \|_1 = I_3I_3 \log(I_2)/5 \) random elements are observed from \( \mathcal{O} \). We use 50% of them for training, and the remaining 50% for validation. Testing is evaluated on the unobserved elements in \( \mathcal{O} \).

Three nonconvex penalties as used: capped-\( \ell_1 \) (Zhang, 2010a), LSP (Candès et al., 2008) and TNN (Hu et al., 2013). The proposed NORT algorithm is compared with (i) its slower variant without adaptive momentum (denoted sNORT); (ii) GDPAN (Zhong & Kwok, 2014), which directly applies PA algorithm to (10) as described in (14)- (16); and (iii) PA-APG (Yu, 2013), which solves the convex overlapped nuclear norm minimization problem. For NORT, \( \tau \) has to be larger than \( \rho + DL \) (Corollary 3.6). However, a large \( \tau \) leads to slow convergence (Remark 3.2). Hence, we set \( \tau = 1.01(\rho + DL) \). Moreover, we set \( \gamma_1 = 0.1 \) and \( p = 0.5 \) as in (Li et al., 2017). Besides, \( F_r \) in step 5 of Algorithm 1 is hard to evaluate, and we use \( F \) instead as in (Zhong & Kwok, 2014). All algorithms are implemented in Matlab, with sparse tensor and matrix operations in C. Experiments are performed on a PC with Intel-i8 CPU and 32GB memory.

Following (Lu et al., 2016; Yao et al., 2017; 2018a), performance is evaluated by (i) root-mean-square-error on the unobserved elements: \( \text{RMSE} = \| P_\Omega(\mathcal{X} - \hat{\mathcal{O}}) \|_F / \| \Omega \|_1^{0.5} \), where \( \hat{\mathcal{X}} \) is the low-rank tensor recovered, and \( \hat{\Omega} \) contains the unobserved elements in \( \mathcal{O} \); and (ii) CPU time. To reduce statistical variation, results are averaged over five repetitions.

4.1.1. Small \( I_3 (D = 2) \)

Recall that for the third-order tensor considered here, we assume that \( I_1 \geq I_2 \geq I_3 \). In this experiment, we first study the case where \( I_3 \) is small. We set \( I_1 = I_2 = 25\hat{c} \), where \( \hat{c} = 100 \), \( I_3 = 5 \), and \( D = 2 \).

Table 2 shows the results. As can be seen, PA-APG, which is based on the convex overlapped nuclear norm, has much higher testing RMSEs than those with nonconvex regularization. Besides, the various nonconvex penalties (capped-\( \ell_1 \), LSP and TNN) have similar empirical testing RMSEs, as is also observed in (Lu et al., 2016; Yao et al., 2017; 2018a). As for space, NORT and its variant sNORT need much less memory than PA-APG and GDPAN, as they do not explicitly construct dense tensors during iterations. As for time, NORT is the fastest. Figure 1 shows convergence of the objective. As can be seen, sNORT and GDPAN have similar speeds w.r.t. the number of iterations. However, sNORT is much faster when measured against time, as it utilizes the “sparse plus low-rank” structure. NORT is even faster due to usage of adaptive momentum.

4.1.2. Large \( I_3 (D = 3) \)

In this experiment, we set \( I_1 = I_2 = I_3 = 10\hat{c} \), where \( \hat{c} = 40 \); and \( D = 3 \) is used here. Results are shown in Table 2. Again, the capped-\( \ell_1 \), LSP and TNN regularizers yield the same RMSE. GDPAN, sNORT and NORT all have much lower RMSEs than PA-APG. Convergence of the objective value is shown in Figure 2. Again, NORT is the fastest, and GDPAN is the slowest. NORT and sNORT need much less memory than GDPAN.

![Figure 1: Convergence of the objective on the synthetic data for small \( I_3 \) (capped-\( \ell_1 \) regularizer).](image)

![Figure 2: Convergence of the objective on synthetic data for large \( I_3 \) (capped-\( \ell_1 \) regularizer).](image)
4.1.3. $D = 2$ vs $D = 3$

Recall that $D$ in (9) can be either 2 and 3. We expect $D = 2$ to be better when $I_3$ is small, and vice versa. This will be verified in this section. The setup is the same as in Sections 4.1.1 and 4.1.2. We only experiment with NORT, as the other baselines are less efficient.

Results are shown in Table 3. As can be seen, when $I_3$ is small, $D = 2$ and $D = 3$ yield similar RMSEs. However, $D = 3$ is much slower. When $I_3$ is small, the third mode is not low-rank. Thus, the proximal step for the third mode is much more expensive than those for the first two modes as we cannot have $k_3^2 < I_3$. Moreover, $D = 3$ requires much larger space than $D = 2$. When $I_3$ is large, it is slightly more expensive on CPU time and space. However, $D = 2$ has much worse testing RMSE than $D = 3$, as it cannot capture the low-rank property on the third mode.

Table 3. NORT with different $D$'s in (9) on the synthetic data.

<table>
<thead>
<tr>
<th>$D$</th>
<th>small $I_3$: $c = 100$, sparsity: 3.09%</th>
<th>large $I_3$: $c = 40$, sparsity: 2.70%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE (MB)</td>
<td>space (MB)</td>
</tr>
<tr>
<td>capped-$I_1$</td>
<td>0.0013</td>
<td>14.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0013</td>
<td>78.7</td>
</tr>
<tr>
<td>3</td>
<td>0.0013</td>
<td>14.1</td>
</tr>
<tr>
<td>0.0013</td>
<td>78.7</td>
<td>899.7</td>
</tr>
<tr>
<td>LSP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0014</td>
<td>14.4</td>
</tr>
<tr>
<td>3</td>
<td>0.0014</td>
<td>77.8</td>
</tr>
<tr>
<td>TNN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2. Real-World Data sets

As different nonconvex regularizers have similar performance, we will only use LSP. Moreover, based on the observations in Section 4.1.3, we use $D = 2$ if $I_3$ is ≤ 10, and $D = 3$ otherwise. Besides comparing with GDPAN, the proposed NORT algorithm is also compared with: (i) algorithms for various convex regularizers including ADMM (Boyd et al., 2011), FaLRTC (Liu et al., 2013), PA-APG (Yu, 2013), FFW (Guo et al., 2017), TenNN (Zhang & Aeron, 2017), and TR-MM (Nimishakavi et al., 2018); (ii) factorization-based algorithms including RP (Kasai & Mishra, 2016), TMac (Xu et al., 2013), CP-WOPT (Acar et al., 2011), and TMac-TT (Bengu et al., 2017).

We do not compare with sNORT as it is slower than NORT. Neither do we compare with (Bahadori et al., 2014), which is inferior to FFW above (Guo et al., 2017); and (Rauhut et al., 2017), whose its code is not publicly available and the more recent TMac-TT solves the same problem.

4.2.1. Color Images

We use windows, tree and rice from (Hu et al., 2013), which are resized to $1000 \times 1000 \times 3$ (Appendix C.2). Each pixel is normalized to $[0, 1]$. We randomly sample 10% of the pixels for training, which are then corrupted by Gaussian noise $\mathcal{N}(0, 0.01)$. Half of the training pixels are used for validation. The remaining unseen clean pixels are used for testing. Hyper-parameters of the various methods are tuned using the validation set. Performance is measured by the testing RMSE and CPU time. To reduce statistical variation, results are averaged over five repetitions.

Table 4. Testing RMSEs ($\times 10^{-4}$) on color images.

<table>
<thead>
<tr>
<th>factorization</th>
<th>convex</th>
<th>GDPAN</th>
<th>NORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMM</td>
<td>0.680±0.003</td>
<td>0.915±0.005</td>
<td>0.709±0.004</td>
</tr>
<tr>
<td>PA-APG</td>
<td>0.583±0.016</td>
<td>0.488±0.007</td>
<td>0.585±0.002</td>
</tr>
<tr>
<td>FaLRTC</td>
<td>0.576±0.004</td>
<td>0.494±0.011</td>
<td>0.567±0.005</td>
</tr>
<tr>
<td>FFW</td>
<td>0.634±0.003</td>
<td>0.599±0.005</td>
<td>0.772±0.004</td>
</tr>
<tr>
<td>TR-MM</td>
<td>0.596±0.005</td>
<td>0.513±0.011</td>
<td>0.634±0.002</td>
</tr>
<tr>
<td>TenNN</td>
<td>0.647±0.004</td>
<td>0.562±0.004</td>
<td>0.586±0.003</td>
</tr>
<tr>
<td>-convex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMac</td>
<td>1.923±0.005</td>
<td>1.750±0.006</td>
<td>1.313±0.005</td>
</tr>
<tr>
<td>CP-OPT</td>
<td>0.912±0.086</td>
<td>0.733±0.060</td>
<td>0.964±0.102</td>
</tr>
<tr>
<td>TMac-TT</td>
<td>0.729±0.022</td>
<td>0.697±0.147</td>
<td>1.045±0.107</td>
</tr>
</tbody>
</table>

Table 4 shows the results. As can be seen, the best convex methods (PA-APG and FaLRTC) are based on the overlapping nuclear norm. This agrees with our motivation to build a nonconvex regularizer based on the overlapping nuclear norm. GDPAN and NORT have similar RMSEs, which are lower than those by convex regularization and factorization approach. Convergence of the testing RMSE is shown in Figure 3. As can be seen, while ADMM solves...
the same convex model as PA-APG and FaLRTC, it has slower convergence. FFW, RP and TR-MM are very fast but their testing RMSEs are higher than that of NORT. By utilizing the “sparse plus low-rank” structure and adaptive momentum, NORT is more efficient than GDPAN.

4.2.2. REMOTE SENSING DATA

Experiments are performed on three hyper-spectral data sets (Appendix C.3): Cabbage (1312×432×49), Scene (1312×951×49) and Female (592×409×148). The third dimension is for the bands of images. We use the same setup as in Section 4.2.1, and hyper-parameters are tuned with the validation set. ADMM, TenNN, GDPAN, and TMac-TT are slow and so not compared.

Results are shown in Table 5. Again, NORT achieves much lower testing RMSE than convex regularization and factorization approach. Figure 4 shows convergence of the testing RMSE. As can be seen, NORT is fast.

4.2.3. SOCIAL NETWORKS

In this section, we perform multi-relational link prediction (Guo et al., 2017) as a tensor completion problem on the YouTube data set (Lei et al., 2009). It contains 15,088 users, and describes five types of user interactions. Thus, it forms a 15088×15088×5 tensor, with a total of 27,257,790 nonzero elements. Besides the full set, we also experiment with a YouTube subset obtained by randomly selecting 1,000 users (leading to 12,101 observations). We use 50% of the observations for training, another 25% for validation and the rest for testing. Experiments are repeated five times. Table 6 shows the testing RMSE, and Figure 5 shows the convergence. As can be seen, NORT achieves low RMSE and is also much faster.

5. Conclusion

In this paper, we propose a low-rank tensor completion model with nonconvex regularization. An efficient non-convex proximal average algorithm is developed, which maintains the “sparse plus low-rank” structure throughout the iterations and also incorporates adaptive momentum. Convergence to critical points is guaranteed. Experimental results show that the proposed algorithm is more efficient and more accurate than existing approaches. As a future work, it will be interesting to utilize automated machine learning (AutoML) (Yao et al., 2018b) for adaptive tuning hyper-parameters of the proposed approach.

6. Acknowledgment

The first author would like to thank Weiwei Tu and Xiawei Guo from 4Paradigm Inc for their valuable feedbacks.
References


Hillar, C. and Lim, L.-H. Most tensor problems are NP-hard. *JACM*, 60(6), 2013.


