Efficient Nonconvex Regularized Tensor Completion with Structure-aware Proximal Iterations

Quanming Yao [1 2] James T. Kwok [2] Bo Han [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).

[1]4Paradigm Inc, Beijing, China [2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong [3]Center for Advanced Intelligence Project, RIKEN, Japan. Correspondence to: Quanming Yao <yaoquanming@4paradigm.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

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-ℓ_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 et al., 2018a) and statistical guarantee (Gui et al., 2016).

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 $\mathbf{A} \in \mathbb{R}^{m \times n}$ (assume that $m \geq n$) with singular values σ_i's, its nuclear norm is $\|\mathbf{A}\|_* = \sum_i \sigma_i$. For tensors, we follow the notation in (Kolda & Bader, 2009). For a third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ (without loss of generality, we assume $I_1 \geq I_2 \geq I_3$), its (i_1, i_2, i_3)th entry is $\mathcal{X}_{i_1 i_2 i_3}$. One can *unfold* $\mathcal{X}$ along its dth mode to obtain the matrix $\mathbf{X}_{\langle d \rangle} \in \mathbb{R}^{I_d \times (I_\times / I_d)}$, whose (i_d, j) entry (where $j = 1 + \sum_{l=1, l \neq d}^{3} (i_l - 1) \prod_{m=1, m \neq d}^{l-1} I_m$) is $\mathcal{X}_{i_1 i_2 i_3}$. One can also *fold* a matrix $\mathbf{X}$ back to a tensor $\mathcal{X} = \mathbf{X}^{\langle d \rangle}$, with $\mathcal{X}_{i_1 i_2 i_3} = \mathbf{X}_{i_d j}$, and j as defined above. The inner product of two third-order tensors $\mathcal{X}$ and $\mathcal{Y}$ is $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} \mathcal{X}_{i_1 i_2 i_3} \mathcal{Y}_{i_1 i_2 i_3}$. The Frobenius norm of $\mathcal{X}$ is $\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$. For a proper and lower-semi-continuous function f, ∂f denotes its Frechet subdifferential (Attouch et al., 2013). If f is ρ-Lipschitz smooth, then $\|\nabla f(\mathbf{X}) - \nabla f(\mathbf{Y})\|_F^2 \leq \rho \|\mathbf{X} - \mathbf{Y}\|_F^2$ for any $\mathbf{X}$ and $\mathbf{Y}$.

2. Related Works

2.1. Low-Rank Matrix Learning

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

$$\min_{\mathbf{X}} f(\mathbf{X}) + \lambda r(\mathbf{X}), \tag{1}$$

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 ρ-Lipschitz smooth loss. Using the proximal algorithm (Parikh & Boyd, 2013), the iterate is given by $\mathbf{X}_{t+1} = \text{prox}_{\frac{\lambda}{\tau}\|\cdot\|_*}(\mathbf{Z}_t)$, where

$$\mathbf{Z}_t = \mathbf{X}_t - \frac{1}{\tau} \nabla f(\mathbf{X}_t), \tag{2}$$

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

$$\text{prox}_{\frac{\lambda}{\tau}\|\cdot\|_*}(\mathbf{Z}) \equiv \arg\min_{\mathbf{X}} \frac{1}{2} \|\mathbf{X} - \mathbf{Z}\|_F^2 + \frac{\lambda}{\tau} \|\mathbf{X}\|_* \tag{3}$$

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

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

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 $\mathbf{O}$:

$$\min_{\mathbf{X} \in \mathbb{R}^{m \times n}} \frac{1}{2} \|P_\Omega(\mathbf{X} - \mathbf{O})\|_F^2 + \lambda \|\mathbf{X}\|_*. \tag{4}$$

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

$$\mathbf{Z}_t = \mathbf{X}_t - \frac{1}{\tau} P_\Omega(\mathbf{X}_t - \mathbf{O}). \tag{5}$$

This has a "sparse plus low-rank" structure, with a low-rank component $\mathbf{X}_t$ and a sparse component $\frac{1}{\tau} P_\Omega(\mathbf{X}_t - \mathbf{O})$. 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 $\mathbf{Z}_t$'s SVD, most effort is spent on multiplications of the form $\mathbf{Z}_t \mathbf{b}$ and $\mathbf{a}^\top \mathbf{Z}_t$ (where $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$). Let $\mathbf{X}_t$ in (5) be low-rank factorized as $\mathbf{U}_t \mathbf{V}_t^\top$ with rank k_t. Computing

$$\mathbf{Z}_t \mathbf{b} = \mathbf{U}_t (\mathbf{V}_t^\top \mathbf{b}) - \frac{1}{\tau} P_\Omega(\mathbf{Y}_t - \mathbf{O}) \mathbf{b} \tag{6}$$

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 $\mathbf{Z}_t$ and $\mathbf{b}$, which takes $O(mn)$ time. The same holds for $\mathbf{a}^\top \mathbf{Z}_t$. 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 $O(mnk_{t+1})$ time. Besides, as only $P_\Omega(\mathbf{X}_t)$ and the factorized form of $\mathbf{X}_t$ 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., 2016; Gu et al., 2017; Yao et al., 2018a):

$$\phi(\mathbf{X}) = \sum_{i=1}^{n} \kappa(\sigma_i(\mathbf{X})), \tag{7}$$

where κ is nonconvex and possibly nonsmooth. We assume the following on κ.

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

Examples of κ include the capped-ℓ_1 penalty: $\kappa(\sigma) = \log(\frac{1}{\theta}\sigma + 1)$ (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

$$\text{prox}_{\frac{\lambda}{\tau}\phi}(\mathbf{Z}) \equiv \arg\min_{\mathbf{X}} \frac{1}{2} \|\mathbf{X} - \mathbf{Z}\|_F^2 + \frac{\lambda}{\tau}\phi(\mathbf{X}) \tag{8}$$

can be obtained as follows.

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

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 $\mathcal{X}$, the* overlapped nuclear norm *is $\|\mathcal{X}\|_{\textit{overlap}} = \sum_{m=1}^{M} \lambda_m \|\mathcal{X}_{\langle m \rangle}\|_*$, where $\{\lambda_m \geq 0\}$ are hyperparameters.*

Factorization methods, such as the Tucker/CP (Kolda & Bader, 2009) and tensor-train decompositions (Oseledets, 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 ϕ in (1) with the overlapped nuclear norm, the tensor completion problem becomes

$$\min_{\mathcal{X}} F(\mathcal{X}) \equiv \frac{1}{2} \|P_\Omega(\mathcal{X} - \mathcal{O})\|_F^2 + \sum_{d=1}^{D} \frac{\lambda_d}{D} \phi(\mathcal{X}_{\langle d \rangle}). \tag{9}$$

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 $\min_{\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 I_3}} \frac{1}{2}\|P_\Omega(\mathbf{X} - \mathcal{O}_{\langle 1 \rangle})\|_F^2 + \lambda_1 \phi(\mathbf{X})$, 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 κ 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, we solve (9) with the proximal algorithm. However, the proximal step for $\sum_{d=1}^{D} \lambda_i \phi(\mathcal{X}_{\langle d \rangle})$ is not simple. To address this problem, we use the proximal average (PA) algorithm (Bauschke et al., 2008; Yu, 2013).

Let $\mathcal{H}$ be a Hilbert space. Consider the following problem with K possibly nonsmooth regularizers, whose individual proximal steps are assumed to be easily computable.

$$\min_{\mathcal{X} \in \mathcal{H}} f(\mathcal{X}) + \sum_{i=1}^{K} \frac{\lambda_i}{K} g_i(\mathcal{X}), \tag{10}$$

where f is convex and Lipschitz-smooth, while each g_i is convex but possibly nonsmooth. The PA algorithm generates the iterates $\mathcal{X}_t$'s as

$$\mathcal{X}_t = \frac{1}{K} \sum_{i=1}^{K} \mathcal{Y}_t^i, \tag{11}$$

$$\mathcal{Z}_t = \mathcal{X}_t - \frac{1}{\tau} \nabla f(\mathcal{X}_t), \tag{12}$$

$$\mathcal{Y}_{t+1}^i = \text{prox}_{\frac{\lambda_i}{\tau} g_i}(\mathcal{Z}_t), \quad i = 1, \dots, K. \tag{13}$$

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 ϕ in (7) admits a difference-of-convex decomposition (Yao et al., 2018a), the PA algorithm can be used to generate the iterates as:

$$\mathcal{X}_t = \frac{1}{D} \sum_{i=1}^{D} \mathcal{Y}_t^i, \tag{14}$$

$$\mathcal{Z}_t = \mathcal{X}_t - \frac{1}{\tau} P_\Omega\left(\mathcal{X}_t - \mathcal{O}\right). \tag{15}$$

However, as the regularizer ϕ is imposed on the unfolded matrix $\mathcal{X}_{\langle d \rangle}$, not on $\mathcal{X}$ directly, the proximal steps need to be performed as follows.

[1]In other words, each g_i can be decomposed as $g_i = \bar{g}_i - \hat{g}_i$ where $\bar{g}_i$ and $\hat{g}_i$ are two convex functions.

Proposition 3.1. *For problem* (9)*, step* (13) *of the PA algorithm can be performed as*

$$\mathcal{Y}^i_{t+1} = \left[\text{prox}_{\frac{\lambda_i}{\tau}\phi}([\mathcal{Z}_t]_{\langle i\rangle})\right]^{\langle i\rangle}. \quad (16)$$

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 $O(I_\times)$ space and $O(I_\times I_+)$ time per iteration, where $I_\times = \prod_{i=d}^3 I_d$ and $I_+ = \sum_{i=d}^3 I_d$, 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 $\mathbf{Y}^i_{t+1} = \text{prox}_{\frac{\lambda_i}{\tau}\phi}(\mathbf{Z}^i_t)$, where $\mathbf{Z}^i_t = [\mathcal{Z}_t]_{\langle i\rangle}$. Recall that $\mathbf{Y}^i_t$ is low-rank. Let its rank be k^i_t. In each iteration, we avoid constructing the dense $\mathcal{Y}^i_t$ by storing $\mathbf{Y}^i_t$ as $\mathbf{U}^i_t(\mathbf{V}^i_t)^\top$, where $\mathbf{U}^i_t \in \mathbb{R}^{I_i\times k^i_t}$ and $\mathbf{V}^i_t \in \mathbb{R}^{(I_\times/I_i)\times k^i_t}$. We also avoid getting $\mathcal{X}_t$ in (14) by storing it implicitly as

$$\mathcal{X}_t = \frac{1}{D}\sum_{i=1}^{D}\left(\mathbf{U}^i_t(\mathbf{V}^i_t)^\top\right)^{\langle i\rangle}. \quad (17)$$

3.1.2. Maintaining "Sparse plus Low-Rank"

Using (17), $\mathcal{Z}_t$ in (15) can be rewritten as

$$\mathcal{Z}_t = \frac{1}{D}\sum_{i=1}^{D}(\mathbf{U}^i_t(\mathbf{V}^i_t)^\top)^{\langle i\rangle} - \frac{1}{\tau}P_\Omega(\mathcal{X}_t - \mathcal{O}). \quad (18)$$

The sparse tensor $P_\Omega(\mathcal{X}_t - \mathcal{O})$ can be constructed efficiently by using the coordinate format[2] (Bader & Kolda, 2007). As $\sum_{i=1}^D(\mathbf{U}^i_t(\mathbf{V}^i_t)^\top)^{\langle i\rangle}$ is a sum of tensor (folded from low-rank matrices) and $\frac{1}{\tau}P_\Omega(\mathcal{X}_t - \mathcal{O})$ is sparse, $\mathcal{Z}_t$ is also "sparse plus low-rank".

Recall that the proximal step in (16) requires SVD, which involves matrix multiplications of the form $(\mathcal{Z}_t)_{\langle i\rangle}\mathbf{b}$ (where $\mathbf{b}\in\mathbb{R}^{I_\times/I_i}$) and $\mathbf{a}^\top(\mathcal{Z}_t)_{\langle i\rangle}$ (where $\mathbf{a}\in\mathbb{R}^{I_i}$). Using the "sparse plus low-rank" structure in (18),

$$(\mathcal{Z}_t)_{\langle i\rangle}\mathbf{b} = \frac{1}{D}\mathbf{U}^i_t[(\mathbf{V}^i_t)^\top\mathbf{b}] + \frac{1}{D}\sum_{j\neq i}[(\mathbf{U}^j_t(\mathbf{V}^j_t)^\top)^{\langle j\rangle}]_{\langle i\rangle}\mathbf{b} - \frac{1}{\tau}[P_\Omega(\mathcal{X}_t - \mathcal{O})]_{\langle i\rangle}\mathbf{b}, \quad (19)$$

$$\mathbf{a}^\top(\mathcal{Z}_t)_{\langle i\rangle} = \frac{1}{D}(\mathbf{a}^\top\mathbf{U}^i_t)(\mathbf{V}^i_t)^\top + \frac{1}{D}\sum_{j\neq i}\mathbf{a}^\top[(\mathbf{U}^j_t(\mathbf{V}^j_t)^\top)^{\langle j\rangle}]_{\langle i\rangle} - \frac{1}{\tau}\mathbf{a}^\top[P_\Omega(\mathcal{X}_t - \mathcal{O})]_{\langle i\rangle}. \quad (20)$$

The first term in both (19) and (20) can be easily computed in $O((I_\times/I_i + I_i)k^i_t)$ space and time. $[P_\Omega(\mathcal{X}_t - \mathcal{O})]_{\langle i\rangle}\mathbf{b}$ and $\mathbf{a}^\top[P_\Omega(\mathcal{X}_t - \mathcal{O})]_{\langle i\rangle}$ are sparse. Using sparse tensor packages such as the Tensor Toolbox (Bader & Kolda, 2007), $[P_\Omega(\mathcal{X}_t - \mathcal{O})]_{\langle i\rangle}\mathbf{b}$ and $\mathbf{a}^\top[P_\Omega(\mathcal{X}_t - \mathcal{O})]_{\langle i\rangle}$ can be computed in $O(\|\Omega\|_1)$ space and time.

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

Proposition 3.2. *Let* $\mathbf{U}\in\mathbb{R}^{I_i\times k}$*,* $\mathbf{V}\in\mathbb{R}^{I_\times/I_i\times k}$*, and* $\mathbf{u}_p$ *(resp.*$\mathbf{v}_p$*) be the pth column of* $\mathbf{U}$ *(resp.*$\mathbf{V}$*). For any* $\mathbf{a}\in\mathbb{R}^{I_j}$ *and* $\mathbf{b}\in\mathbb{R}^{I_\times/I_j}$*, we have*

(i) $\mathbf{a}^\top[(\mathbf{U}\mathbf{V}^\top)^{\langle i\rangle}]_{\langle j\rangle} = \sum_{p=1}^k \mathbf{u}_p^\top \otimes [\mathbf{a}^\top\text{mat}(\mathbf{v}_p)]$*;*

(ii) $[(\mathbf{U}\mathbf{V}^\top)^{\langle i\rangle}]_{\langle j\rangle}\mathbf{b} = \sum_{p=1}^k \text{mat}(\mathbf{v}_p)\text{mat}(\mathbf{b})^\top\mathbf{u}_p$*;*

where $\otimes$ *is the Kronecker product, and* mat($\mathbf{c}$) *reshapes vector* $\mathbf{c}\in\mathbb{R}^{I_iI_j}$ *into a* $I_i\times I_j$ *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* $\mathbf{a}^\top[(\mathbf{U}\mathbf{V}^\top)^{\langle 1\rangle}]_{\langle j\rangle} = \sum_{p=1}^k\mathbf{u}_p^\top(\mathbf{a}^\top\mathbf{v}_p) = (\mathbf{a}^\top\mathbf{U})\mathbf{V}^\top$, $[(\mathbf{U}\mathbf{V}^\top)^{\langle 1\rangle}]_{\langle j\rangle}\mathbf{b} = \sum_{p=1}^k\mathbf{v}_p(\mathbf{b}^\top\mathbf{u}_p) = \mathbf{U}(\mathbf{V}^\top\mathbf{b})$*, and* (19) *reduces to* (6)*.*

Computation of $\mathbf{a}^\top[(\mathbf{U}^i_t(\mathbf{V}^i_t)^\top)^{\langle j\rangle}]_{\langle i\rangle}$ takes a total of $O((\frac{1}{I_i}+\frac{1}{I_j})k^i_tI_\times)$ time and $O((\frac{1}{I_i}+\frac{1}{I_j})I_\times)$ space. The same holds for computation of $[(\mathbf{U}^i_t(\mathbf{V}^i_t)^\top)^{\langle j\rangle}]_{\langle i\rangle}\mathbf{b}$. This is much less expensive than direct evaluation, which takes $O(k^i_tI_\times)$ time and $O(I_\times)$ space.

Combining the above, and noting that we have to keep the factorized form $\mathbf{U}^i_t(\mathbf{V}^i_t)^\top$ of $\mathbf{Y}^i_t$, computing proximal steps in (16) takes $O(\sum_{i=1}^D\sum_{j\neq i}(\frac{1}{I_i}+\frac{1}{I_j})k^i_tI_\times + \|\Omega\|_1)$ space and $O(\sum_{i=1}^D\sum_{j\neq i}(\frac{1}{I_i}+\frac{1}{I_j})k^i_tk^i_{t+1}I_\times + \|\Omega\|_1(k^i_t + k^i_{t+1}))$ time.

3.1.3. Complexities

In each PA iteration, D proximal steps are performed in (16). The whole PA algorithm thus takes a total of $O(\sum_{i=1}^D\sum_{j\neq i}(\frac{1}{I_i}+\frac{1}{I_j})k^i_tI_\times + \|\Omega\|_1)$ space and $O(\sum_{i=1}^D\sum_{j\neq i}(\frac{1}{I_i}+\frac{1}{I_j})k^i_tk^i_{t+1}I_\times + \|\Omega\|_1(k^i_t + k^i_{t+1}))$ time for each iteration. As k^i_t, $k^i_{t+1} \ll I_i$, these are much lower than those of a direct implementation (Table 1). Moreover, the PA algorithm has a convergence rate of $O(1/T)$, where T is the number of iterations (Zhong & Kwok, 2014).

[2]For a sparse third-order tensor, its pth 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 $P_\Omega(\mathcal{X}_t - \mathcal{O})$ can be computed by finding the corresponding rows in $\mathbf{U}^i_t$ and $\mathbf{V}^i_t$, which takes $O(\sum_{i=1}^D k^i_t)$ time.

Table 1. Comparison of the proposed NORT (Algorithm 1) and direct implementations of the PA algorithm.

	per-iteration time complexity	space	convergence
direct	$O(I_\times \sum_{i=1}^{D} I_i)$	$O(I_\times)$	slow
NORT	$O(\sum_{i=1}^{D} \sum_{j\neq i}(\frac{1}{I_i}+\frac{1}{I_j})k_t^i k_{t+1}^i I_\times + \|\Omega\|_1 (k_t^i + k_{t+1}^i))$	$O(\sum_{i=1}^{D} \sum_{j\neq i}(\frac{1}{I_i}+\frac{1}{I_j})k_t^i I_\times + \|\Omega\|_1)$	fast

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 procedure, shown in Algorithm 1, will be called NOncvx Regularized Tensor (NORT). Note that even when step 6 is performed, $\mathcal{Z}_t$ in step 10 still has the "sparse plus low-rank" structure on $\mathcal{Z}_t$, since $\mathcal{Z}_t = \frac{1+\gamma_t}{D}\sum_{i=1}^{D}(\mathbf{U}_t^i(\mathbf{V}_t^i)^\top)^{\langle i\rangle} - \frac{\gamma_t}{D}\sum_{i=1}^{D}(\mathbf{U}_{t-1}^i(\mathbf{V}_{t-1}^i)^\top)^{\langle i\rangle} - \frac{1}{\tau}P_\Omega\left(\bar{\mathcal{X}}_t - \mathcal{O}\right)$. The rank of each $\mathbf{X}_{t+1}^i$ 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 NOnconvex Regularized Tensor (NORT).

```
1: initialize $\mathcal{X}_0 = \mathcal{X}_1 = 0$, $\tau > \rho + DL$ and $\gamma_1, p \in (0,1)$;
2: for $t = 1, \dots, T$ do
3:     $\mathcal{X}_{t+1} = \frac{1}{D}\sum_{i=1}^{D}(\mathbf{U}_{t+1}^i(\mathbf{V}_{t+1}^i)^\top)^{\langle i\rangle}$;
4:     $\bar{\mathcal{X}}_t = \mathcal{X}_t + \gamma_t(\mathcal{X}_t - \mathcal{X}_{t-1})$;
5:     if $F_\tau(\bar{\mathcal{X}}_t) \le F_\tau(\mathcal{X}_t)$ then
6:         $\mathcal{V}_t = \bar{\mathcal{X}}_t$, $\gamma_{t+1} = \min(\frac{\gamma_t}{p}, 1)$;
7:     else
8:         $\mathcal{V}_t = \mathcal{X}_t$, $\gamma_{t+1} = p\gamma_t$;
9:     end if
10:    $\mathcal{Z}_t = \mathcal{V}_t - \frac{1}{\tau}P_\Omega(\mathcal{V}_t - \mathcal{O})$;
       // compute $P_\Omega(\mathcal{V}_t - \mathcal{O})$ using sparse tensor format;
11:    for $i = 1, \dots, D$ do
12:        $\mathbf{X}_{t+1}^i = \text{prox}_{\frac{\lambda_i}{\tau}\phi}((\mathcal{Z}_t)_{\langle i\rangle})$; // keep as $\mathbf{U}_t^i(\mathbf{V}_t^i)^\top$;
13:    end for
14: end for
output $\mathcal{X}_{T+1}$.
```

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* $\text{prox}_{\frac{1}{\tau}\bar{g}}(\mathcal{Z}) = \frac{1}{D}\sum_{i=1}^{D}[\text{prox}_{\frac{\lambda_i}{\tau}\phi}([\mathcal{Z}]_{\langle i\rangle})]^{\langle i\rangle}$ *for any* $\tau > 0$.

Let the objective with the new regularizer be $F_\tau(\mathcal{X}) = f(\mathcal{X}) + \bar{g}(\mathcal{X})$. The following bounds the difference between the optimal values ($F^{\min}$ and $F_\tau^{\min}$, respectively) of the objectives F in (9) and F_τ.

Proposition 3.4. $0 \le F^{\min} - F_\tau^{\min} \le \frac{L^2}{2\tau D}\sum_{d=1}^{D}\lambda_d^2$.

3.3.1. With Smooth Assumption

As in Section 2.1, we assume that f is L-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 $\{\mathcal{X}_t\}$ generated from Algorithm 1 has at least one limit point, and all limits points are critical points of $F_\tau(\mathcal{X})$.*

Corollary 3.6. *(i) If $\mathcal{X}_{t+1} = \mathcal{V}_t$, then $\mathcal{X}_{t+1}$ is a critical point of F_τ; (ii) Let $\eta = \tau - \rho - DL$. Then* $\min_{t=1,\dots,T}\frac{1}{2}\|\mathcal{X}_{t+1} - \mathcal{V}_t\|_F^2 \le \frac{1}{\eta T}[F_\tau(\mathcal{X}_1) - F_\tau^{\min}]$;

Remark 3.2. *A larger τ 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 $\mathcal{S} \in$ dom(h) on which h is a constant, there exist ϵ, $c > 0$ such that for all $\mathbf{u} \in \mathcal{S}$ and all $\bar{\mathbf{u}} \in \{\mathbf{u} : \min_{\mathbf{v}\in\mathcal{S}}\|\mathbf{u}-\mathbf{v}\|_2 \le \epsilon\} \cap \{\mathbf{u} : f(\bar{\mathbf{u}}) < f(\mathbf{u}) < f(\bar{\mathbf{u}}) + c\}$, one has $\psi'(f(\mathbf{u}) - f(\bar{\mathbf{u}}))\min_{\mathbf{v}\in\partial f(\mathbf{u})}\|\mathbf{v}\|_F > 1$, where $\psi(\alpha) = \frac{C}{\beta}\alpha^\beta$ for some $C > 0$, $\alpha \in [0, c)$ and $\beta \in (0, 1]$.*

The following extends this to Algorithm 1.

Theorem 3.7. *Let $r_t = F_\tau(\mathcal{X}_t) - F_\tau^{\min}$. If F_τ has the uniformized KL property, for a sufficiently large t_0, we have*

a) If $\beta = 1$, r_t reduces to zero in finite steps;
b) If $\beta \in [\frac{1}{2}, 1)$, $r_t \le (\frac{d_1 C^2}{1+d_1 C^2})^{t-t_0} r_{t_0}$ where $d_1 = \frac{2(\tau+\rho)^2}{\eta}$;
c) If $\beta \in (0, \frac{1}{2})$, $r_t \le (\frac{C}{(t-t_0)d_2(1-2\beta)})^{1/(1-2\beta)}$ where $d_2 = \min\{\frac{1}{2d_1 C}, \frac{C}{1-2\beta}(2^{\frac{2\beta-1}{2\beta-2}} - 1)r_{t_0}\}$.

Though the convergence rates in Corollary 3.6 and Theorem 3.7 are the same as when momentum is not used, the proposed algorithm does have faster convergence empirically, as will be shown in Section 4. This also agrees with previous studies showing that adaptive momentum can significantly accelerate empirical convergence (Duchi et al., 2011; Kingma & Ba, 2015; Li & Lin, 2015; Li et al., 2017; Yao et al., 2017).

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 $\bar{\mathfrak{O}} = \sum_{i=1}^{5} s_i(\mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i)$, where $\mathbf{a}_i \in \mathbb{R}^{I_1}$, $\mathbf{b}_i \in \mathbb{R}^{I_2}$ and $\mathbf{c}_i \in \mathbb{R}^{I_3}$, and $\circ$ denotes the outer product (i.e., $[\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}]_{ijk} = a_i b_j c_k$). All elements in $\mathbf{a}_i$'s, $\mathbf{b}_i$'s, $\mathbf{c}_i$'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 $\mathfrak{O}$. A total of $\|\Omega\|_1 = I_+ I_3 \log(I_\times)/5$ random elements are observed from $\mathfrak{O}$. We use 50% of them for training, and the remaining 50% for validation. Testing is evaluated on the unobserved elements in $\bar{\mathfrak{O}}$.

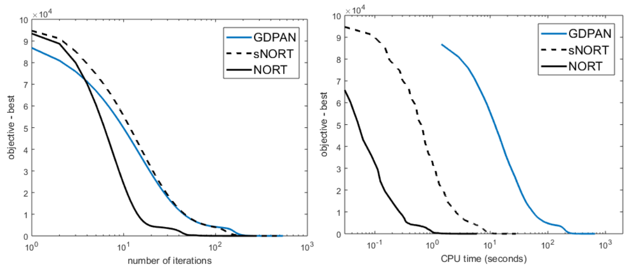
Three nonconvex penalties as used: capped-ℓ_1 (Zhang, 2010a), LSP (Candès et al., 2008) and TNN (Hu et al., 2013). The proposed NORT algorithm is compared with (i) its slower variant without adaptive momentum (denoted sNORT); (ii) GDPAN (Zhong & Kwok, 2014), which directly applies PA algorithm to (10) as described in (14)-(16); and (iii) PA-APG (Yu, 2013), which solves the convex overlapped nuclear norm minimization problem. For NORT, τ has to be larger than $\rho + DL$ (Corollary 3.6). However, a large τ 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_τ 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: RMSE $= \|P_{\bar{\Omega}}(\mathfrak{X} - \bar{\mathfrak{O}})\|_F / \|\bar{\Omega}\|_1^{0.5}$, where $\mathfrak{X}$ is the low-rank tensor recovered, and $\bar{\Omega}$ contains the unobserved elements in $\bar{\mathfrak{O}}$; 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 $I_1 = I_2 = 25\bar{c}$, where $\bar{c} = 100$, $I_3 = 5$, 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-ℓ_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.[3] 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.

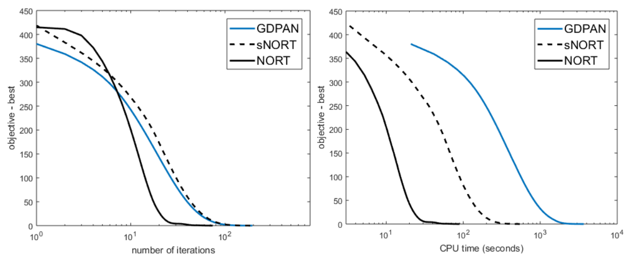


(a) vs iterations. (b) vs time.

Figure 1. Convergence of the objective on the synthetic data for small I_3 (capped-ℓ_1 regularizer).

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-ℓ_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.



(a) vs iterations. (b) vs time.

Figure 2. Convergence of the objective on synthetic data for large I_3 (capped-ℓ_1 regularizer).

[3]Plots for LSP and TNN are similar, and so are not shown because of the lack of space.

Table 2. Testing RMSE, CPU time and space required on the synthetic data set. The left is for small I_3, and the right is for large I_3. Results for $\bar{c} = 50$ (small I_3) and $\hat{c} = 20$ (large I_3) are in Appendix C. In all tables, the best and comparable performances (according to the pairwise t-test with 95% confidence) are highlighted.

		small I_3: $\bar{c} = 100$, sparsity: 3.09%			large I_3: $\hat{c} = 40$, sparsity:2.70%		
		RMSE	space (MB)	time (sec)	RMSE	space (MB)	time (sec)
convex	PA-APG	0.0149±0.0011	302.4±0.1	2131.7±419.9	0.0098±0.0001	4804.5±598.2	6196.4±2033.4
(nonconvex) capped-ℓ_1	GDPAN	**0.0103±0.0001**	171.5±2.2	665.4±99.8	**0.0006±0.0001**	3243.3±489.6	3670.4±225.8
	sNORT	**0.0103±0.0001**	**14.0±0.8**	27.9±5.1	**0.0006±0.0001**	**44.6±0.3**	575.9±70.9
	NORT	**0.0103±0.0001**	14.9±0.9	**5.9±1.6**	**0.0006±0.0001**	66.3±0.6	89.4±13.4
(nonconvex) LSP	GDPAN	0.0104±0.0001	172.2±1.5	654.1±214.7	**0.0006±0.0001**	3009.3±376.2	3794.0±419.5
	sNORT	0.0104±0.0001	14.4±0.1	27.9±5.7	**0.0006±0.0001**	**44.6±0.2**	544.2±75.5
	NORT	0.0104±0.0001	15.1±0.1	**5.8±2.8**	**0.0006±0.0001**	62.1±0.5	**81.3±24.9**
(nonconvex) TNN	GDPAN	0.0104±0.0001	172.1±1.6	615.0±140.9	**0.0006±0.0001**	3009.2±412.2	3922.9±280.1
	sNORT	0.0104±0.0001	14.4±0.1	26.2±4.0	**0.0006±0.0001**	**44.7±0.2**	554.7±44.1
	NORT	**0.0103±0.0001**	15.1±0.1	**5.3±1.5**	**0.0006±0.0001**	63.1±0.6	**78.0±9.4**

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 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 $k_t^3 \ll I_3$. 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.

		small I_3 ($\bar{c} = 100$)			large I_3 ($\hat{c} = 40$)		
	D	RMSE	space (MB)	time (sec)	RMSE	space (MB)	time (sec)
capped -ℓ_1	2	**0.0103**	**14.0**	**5.9**	0.0009	46.7	**40.0**
	3	**0.0103**	78.7	918.7	**0.0006**	**66.3**	89.4
LSP	2	**0.0104**	**14.1**	**5.8**	0.0010	45.2	50.8
	3	**0.0103**	78.7	899.7	**0.0006**	**62.1**	81.3
TNN	2	**0.0103**	**14.4**	**5.3**	0.0009	46.8	**39.3**
	3	**0.0104**	77.8	615.5	**0.0006**	**63.1**	78.0

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 (Bengua 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.

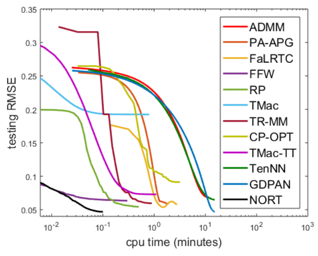
4.2.1. COLOR IMAGES

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 ($\times 10^{-1}$) on color images.

		rice	*tree*	*windows*
convex	ADMM	0.680±0.003	0.915±0.005	0.709±0.004
	PA-APG	0.583±0.016	0.488±0.007	0.585±0.002
	FaLRTC	0.576±0.004	0.494±0.011	0.567±0.005
	FFW	0.634±0.003	0.599±0.005	0.772±0.004
	TR-MM	0.596±0.005	0.515±0.011	0.634±0.002
	TenNN	0.647±0.004	0.562±0.004	0.586±0.003
factor-ization	RP	0.541±0.011	0.524±0.010	0.388±0.026
	TMac	1.923±0.005	1.750±0.006	1.313±0.005
	CP-OPT	0.912±0.086	0.733±0.060	0.964±0.102
	TMac-TT	0.729±0.022	0.697±0.147	1.045±0.107
non -convex	GDPAN	**0.467±0.002**	**0.388±0.012**	**0.296±0.007**
	NORT	**0.468±0.001**	**0.386±0.009**	**0.297±0.007**

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.


(a) *rice*.

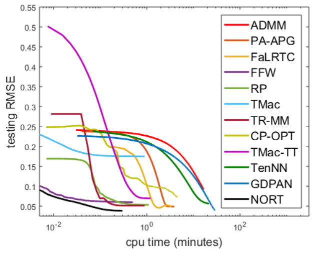

(b) *tree*.

Figure 3. Testing RMSE vs CPU time (seconds) on color images. The plot for *windows* is similar, and thus not shown.

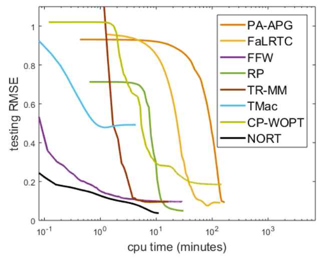
4.2.2. Remote Sensing Data

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.

Table 5. Testing RMSEs ($\times 10^{-2}$) on remote sensing data.

		Cabbage	*Scene*	*Female*
convex	PA-APG	9.13±0.06	19.65±0.02	11.57±0.03
	FaLRTC	9.09±0.02	19.20±0.01	11.33±0.04
	FFW	9.62±0.04	20.37±0.02	20.96±0.06
	TR-MM	9.59±0.01	19.65±0.02	13.97±0.06
factor-ization	RP	4.91±0.11	18.04±0.05	6.47±0.03
	TMac	49.19±0.59	59.70±0.29	198.97±0.06
	CP-OPT	18.46±5.14	48.11±0.82	18.68±0.13
noncvx	NORT	**3.76±0.04**	**17.14±0.12**	**5.92±0.02**


(a) *Cabbage*.

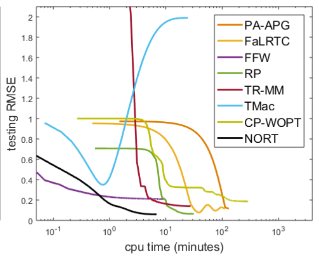

(b) *Female*.

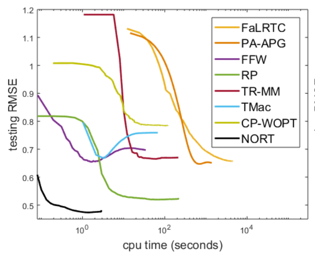
Figure 4. Testing RMSE vs CPU time (minutes) on remote sensing data. The plot for *Scene41* is similar, and thus not shown.

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.

Table 6. Testing RMSEs on *Youtube* data set. FaLRTC, PA-APG, TR-MM and CP-OPT are slow, and thus not run on the full set.

		subset	full set
convex	FaLRTC	0.657±0.060	—
	PA-APG	0.651±0.047	—
	FFW	0.697±0.054	0.395±0.001
	TR-MM	0.670±0.098	—
factorization	RP	0.522±0.038	0.410±0.001
	TMac	0.795±0.033	0.611±0.007
	CP-OPT	0.785±0.040	—
nonconvex	NORT	**0.482±0.030**	**0.370±0.001**


(a) Subset.

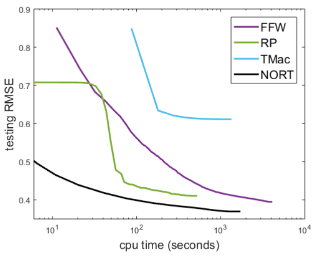

(b) Full set.

Figure 5. Testing RMSE vs CPU time (seconds) on *Youtube*.

5. Conclusion

In this paper, we propose a low-rank tensor completion model with nonconvex regularization. An efficient nonconvex 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

Acar, E., Dunlavy, D., Kolda, T., and Mørup, M. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

Attouch, H., Bolte, J., and Svaiter, B. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss-Seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

Bader, B. and Kolda, T. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, 2007.

Bahadori, M., Yu, Q., and Liu, Y. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *NeurIPS*, pp. 3491–3499, 2014.

Bauschke, H., Goebel, R., Lucet, Y., and Wang, X. The proximal average: basic theory. *JOPT*, 19(2):766–785, 2008.

Bengua, J., Phien, H., Tuan, H., and Do, M. Efficient tensor completion for color image and video recovery: Low-rank tensor train. *TIP*, 26(5):2466–2479, 2017.

Bolte, J., Sabach, S., and Teboulle, M. Proximal alternating linearized minimization or nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2009.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Cai, J.-F., Candès, E., and Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

Candès, E. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

Candès, E., Wakin, M., and Boyd, S. Enhancing sparsity by reweighted ℓ_1 minimization. *JFAA*, 14(5-6):877–905, 2008.

Cheng, H., Yu, Y., Zhang, X., Xing, E., and Schuurmans, D. Scalable and sound low-rank tensor learning. In *AISTAT*, pp. 1114–1123, 2016.

Cichocki, A., Mandic, D., De Lathauwer, L., Zhou, G., Zhao, Q., Caiafa, C., and Phan, H. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *SPM*, 32(2): 145–163, 2015.

Davis, D., Lichtenwalter, R., and Chawla, N. V. Multi-relational link prediction in heterogeneous information networks. In *ASONAM*, pp. 281–288. IEEE, 2011.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul):2121–2159, 2011.

Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *JASA*, 96 (456):1348–1360, 2001.

Gandy, S., Recht, B., and Yamada, I. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

Ghadimi, S. and Lan, G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *MathProg*, 156(1-2):59–99, 2016.

Gu, S., Xie, Q., Meng, D., Zuo, W., Feng, X., and Zhang, L. Weighted nuclear norm minimization and its applications to low level vision. *IJCV*, 121(2):183–208, 2017.

Gui, H., Han, J., and Gu, Q. Towards faster rates and oracle property for low-rank matrix estimation. In *ICML*, pp. 2300–2309, 2016.

Guo, X., Yao, Q., and Kwok, J. Efficient sparse low-rank tensor completion using the Frank-Wolfe algorithm. In *AAAI*, pp. 1948–1954, 2017.

Hare, W. and Sagastizábal, C. Computing proximal points of nonconvex functions. *Mathematical Programming*, 116(1-2):221–258, 2009.

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

Hu, Y., Zhang, D., Ye, J., Li, X., and He, X. Fast and accurate matrix completion via truncated nuclear norm regularization. *TPAMI*, 35(9):2117–2130, 2013.

Kasai, H. and Mishra, B. Low-rank tensor completion: a Riemannian manifold preconditioning approach. In *ICML*, pp. 1012–1021, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kolda, T. and Bader, B. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Lei, T., Wang, X., and Liu, H. Uncoverning groups via heterogeneous interaction analysis. In *ICDM*, pp. 503–512, 2009.

Li, H. and Lin, Z. Accelerated proximal gradient methods for nonconvex programming. In *NIPS*, pp. 379–387, 2015.

Li, Q., Zhou, Y., Liang, Y., and Varshney, P. Convergence analysis of proximal gradient with momentum for nonconvex optimization. In *ICML*, pp. 2111–2119, 2017.

Liu, J., Musialski, P., Wonka, P., and Ye, J. Tensor completion for estimating missing values in visual data. *TPAMI*, 35(1):208–220, 2013.

Lu, C., Tang, J., Yan, S., and Lin, Z. Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm. *TIP*, 25(2):829–839, 2016.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *JMLR*, 11:2287–2322, 2010.

Mu, C., Huang, B., Wright, J., and Goldfarb, D. Square deal: Lower bounds and improved relaxations for tensor recovery. In *ICML*, pp. 73–81, 2014.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013.

Nimishakavi, M., Jawanpuria, P., and Mishra, B. A dual framework for low-rank tensor completion. In *NeurIPS*, 2018.

Oseledets, I. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Parikh, N. and Boyd, S. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231, 2013.

Rauhut, H., Schneider, R., and Stojanac, Ž. Low rank tensor recovery via iterative hard thresholding. *Linear Algebra and its Applications*, 523:220–262, 2017.

Rendle, S. and Schmidt-Thieme, L. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pp. 81–90, 2010.

Signoretto, M., Van de Plas, R., De Moor, B., and Suykens, J. Tensor versus matrix completion: a comparison with application to spectral data. *SPL*, 18(7):403–406, 2011.

Song, Q., Ge, H., Caverlee, J., and Hu, X. Tensor completion algorithms in big data analytics. Technical report, Department of Computer Science and Engineering, Texas A&M University, 2017.

Tomioka, R. and Suzuki, T. Convex tensor decomposition via structured schatten norm regularization. In *NIPS*, pp. 1331–1339, 2013.

Tomioka, R., Hayashi, K., and Kashima, H. Estimation of low-rank tensors via convex optimization. *arXiv preprint*, 2010.

Tomioka, R., Suzuki, T., Hayashi, K., and Kashima, H. Statistical performance of convex tensor decomposition. In *NIPS*, pp. 972–980, 2011.

Vasilescu, A. and Terzopoulos, D. Multilinear analysis of image ensembles: Tensorfaces. In *ECCV*, pp. 447–460, 2002.

Xu, Y., Hao, R., Yin, W., and Su, Z. Parallel matrix factorization for low-rank tensor completion. *IPI*, 9(2): 601–624, 2013.

Yao, Q., Kwok, J., Gao, F., Chen, W., and Liu, T.-Y. Efficient inexact proximal gradient algorithm for nonconvex problems. In *IJCAI*, pp. 3308–3314, 2017.

Yao, Q., Kwok, J., Wang, T., and Liu, T.-Y. Large-scale low-rank matrix learning with nonconvex regularizers. *TPAMI*, 2018a.

Yao, Q., Wang, M., Chen, Y., Dai, W., Y., H., Li, Y., Tu, W., Yang, Q., and Yu, Y. Taking human out of learning applications: A survey on automated machine learning. Technical report, arXiv preprint, 2018b.

Yu, Y.-L. Better approximation and faster algorithm using the proximal average. In *NIPS*, pp. 458–466, 2013.

Zhang, C. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38(2): 894–942, 2010a.

Zhang, T. Analysis of multi-stage convex relaxation for sparse regularization. *JMLR*, 11:1081–1107, 2010b.

Zhang, Z. and Aeron, S. Exact tensor completion using t-SVD. *TSP*, 65(6):1511–1526, 2017.

Zhong, W. and Kwok, J. Gradient descent with proximal average for nonconvex and composite regularization. In *AAAI*, pp. 2206–2212, 2014.