Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Efficient Sparse Low-Rank Tensor Completion Using the Frank-Wolfe Algorithm

Xiawei Guo, Quanming Yao, James T. Kwok
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{xguoae, qyaoaa, jamesk}@cse.ust.hk

Abstract

Most tensor problems are NP-hard, and low-rank tensor completion is much more difficult than low-rank matrix completion. In this paper, we propose a time and space-efficient low-rank tensor completion algorithm by using the scaled latent nuclear norm for regularization and the Frank-Wolfe (FW) algorithm for optimization. We show that all the steps can be performed efficiently. In particular, FW's linear subproblem has a closed-form solution which can be obtained from rank-one SVD. By utilizing sparsity of the observed tensor, we only need to maintain sparse tensors and a set of small basis matrices. Experimental results show that the proposed algorithm is more accurate, much faster and more scalable than the state-of-the-art.

Introduction

Tensors have been commonly used to describe the linear and multilinear relationships in the data. For example, in remote sensing applications, a hyperspectral image with multiple bands can be naturally represented as a 3-dimensional tensor. A multidimensional social network can also be modeled as a 3-dimensional tensor, where the third mode may represent different type of relations. Higher-dimensional tensors are also useful. For example, a multi-mode social network (such as the DBLP network) with heterogeneous actors (papers, authors, terms and venues) can be represented by a 4-order tensor, and a relation can connect these four kinds of entities (Tang, Wang, and Liu 2009).

Analogous to matrix completion (Candès and Recht 2009), tensor completion attempts to recover a low-rank tensor that best approximates a partially observed data tensor. For example, in recommender systems, users can rate an item based on different criteria (e.g., story, visual effects, actors). By treating these attributes as another dimension, rating prediction becomes a tensor completion problem on a 3-dimensional tensor (Adomavicius, Manouselis, and Kwon 2011). Similarly, in hyperspectral imaging, as some bands may be partially missing due to sensor problems, tensor completion can be used to inpaint the incomplete image. However, while matrix completion has attracted a lot of interest, these matrix techniques cannot be readily adopted for tensors. Indeed, most tensor problems, even computing the tensor rank, is NP-hard (Hillar and Lim 2013).

To impose a low-rank structure on tensors, CP and Tucker decompositions assume that the tensor can be decomposed into low-rank factor matrices (Kolda and Bader 2009). This can then be learned by alternating least squares or coordinate descent (Acar et al. 2010). Recently, Kressner, Steinlechner, and Vandereycken (2014) proposed to utilize the Riemannian structure on the manifold of tensors with fixed multilinear rank, and then perform nonlinear conjugate gradient descent. It can be speeded up by preconditioning (Kasai and Mishra 2016). However, these models are non-convex, can suffer from the problem of local minimum, and has no theoretical guarantee on the convergence rate. Moreover, its per-iteration cost depends on the product of all the mode ranks, and so can be expensive.

Another popular approach is to unfold the tensor and apply low-rank matrix factorization techniques on all the resultant matricizations (Tomioka, Hayashi, and Kashima 2010; Xu et al. 2013). However, the low-rank constraint is not directly enforced on the tensor and can be misleading (Cheng et al. 2016). To alleviate this problem, one can use instead convex low-rank regularizers as in matrix completion. While the matrix nuclear norm is the tightest convex envelope of the matrix rank (Candès and Recht 2009), there are a number of norms that induce low-rank tensors. Common examples include the tensor trace norm (Chandrasekaran et al. 2012), overlapped nuclear norm (Liu et al. 2013; Tomioka, Hayashi, and Kashima 2010), latent nuclear norm (Tomioka, Hayashi, and Kashima 2010) and scaled latent nuclear norm (Wimalawarne, Sugiyama, and Tomioka 2014). By using these convex low-rank tensor regularizers, the resulting optimization problem can be solved by standard convex optimizers. For example, the FaLRTC algorithm (Liu et al. 2013) considers the overlapped nuclear norm regularizer, and uses Nesterov's smoothing (Nesterov 2005) and accelerated proximal algorithm (Beck and Teboulle 2009) for optimization. Though convergence can be guaranteed, these algorithms do not utilize sparsity of the observed tensor. In each iteration, they have to operate on the full-sized tensor. When the tensor is large, it may not even be able to fit into memory. Moreover, low-rank tensor regularization are more complicated and difficult to optimize. Typically, expensive multiple partial SVD operations on a large dense matrix are required.

On the other hand, the Frank-Wolfe (FW) algorithm has a simple update rule and good convergence guarantees. Recently, it has regained popularity in machine learning (Jaggi 2013). In particular, it has been successfully used for matrix completion with nuclear norm regularization (Zhang, Schuurmans, and Yu 2012). However, its usefulness on the more complicated tensor completion problem is less clear. Yang, Feng, and Suykens (2015), Cheng et al. (2016) recently applied the FW algorithm for tensor completion with the tensor trace norm. However, the linear subproblem in FW can only be solved approximately, and the resultant convergence result is weak.

In this paper, we show that the scaled latent nuclear norm, together with the FW algorithm, is a more appropriate combination for low-rank tensor completion. The FW linear subproblem then has a closed-form solution which can be obtained efficiently from rank-one SVD. Moreover, both the linear subproblem and line search only need to access the observed entries, and the sparsity structure of the observed tensor can be efficiently utilized. Besides, instead of explicitly handling the full tensors during iterations, we only need to store the sparse tensors and a set of basis matrices for recovering the solution tensor. The resultant algorithm is efficient in terms of both space and time, and converges to the optimal solution at a rate of $\mathcal{O}(1/T)$, where T is the number of iterations. Empirically, it is more accurate, much faster and more scalable than state-of-the-art tensor completion algorithms.

Notation: In the sequel, vectors are denoted by lowercase boldface, matrices by uppercase boldface, and tensors by boldface Euler. For a matrix $\mathbf{A}$ with singular values σ_i's, its nuclear norm is $\|\mathbf{A}\|_* = \sum_i \sigma_i$. We follow the tensor notations in (Kolda and Bader 2009). For a D-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$, its $(i_1, i_2, \dots, i_D)$th entry is $x_{i_1 i_2 \dots i_D}$. We use $[D]$ to denote the range $\{1, 2, \dots, D\}$, and $I_{D\backslash d} = \prod_{j=1, j\neq d}^{D} I_j$. The mode-$d$ matricizations $\mathcal{X}_{\langle d \rangle}$ of $\mathcal{X}$ is a $I_d \times I_{D\backslash d}$ matrix with $(\mathcal{X}_{\langle d \rangle})_{i_d j} = x_{i_1 i_2 \cdots i_D}$, and $j = 1 + \sum_{l=1, l\neq d}^{D} (i_l - 1) \prod_{m=1, m\neq d}^{l-1} I_m$. Given a matrix $\mathbf{A}$, its mode-d tensorization $\mathbf{A}^{\langle d \rangle}$ is a tensor $\mathcal{X}$ with elements $x_{i_1 i_2 \cdots i_D} = a_{i_d j}$, and j is as defined before. The inner product of two tensors $\mathcal{X}$ and $\mathcal{Y}$ is $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_D=1}^{I_D} x_{i_1 i_2 \dots i_D} y_{i_1 i_2 \dots i_D}$, and the Frobenius norm of $\mathcal{X}$ is $\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$.

Related Work

Low-Rank Tensor Completion

Given a partially observed D-order tensor $\mathcal{A}$, tensor completion attempts to recover a low-rank tensor $\mathcal{X}$ that best approximates $\mathcal{A}$ on the observed entries. Let the positions of the observed entries be indicated by Ω. Tensor completion can be formulated as the following optimization problem:

$$\min_{\mathcal{X}} F(\mathcal{X}) \equiv \frac{1}{2} \|\mathcal{P}_\Omega(\mathcal{X} - \mathcal{A})\|_F^2 \; : \; R(\mathcal{X}) \leq \tau, \quad (1)$$

where $P_\Omega(\mathcal{X})$ is a tensor with $[\mathcal{P}_\Omega(\mathcal{X})]_{i_1 i_2 \dots i_D} = x_{i_1 i_2 \dots i_D}$ if $(i_1, i_2, \dots, i_D) \in \Omega$, and 0 otherwise, $R(\mathcal{X})$ is a low-rank regularizer, and τ is a given parameter. In matrix completion, the nuclear norm is often used as a convex surrogate for the matrix rank. For tensors, several definitions of the norm exist. The most common ones are the overlapped nuclear norm (Liu et al. 2013) and the (scaled) latent nuclear norm (Wimalawarne, Sugiyama, and Tomioka 2014; Tomioka, Hayashi, and Kashima 2010).

Definition 1. *For a D-order tensor $\mathcal{X}$, the* overlapped nuclear norm *is $\|\mathcal{X}\|_{overlap} = \sum_{d=1}^{D} \|\mathcal{X}_{\langle d \rangle}\|_*$, and the* scaled latent nuclear norm *is $\|\mathcal{X}\|_{scaled} = \min_{\mathcal{X}_1, \dots, \mathcal{X}_D : \sum_{d=1}^{D} \mathcal{X}_d = \mathcal{X}} \sum_{d=1}^{D} \frac{1}{\sqrt{I_d}} \|(\mathcal{X}_d)_{\langle d \rangle}\|_*$. On dropping the weight $1/\sqrt{I_d}$, this reduces to the* latent nuclear norm.

The overlapped nuclear norm regularizer penalizes nuclear norms on all modes. On the other hand, the latent nuclear norm regularizer is more appropriate when the target tensor can be decomposed into a set of tensors, each of which is low-rank in a specific mode. When only several modes are low-rank, decomposition with the latent nuclear norm generalizes better than the overlapped nuclear norm (Tomioka and Suzuki 2013). The scaled latent nuclear norm also performs better than its unscaled version when the tensor dimensions or ranks are heterogeneous (Wimalawarne, Sugiyama, and Tomioka 2014). A comparison of their sample complexities can also be found in (Wimalawarne, Sugiyama, and Tomioka 2014).

Frank-Wolfe Algorithm

Recently, the Frank-Wolfe (FW) algorithm has been popularly used in machine learning (Jaggi 2013). It can be used for solving problems of the form: $\min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x})$, where f is convex and continuously differentiable, and $\mathcal{D}$ is convex and compact. In particular, Zhang, Schuurmans, and Yu (2012) applied FW on matrix completion with the nuclear norm regularizer. The linear subproblem (step 3 in Algorithm 1) then has a closed-form solution which can be efficiently obtained by rank-one SVD. Moreover, the FW algorithm converges at a rate of $\mathcal{O}(1/T)$, where T is the number of iterations (Jaggi 2013).

Algorithm 1 Frank-Wolfe (FW) algorithm.

```
1: Initialize $\mathbf{x}_0$;
2: for $t = 0, 1, \dots, T$ do
3:     $\mathbf{s}^{(t+1)} = \min_{\mathbf{s} \in \mathcal{D}} \langle \mathbf{s}, \nabla f(\mathbf{x}^{(t)}) \rangle$;
4:     $\gamma^{(t+1)} = \arg\min_{\gamma \in [0,1]} f(\mathbf{x}^{(t)} + \gamma(\mathbf{s}^{(t+1)} - \mathbf{x}^{(t)}))$;
5:     $\mathbf{x}^{(t+1)} = (1 - \gamma^{(t+1)})\mathbf{x}^{(t)} + \gamma^{(t+1)}\mathbf{s}^{(t+1)}$;
6: end for
output $\mathbf{x}_T$.
```

FW Algorithm for Tensor Completion

While the FW algorithm has been successfully used for matrix completion, it becomes more complicated for tensor completion. First, there are several tensor norms which are more sophisticated. Second, previous attempts cannot solve the FW linear subproblem and line search efficiently. For example, Yang, Feng, and Suykens (2015), Cheng et al. (2016) recently applied the FW algorithm to tensor learning with the tensor trace norm (TTN) regularizer, and the corresponding FW linear subproblem can only be solved approximately. Instead of having $\mathcal{S}^{(t+1)} = \arg\min_{\mathcal{S}:\|\mathcal{S}\|_{\text{TTN}}\le 1} \langle \mathcal{S}, \nabla F(\mathcal{X}^{(t)})\rangle$, where $\|\cdot\|_{\text{TTN}}$ denotes the TTN regularizer, they can only guarantee that $\langle \mathcal{S}^{(t+1)}, \nabla F(\mathcal{X}^{(t)})\rangle \le \alpha \min_{\mathcal{S}:\|\mathcal{S}\|_{\text{TTN}}\le 1} \langle \mathcal{S}, \nabla F(\mathcal{X}^{(t)})\rangle$ where $\alpha = \prod_{d=1}^{D-2} I_d^{-1/2} \in [0,1]$.

In the following, we will show that by using the scaled latent nuclear norm for low-rank tensor regularization, all the steps in Algorithm 1 can be performed efficiently.

Efficient FW Linear Subproblem

In the linear subproblem, $\mathcal{S}^{(t+1)} = \arg\min_{\mathcal{S}:\|\mathcal{S}\|_{\text{scaled}}\le\tau} \langle \mathcal{S}, \nabla F(\mathcal{X}^{(t)})\rangle$. The following Proposition shows that it can be obtained from rank-one SVDs of the d matricizations of $-\nabla F(\mathcal{X}^{(t)})$. Note that $\nabla F(\mathcal{X}^{(t)}) = \mathcal{P}_\Omega(\mathcal{X}^{(t)} - \mathcal{A})$ is sparse, and rank-one SVD can be computed efficiently by the power method (Halko, Martinsson, and Tropp 2011).

Proposition 1. $\mathcal{S}^{(t+1)} = (\tau\sqrt{I_{d^*}}\mathbf{u}_{d^*}\mathbf{v}_{d^*}^\top)^{\langle d^*\rangle}$, *where* $d^* = \arg\max_{d\in[D]} \sqrt{I_d}\sigma_{\max}(-\nabla F(\mathcal{X}^{(t)})_{\langle d\rangle})$, $\mathbf{u}_{d^*}, \mathbf{v}_{d^*}$ *are the leading left and right singular vectors of* $-\nabla F(\mathcal{X}^{(t)})_{\langle d^*\rangle}$, *and* $\sigma_{\max}(\cdot)$ *is the largest singular value.*

Many related low-rank tensor learning algorithms (Liu et al. 2013; Tomioka, Hayashi, and Kashima 2010) require performing rank-k SVD, where k is lower-bounded by the maximum mode rank of the solution. Typically, performing rank-k SVD is k times slower than rank-one SVD. In our experiments, k is at least 20.

Efficient Line Search

With F in (1), the line search (step 4 of Algorithm 1) has the following simple closed-form solution:

$$\begin{aligned}\gamma^{(t+1)} &= \arg\min_{\gamma\in[0,1]} \|\mathcal{P}_\Omega(\mathcal{X}^{(t)} + \gamma(\mathcal{S}^{(t+1)} - \mathcal{X}^{(t)}) - \mathcal{A})\|_F^2 \\ &= \begin{cases} 0 & -\frac{b}{2a} \in (-\infty, 0) \\ -\frac{b}{2a} & -\frac{b}{2a} \in [0,1] \\ 1 & -\frac{b}{2a} \in (1,\infty) \end{cases}, \end{aligned} \quad (2)$$

where $a = \|\mathcal{P}_\Omega(\mathcal{X}^{(t)} - \mathcal{S}^{(t+1)})\|_F^2$, and $b = 2\langle \mathcal{P}_\Omega(\mathcal{X}^{(t)} - \mathcal{A}), \mathcal{P}_\Omega(\mathcal{S}^{(t+1)} - \mathcal{X}^{(t)})\rangle$. Both a and b can be computed in $\mathcal{O}(\|\Omega\|_1)$ time.

Efficient Use of Sparse Structure

Note that the linear subproblem and line search only need to access the observed entries of $\mathcal{X}^{(t)}$ and $\mathcal{S}^{(t+1)}$. Hence, instead of storing the whole tensors during iterations, we only calculate and store their entries indexed by Ω (sometimes explicitly denoted as $\mathcal{X}^{(t)}|_\Omega$ and $\mathcal{S}^{(t+1)}|_\Omega$). For $\mathcal{S}^{(t+1)}|_\Omega$ in Proposition 1, this reduces the time complexity of computing $(\tau\sqrt{I_{d^*}}\mathbf{u}_{d^*}\mathbf{v}_{d^*}^\top)^{\langle d^*\rangle}$ from $\mathcal{O}(\prod_{d=1}^D I_d)$ to $\mathcal{O}(\|\Omega\|_1)$.

However, $\mathcal{S}^{(t+1)}$, and consequently the new iterate

$$\mathcal{X}^{(t+1)} = (1-\gamma^{(t+1)})\mathcal{X}^{(t)} + \gamma^{(t+1)}\mathcal{S}^{(t+1)} \quad (3)$$

in step 5 of Algorithm 1, may have nonzero entries outside Ω. Hence, we also maintain $\mathcal{X}^{(t)}$ as $\sum_{d=1}^D (\mathbf{U}_d\boldsymbol{\Sigma}_d\mathbf{V}_d{}^\top)^{\langle d\rangle}$. These matrices can be efficiently updated as follows. First, all $\mathbf{U}_d, \boldsymbol{\Sigma}_d, \mathbf{V}_d$'s are initialized to empty matrices. At the tth iteration, for $d \ne d^*$ in Proposition 1,

$$\boldsymbol{\Sigma}_d \leftarrow (1-\gamma^{(t+1)})\boldsymbol{\Sigma}_d, \;\; \mathbf{U}_d \leftarrow \mathbf{U}_d, \;\; \mathbf{V}_d \leftarrow \mathbf{V}_d; \quad (4)$$

and

$$\begin{aligned}\boldsymbol{\Sigma}_{d^*} &\leftarrow \begin{bmatrix} (1-\gamma^{(t+1)})\boldsymbol{\Sigma}_{d^*} & \mathbf{0} \\ \mathbf{0} & \gamma^{(t+1)}\tau\sqrt{I_{d^*}} \end{bmatrix}, \\ \mathbf{U}_{d^*} &\leftarrow [\mathbf{U}_{d^*} \;\; \mathbf{u}_{d^*}], \;\; \mathbf{V}_{d^*} \leftarrow [\mathbf{V}_{d^*} \;\; \mathbf{v}_{d^*}]. \end{aligned} \quad (5)$$

It is easy to see that this satisfies (3). Moreover, $\mathcal{X}^{(t)}$ needs to be explicitly computed only when the algorithm ends or after basis reduction (which will be discussed in the sequel).

Related low-rank tensor learning methods (Liu et al. 2013; Xu et al. 2013; Tomioka, Hayashi, and Kashima 2010) need to store the whole tensor, and so require at least $\mathcal{O}(\prod_{d=1}^D I_d)$ time and space. When the tensor is very large, it cannot even be fit into memory. In contrast, we only store the sparse tensors $\mathcal{S}^{(t+1)}|_\Omega, \mathcal{X}^{(t)}|_\Omega$ and the basis matrices $\{\mathbf{U}_d \in \mathbb{R}^{I_d\times k_d}, \boldsymbol{\Sigma}_d \in \mathbb{R}^{k_d\times k_d}, \mathbf{V}_d \in \mathbb{R}^{I_{D\setminus d}\times k_d}\}_{d\in[D]}$. It may appear that the difference between $\prod_{d=1}^D I_d$ and $I_{D\setminus d}$ $(= \prod_{j=1, j\ne d}^D I_j)$ is small. However, from Proposition 1, modes with small I_d's are unlikely to be selected as d^*. Hence, most of the non-empty matrices in $\{\mathbf{U}_d, \boldsymbol{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]}$ are for d's with large I_d's.

Recall that Yang, Feng, and Suykens (2015) used the tensor trace norm instead. Not only is the FW linear subproblem hard to solve, also sparsity can no longer be utilized. Specifically, their subroutine 2 extracts singular vectors from the input (matricized) tensor, folds it to a matrix, and then repeats. Even when the tensor is sparse, its singular vectors are typically dense, and subsequent steps then involve dense matrices.

Reducing the Size of the Basis

Let k_d be the number of basis vectors in $\mathbf{U}_d$ (or $\mathbf{V}_d$). While storing $\{\mathbf{U}_d, \boldsymbol{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]}$ avoids explicit handling of the full tensor $\mathcal{X}^{(t)}$, one of the k_d's is increased by one in each iteration. Thus, the matrices $\{\mathbf{U}_d, \boldsymbol{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]}$ gradually increase in size and may cause memory problems (especially for $\mathbf{V}_d$'s). In this section, we propose to "compress" these matrices when $\sum_{d=1}^D k_d$ exceeds a given threshold K.

We consider the modes one at a time. For a particular d, let $\mathbf{Q}_\mathbf{U}\mathbf{R}_\mathbf{U}$ (resp. $\mathbf{Q}_\mathbf{V}\mathbf{R}_\mathbf{V}$) be the QR decomposition of $\mathbf{U}_d$ (resp. $\mathbf{V}_d$). Thus, $\mathbf{U}_d\boldsymbol{\Sigma}_d\mathbf{V}_d^\top = \mathbf{Q}_\mathbf{U}\mathbf{J}_0\mathbf{Q}_\mathbf{V}^\top$, where $\mathbf{J}_0 = \mathbf{R}_U\boldsymbol{\Sigma}_d\mathbf{R}_V^\top$. The objective in (1) can be rewritten as $F(\mathcal{X}) = \frac{1}{2}\|\mathcal{P}_\Omega((\mathbf{Q}_\mathbf{U}\mathbf{J}_0\mathbf{Q}_\mathbf{V}^\top)^{\langle d\rangle} + \mathcal{B}_d)\|_F^2$, where $\mathcal{B}_d = \mathcal{P}_\Omega(\sum_{i\ne d}(\mathbf{U}_i\boldsymbol{\Sigma}_i\mathbf{V}_i^\top)^{\langle i\rangle} - \mathcal{A})$. Recall that the (matrix) nuclear norm is orthogonally invariant (Parikh and Boyd 2014), i.e., $\|\mathbf{Q}_\mathbf{U}\mathbf{J}_0\mathbf{Q}_\mathbf{V}^\top\|_* = \|\mathbf{J}_0\|_*$. We replace $\mathbf{Q}_\mathbf{U}\mathbf{J}_0\mathbf{Q}_\mathbf{V}^\top$ by $\mathbf{Q}_\mathbf{U}\mathbf{J}\mathbf{Q}_\mathbf{V}^\top$ and minimize $F(\mathcal{X})$ such that $\|\mathbf{Q}_\mathbf{U}\mathbf{J}\mathbf{Q}_\mathbf{V}^\top\|_* \le \|\mathbf{Q}_\mathbf{U}\mathbf{J}_0\mathbf{Q}_\mathbf{V}^\top\|_*$, i.e., $\min_\mathbf{J} \|\mathcal{P}_\Omega((\mathbf{Q}_\mathbf{U}\mathbf{J}\mathbf{Q}_\mathbf{V}^\top)^{\langle d\rangle} + \mathcal{B}_d)\|_F^2 : \|\mathbf{J}\|_* \le \|\mathbf{J}_0\|_*$. This can be solved by projected gradient descent[1], in which projection onto the nuclear-norm ball has closed-form solution (Parikh and Boyd 2014). Algorithm 2 shows the whole basis reduction procedure.

Algorithm 2 Reducing the size of $\{\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]}$.

```
input $\{\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]}$;
1: for $d = 1, 2, \dots, D$ do
2:     $[\mathbf{Q}_\mathbf{U}, \mathbf{R}_\mathbf{U}] = \text{QR}(\mathbf{U}_d)$,   $[\mathbf{Q}_\mathbf{V}, \mathbf{R}_V] = \text{QR}(\mathbf{V}_d)$;
3:     $\mathbf{J}_0 = \mathbf{R}_U \mathbf{\Sigma}_d \mathbf{R}_V^\top$;
4:     $\mathbf{J} = \arg\min_{\mathbf{J}:\|\mathbf{J}\|_* \le \|\mathbf{J}_0\|_*} \|\mathcal{P}_\Omega((\mathbf{Q}_\mathbf{U}\mathbf{J}\mathbf{Q}_\mathbf{V}^\top)^{\langle d\rangle} + \mathcal{B}_d)\|_F^2$;
5:     $\mathbf{U}_\mathbf{J}\mathbf{\Sigma}_\mathbf{J}\mathbf{V}_\mathbf{J}^\top \leftarrow \text{SVD}(\mathbf{J})$;
6:     $\mathbf{U}_d = \mathbf{Q}_\mathbf{U}\mathbf{U}_\mathbf{J}$,   $\mathbf{V}_d = \mathbf{Q}_\mathbf{V}\mathbf{V}_\mathbf{J}$,   $\mathbf{\Sigma}_d = \mathbf{\Sigma}_\mathbf{J}$;
7:     $k_d$ = number of nonzero elements in $\mathbf{\Sigma}_\mathbf{J}$;
8: end for
output $\{\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d, k_d\}_{d\in[D]}$.
```

Obviously, the size of the processed basis will not be increased after basis reduction, and in practice it can be much smaller. Moreover, the objective is reduced as shown by the following Proposition.

Proposition 2. *$F(\sum_{d=1}^D (\mathbf{U}'_d\mathbf{\Sigma}'_d{\mathbf{V}'_d}^\top)^{\langle d\rangle}) \le F(\sum_{d=1}^D (\mathbf{U}_d\mathbf{\Sigma}_d\mathbf{V}_d^\top)^{\langle d\rangle})$, where $\mathbf{U}'_d, \mathbf{\Sigma}'_d, \mathbf{V}'_d$ are the outputs from Algorithm 2.*

The Complete Procedure

The whole procedure is shown in Algorithm 3. It is known that FW converges at a rate of $\mathcal{O}(1/T)$ (Jaggi 2013). However, because of the extra basis reduction step, the standard convergence results cannot be directly used.

Algorithm 3 Fast FW algorithm for tensor completion (FFWTensor).

```
1: Initialize $\mathcal{X}^{(0)} = \mathbf{0}, k_1 = \cdots = k_D = 0$, $\{\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]} = []$;
2: for $t = 0, 1, \dots, T$ do
3:     $d^* = \arg\max_{d\in[D]} \sqrt{I_d}\sigma_{\max}(-\nabla F(\mathcal{X}^{(t)})_{\langle d\rangle})$;
4:     $[\mathbf{u}_{d^*}, \mathbf{v}_{d^*}]$ = leading left and right singular vectors of $-\nabla F(\mathcal{X}^{(t)})_{\langle d^*\rangle}$;
5:     $\mathcal{S}^{(t+1)}|_\Omega = ((\tau\sqrt{I_{d^*}}\mathbf{u}_{d^*}\mathbf{v}_{d^*}^\top)^{\langle d^*\rangle})|_\Omega$;
6:     update $\gamma^{(t+1)}$ by (2);
7:     $\mathcal{X}^{(t+1)}|_\Omega = (1-\gamma^{(t+1)})\mathcal{X}^{(t)}|_\Omega + \gamma^{(t+1)}\mathcal{S}^{(t+1)}|_\Omega$;
8:     update $\{\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d\}_{d\in[D]}$ using (4), (5);
9:     $k_{d^*} \leftarrow k_{d^*} + 1$;
10:    if $\sum_{d=1}^D k_d \ge K$ then        // basis reduction
11:        shrink $\{\mathbf{U}_d, \mathbf{\Sigma}_d, \mathbf{V}_d, k_d\}_{d\in[D]}$ by Algorithm 2;
12:        $\mathcal{X}^{(t+1)}|_\Omega = \left(\sum_{d=1}^D (\mathbf{U}_d\mathbf{\Sigma}_d\mathbf{V}_d^\top)^{\langle d\rangle}\right)|_\Omega$;
13:    end if
14: end for
output $\mathcal{X} = \sum_{d=1}^D (\mathbf{U}_d\mathbf{\Sigma}_d\mathbf{V}_d^\top)^{\langle d\rangle}$.
```

[1]In the experiments, we only perform one descent iteration.

Let $\mathcal{X}^*$ be the optimal solution of (1), and C_F be the "curvature" of F (Jaggi 2013). The following Theorem shows that we still have the same $\mathcal{O}(1/T)$ rate. Moreover, it can be shown that $C_F \le 4\tau^2$ here, which is independent of the tensor dimensionality.

Theorem 1. *The sequence $\{\mathcal{X}^{(t)}\}$ generated by Algorithm 3 satisfies $F(\mathcal{X}^{(t)}) - F(\mathcal{X}^*) \le 2C_F/(t+2)$.*

Recall that $\mathcal{X}^{(t)} = \sum_{d=1}^D (\mathbf{U}_d^{(t)}\mathbf{\Sigma}_d^{(t)}{\mathbf{V}_d^{(t)}}^\top)^{\langle d\rangle}$ (here, we have explicitly included the iteration index t). The following Proposition shows that $(\mathbf{U}_d^{(t)}\mathbf{\Sigma}_d^{(t)}{\mathbf{V}_d^{(t)}}^\top)^{\langle d\rangle}$'s converge to the latent factors of $\mathcal{X}^{(t)}$ (as defined in the scaled latent nuclear norm).

Proposition 3. *Assume that $\lim_{t\to\infty} \|\mathcal{X}^{(t)}\|_{scaled} = \tau$. When $t \to \infty$, $\{(\mathbf{U}_d^{(t)}\mathbf{\Sigma}_d^{(t)}{\mathbf{V}_d^{(t)}}^\top)^{\langle d\rangle}\}_{d\in[D]}$ is a solution of $\arg\min_{\mathcal{X}_1,\dots,\mathcal{X}_D:\sum_{d=1}^D \mathcal{X}_d = \mathcal{X}^{(t)}} \sum_{d=1}^D \frac{1}{\sqrt{I_d}}\|(\mathcal{X}_d)_{\langle d\rangle}\|_*$.*

In contrast, with the TTN regularizer, the FW linear subproblem in (Yang, Feng, and Suykens 2015; Cheng et al. 2016) can only be solved approximately. Consequently, it has the weaker convergence result $F(\mathcal{X}^{(t)}) - \frac{F(\mathcal{X}^*)}{\alpha} \le \frac{4C}{t+1}$, where C is some constant (Cheng et al. 2016). For a large tensor, $\alpha = \prod_{d=1}^{D-2} I_d^{-1/2}$ is close to zero, and this bound can be very loose. Yang, Feng, and Suykens (2015) solves the same FW linear subproblem as (Cheng et al. 2016), but uses a different approximate solver. Unfortunately, its convergence guarantee is even weaker.

Post-Processing

In matrix completion, the (matrix) nuclear norm may over-penalize the singular values of the solution (Mazumder, Hastie, and Tibshirani 2010). For tensor completion, a similar over-penalization may occur.

To alleviate this problem, we undo some of the shrinkage by adding a post-processing step as in (Mazumder, Hastie, and Tibshirani 2010), which re-fits the basis without the nuclear norm constraint. Recall that $\mathcal{X}^{(T)} = \sum_{d=1}^D (\mathbf{U}_d\mathbf{\Sigma}_d\mathbf{V}_d^\top)^{\langle d\rangle}$. Let $\mathbf{u}_{dl}$ (resp. $\mathbf{v}_{dl}$) be the lth column of $\mathbf{U}_d$ (resp. $\mathbf{V}_d$), and σ_{dl} the lth diagonal element of $\mathbf{\Sigma}_d$. $\mathcal{X}^{(T)}$ can be rewritten as $\mathcal{X}^{(T)} = \sum_{dl}\sigma_{dl}\mathcal{W}_{dl}$, where $\mathcal{W}_{dl} = (\mathbf{u}_{dl}\mathbf{v}_{dl}^\top)^{\langle d\rangle}$. We then minimize $\frac{1}{2}\|\mathcal{P}_\Omega(\sum_{dl}\sigma_{dl}\mathcal{W}_{dl} - \mathcal{A})\|_F^2$ w.r.t. $\boldsymbol{\sigma} = \{\sigma_{dl}\}$, which leads to a simple least-squares problem.

Experiments

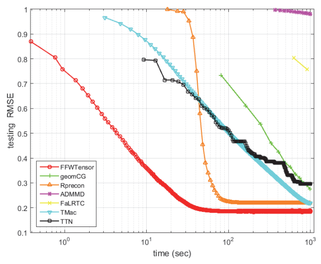
Color Image Inpainting

In this section, we perform experiments on a 2901×3000 RGB image (Figure 1(a)). 5% of the $2901 \times 3000 \times 3$ data tensor entries are randomly sampled as the training set, another 20% as validation set (for parameter tuning), and the rest as testing set. The experiment is repeated 10 times.

The proposed FFWTensor is compared with the following state-of-the-art: (i) GeomCG[2] (Kressner, Steinlechner, and

[2]Downloaded from http://anchp.epfl.ch/

(a) Image used.



(b) Testing RMSE vs CPU time.

Figure 1: Experiment on the color image.

Vandereycken 2014) based on Riemannian optimization on the manifold of fixed-rank tensors; (ii) Rprecon[3] (Kasai and Mishra 2016), which exploits preconditioning on the Riemannian manifold; (iii) Alternating direction methods of multipliers (ADMM)[4] applied on the dual form of latent scaled norm (Tomioka, Hayashi, and Kashima 2010); (iv) FalRTC[5] (Liu et al. 2013), which performs accelerated proximal gradient descent with the smoothed overlapped nuclear norm[6]; (v) TMac[7] (Xu et al. 2013), which performs simultaneous low-rank matrix factorizations to all mode matricizations; (vi) TTN (Cheng et al. 2016), which uses FW with the tensor nuclear norm; (vii) ROne (Yang, Feng, and Suykens 2015), which is similar to TTN but uses a different procedure for the FW linear subproblem. All the codes are in Matlab. Experiments are performed on a PC with Intel i7 CPU and 16GB RAM. Since the algorithms are based on different models, we use the same stopping criterion that allows each to run for 1000 seconds. The testing root-mean-squared error (RMSE) is used for performance evaluation.

Table 1 shows the testing RMSE obtained, and Figure 1(b) shows the convergence. The proposed FFWTensor has the best RMSE and fastest convergence (about 20 times faster). Table 2 shows the numbers of basis vectors (k_d) in $\{\mathbf{U}_d\}_{d\in[D]}$. As discussed earlier, mode 3 (with $I_3 = 3$) is never selected into the basis, and so we do not need to store large basis matrices of size 2901×3000. Table 2 also shows the sizes of the data tensor $\mathcal{X}$, the sparsified data tensor $\mathcal{X}|_\Omega$, and the basis matrices ($\mathbf{U}_d$'s, $\mathbf{\Sigma}_d$'s and $\mathbf{V}_d$'s). As can be seen, FFWTensor is much more space-efficient than processing the whole data tensor directly.

[3]Downloaded from https://bamdevmishra.com/codes/tensorcompletion/

[4]Downloaded from http://ttic.uchicago.edu/~ryotat/softwares/tensor/

[5]Downloaded from http://www.cs.rochester.edu/u/jliu/code/TensorCompletion.zip

[6]As suggested in (Liu et al. 2013), we use the scaled overlapped version $\sum_{d=1}^{D} \|\mathcal{X}_{\langle d \rangle}\|_* / \sqrt{I_d}$.

[7]Downloaded from http://www.math.ucla.edu/~wotaoyin/papers/codes/TMac.zip

Table 1: Testing RMSE on the color image. ROne runs out of memory and is thus not reported. Result that is better than the others, according to the paired t-test at 95% significance level, is highlighted.

FFWTensor	$\mathbf{0.186 \pm 0.001}$
GeomCG	0.276 ± 0.001
Rprecon	0.219 ± 0.002
ADMM	0.981 ± 0.001
FalRTC	0.757 ± 0.004
TMac	0.218 ± 0.002
TTN	0.297 ± 0.002

Multi-Relational Link Prediction

Two data sets are used. The first one is ClimateNet[8], which is constructed from the $5^\circ \times 5^\circ$ latitude-longitude gridded climate data set. There are 1773 nodes (physical locations), and 7 types of node similarities (e.g., temperature, sea-level pressure). As in (Davis, Lichtenwalter, and Chawla 2011), we binarize the similarities so that 27.3% of the node pairs are linked. 5% of the entries in the $1773 \times 1773 \times 7$ data tensor are sampled as observed, another 20% for validation and the rest for testing. The second one is YouTube data set[9] (Tang, Wang, and Liu 2009), with $15,088$ users and 5 types of Boolean interactions. From the $15088 \times 15088 \times 5$ data tensor, we randomly sample 0.8% of the entries as observed, another 0.1% for validation and 0.15% for testing. Many baselines (ADMM, FaLRTC, TMac, TTN) need to maintain the whole target tensor, and run out of memory on this large data set.[10] Hence, we also experiment with a subset having only the $1,509$ users with the most number of links. From the $1509 \times 1509 \times 5$ data tensor, we sample 5% of the entries as observed, another 20% for validation, and the rest for testing. The experiment is repeated 10 times.

For performance evaluation, we randomly select 10^6 pairs to form $S = \{((i,j,k),(p,q,r))\}$, where tensor entry (i,j,k) is a positive link, and (p,q,r) is negative. The AUC (area under the ROC curve) is then estimated as $\frac{1}{|S|}\sum_{((i,j,k),(p,q,r))\in S}[\mathbb{I}(\mathcal{X}_{ijk} > \mathcal{X}_{pqr}) + 0.5\mathbb{I}(\mathcal{X}_{ijk} = \mathcal{X}_{pqr})]$ (Lü and Zhou 2011), where $\mathbb{I}(\cdot)$ is the indicator function.

Table 3 shows the testing AUCs obtained, and Figure 2 shows the convergence. The proposed FFWTensor achieves the best AUC, is much faster and more scalable. Moreover, as in the image inpainting experiment, mode 3 (with a small dimensionality) is never selected into the basis, and FFWTensor is thus much more space-efficient (Table 2).

Figure 3 shows the basis size obtained by FFWTensor. As can be seen, it can be effectively controlled by basis reduction (which is triggered when the size exceeds $K = 100$). The basis size after each run of basis reduction remains relatively stable, which agrees with the fact that the model has almost converged (Figure 2).

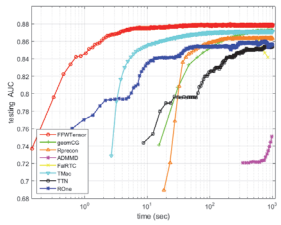
[8]http://www.nd.edu/dial/software/climateNet.zip

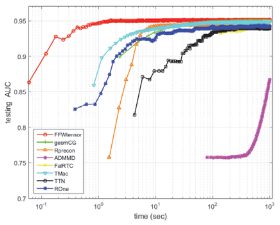
[9]http://leitang.net/data/youtube-data.tar.gz

[10]As shown in Table 2, storing the whole data tensor alone already takes 4.391GB memory.

Table 2: Values of basis vectors (k_d) in the various modes, and sizes of some relevant tensors/matrices.

	number of basis vectors (k_d)			storage				
	mode 1	mode 2	mode 3	$\mathcal{X}$	$\mathcal{X}\vert_\Omega$	$\mathbf{U}_d$'s	$\mathbf{\Sigma}_d$'s	$\mathbf{V}_d$'s
color image	39	42	0	100M	14M	1M	0.39K	3M
ClimateNet	48	47	0	84M	12M	0.65M	0.39K	5M
YouTube (subset)	34	36	0	43M	6.5M	0.57M	0.39K	3M
YouTube (full set)	43	40	0	4.391G	102M	5.7M	0.39K	29M


(a) ClimateNet.


(b) YouTube (subset).

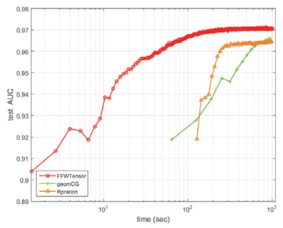
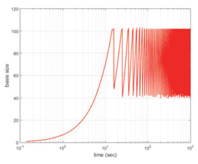
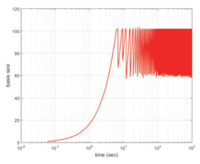

(c) YouTube (full set).

Figure 2: Testing AUC vs CPU time on the link prediction data sets.

Table 3: Testing AUCs on link prediction (all numbers have standard deviation of 0.001). Note that only FFWTensor and GeomCG can be run on the full YouTube set.

	Climate	YouTube (subset)	YouTube (full)
FFWTensor	**0.877**	**0.950**	**0.970**
GeomCG	0.873	**0.949**	0.966
Rprecon	0.863	0.944	0.964
ADMM	0.751	0.867	-
FalRTC	0.842	**0.950**	-
TMac	0.871	0.948	-
TTN	0.854	0.939	-
ROne	0.857	0.942	-


(a) ClimateNet.


(b) YouTube (subset).

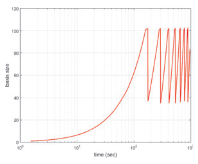
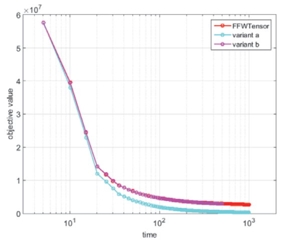
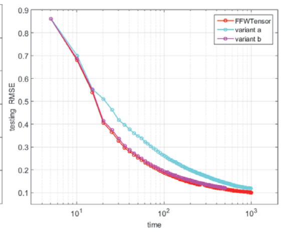
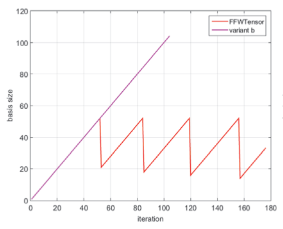

(c) YouTube (full).

Figure 3: Basis size obtained by FFWTensor.


(a) Objective vs CPU time.


(b) RMSE vs CPU time.


(c) Basis size vs iterations.

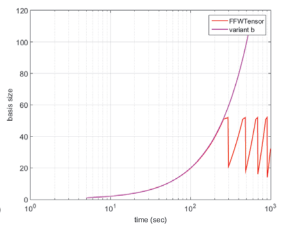

(d) Basis size vs time (sec).

Figure 4: FFWTensor and its variants on synthetic data.

Synthetic Data

Finally, we experiment with FFWTensor and its variants on a synthetic data set. The data are generated as $\mathcal{A} = \mathcal{G} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3$, where $\mathcal{G} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$, $\mathbf{A}_1 \in \mathbb{R}^{3000 \times k_1}$, $\mathbf{A}_2 \in \mathbb{R}^{3000 \times k_2}$, $\mathbf{A}_3 \in \mathbb{R}^{3000 \times k_3}$, and $k_1 = 5, k_2 = 50, k_3 = 2000$. Entries of $\mathcal{G}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ are generated from the standard normal distribution. $\mathcal{A}$ is then normalized to zero mean and unit variance, and Gaussian noise $\mathcal{N}(0, 0.05)$ is added. 0.6% of the entries in $\mathcal{A}$ are randomly sampled as observed, 0.1% for validation, and another 0.1% for testing. The experiment is repeated 10 times. For performance evaluation, we use the testing RMSE.

Without utilizing sparsity, variant c requires a lot of memory and cannot be run. Post-processing (FFWTensor vs variant a) slightly slows down the algorithm (Figure 4(a)), but can improve RMSE (Figure 4(b) and Table 4). Without basis reduction, variant b can only run for 103 iterations and terminates early (Figure 4(c)), leading to inferior RMSE (Table 4). Basis reduction significantly reduces the basis size, and has little impact on the CPU time (Figure 4(d)).

Conclusion

In this paper, we proposed a novel low-rank tensor completion algorithm by using the scaled latent nuclear norm for regularization and the Frank-Wolfe algorithm for optimization. All the steps can be performed efficiently, and

Table 4: Testing RMSE of FFWTensor variants on synthetic data (all numbers have standard deviation of 0.001).

	sparse structure	basis reduction	post-processing	RMSE
FFWTensor	√	√	√	**0.100**
variant a	√	√	×	0.120
variant b	√	×	√	0.121
variant c	×	×	×	-

can also take advantage of the sparsity structure of the observed incomplete tensor. Experimental results show that the proposed method is more accurate, much faster and more scalable than the state-of-the-art.

Acknowledgments

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

References

Acar, E.; Dunlavy, D.; Kolda, T.; and Mørup, M. 2010. Scalable tensor factorizations with missing data. In *Proceedings of the International Conference on Data Mining*, 701–712.

Adomavicius, G.; Manouselis, N.; and Kwon, Y. 2011. Multi-criteria recommender systems. In *Recommender Systems Handbook*. Springer. 769–803.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

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

Chandrasekaran, V.; Recht, B.; Parrilo, P.; and Willsky, A. 2012. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics* 12(6):805–849.

Cheng, H.; Yu, Y.; Zhang, X.; Xing, E.; and Schuurmans, D. 2016. Scalable and sound low-rank tensor learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 1114–1123.

Davis, D.; Lichtenwalter, R.; and Chawla, N. 2011. Multi-relational link prediction in heterogeneous information networks. In *Proceedings of the international conference on Advances in Social Network Analysis and Mining*, 281–288.

Halko, N.; Martinsson, P.-G.; and Tropp, J. A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review* 53(2):217–288.

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

Jaggi, M. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the International Conference on Machine Learning*, 427–435.

Kasai, H., and Mishra, B. 2016. Low-rank tensor completion: A Riemannian manifold preconditioning approach. In *Proceedings of the International Conference on Machine Learning*.

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

Kressner, D.; Steinlechner, M.; and Vandereycken, B. 2014. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics* 54(2):447–468.

Liu, J.; Musialski, P.; Wonka, P.; and Ye, J. 2013. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1):208–220.

Lü, L., and Zhou, T. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* 390(6):1150–1170.

Mazumder, R.; Hastie, T.; and Tibshirani, R. 2010. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research* 11:2287–2322.

Nesterov, Y. 2005. Smooth minimization of non-smooth functions. *Mathematical programming* 103(1):127–152.

Parikh, N., and Boyd, S. 2014. Proximal algorithms. *Foundations and Trends in Optimization* 1(3):127–239.

Tang, L.; Wang, X.; and Liu, H. 2009. Uncovering groups via heterogeneous interaction analysis. In *Proceedings of the International Conference on Data Mining*, 503–512.

Tomioka, R., and Suzuki, T. 2013. Convex tensor decomposition via structured Schatten norm regularization. In *Neural Information Processing Systems*, 1331–1339.

Tomioka, R.; Hayashi, K.; and Kashima, H. 2010. Estimation of low-rank tensors via convex optimization. Technical Report arXiv:1010.0789.

Wimalawarne, K.; Sugiyama, M.; and Tomioka, R. 2014. Multitask learning meets tensor factorization: Task imputation via convex optimization. In *Neural Information Processing Systems*, 2825–2833.

Xu, Y.; Hao, R.; Yin, W.; and Su, Z. 2013. Parallel matrix factorization for low-rank tensor completion. Technical Report arXiv:1312.1254.

Yang, Y.; Feng, Y.; and Suykens, J. 2015. A rank-one tensor updating algorithm for tensor completion. *Signal Processing Letters* 22(10):1633–1637.

Zhang, X.; Schuurmans, D.; and Yu, Y. 2012. Accelerated training for matrix-norm regularization: A boosting approach. In *Neural Information Processing Systems*, 2906–2914.