Multi-version Tensor Completion for Time-delayed Spatio-temporal Data

Cheng Qian[1,*], **Nikos Kargas**[2,*], **Cao Xiao**[1], **Lucas Glass**[1], **Nicholas Sidiropoulos**[3] and **Jimeng Sun**[4]

[1]Analytics Center of Excellence, IQVIA
[2]Department of Electrical and Computer Engineering, University of Minnesota Twin Cities
[3]Department of Electrical and Computer Engineering, University of Virginia
[4]Department of Computer Science, University of Illinois Urbana-Champaign
{alextoqc,danicaxiao,jimeng.sun}@gmail.com, karga005@umn.edu, lucas.Glass@iqvia.com, nikos@virginia.edu

Abstract

Real-world spatio-temporal data is often incomplete or inaccurate due to various data loading delays. For example, a location-disease-time tensor of case counts can have multiple delayed updates of recent temporal slices for some locations or diseases. Recovering such missing or noisy (under-reported) elements of the input tensor can be viewed as a generalized tensor completion problem. Existing tensor completion methods usually assume that i) missing elements are randomly distributed and ii) noise for each tensor element is i.i.d. zero-mean. Both assumptions can be violated for spatio-temporal tensor data. We often observe multiple versions of the input tensor with different under-reporting noise levels. The amount of noise can be time- or location-dependent as more updates are progressively introduced to the tensor. We model such dynamic data as a multi-version tensor with an extra tensor mode capturing the data updates. We propose a low-rank tensor model to predict the updates over time. We demonstrate that our method can accurately predict the ground-truth values of many real-world tensors. We obtain up to 27.2% lower root mean-squared-error compared to the best baseline method. Finally, we extend our method to track the tensor data over time, leading to significant computational savings.

1 Introduction

Data completion is the process of estimating missing elements of an N-dimensional array using the observed elements and some presumed structural properties of the data, such as non-negativity and/or low rank. In many real-world spatio-temporal data, missing values are often due to delays or failures in the data acquisition or reporting process. Low-rank tensor factorization and completion is a popular approach to data denoising and completion, with applications ranging from image restoration [Liu *et al.*, 2012; Xu and Yin, 2013] to healthcare data completion [Acar *et al.*, 2011; Wang *et al.*, 2015], recommendation systems [Almutairi *et al.*, 2017; Song *et al.*, 2017], and link prediction [Lacroix *et al.*, 2018]. Low-rank tensor factorization models have also been proposed for analyzing and making predictions based on spatio-temporal data [Bahadori *et al.*, 2014; Araujo *et al.*, 2017; Kargas *et al.*, 2021].

Existing methods usually assume that missing elements are randomly distributed, and the noise for each tensor element is zero-mean i.i.d. (independent and identically distributed). However, in many real-world tensor data, such as medical claims data and public health reporting data, both assumptions are violated. For example, insurance claim processing centers receive daily data from third-party providers, where there is often a time lag between data generation and data loading. There are time delays when hospitals report COVID case counts to the government. Such data can be modeled as multiple input tensors with different time-dependent noise levels, where more recent data are more unreliable due to the data acquisition delays. Periodically, updates on tensor elements arrive to produce a new version of the tensor. Over time, we observe multiple versions of the underlying tensor. To model such multi-version tensors, we have to deal with the following challenges.

- **Atypical noise model:** In our context, the main source of noise is under-reporting. Thus noise is non-positive *and* negatively correlated in the update mode (if one update is unusually large, the next is likely to be smaller). Under-reporting also exhibits predictable patterns (e.g., holidays and weekends).
- **Devising efficient schemes for incremental updates:** As data updates arrive incrementally over time for both new and historical tensor slices, it is challenging to update the tensor factorization efficiently. Existing works [Sun *et al.*, 2006; Zhou *et al.*, 2016] in dynamic tensor factorization assume that the historical data slices remain unchanged, and only new slices are added.

To address these challenges, we propose a Multi-version Tensor Completion (MTC) framework. Our key contributions are as follows:

- **Multi-version tensor model:** Instead of modeling the different versions of the data tensor *per se*, we introduce an

*Equal contribution.

extra tensor mode capturing the *data updates* across multiple versions of the data tensor. This allows us to explicitly model the update dynamics in latent space.

- **Online factorization:** We propose a low-rank tensor model to accurately track the updates over time. To estimate the tensor values at any given time point, we perform marginalization over the extra tensor mode. We show how we can extend our approach to track the tensor data over time, leading to significant computational savings.
- We compare MTC against several tensor and time series baselines on multiple real-world spatio-temporal datasets. We observe that MTC can accurately predict the ground-truth values of many real-world tensors and achieve up to 27.2% lower root mean-squared-error than the best baseline method.
- Finally, we extend our method to track the tensor data over time, leading to significant computational savings.

2 Related Work

CANDECOMP/PARAFAC decomposition (CPD) [Harshman, 1970] and Tucker decomposition [Tucker, 1966] have been used for imputing missing data for image restoration [Liu *et al.*, 2012; Xu and Yin, 2013], grade prediction [Almutairi *et al.*, 2017], link prediction [Lacroix *et al.*, 2018], and healthcare data completion [Acar *et al.*, 2011; Wang *et al.*, 2015]. Under certain low-rank assumptions, it is possible to recover the missing entries [Yuan and Zhang, 2016]. Tensor models have also been used for joint tracking and imputation [Sun *et al.*, 2006; Sun *et al.*, 2008; Nion and Sidiropoulos, 2009; Mardani *et al.*, 2015; Song *et al.*, 2017] which enables dynamic tensor factorization with streaming data via multi-linear decomposition. Recently, several nonlinear tensor factorization models have been proposed. NeuralCP [Liu *et al.*, 2018] employs two neural networks where the first one predicts the entry value and the second one estimates the noise variance. COSTCO [Liu *et al.*, 2019] is a convolutional tensor framework that models the nonlinear relationship using the local embedding features extracted by two convolutional layers. SPIDER [Fang *et al.*, 2020] is a probabilistic deep tensor factorization method suitable for streaming data.

These methods assume that past slices remain unchanged and the noise in the tensor elements is i.i.d. zero-mean. These assumptions are not appropriate for modeling noise due to under-reporting, as explained earlier, which motivates our Multi-version Tensor Completion (MTC) framework.

3 Problem Statement

Given a spatio-temporal tensor of I locations and J features over time, we first introduce two time concepts:

Generation date (GD) is the time when data items are generated.

Loading date (LD) is when we receive the data items. On an LD t, we can receive data items related to multiple locations and signals that have been generated in different GDs.

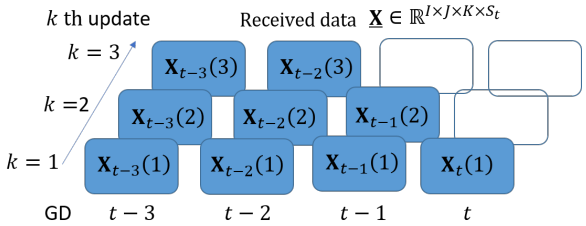

Figure 1: The updates for the data on GD t.

Next, we will introduce several tensors at loading date t: the observed tensor $\underline{\mathbf{Z}}_t$, the ground-truth tensor $\underline{\tilde{\mathbf{Z}}}_t$, and the update tensor $\underline{\mathbf{X}}_t$ all up to time t.

On each LD t, we receive data updates $\{x_{ijs}(t)\}_{s\in\mathcal{S}_t}$ for location i, feature j and GDs $\mathcal{S}_t$ which are earlier than t i.e., $s \leq t\ \forall s \in \mathcal{S}_t$. The total number of data that are generated on GD s and are available on LD t is computed by summing over all data received in the interval $[s, t]$, i.e., $z_{ijs}(t) = \sum_{t'=s}^{t} x_{ijs}(t')$. Stacking all $\{z_{ijs}(t)\}$ into a 3-way tensor, yields $\underline{\mathbf{Z}}_t \in \mathbb{R}_+^{I\times J\times S_t}$, where $\underline{\mathbf{Z}}_t(i, j, s) = z_{ijs}(t)$. Tensor $\underline{\mathbf{Z}}_t$ contains the aggregated number of updates which have been received at time t. Because of delays in the data acquisition process, data corresponding to the most recent GDs are usually under-reported. Let $\underline{\widetilde{\mathbf{Z}}}_t$ be the ground truth tensor at time t. The signal model can then be described by

$$\underline{\mathbf{Z}}_t = \underline{\widetilde{\mathbf{Z}}}_t + \underline{\mathbf{N}}_t,\ t = 1, 2, \ldots, \tag{1}$$

where $\mathbf{N}_t \in \mathbb{R}^{I\times J\times S_t}$ denotes a noise tensor. Our goal is to estimate the groundtruth tensor $\underline{\widetilde{\mathbf{Z}}}_t$. We identify three main challenges in estimating $\underline{\widetilde{\mathbf{Z}}}_t$.

1. The values corresponding to the latest GDs in $\underline{\mathbf{Z}}_t$, i.e., the latest frontal slabs of $\underline{\mathbf{Z}}_t$ are under-reported and thus very noisy. We need to unveil the update patterns from past data and correct these highly corrupted slabs.
2. The noise distribution is unknown in practice. The number of noise changes over time as more corrections to the tensor is introduced. Also, there are patterns in the noise. The method needs to be flexible enough so that it can capture different noise distributions.
3. The dimension corresponding to the GDs in $\underline{\mathbf{Z}}_t$ is gradually growing as t increases and more data are introduced. Note that new updates arrive continuously. Therefore, the method needs to be adaptive such that it cannot only correct the under-reported numbers but also refine itself efficiently as soon as new data are received.

To tackle these challenges, the key idea is to track the update tensor. We may assume that data corresponding to a given GD is updated at most K times. This implies that at time t we receive data for GDs $(t - K + 1)$ to t. An example is shown in Figure 1, where we assume $K = 3$ and $\mathbf{X}_t(k)$ denotes the kth update of the data for I location and J features generated on GD t. Each column in Figure 1 shows the updates for the data on GD t. In this example, GD $(t - 3)$ has received three updates at time $t-3$, $t-2$ and $t-1$ respectively. By summing over the three updates, we have the ground

truth values. Therefore, we have transformed the problem into an equivalent 4-way tensor completion problem. The final estimate of the target tensor $\widetilde{\underline{\mathbf{Z}}}_t$ can be estimated by first imputing the missing values of $\underline{\mathbf{X}}_t$, and then marginalizing over the third mode

$$\hat{\underline{\mathbf{Z}}}_t = \sum_{k=1}^{K} \underline{\mathbf{X}}_t(:,:,k,:). \tag{2}$$

where $\underline{\mathbf{X}}_t(:,:,k,:)$ is the MATLAB notation that selects data from a tensor.

4 Multi-version Tensor Completion (MTC)

Formulation In this work, we approximate $\underline{\mathbf{X}}$ using a low-rank CPD model [Harshman, 1970]. For the ease of notation, w drop the underscript t. CPD expresses tensor $\underline{\mathbf{X}}$ as a sum of F rank-1 tensors,

$$\underline{\mathbf{X}} = \sum_{f=1}^{F} \mathbf{A}(:,f) \circ \mathbf{B}(:,f) \circ \mathbf{C}(:,f) \circ \mathbf{D}(:,f), \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{I\times F}$, $\mathbf{B} \in \mathbb{R}^{J\times F}$,$\mathbf{C} \in \mathbb{R}^{K\times F}$ and $\mathbf{D} \in \mathbb{R}^{S\times F}$ are the factor matrices for location, feature, loading date (LD) and and generation date (GD), respectively. This kind of low-rank model is intuitive in many applications: Counties in the same state can share the same data loading schedule, and thus they have similar loading patterns. This is also true for updates data collected and sent through the same insurance company. Therefore, there is a high correlation among the data points for similar locations and features, indicating low-rank structures in the tensor. Suppose the optimal low-rank approximation of $\underline{\mathbf{X}}$ is $\underline{\mathbf{X}} \approx [\![\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}]\!]$.

Recall that the received data can be split into two parts – a fully observed tensor corresponding to GDs from 1 to $(S-K+1)$ and an incomplete tensor corresponding to GDs from $(S-K+2)$ to S. The loss function is then expressed as

$$\mathcal{F}(\boldsymbol{\theta}) = \underbrace{\alpha\mathcal{F}_1(\boldsymbol{\theta})}_{\text{fully observed}} + \underbrace{(1-\alpha)\mathcal{F}_2(\boldsymbol{\theta})}_{\text{incomplete}} \tag{4}$$

where

$$\mathcal{F}_1(\boldsymbol{\theta}) = \sum_{s=1}^{S-K+1} \|\underline{\mathbf{X}}(:,:,:,s) - [\![\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{d}_s]\!]\|_F^2,$$

$$\mathcal{F}_2(\boldsymbol{\theta}) = \sum_{s=S-K+2}^{S} \|\mathcal{P}_{\Omega_s}(\underline{\mathbf{X}}(:,:,:,s) - [\![\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{d}_s]\!])\|_F^2$$

where $\|\cdot\|_F$ is the Frobenius norm, $\mathbf{d}_s$ denotes the s-th row of $\mathbf{D}$, Ω_s denotes the index set of the observed entries of $\underline{\mathbf{X}}(:,:,:,s)$, and $\mathcal{P}_{\Omega_s}(\cdot)$ is the operation for tensor completion by only keeping the entries from index set Ω_s and zero-out the remaining entries, and $\boldsymbol{\theta} = \{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$.

In the context of spatio-temporal analysis, we would like to keep the latent factors interpretable. Towards this end, we employ a graph regularizer which regularizes the latent subspace of locations according to its graph Laplacian $\mathbf{L}$, such that two locations are sufficiently close to each other in the latent subspace if they are connected in the graph and have similar loading patterns. This regularization term is

$$\mathcal{R}(\mathbf{A}) = \rho_A \text{tr}\left(\mathbf{A}^T\mathbf{L}\mathbf{A}\right),$$

where ^{T} is the transpose and ρ_A is the regularization parameter. Furthermore, for the LD factor $\mathbf{C}$ and GD factor $\mathbf{D}$,we impose smoothness regularization, i.e.,

$$\mathcal{R}(\mathbf{C}) = \rho\|\boldsymbol{\Gamma}\mathbf{C}\|_F^2,\ \mathcal{R}(\mathbf{D}) = \rho\|\boldsymbol{\Gamma}\mathbf{D}\|_F^2.$$

We choose $\boldsymbol{\Gamma}$ to be a Toeplitz matrix holding $(-1,2,-1)$ as its principal three diagonals and zeros elsewhere. $\boldsymbol{\Gamma}$ promotes smoothness on the latent factors so that data corresponding to adjacent time points has a smooth transition in LD and GD dimensions. Let $\mathcal{R}(\boldsymbol{\theta}) = \mathcal{R}(\mathbf{A})+\mathcal{R}(\mathbf{C})+\mathcal{R}(\mathbf{D})$. We propose solving

$$\min_{\boldsymbol{\theta}}\ \mathcal{F}(\boldsymbol{\theta}) + \mathcal{R}(\boldsymbol{\theta}),\ \text{s. t. } \boldsymbol{\theta} \geq 0, \tag{5}$$

where the constraint $\boldsymbol{\theta} \geq 0$ is selected to ensure that the imputed values are non-negative.

Initialization Due to the special structure of the tensor completion problem in Equation (4) we can obtain a good initialization for our algorithm by first solving

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}} \mathcal{F}_1(\boldsymbol{\theta}),\ \text{s.t. } \mathbf{A} \geq 0, \mathbf{B} \geq 0, \mathbf{C} \geq 0, \mathbf{d}_s \geq 0,$$

for $s = 1,\ldots,S-K+1$. This is a standard tensor factorization problem which can be solved efficiently [Xu and Yin, 2013; Fu *et al.*, 2020]. By solving the above problem we obtain initial estimates of $\mathbf{A},\mathbf{B},\mathbf{C}$. Then, $\mathcal{F}_2(\boldsymbol{\theta})$ reduces to a non-negative least squares problem, i.e.,

$$\min_{\mathbf{D}} \|\mathcal{P}_{\Omega_s}(\text{vec}(\underline{\mathbf{X}}(:,:,:,s)) - (\mathbf{A}\odot\mathbf{B}\odot\mathbf{C})\mathbf{d}_s^T)\|_2^2,\ \mathbf{d}_s \geq 0$$

for $s = S-K+2,\ldots,S$, a convex optimization problem which can be solved optimally, where $\text{vec}(\cdot)$ is the vectorization operator.

Optimization Having obtained initial estimates for $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ we turn our attention to optimization Problem (5). A common way of tackling Problem (5) is through alternating optimization. We introduce an auxiliary variable $\underline{\mathbf{Y}} \in \mathbb{R}^{I\times J\times K\times S}$ of the same size as $\underline{\mathbf{X}}$ and alternatively estimate the factor matrices while imputing the missing data leading to an expectation maximization (EM)-like approach. Now let us express optimization problem (5) as

$$\begin{aligned} &\min_{\boldsymbol{\theta},\underline{\mathbf{Y}}}\ \mathcal{F}(\boldsymbol{\theta},\underline{\mathbf{Y}}) + \mathcal{R}(\boldsymbol{\theta}) \\ &\text{s. t. } \boldsymbol{\theta} \geq 0, \mathcal{P}_{\Omega_\mathbf{s}}(\underline{\mathbf{Y}}(:,:,:,s)) = \mathcal{P}_{\Omega_\mathbf{s}}(\underline{\mathbf{X}}(:,:,:,s)), \\ &\quad \forall \mathbf{s} = S-K+2,\ldots,S, \end{aligned} \tag{6}$$

where

$$\mathcal{F}(\boldsymbol{\theta},\underline{\mathbf{Y}}) = \alpha\mathcal{F}_1(\boldsymbol{\theta},\underline{\mathbf{Y}}) + (1-\alpha)\mathcal{F}_2(\boldsymbol{\theta},\underline{\mathbf{Y}}),$$

$$\mathcal{F}_1(\boldsymbol{\theta},\underline{\mathbf{Y}}) = \sum_{s=1}^{S-K+1} \|\underline{\mathbf{Y}}(:,:,:,s) - [\![\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{d}_s]\!]\|_F^2,$$

$$\mathcal{F}_2(\boldsymbol{\theta},\underline{\mathbf{Y}}) = \sum_{s=S-K+2}^{S} \|\underline{\mathbf{Y}}(:,:,:,s) - [\![\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{d}_s]\!]\|_F^2.$$

Optimization problems (5) and (6) are equivalent [Xu and Yin, 2013]. Problem (6) can be readily handled through alternating optimization. At each iteration we fix all variables except for one. At τ-th iteration, the subproblem with respect to $\mathbf{A}$ is given by

$$\min_{\mathbf{A}\geq\mathbf{0}} \mathcal{F}(\mathbf{A}) + \mathcal{R}(\mathbf{A}). \tag{7}$$

To proceed, we define $\bar{\underline{\mathbf{Y}}}^{(\tau)} = \underline{\mathbf{Y}}^{(\tau)} \times_4 \mathbf{W}$, $\bar{\mathbf{D}}^{(\tau)} = \mathbf{W}\mathbf{D}^{(\tau)}$, with $\mathbf{W} = \operatorname{diag}\big(\sqrt{\alpha}, \cdots, \sqrt{\alpha}, \sqrt{1-\alpha}, \cdots, \sqrt{1-\alpha}\big)$. Using the mode-1 unfolding matrix unfolding [Sidiropoulos *et al.*, 2017] we equivalently express $\mathcal{F}(\mathbf{A})$ as

$$\mathcal{F}(\mathbf{A}) = \left\| \bar{\mathbf{Y}}_{(1)}^{(\tau)} - \mathbf{A}(\mathbf{H}_A^{(\tau)})^T \right\|_F^2, \tag{8}$$

where $\mathbf{H}_A^{(\tau)} = \mathbf{B}^{(\tau)} \odot \mathbf{C}^{(\tau)} \odot \bar{\mathbf{D}}^{(\tau)}$ and matrix $\bar{\mathbf{Y}}_{(1)}^{(\tau)} \in \mathbb{R}_+^{I\times JKS}$ is the mode-1 matrix unfolding which stores the mode-1 fibers of the tensor as its columns.

Let $f(\mathbf{A}) = \mathcal{F}(\mathbf{A}) + \mathcal{R}(\mathbf{A})$ which can be approximated by the following quadratic function locally around $\hat{\mathbf{A}}^{(\tau)}$

$$f(\mathbf{A}; \hat{\mathbf{A}}^{(\tau)}) = f(\hat{\mathbf{A}}^{(\tau)}) + \left\langle \nabla_{\mathbf{A}}\big(f(\hat{\mathbf{A}}^{(\tau)})\big), \mathbf{A} - \hat{\mathbf{A}}^{(\tau)} \right\rangle + \frac{\gamma_A^{(\tau)}}{2} \left\| \mathbf{A} - \hat{\mathbf{A}}^{(\tau)} \right\|_F^2, \tag{9}$$

where $\langle\cdot\rangle$ is the inner product, $\gamma_A^{(\tau)}$ is a parameter associated with the step-size. The gradient $\nabla_{\mathbf{A}}\big(f(\hat{\mathbf{A}}^{(\tau)})\big)$ is given by $\nabla_{\mathbf{A}}\big(f(\hat{\mathbf{A}}^{(\tau)})\big) = \Big(\hat{\mathbf{A}}^{(\tau)}(\mathbf{H}_A^{(\tau)})^T - \mathbf{Y}_{(1)}^{(\tau)}\Big)\mathbf{H}_A^{(\tau)} + \lambda\mathbf{L}\hat{\mathbf{A}}^{(\tau)}$ and

$$\hat{\mathbf{A}}^{(\tau)} = \mathbf{A}^{(\tau)} + \nu^{(\tau)}(\mathbf{A}^{(\tau)} - \mathbf{A}^{(\tau-1)}) \tag{10}$$

denotes an extrapolated point, $\nu^{(\tau-1)} \geq 0$ is the extrapolation weight which can be updated as [Beck and Teboulle, 2009] $\nu^{(\tau)} = \frac{1-e^{(\tau)}}{e^{(\tau+1)}}$ where $e^{(0)} = 0$ and $e^{(\tau)} = \frac{1+\sqrt{4(e^{(\tau-1)})^2+1}}{2}$. Now the subproblem w.r.t. $\mathbf{A}$ reduces to

$$\min_{\mathbf{A}\geq\mathbf{0}} f(\mathbf{A}; \hat{\mathbf{A}}^{(\tau)}) \tag{11}$$

which admits a closed-form expression as

$$\mathbf{A}^{(\tau+1)} = \mathbf{A}^{(\tau)} - \nabla_{\mathbf{A}}\big(f(\hat{\mathbf{A}}^{(\tau)})\big)/\gamma_A^{(\tau)}. \tag{12}$$

Following the above analysis, we can easily write down the updates for $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, i.e.,

$$\mathbf{B}^{(\tau+1)} = \mathbf{B}^{(\tau)} - \nabla_{\mathbf{B}}\big(f(\hat{\mathbf{B}}^{(\tau)})\big)/\gamma_B^{(\tau)} \tag{13}$$

$$\mathbf{C}^{(\tau+1)} = \mathbf{C}^{(\tau)} - \nabla_{\mathbf{C}}\big(f(\hat{\mathbf{C}}^{(\tau)})\big)/\gamma_C^{(\tau)} \tag{14}$$

$$\mathbf{D}^{(\tau+1)} = \mathbf{D}^{(\tau)} - \nabla_{\mathbf{D}}\big(f(\hat{\mathbf{D}}^{(\tau)})\big)/\gamma_D^{(\tau)} \tag{15}$$

where $\hat{\mathbf{B}}^{(\tau)}$, $\hat{\mathbf{C}}^{(\tau)}$ and $\hat{\mathbf{D}}^{(\tau)}$ have the same definition as $\hat{\mathbf{A}}^{(\tau)}$ in (10) [1]. Finally, the update for $\underline{\mathbf{Y}}$ is given by

$$\underline{\mathbf{Y}}^{(\tau+1)} = \mathcal{P}_{\Omega^c}\left(\hat{\underline{\mathbf{X}}}^{(\tau+1)}\right) + \mathcal{P}_{\Omega}\left(\underline{\mathbf{X}}\right), \tag{16}$$

[1] See Supplementary 1 for the analytic expressions of the gradients in (13)-(15).

where $\hat{\underline{\mathbf{X}}}^{(\tau+1)} = [\![\mathbf{A}^{(\tau+1)}, \mathbf{B}^{(\tau+1)}, \mathbf{C}^{(\tau+1)}, \mathbf{D}^{(\tau+1)}]\!]$, Ω denotes the index set of the observed entries of $\underline{\mathbf{X}}$ and Ω^c is the complementary set of Ω.

Summary of MTC Note that for the factor matrices, we perform local prox-linear approximation to update them. The benefits of doing this are that each subproblem admits a closed-form solution, and the algorithm is guaranteed to converge [Xu and Yin, 2013].

As an example, consider the update for $\mathbf{A}$. Since the original subproblem is a least squares problem, the prox-linear mapping is known to have a Lipschitz continuous gradient. The smallest Lipschitz constant is the maximum eigenvalue of $(\mathbf{H}_A^{(\tau)T}\mathbf{H}_A^{(\tau)} + \lambda\mathbf{L})$. This is also true for the update of $\mathbf{B}, \mathbf{C}, \mathbf{D}$, where their respective smallest Lipschitz constants are the largest eigenvalues of $\mathbf{H}_B^{(\tau)T}\mathbf{H}_B^{(\tau)}$, $(\mathbf{H}_C^{(\tau)T}\mathbf{H}_C^{(\tau)} + \rho\mathbf{\Gamma}^T\mathbf{\Gamma})$ and $(\mathbf{H}_D^{(\tau)T}\mathbf{H}_D^{(\tau)} + \rho\mathbf{\Gamma}^T\mathbf{\Gamma})$, respectively. In the above, $\mathbf{H}_B^{(\tau)} = \mathbf{A}^{(\tau+1)} \odot \mathbf{C}^{(\tau)} \odot \bar{\mathbf{D}}^{(\tau)}$, $\mathbf{H}_C^{(\tau)} = \mathbf{A}^{(\tau+1)} \odot \mathbf{B}^{(\tau+1)} \odot \bar{\mathbf{D}}^{(\tau)}$ and $\mathbf{H}_D^{(\tau)} = \mathbf{A}^{(\tau+1)} \odot \mathbf{B}^{(\tau+1)} \odot \mathbf{C}^{(\tau+1)}$. Furthermore, the subproblem w.r.t. $\underline{\mathbf{Y}}_\tau$ is also convex with closed-form solution. It follows from [Xu and Yin, 2013](Theorem 2.8):

Proposition 1. *MTC converges to a stationary point of* (6).

The detailed steps of MTC are summarized in Algorithm 1 in the supplementary material. We note that MTC belongs to a family of block coordinate descent (BCD) algorithms. The computational bottleneck is the matricized tensor times Khatri-Rao product (MTTKRP) in the gradient calculation, which dominates the per-iteration complexity. This can be alleviated by exploiting sparsity [Smith *et al.*, 2015]. The overall complexity per iteration of MTC is $\mathcal{O}(IJKSF)$.

5 Adaptive Update

The MTC algorithm leverages all data to complete the missing values, but this is at the expense of high complexity. It requires solving the tensor completion problem exactly at any instant time t. To avoid restarting MTC, we propose an efficient forward-backward propagation based strategy to estimate the missing values and update all factors simultaneously. The resulting algorithm is termed MTC-online.

Let $\underline{\mathbf{X}}_{t+1} \in \mathbb{R}_+^{I\times J\times K\times S_{t+1}}$ be obtained after appending newly observed data $\breve{\underline{\mathbf{X}}}_{t+1} \in \mathbb{R}_+^{I\times J\times K}$ in the fourth dimension, where $S_{t+1} = S_t + 1$. In this case, some of the missing values in $\underline{\mathbf{X}}_t$ have been completed, but new missing values are included. Therefore, the index set Ω_t changes to Ω_{t+1} and the factor matrices should be refined accordingly. Assume that the best low-rank approximation of the missing values is

$$\underline{\mathbf{X}}_{t+1} \approx [\![\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}, \mathbf{D}_{t+1}]\!],$$

where the dimension of $\mathbf{A}_{t+1}$, $\mathbf{B}_{t+1}$ and $\mathbf{C}_{t+1}$ are the same as their counterparts obtained at time t while the number of rows in $\mathbf{D}_{t+1}$ is now expanded to $(S_t + 1)$.

To develop a linear complexity algorithm, it is necessary to impose a proper approximation for the new data. The choice of such an approximation has a major effect on the performance of the resulting algorithm. One such option is to assume slow variations between data from t to $(t + 1)$. This

means that the (i, j, k)-th entry in $\underline{\breve{\mathbf{X}}}_{t+1}$ can be well approximated using the factors from the previous time step. Thus, we have

$$\text{vec}(\underline{\breve{\mathbf{X}}}_{t+1}) \approx (\mathbf{A}_t \odot \mathbf{B}_t \odot \mathbf{C}_t)\big(\mathbf{D}_{t+1}(S_{t+1}, :)\big)^T. \quad (17)$$

We then employ a two-step approach to refine the factor matrices, including one forward-propagation (FP) update and one backward-propagation (BP) update, where in each propagation, only one iteration is employed to refine the factor matrices. Specifically, in the FP step, we update the row of the latent factor corresponding to the new GD. The associated optimization problem is described as

$$\min_{\mathbf{d}\geq 0} \|\text{vec}(\underline{\breve{\mathbf{X}}}_{t+1}) - (\mathbf{A}_t \odot \mathbf{B}_t \odot \mathbf{C}_t)\mathbf{d}\|_2^2, \quad (18)$$

which can be solved optimally using projected gradient descent with light-weight operations. After we obtain $\mathbf{d}$, we perform the BP. We first append $\mathbf{d}$ to $\mathbf{D}_t$ and then complete all missing values via

$$\underline{\mathbf{Y}}_{t+1} = \mathcal{P}_{\Omega^c_{t+1}}(\underline{\mathbf{X}}_{t+1}) + \mathcal{P}_{\Omega_{t+1}}\big(\llbracket \mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t, [\mathbf{D}_t^T, \mathbf{d}]^T \rrbracket\big) \quad (19)$$

where Ω_{t+1} is the index set of the the missing values at time $(t+1)$ and Ω^c_{t+1} is its complement. We refine all factor matrix through one iteration of the gradient updates in (12)-(15) in a coordinate manner.

6 Experiments

We evaluate the MTC's performance in terms of prediction accuracy and scalability. We use the following metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 score. To evaluate scalability, we record the CPU running time. All methods were trained on 2.6 GHz 6-Core Intel Core i7, 16 GB Memory, and 256 GB SSD.

6.1 Experimental Setup

Data We consider the following datasets for evaluation.

- Chicago-Crime dataset [Smith *et al.*, 2017] includes 5,884,473 crime reports in Chicago, ranging from 2001 to 2017. We us biweekly aggregation of the reports resulting in a $77 \times 32 \times 442 \times 10$ tensor of density is 0.219. These are 77 locations, 32 crime categories, 442 GDs, and 10 LDs.
- COVID-19 dataset [Dong *et al.*, 2020] summarizes the COVID-19-19 daily reports from Johns Hopkins University. It records 3 categories, i.e., new cases, deaths, and hospitalization of 51 states in the United States between April 15, 2020, to October 31, 2020. The tensor's size is $51 \times 3 \times 200 \times 8$, and its density is 0.616, where there are 51 states, 3 features, 200 GDs, and 8 LDs.
- Patient-Claims dataset extract patient claims data from a proprietary claims database from 2018. Each data sample consists of a zip code, diagnosis (i.e., disease), date of diagnosis (i.e., GD), and date that claims loaded to the database (i.e., LD). We transform the zip codes to US counties and map the diagnosis codes to 22 diseases according to the ICD-10-CM category[2]. Finally, we count the claims for a given county, disease, GD, and LD and create a 4-way tensor. The tensor's size is $3027 \times 22 \times 52 \times 12$ (over 41m entries) and its density is 0.534, where there are 3027 counties, 22 diseases, 52 GDs and 12 LDs.

[2]https://www.icd10data.com/ICD10CM/Codes

Evaluation Strategy We consider both static and dynamic cases. We select the first S GDs from each dataset for the static case while setting the remaining for the dynamic case. Here, S is set to 421, 168, and 46 for Chicago-Crime, COVID-19, and Patient-Claims, respectively. Therefore, their corresponding numbers of GDs for the dynamic case are 21, 32, and 6, respectively. In the static case, the data corresponding to the first $(S-K+1)$ GDs are fully observed, and the last $(K-1)$ GDs have missing LDs, so the corresponding true values $\{\tilde{z}_{ijs}\}$ in $\underline{\widetilde{\mathbf{Z}}}$ are under-reported. Our target is to predict $\{\tilde{z}_{ijs}\}$ corresponding to the last $(K-1)$ GDs.

Baselines We compare our method against low-rank tensor factorization methods, a neural network tensor completion method, and deep sequential models.

- Naive: We use the received data until t to benchmark the gap between the received data $\underline{\mathbf{Z}}_t$ and the actual data $\underline{\widetilde{\mathbf{Z}}}_t$.
- SDF (3-way): This is a 3-way tensor completion approach using the Structured Data Fusion (SDF) framework in Tensorlab [Vervliet *et al.*, 2016]. We complete the missing data of $\underline{\mathbf{Z}}_t$ and use the learned model as our estimate for $\underline{\widetilde{\mathbf{Z}}}_t$.
- SDF (4-way): We consider a variation of MTC for comparison. This is a 4-way tensor completion approach using SDF/Tensorlab [Vervliet *et al.*, 2016] without smoothness and Laplacian regularization.
- COSTCO [Liu *et al.*, 2019]: This is a convolutional neural network-based model for tensor completion.
- Auto Regressive Integrated Moving Average (ARIMA): We train distinct ARIMA models for different pairs of location and attributes/signals.
- Long Short Term Memory (LSTM) network: We train LSTM to predict the values for the subsequent GDs.

The implementation details are provided in the appendix.

6.2 Results

Table 1 summarizes the results of the static case. The Naive method performs poorly, which indicates that the time-delayed data differ significantly from the ultimate ground truth. The poor performance of SDF (3-way) also verifies such an observation. The 3-way completion approach can complete the missing values of the latest slabs of $\underline{\mathbf{Z}}_t$ but cannot correct the under-reported cases. MTC has consistently lower MAE and RMSE than the other methods in the three datasets. Note that in the Patient-Claims dataset, which is the most challenging one, MTC performs the best, and its performance is followed by the SDF (4-way) algorithm. Since both MTC and SDF (4-way) are 4-way tensor completion approaches, the performance improvement of MTC is achieved by the well-designed prox-linear gradient approach and the regularization terms. We empirically observed that COSTCO was more susceptible to overfitting and did not perform well.

Accuracy In Table 2, we calculate the RMSE, MAE, and R^2 for each method in the dynamic case. The values were averaged over all GDs in each dataset. It can be seen that the proposed methods achieve the best RMSE and R^2 in the three datasets. We observed that COSTCO could easily overfit the

Method	Patient-Claims			Covid-19			Chicago-Crime		
	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
MTC	**220.4**	**29.7**	**0.997**	**74.2**	**26.0**	**0.986**	**1.42**	0.57	**0.983**
Naive	1113.5	107.2	0.896	290.1	97.1	0.559	4.98	1.24	0.594
SDF (3-way)	1149.1	146.2	0.905	291.9	105.0	0.551	4.70	1.24	0.648
SDF (4-way)	278.7	31.5	0.995	101.7	31.8	0.974	1.46	**0.55**	0.981
COSTCO	633.5	96.4	0.972	203.1	99.3	0.877	2.43	0.66	0.908
ARIMA	524.2	66.5	0.981	283.2	99.8	0.780	3.63	1.55	0.915
LSTM	400.6	58.5	0.989	343.9	111.8	0.736	3.65	1.41	0.876

Table 1: Performance comparison in static case.

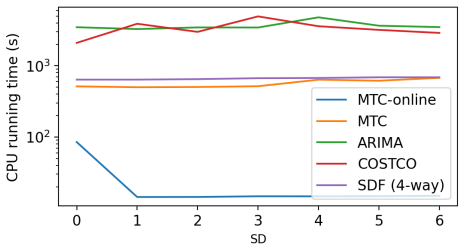

Figure 2: Complexity comparison.

Method	RMSE	MAE	R^2
Patient-Claims dataset			
MTC	$\mathbf{237.8 \pm 40.3}$	32.5 ± 3.0	$\mathbf{0.996 \pm 0.001}$
MTC-online	238.5 ± 37.5	$\mathbf{32.2 \pm 2.8}$	$\mathbf{0.996 \pm 0.001}$
SDF (4-way)	253.4 ± 59.5	34.6 ± 4.7	0.996 ± 0.002
Naive	$1,017.5 \pm 69.3$	99.8 ± 6.3	0.912 ± 0.012
SDF (3-way)	$1,052.8 \pm 89.1$	131.0 ± 15.3	0.904 ± 0.015
COSTCO	580.5 ± 37.2	87.9 ± 5.3	0.977 ± 0.003
ARIMA	553.1 ± 31.5	74.6 ± 5.2	0.979 ± 0.002
LSTM	692.1 ± 232.2	98.7 ± 31.8	0.952 ± 0.039
Covid-19 dataset			
MTC	$\mathbf{105.4 \pm 26.7}$	$\mathbf{34.9 \pm 8.2}$	$\mathbf{0.983 \pm 0.005}$
MTC-online	112.2 ± 23.0	35.4 ± 6.5	0.980 ± 0.009
SDF (4-way)	147.5 ± 104.4	42.5 ± 14.6	0.974 ± 0.046
Naive	411.4 ± 98.13	139.2 ± 32.5	0.450 ± 0.123
SDF (3-way)	525.2 ± 218.6	179.0 ± 61.2	0.391 ± 0.187
COSTCO	310.0 ± 171.0	98.7 ± 35.4	0.880 ± 0.082
ARIMA	399.2 ± 134.7	137.0 ± 33.1	0.597 ± 0.256
LSTM	386.4 ± 96.2	129.5 ± 19.4	0.791 ± 0.065
Chicago-Crime dataset			
MTC	$\mathbf{1.45 \pm 0.09}$	0.58 ± 0.02	$\mathbf{0.985 \pm 0.002}$
MTC-online	1.47 ± 0.10	0.59 ± 0.02	0.984 ± 0.002
SDF (4-way)	1.50 ± 0.11	$\mathbf{0.56 \pm 0.02}$	0.984 ± 0.003
Naive	5.46 ± 0.48	1.30 ± 0.08	0.554 ± 0.078
SDF (3-way)	5.12 ± 0.46	1.30 ± 0.07	0.626 ± 0.058
COSTCO	2.63 ± 0.38	0.73 ± 0.10	0.947 ± 0.014
ARIMA	3.77 ± 0.43	1.42 ± 0.07	0.893 ± 0.032
LSTM	4.10 ± 0.50	1.43 ± 0.07	0.848 ± 0.034

Table 2: Performance comparison in dynamic case: (mean $\pm$ std) of each metric, calculated over all the GDs in each dataset.

data, so it did not perform well. ARIMA, LSTM, and SDF (3-way) did not use the LD information, thus failing to achieve satisfactory results. This indicated that using an additional mode – LD can help solve the data delayed issue in spatio-temporal tensors. In the Patient-Claims dataset, MTC-online has a smaller MAE than MTC. In all other cases, MTC has slightly better performance than MTC-online. However, this is at the expense of high complexity, since MTC needs to be restarted once new data received. This is true for the other baselines as well. On the other hand, MTC-online updates all factors in an inexact way with a single proximal gradient update, resulting in very low-complexity.

Scalability Figure 2 shows the CPU running time of MTC-online, MTC, ARIMA, and SDF (4-way) on the Patient-Claims dataset. We did not include LSTM for comparison because it had a much higher time complexity for training. Figure 2 shows COSTCO and ARIMA has the highest complexity – the CPU time for both methods was close to one hour. ARIMA is slow we need to train a separate model for each location. SDF (4-way) and MTC have similar running

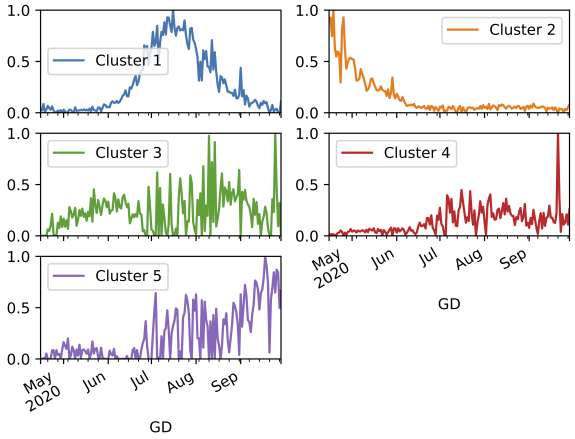

Figure 3: Latent components in GD mode of Covid-19 dataset.

time. MTC-online is an order of magnitude faster than the best baseline due to the approximate gradient updates.

Model interpretation Figure 3 shows the latent components corresponding of the GD mode in the COVID-19 dataset. Figure 3 shows that Cluster 2 consisted of the states hit by Covid-19 in the first wave in the United States. The Cluster-1 had about two months delay, exactly for the second wave, including FL, CA, TX, AZ, etc. The five clusters have similar exponential decay trend. Interestingly, Cluster 4 had the lowest loading speed, meaning that data corresponding to this cluster took a longer time to be received than those in the other components. Four out of the top five counties in Cluster 4 were from California, where the laws on processing/collecting claims data are more strict than the other states. More results are reported in the supplement.

7 Conclusion

This paper studies the time-delayed spatio-temporal tensor data, which corresponds to many real-world data aggregation applications. We formulated the problem as a multi-version tensor completion (MTC) problem by introducing an extra mode to capture the data updates. The final estimate of the target tensor is estimated by first imputing the missing values and then marginalizing over the extra mode. We proposed static and online version of MTC algorithms to tackle this problem. The proposed methods employ a graph regularization to reserve the location correlations in latent subspace and smoothness. The experimental results on several real datasets have demonstrated the advantages of the proposed methods.

References

[Acar *et al.*, 2011] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[Almutairi *et al.*, 2017] Faisal M Almutairi, Nicholas D Sidiropoulos, and George Karypis. Context-aware recommendation-based learning analytics using tensor and coupled matrix factorization. *IEEE Journal of Selected Topics in Signal Processing*, 11(5):729–741, 2017.

[Araujo *et al.*, 2017] Miguel Araujo, Pedro Ritiro, and Christos Faloutsos. Tensorcast: Forecasting with context using coupled tensors. In *IEEE ICDM*, pages 71–80, 2017.

[Bahadori *et al.*, 2014] Mohammad Taha Bahadori, Qi (Rose) Yu, and Yan Liu. Fast multivariate spatio-temporal analysis via low rank tensor learning. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *NeurIPS*, volume 27, 2014.

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2(1):183–202, 2009.

[Dong *et al.*, 2020] E. Dong, H. Du, and L. Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet Infect. Diseas.*, 20(5):533–534, 2020.

[Fang *et al.*, 2020] Shikai Fang, Zheng Wang, Zhimeng Pan, Ji Liu, and Shandian Zhe. Streaming probabilistic deep tensor factorization. *arXiv:2007.07367*, 2020.

[Fu *et al.*, 2020] Xiao Fu, Nico Vervliet, Lieven De Lathauwer, Kejun Huang, and Nicolas Gillis. Computing large-scale matrix and tensor decomposition with structured factors: A unified nonconvex optimization perspective. *IEEE Signal Processing Magazine*, 37(5):78–94, 2020.

[Harshman, 1970] Richard A Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers Phonetics*, 1970.

[Kargas *et al.*, 2021] Nikos Kargas, Cheng Qian, Nicholas D Sidiropoulos, Cao Xiao, Lucas M Glass, and Jimeng Sun. STELAR: Spatio-temporal tensor factorization with latent epidemiological regularization. In *Proc. of AAAI*, 2021.

[Lacroix *et al.*, 2018] Timothee Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In Jennifer Dy and Andreas Krause, editors, *Proc. of ICML*, volume 80, pages 2863–2872. PMLR, 10–15 Jul 2018.

[Liu *et al.*, 2012] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Trans. PAMI*, 35(1):208–220, 2012.

[Liu *et al.*, 2018] B. Liu, L. He, Y. Li, S. Zhe, and Z. Xu. Neuralcp: Bayesian multiway data analysis with neural tensor decomposition. *Cognitive Computation*, 10(6):1051–1061, 2018.

[Liu *et al.*, 2019] H. Liu, Y. Li, M. Tsang, and Y. Liu. Costco: A neural tensor completion model for sparse tensors. In *Proc. of KDD*, page 324–334, New York, NY, USA, 2019.

[Mardani *et al.*, 2015] Morteza Mardani, Gonzalo Mateos, and Georgios B Giannakis. Subspace learning and imputation for streaming big data matrices and tensors. *IEEE Trans. Signal Processing*, 63(10):2663–2677, 2015.

[Nion and Sidiropoulos, 2009] Dimitri Nion and Nicholas D Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *IEEE Trans. Signal Processing*, 57(6):2299–2310, 2009.

[Sidiropoulos *et al.*, 2017] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Trans. Signal Processing*, 65(13):3551–3582, 2017.

[Smith *et al.*, 2015] S. Smith, N. Ravindran, N. D. Sidiropoulos, and G. Karypis. Splatt: Efficient and parallel sparse tensor-matrix multiplication. In *Proc. of IEEE PDPS*, pages 61–70, 2015.

[Smith *et al.*, 2017] S. Smith, J. W. Choi, J. Li, R. Vuduc, J. Park, X. Liu, and G. Karypis. FROSTT: The formidable repository of open sparse tensors and tools. *frostt.io*, 2017.

[Song *et al.*, 2017] Q. Song, X. Huang, H. Ge, J. Caverlee, and X. Hu. Multi-aspect streaming tensor completion. In *Proc. of KDD*, page 435–443, New York, NY, USA, 2017.

[Sun *et al.*, 2006] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proc. of KDD*, page 374–383, New York, NY, USA, 2006.

[Sun *et al.*, 2008] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM TKDD*, 2(3):1–37, 2008.

[Tucker, 1966] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[Vervliet *et al.*, 2016] Nico Vervliet, Otto Debals, Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. Tensorlab 3.0. *www. tensorlab. net*, 2016.

[Wang *et al.*, 2015] Y. Wang, R. Chen, J. Ghosh, J. C. Denny, A. Kho, Y. Chen, B. A. Malin, and J. Sun. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *Proc. of KDD*, page 1265–1274, New York, NY, USA, 2015.

[Xu and Yin, 2013] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imag. Sci.*, 6(3):1758–1789, 2013.

[Yuan and Zhang, 2016] Ming Yuan and Cun-Hui Zhang. On tensor completion via nuclear norm minimization. *FoCM*, 16(4):1031–1068, 2016.

[Zhou *et al.*, 2016] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating online cp decompositions for higher order tensors. In *Proc. of KDD*, page 1375–1384, New York, NY, USA, 2016.