Multi-Aspect Streaming Tensor Completion

Qingquan Song,[†] Xiao Huang,[†] Hancheng Ge,[†] James Caverlee,[†] Xia Hu[†,‡]
[†]Department of Computer Science and Engineering, Texas A&M University
[‡]Center for Remote Health Technologies and Systems, Texas A&M Engineering Experiment Station
{song_3134,xhuang,hge,caverlee,xiahu}@tamu.edu

ABSTRACT

Tensor completion has become an effective computational tool in many real-world data-driven applications. Beyond traditional static setting, with the increasing popularity of high velocity streaming data, it requires efficient online processing without reconstructing the whole model from scratch. Existing work on streaming tensor completion is usually built upon the assumption that tensors only grow in one mode. Unfortunately, the assumption does not hold in many real-world situations in which tensors may grow in multiple modes, i.e., multi-aspect streaming tensors. Efficiently modeling and completing these incremental tensors without sacrificing its effectiveness remains a challenging task due to the uncertainty of tensor mode changes and complex data structure of multi-aspect streaming tensors. To bridge this gap, we propose a Multi-Aspect Streaming Tensor completion framework (MAST) based on CANDECOMP/PARAFAC (CP) decomposition to track the subspace of general incremental tensors for completion. In addition, we investigate a special situation where time is one mode of the tensors, and leverage its extra structure information to improve the general framework towards higher effectiveness. Experimental results on four datasets collected from various real-world applications demonstrate the effectiveness and efficiency of the proposed framework.

1 INTRODUCTION

Tensors are multidimensional or N-way extensions of matrices. Given the increasing popularity of multi-modal information, data instances are often indexed with multiple variables and naturally derive high-order tensor objects [20]. Tensor completion, which aims at filling the missing entries of partially observed tensors, has become an effective computational tool in many real-world data mining applications such as social network analysis [7, 11], recommender systems [25, 26], image recovery [17], and clinical data analysis [9, 36].

Beyond traditional static setting, real-world applications are often imbued with high velocity streaming data. For example, due to the popularity of online information systems, large amounts of new data are produced rapidly. Facebook users update 684,478 messages and Twitter users post over 100,000 tweets every single minute.[1] With the large number of newly created data instances and features, model reconstruction from scratch is computationally expensive due to high time and space complexity [8]. The fast-evolving data motivates us to investigate dynamic tensor completion methods to timely model and incorporate newly emerging patterns.

[1]http://mashable.com/2012/06/22/data-created-every-minute/

KDD'17, August 13–17, 2017, Halifax, NS, Canada.
© 2017 ACM. 978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098007

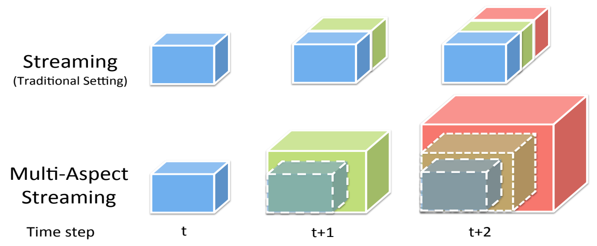


Figure 1: An illustration of traditional streaming tensors and multi-aspect streaming tensors.

Existing work has been conducted on developing dynamic tensor completion methods based on a widely used assumption that tensors will be developed in one mode (or dimension). Following this assumption, online completion methods have been proposed based on CP decomposition [15, 20]. The key idea is that, by iteratively updating the decompositions of static modes with partial decomposition of the incremental mode, the developed methods could efficiently track the low-rank subspace over time, and achieve comparable performance with batch learning methods for data imputation. The developed methods have shown effectiveness in dynamic data analysis applications such as video streams [15].

Unfortunately, in many real-world applications, a tensor may develop in multiple dimensions and traditional assumption does not hold. For example, given a dynamic tensor in recommender systems structured as *user* × *movie* × *actor*, the number of registered users, movies, and actors may all increase as time goes. This incremental property gives rise to a new streaming pattern, which we call multi-aspect streaming. Figure 1 depicts traditional streaming tensors and its generalized multi-aspect streaming tensors counterpart. As we can observe from the figure, from time t to $t + 2$, each mode of the multi-aspect streaming tensor increases while only one mode of the streaming tensor grows. To deal with the tensors with distinct data characteristics, in this paper, we propose to study the novel problem of effective multi-aspect streaming tensor completion.

Multi-aspect streaming tensor completion is a challenging task due to the following reasons. First, coordinating multidimensional dynamics is difficult given the uncertainty of tensor mode changes. It is too arbitrary to fix partial structure of a certain mode without accessing the data for an update. Second, the incremental part of multi-aspect streaming data may not be a complete tensor. This is different from streaming tensors in which the incremental part is well structured and can be directly decomposed. Third, the multi-aspect streaming pattern leads to much higher time and space complexity. Although some distributed and parallel algorithms could partially address the scalability issue [4, 24, 28], the problem of non-adaptability still exists, meaning that the completion process of new data cannot directly benefit from the existing model without complete model reconstruction.

To tackle the aforementioned challenges, in this paper, we propose to investigate how to complete multi-aspect streaming tensors based on CP decomposition. Specifically, we mainly study: (1) How to build up an updatable framework based on CP decomposition for efficiently modeling multi-aspect streaming tensors? (2) How to effectively capture the low-rank subspace for completion purpose? Through answering the two questions, we propose a general Multi-Aspect Streaming Tensor completion algorithm (MAST). The main contributions are summarized as follows:

- Formally define the problem of multi-aspect streaming tensor completion;
- Propose a CP-based general algorithm MAST for multi-aspect streaming tensor completion;
- Propose a modified model T-MAST based on MAST for temporal multi-aspect streaming tensor completion, which is a special case of the general problem; and
- Empirically validate the effectiveness and efficiency of the proposed models on four real-world datasets with different real-world applications.

2 PRELIMINARIES

Notations and definitions in this paper are presented as follows. **Notations**: An N^{th}-order tensor is an N-way array, also known as N-dimensional or N-mode tensor. In this paper, tensors, matrices, vectors are denoted by Euler script letters (e.g., $\mathcal{X}$), boldface uppercase letters (e.g., $\mathbf{A}$), and boldface lowercase letters (e.g., $\mathbf{a}$), respectively. An entry of tensor $\mathcal{X}$ indexed by $[i_1^t, \ldots, i_N^t]$ is denoted as $\mathcal{X}[i_1^t, \ldots, i_N^t]$. We use $\mathbf{A}^{-1}$, $\mathbf{A}^\top$, and $\|\mathbf{A}\|_F$ to denote the transpose, inverse, and Frobenius norm of matrix $\mathbf{A}$, respectively. Let $[\![\cdot]\!]$ denote the Kruskal operator. Notations $\odot$ and $\circledast$ are used to represent the Khatri-Rao product and Hadamard product, respectively. The main symbols and operations are listed in Table 1.

Definition 1 (CP Decomposition). Given an N^{th}-order tensor $\mathcal{X}$, its CP decomposition is an approximation of N loading matrices $\mathbf{A}_n \in \mathbb{R}^{I_n \times R}$, $n = 1, \ldots, N$, such that,

$$\mathcal{X} \approx [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!], \tag{1}$$

where R is a positive integer denoting an upper bound of the rank of $\mathcal{X}$. We can further unfold $\mathcal{X}$ along its n^{th} mode as follows,

$$\begin{aligned}\mathcal{X} \overset{\text{unfold}}{\Longrightarrow} \quad \mathbf{X}_{(n)} &\approx \mathbf{A}_n(\mathbf{A}_N \odot \ldots \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \ldots \odot \mathbf{A}_1)^\top \\ &= \mathbf{A}_n[(\mathbf{A}_k)^{\odot k \neq n}]^\top.\end{aligned} \tag{2}$$

Definition 2 (Coupled Tensor). If two tensors share one or more modes, then they are coupled with each other and called coupled tensors. For example, in recommender systems, two tensors structured as $user \times movie \times time$ and $movie \times actor \times director$ are coupled on the *movie* mode.

Definition 3 (Tensor Sequence). A sequence of N^{th}-order tensors $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(T)}, \ldots$ is called a tensor sequence denoted as $\{\mathcal{X}^{(T)}\}$, where each $\mathcal{X}^{(T)} \in \mathbb{R}^{I_1^T \times I_2^T \times \ldots \times I_N^T}$, $T \in \mathbb{Z}^+$.

Definition 4 (Multi-aspect streaming Tensor Sequence). A sequence of N^{th}-order tensors $\{\mathcal{X}^{(T)}\}$ is called multi-aspect streaming

Notations	Definitions
$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$	N^{th}-order tensor
$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (\Pi_{i \neq n}^N I_i)}$	Mode-n unfolding matrix of tensor $\mathcal{X}$
$[\![\cdot]\!]$	Kruskal operator, e.g., $\mathcal{X} \approx [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$
$\odot$	Khatri-Rao product
$\circledast$	Hadamard product
$(\mathbf{A}_k)^{\odot k \neq n}$	$\mathbf{A}_N \odot \ldots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \ldots \odot \mathbf{A}_1$
$(\mathbf{A}_k)^{\circledast k \neq n}$	$\mathbf{A}_N \circledast \ldots \circledast \mathbf{A}_{n+1} \circledast \mathbf{A}_{n-1} \circledast \ldots \circledast \mathbf{A}_1$

Table 1: Main symbols and operations.

tensor sequence if for any $T \in \mathbb{Z}^+$, $\mathcal{X}^{(T-1)}$ is the sub-tensor of $\mathcal{X}^{(T)}$, denoted as $\mathcal{X}^{(T-1)} \subseteq \mathcal{X}^{(T)}$. T increases with time, and $\mathcal{X}^{(T)}$ is the snapshot tensor of this sequence taken at time T.

In fact, any tensor sequence $\{\mathcal{X}^{(T)}\}$ can be modeled as a multi-aspect streaming one by adding an additional mode, i.e., defining an $(N+1)^{\text{th}}$-order new tensor sequence $\{\mathcal{Y}^{(T)}\}$, where $\mathcal{Y}^{(T)} \in \mathbb{R}^{J_1^T \times J_2^T \times \ldots \times J_N^T \times T}$, $J_n = \max_t\{I_n^t\}$. Entries of $\mathcal{Y}^{(T)}$ are defined as:

$$\mathcal{Y}^{(T)}[i_1^t, \ldots, i_N^t, t] = \begin{cases} \mathcal{X}^{(t)}[i_1^t, \ldots, i_N^t], & \text{if } 1 \leq i_n^t \leq I_N^t, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $1 \leq t \leq T$, and then $\mathcal{Y}^{(T-1)} \subseteq \mathcal{Y}^{(T)}$ is satisfied.

Based on the terminologies, the multi-aspect streaming tensor completion problem can be formally defined as follows.

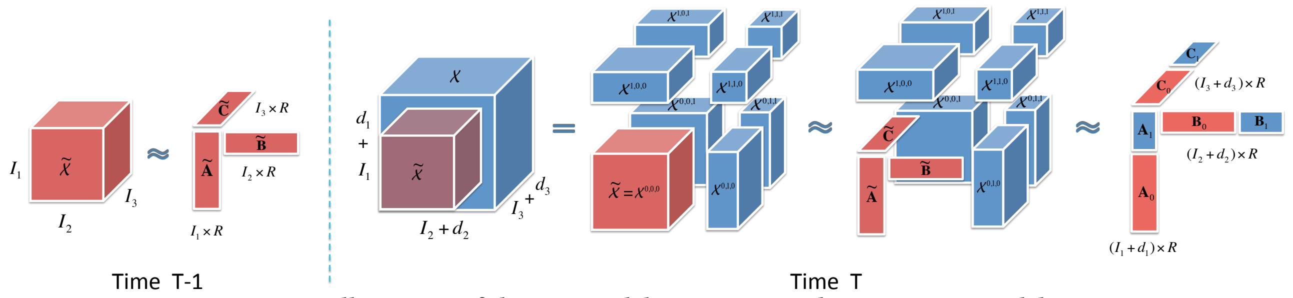
Given a multi-aspect streaming tensor sequence $\{\mathcal{X}^{(T)}\}$ with missing entries, we aim at recovering the missing data in current snapshot $\mathcal{X}^{(T)}$. Since $\mathcal{X}^{(T-1)} \subseteq \mathcal{X}^{(T)}$, and we have recovered $\mathcal{X}^{(T-1)}$ in previous time step, the problem is equivalent to completing the relative complement of $\mathcal{X}^{(T-1)}$ in $\mathcal{X}^{(T)}$ which is denoted as $\mathcal{X}^{(T)} \backslash \mathcal{X}^{(T-1)}$.

3 MULTI-ASPECT STREAMING TENSOR COMPLETION

In this section, we introduce the proposed framework MAST to solve the general multi-aspect streaming completion problem. By taking advantage of the learned model in previous time step, MAST can accelerate the completion process while preserving comparable completion accuracy to traditional batch learning methods. To achieve this, we begin with designing a dynamic tensor decomposition (DTD) algorithm to model the incremental pattern of multi-aspect streaming tensor sequence and track its low-rank subspace. Then, based on the DTD algorithm, we conduct the completion task via the nuclear norm constraint. At the end, we discuss a special case in which "time" is one mode of the tensors and further improve our framework for a better completion effectiveness.

3.1 Dynamic Tensor Decomposition

CP decomposition is a widely used tool for tensor completion [11, 17, 18]. By decomposing a tensor into multiple low-rank matrices using its observed entries, the underlying low-rank subspace structure can be tracked for filling the missing entries. There are two intuitive solutions for our main problem. At each time step, we can either reconstruct the batch CP model for the entire snapshot, or decompose $\mathcal{X}^{(T)} \backslash \mathcal{X}^{(T-1)}$ only. The former solution is both time and space consuming since the snapshots are increasing dramatically. In contrast, the latter fails to exploit the information in $\mathcal{X}^{(T-1)}$ to help achieve desirable imputation.



I_1
$\widetilde{\mathcal{X}}$
I_2
I_3
≈
$\widetilde{C}$ $I_3 \times R$
$\widetilde{B}$ $I_2 \times R$
$\widetilde{A}$ $I_1 \times R$
Time T-1
$d_1 + I_1$
$\mathcal{X}$
$\widetilde{\mathcal{X}}$
$I_2 + d_2$
$I_3 + d_3$
=
$\mathcal{X}^{1,0,1}$ $\mathcal{X}^{1,1,1}$ $\mathcal{X}^{1,0,0}$ $\mathcal{X}^{1,1,0}$ $\mathcal{X}^{0,0,1}$ $\mathcal{X}^{0,1,1}$ $\mathcal{X}^{0,1,0}$
$\widetilde{\mathcal{X}} = \mathcal{X}^{0,0,0}$
≈
$\widetilde{C}$ $\widetilde{B}$ $\widetilde{A}$
≈
C_1 C_0 $(I_3 + d_3) \times R$
A_1 B_0 B_1 $(I_2 + d_2) \times R$
A_0 $(I_1 + d_1) \times R$
Time T

Figure 2: An illustration of the proposed dynamic tensor decomposition model DTD.

To accelerate completion process while preserving effectiveness, a dynamic CP decomposition method DTD is proposed. It is based on two ideas. First, CP decomposition enjoys the good partitioning property which means if $\mathcal{X}^{(T)}$ is approximated by $[\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$, then its sub-tensor $\mathcal{X}^{(T-1)}$ could be approximated by $[\![\widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_N]\!]$ where $\widetilde{\mathbf{A}}_n$ is a sub-matrix of $\mathbf{A}_n$, for $n = 1, \ldots, N$. This enables us to partition the current snapshot $\mathcal{X}^{(T)}$ to take advantage of the foregone acquired decomposition of its sub-tensor $\mathcal{X}^{(T-1)}$. Second, partial calculation complexity of decomposing $\mathcal{X}^{(T)}$ can be reduced one order lower by substituting acquired decompositions for $\mathcal{X}^{(T-1)}$. These two ideas are separately used in following two steps of our proposed method for information preservation and optimization acceleration, respectively.

3.1.1 Tensor partition and substitution. For the ease of presentation, we use a third-order tensor with steady growth on all three modes as an example. It is straightforward to extend to general N^{th}-order tensors. We use $\widetilde{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\mathcal{X} \in \mathbb{R}^{(I_1+d_1)\times(I_2+d_2)\times(I_3+d_3)}$ to represent two consecutive snapshots $\mathcal{X}^{(T-1)}$ and $\mathcal{X}^{(T)}$, $(T \in \mathbb{Z}^+)$, respectively. $\widetilde{\mathcal{X}} = 0$ if $T = 0$. We now introduce the details.

Let $\mathbf{A} \in \mathbb{R}^{(I_1+d_1)\times R}, \mathbf{B} \in \mathbb{R}^{(I_2+d_2)\times R}, \mathbf{C} \in \mathbb{R}^{(I_3+d_3)\times R}$ be the CP factor matrices of $\mathcal{X}$, the CP loss function under gaussian noise is:

$$\mathcal{L}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2. \tag{4}$$

After partitioning $\mathcal{X}$ into eight sub-tensors according to $\widetilde{\mathcal{X}}$, we use a binary tuple $(i, j, k) \in \{0, 1\}^3 \triangleq \Theta$ to denote these eight sub-tensors as shown in Figure 2, with $\widetilde{\mathcal{X}} = \mathcal{X}^{0,0,0}$. Each pair of adjacent sub-tensors are coupled with each other. Based on the partitioning property of CP decomposition, each sub-tensor of $\mathcal{X}$ could also be approximated via the sub-matrices of A, B and C, i.e., $\mathcal{X}^{i,j,k} \approx [\![\mathbf{A}_i, \mathbf{B}_j, \mathbf{C}_k]\!]$. $\mathbf{A}_0 \in \mathbb{R}^{I_1 \times R}$ and $\mathbf{A}_1 \in \mathbb{R}^{d_1 \times R}$ are the partitioned sub-matrices of A, i.e., $\mathbf{A}^\top = [\mathbf{A}_0^\top, \mathbf{A}_1^\top]$. Similarly to $\mathbf{B}_j$ and $\mathbf{C}_k$. Then Equation (4) could be rewritten as follows.

$$\begin{aligned}\mathcal{L}(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= \sum_{(i,j,k)\in\Theta} \|\mathcal{X}^{i,j,k} - [\![\mathbf{A}_i, \mathbf{B}_j, \mathbf{C}_k]\!]\|_F^2 \\ &= \|\mathcal{X}^{0,0,0} - [\![\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0]\!]\|_F^2 + \mathcal{L}_0,\end{aligned} \tag{5}$$

where $\mathcal{L}_0 \triangleq \sum_{(i,j,k)\in\Theta\backslash(0,0,0)} \|\mathcal{X}^{i,j,k} - [\![\mathbf{A}_i, \mathbf{B}_j, \mathbf{C}_k]\!]\|_F^2$.

Let $[\![\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}]\!]$ denote the CP decomposition of $\widetilde{\mathcal{X}}$ obtained at time $T - 1$. It represents the low-rank subspace which preserves the information of former snapshot. Based on this, we can approximately replace $\mathcal{X}^{0,0,0}$ by $[\![\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}]\!]$ and get the loss function as follows:

$$\mathcal{L}(\mathbf{A}, \mathbf{B}, \mathbf{C}) \approx \mu \|[\![\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}]\!] - [\![\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0]\!]\|_F^2 + \mathcal{L}_0, \tag{6}$$

where weight $\mu \in [0, 1]$ is the forgetting factor [20] used to alleviate the influence of the previous decomposition error. If $\mu = 1$, then $[\![\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}]\!]$ is considered as a perfect decomposition of $\widetilde{\mathcal{X}}$. If $\mu = 0$, then the former-step information is considered as independent to current decomposition. As our final goal is to complete this tensor sequence, this inequality $(0 < \mu < 1)$ is quite effective to alleviate the substitution errors especially when the percentage of missing data is high. In other words, this forgetting factor degrades the status of old data and upgrades the power of newly arrived data, so that we can more effectively incorporate newly emerged information to update the foregone decomposition and ensure the substitution error not continuously accumulated.

3.1.2 Alternating least square updating. The optimization is based on alternating least squares (ALS), for its fast and easy implementation with high accuracy [16, 38]. The update rule for $\mathbf{A}_0$ is:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{A}_0} &= -2\mu(\widetilde{\mathbf{A}}(\widetilde{\mathbf{C}} \odot \widetilde{\mathbf{B}})^\top(\mathbf{C}_0 \odot \mathbf{B}_0) - \mathbf{A}_0(\mathbf{C}_0 \odot \mathbf{B}_0)^\top(\mathbf{C}_0 \odot \mathbf{B}_0)) \\ &\quad - 2\sum_{j,k}(\mathbf{X}_{(1)}^{0,j,k}(\mathbf{C}_k \odot \mathbf{B}_j) - \mathbf{A}_0(\mathbf{C}_0 \odot \mathbf{B}_0)^\top(\mathbf{C}_0 \odot \mathbf{B}_0)),\end{aligned}$$

$$\mathbf{A}_0 \leftarrow \frac{\mu\widetilde{\mathbf{A}}\left[(\widetilde{\mathbf{C}}^\top\mathbf{C}_0) \circledast (\widetilde{\mathbf{B}}^\top\mathbf{B}_0)\right] + \sum_{(j,k)\neq(0,0)} \mathbf{X}_{(1)}^{0,j,k}(\mathbf{C}_k \odot \mathbf{B}_j)}{(\sum_{k=0}^1 \mathbf{C}_k^\top\mathbf{C}_k) \circledast (\sum_{j=0}^1 \mathbf{B}_j^\top\mathbf{B}_j)}. \tag{7}$$

The main difference between the above update rule and static CP-ALS method is the first term of the numerator. Following the property of Khatri-Rao product, the space and time complexity of calculating the Khatri-Rao products $\odot$ in equation $(\widetilde{\mathbf{C}} \odot \widetilde{\mathbf{B}})^\top(\mathbf{C}_0 \odot \mathbf{B}_0)$ can be reduced using the element-wise Hadamard product, i.e., $\widetilde{\mathbf{C}}^\top\mathbf{C}_0 \circledast \widetilde{\mathbf{B}}^\top\mathbf{B}_0$. Hence, by substituting $\mathcal{X}^{0,0,0}$ with $[\![\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}]\!]$, this term is changed from $\mathbf{X}_{(1)}^{0,0,0}(\mathbf{C}_0 \odot \mathbf{B}_0)$ to $\mu\widetilde{\mathbf{A}}\left[(\widetilde{\mathbf{C}}^\top\mathbf{C}_0) \circledast (\widetilde{\mathbf{B}}^\top\mathbf{B}_0)\right]$ leading to a reduction of time complexity from $O(RI_1I_2I_3)$ to $O(R^2(I_1+I_2+I_3))$.

The update rules of the incremental matrix $\mathbf{A}_1$ can be derived as:

$$\mathbf{A}_1 \leftarrow \frac{\sum_{j,k} \mathbf{X}_{(1)}^{1,j,k}(\mathbf{C}_k \odot \mathbf{B}_j)}{(\sum_{k=0}^1 \mathbf{C}_k^\top\mathbf{C}_k) \circledast (\sum_{j=0}^1 \mathbf{B}_j^\top\mathbf{B}_j)}. \tag{8}$$

Similar update rules for matrices $\mathbf{B}_0$, $\mathbf{C}_0$, $\mathbf{B}_1$, and $\mathbf{C}_1$ can be easily derived. The proposed DTD model enjoys several nice properties. First, it is capable of handling any-mode-change dynamic pattern. Second, it has a fast updating process for the purpose of scalability. Third, it well preserves the information acquired from old data as well as captures the new data characteristics through the updated process. Besides, the idea of partitioning also benefits methods in solving large-scale tensor decomposition problems [4, 24, 28].

3.2 Proposed Completion Framework - MAST

The proposed DTD method balances the trade-off between efficiency and effectiveness in modeling the multi-aspect streaming pattern. As CP decomposition is an effective way in addressing static tensor completion problem, the DTD model also paves the way of solving the multi-aspect streaming tensor completion problem. In this section, we show how MAST is applied to this problem. We start with introducing the nuclear norm based method on low-rank tensor completion problem in batch setting and then combine it with the DTD model for completing multi-aspect streaming tensor sequences.

3.2.1 Low-rank tensor completion (LRTC). Generalized from matrix completion problem, the LRTC problem can be formulated as an equivalent rank-minimization problem as follows [9]:

$$\underset{\mathcal{X}}{\text{minimize}}\ rank(\mathcal{X}) \quad \text{subject to } \boldsymbol{\Omega} \circledast \mathcal{X} = \mathcal{T}, \tag{9}$$

where $\mathcal{X}$ denotes the complete tensor, $\mathcal{T}$ denotes the practical observations of $\mathcal{X}$. $\boldsymbol{\Omega}$ is a binary tensor with the same size as $\mathcal{X}$ indicating whether each corresponding entry in $\mathcal{X}$ is observed or not.

As calculating the tensor rank is an NP-hard problem [12], one way is to relax the tensor rank by replacing it with the summation of the nuclear norm of its factorized matrices [17, 18]. The modified relaxation objective function is described as follows:

$$\begin{aligned} \underset{\mathcal{X},\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_N}{\text{minimize}} \quad & \sum_{n=1}^{N} \alpha_n \|\mathbf{A}_n\|_* + \|\mathcal{X} - [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]\| \\ \text{subject to} \quad & \boldsymbol{\Omega} \circledast \mathcal{X} = \mathcal{T}, \end{aligned} \tag{10}$$

where $\{\alpha_n\}$ are trade-offs to balance the significance of each mode.

3.2.2 Combine LRTC with DTD. Since multi-aspect streaming tensor completion problem can be treated as a dynamic LRTC problem, a natural solution is to combine DTD with LRTC approach. By extending Equation (6) to N^{th}-order tensor and combining it with Equation (10), the proposed loss function for N^{th}-order multi-aspect streaming tensor completion problem is formulated as:

$$\begin{aligned} \mathcal{L} = & \sum_{(i_1,\ldots,i_N)\in\Theta_N\setminus\{0\}^N} \|\mathcal{X}^{(i_1,\ldots,i_N)} - [\![\mathbf{A}_1^{(i_1)}, \ldots, \mathbf{A}_N^{(i_N)}]\!]\|_F^2 \\ & + \sum_{n=1}^{N} \alpha_n \|\mathbf{A}_n\|_* + \ \mu \|[\![\widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_N]\!] - [\![\mathbf{A}_1^{(0)}, \ldots, \mathbf{A}_N^{(0)}]\!]\|_F^2 \\ \triangleq\ & \mathcal{L}_1 + \sum_{n=1}^{N} \alpha_n \|\mathbf{A}_n\|_*, \end{aligned} \tag{11}$$

where $\mathbf{A}_n = \begin{bmatrix} \mathbf{A}_n^{(0)} \\ \mathbf{A}_n^{(1)} \end{bmatrix} \begin{matrix} \in \mathbb{R}^{I_n \times R} \\ \in \mathbb{R}^{d_n \times R} \end{matrix}$. $[\![\widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_N]\!]$ is the decomposition of recovered tensor $\widetilde{\mathcal{X}}$ in the previous step, and $\Theta_N = \{0, 1\}^N$.

A tensor-based Alternating Direction Method of Multipliers (ADMM) algorithm is employed to solve this optimization problem. ADMM is an efficient optimization scheme for accommodating various constraints [14, 23]. Following the similar optimization scheme described in [11, 18], we induce auxiliary variables $\mathbf{Z}_n = \begin{bmatrix} \mathbf{Z}_n^{(0)} \\ \mathbf{Z}_n^{(1)} \end{bmatrix} \begin{matrix} \in \mathbb{R}^{I_n \times R} \\ \in \mathbb{R}^{d_n \times R} \end{matrix}$, $n = 1, \ldots, N$ into Equation (11), then the corresponding partial augmented Lagrangian function is:

```
Algorithm 1: The proposed framework MAST
    Input: $\mathcal{T}$, $\Omega$, $\{\widetilde{\mathbf{A}}_n\}_{n=1}^N$, $R$, $\{\alpha_n\}_{n=1}^N$, $\mu$, $\eta$, $\eta_{\max}$, $\rho$, $tol$;
    Output: $\mathcal{X}$, $\{\mathbf{A}_n\}_{n=1}^N$
 1  Initialize $\mathbf{A}_n^{(0)} = \widetilde{\mathbf{A}}_n$, $\mathbf{A}_n^{(1)} = rand(\mathbf{d_n}, R)$, $\mathbf{Z}_n = \mathbf{Y}_n = \mathbf{0}$;
 2  repeat
 3      $\eta = \min\{\eta * \rho, \eta_{\max}\}$ (accelerate optimization process);
 4      for $n = 1 : N$ do
 5          Update $\mathbf{A}_n^{(0)}$ and $\mathbf{A}_n^{(1)}$ using Equation (13);
 6      Update $\mathcal{X}$ using Equation (15);
 7      for $n = 1 : N$ do
 8          Update $\mathbf{Z}_n$ using Equation (14);
 9          Update $\mathbf{Y}_n$ using Equation (16);
10  until $\|\mathcal{X}_{pre} - \mathcal{X}\|_F / \|\mathcal{X}_{pre}\|_F < tol$;
```

$$\begin{aligned} \underset{\mathcal{X},\{\mathbf{A}_n\},\{\mathbf{Z}_n\},\{\mathbf{Y}_n\}}{\text{minimize}} \quad & \mathcal{L}_1 + \sum_{n=1}^{N} \Big(\alpha_n \|\mathbf{Z}_n\|_* + < \mathbf{Y}_n, \mathbf{Z}_n - \mathbf{A}_n > + \frac{\eta}{2} \|\mathbf{Z}_n - \mathbf{A}_n\|_F^2 \Big) \\ \text{subject to} \quad & \boldsymbol{\Omega} \circledast \mathcal{X} = \mathcal{T}, \ \ \mathbf{Z}_n = \mathbf{A}_n, \ \ n = 1, 2, \ldots, N, \end{aligned} \tag{12}$$

where $\mathbf{Y}_n = \begin{bmatrix} \mathbf{Y}_n^{(0)} \\ \mathbf{Y}_n^{(1)} \end{bmatrix} \begin{matrix} \in \mathbb{R}^{I_n \times R} \\ \in \mathbb{R}^{d_n \times R} \end{matrix}$ is the Lagrange multiplier matrix, $\eta > 0$ is a penalty parameter. Based on Equations (7) and (8), by calculating the derivatives of matrices $\{\mathbf{A}_n^{(0)}\}$ and $\{\mathbf{A}_n^{(1)}\} (n = 1, 2, \ldots, N)$, we can get their update rules as follows:

$$\begin{aligned} \mathbf{A}_n^{(0)} & \leftarrow \frac{\mu \widetilde{\mathbf{A}}_n \Big[(\widetilde{\mathbf{A}}_k^\top \mathbf{A}_k^{(0)})^{\circledast k \neq n} \Big] + \sum_{i \in S_n^0} \mathbf{X}_{(n)}^i (\mathbf{A}_k^{(i_k)})^{\odot k \neq n} + \eta \mathbf{Z}_n^{(0)} + \mathbf{Y}_n^{(0)}}{(\mathbf{A}_k^{(0)\top} \mathbf{A}_k^{(0)} + \mathbf{A}_k^{(1)\top} \mathbf{A}_k^{(1)})^{\circledast k \neq n} + \eta \mathbf{I}_R}, \\ \mathbf{A}_n^{(1)} & \leftarrow \frac{\sum_{i \in S_n^1} \mathbf{X}_{(n)}^i (\mathbf{A}_k^{(i_k)})^{\odot k \neq n} + \eta \mathbf{Z}_n^{(1)} + \mathbf{Y}_n^{(1)}}{(\mathbf{A}_k^{(0)\top} \mathbf{A}_k^{(0)} + \mathbf{A}_k^{(1)\top} \mathbf{A}_k^{(1)})^{\circledast k \neq n} + \eta \mathbf{I}_R}, \end{aligned} \tag{13}$$

where $S_n^0 = \{(s_1, \ldots, s_N) | \sum_{k=1}^N s_k \neq 0, s_k \in \{0, 1\}, s_n = 0\}$ and $S_n^1 = \{(s_1, \ldots, s_N) | \forall\, k \in \{1, \ldots, N\}\ s_k \in \{0, 1\}, s_n = 1\}$.

In each iteration, the auxiliary matrices $\{\mathbf{Z}_n\}$ has a closed-form solution as follows[6]:

$$\mathbf{Z}_n = SVT_{\frac{\alpha_n}{\eta}} (\mathbf{A}_n - \frac{\mathbf{Y}_n}{\eta}), \ n = 1, 2, \ldots, N. \tag{14}$$

where $SVT_{\frac{\alpha_n}{\eta}}$ is the singular value thresholding operator [6] defined as $SVT_\delta(\mathbf{A}) = \mathbf{U}(diag\{\sigma_i - \delta\})_+ \mathbf{V}^\top$, where $\mathbf{U}(diag\{\sigma_i\}_{1 \le i \le r})\mathbf{V}^\top$ is the singular value decomposition of matrix $\mathbf{A}$. For any matrix $\mathbf{X}$, $\mathbf{X}_+ = \max\{\mathbf{X}, 0\}$, where $\max\{\cdot, \cdot\}$ is an element-wise operator.

To mask the missing values and iteratively completing tensor $\mathcal{X}$, $\mathcal{X}$ is updated as follows:

$$\mathcal{X} \leftarrow \mathcal{T} + \boldsymbol{\Omega}^c \circledast [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]. \tag{15}$$

Finally, $\mathbf{Y}_n$ is updated as follows:

$$\mathbf{Y}_n \leftarrow \mathbf{Y}_n + \eta(\mathbf{Z}_n - \mathbf{A}_n), \ n = 1, 2, \ldots, N. \tag{16}$$

By using this optimization scheme, we can get the decomposition matrices and completed tensor $\mathcal{X}$ simultaneous. The entire optimization process of MAST is summarized in Algorithm 1. The termination criterion is that the relative changing of tensor $\mathcal{X}$ in two contiguous iterations is smaller than the tolerance.

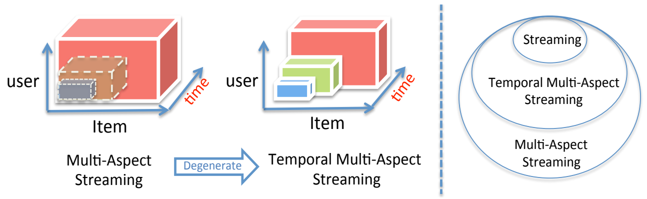


Figure 3: Degeneration of multi-aspect streaming tensor.

3.3 Temporal Multi-Aspect Streaming Tensor Completion

In some real-world applications, time is one mode of each tensor in a multi-aspect streaming tensor sequence. Take a recommender system structured as a *user* × *movie* × *time* tensor as an example. If a new user or a new item joins the system at time T, their past-time information would not exist during $t \in (0, T)$. This information shortage renders the snapshot tensor degenerated to a ladder-type structure shown in Figure 3. If we treat this structure as a combination of slices $\{\mathfrak{X}^{(T)}\}$, then the original multi-aspect streaming tensor sequence can be reconstructed using the incremental tensor slices with zero-paddings. We call these slices $\{\mathfrak{X}^{(T)}\}$ temporal multi-aspect streaming tensor sequence. This special structure provides us a way to improve MAST model for a better effectiveness by alleviating the quantity of substitution on missing entries.

3.3.1 Connection of Different Dynamic Patterns. Figure 3 depicts the relationships between different dynamic patterns. The temporal multi-aspect steaming is a special scenario. If the sizes of the tensor slices are the same at each time step, it will further degenerate into the traditional streaming tensor situation. Although the general framework MAST can be directly applied to the temporal case, the ladder-type structure provides extra information that could help.

We propose to reduce the quantity of substitution on missing entries to improve one of the key manipulations of MAST, i.e., tensor substitution. Different from DTD, because of data missing, the substitution of $\mathfrak{X}^{(T-1)}$ at time T is composed of two parts: (1) using former-step predicted values to impute the missing entries; (2) using fitting values to substitute the existed entries. The substitution of existed part often has much smaller error and usually provides more convincing information than the prediction of missing part. As the degeneration process naturally centralizes the existed data in the ladder-type object, all of the extra parts can be treat as missing data and their substitution errors are easy to reduce. Thus, besides forgetting factor μ, this special structure paves another way to reduce the accumulated error in MAST framework towards effectively capturing fast and dramatically changed subspaces.

3.3.2 Temporal Multi-Aspect Streaming CP Completion Algorithm. Inspired by the coupled tensor factorization [2], we tailor the general MAST framework and use Figure 4 to illustrate the main idea of the variation model T-MAST. Let $\mathfrak{X}^{(T)}$ denote the new tensor slice arrives at time T. $\mathfrak{T}^{(T)}$ represents its observed entries. $\{[\![\widetilde{\mathrm{A}}_1^{(t)}, \begin{bmatrix}\widetilde{\mathrm{A}}_2^{(1)}\\ \vdots \\ \widetilde{\mathrm{A}}_2^{(t)}\end{bmatrix}, \dots, \begin{bmatrix}\widetilde{\mathrm{A}}_N^{(1)}\\ \vdots \\ \widetilde{\mathrm{A}}_N^{(t)}\end{bmatrix}]\!]\}_{(1\le t\le T-1)}$ is the set of CP decompositions of the recovered tensor slices $\{\mathfrak{X}^{(t)}\}_{(1\le t\le T-1)}$ at time

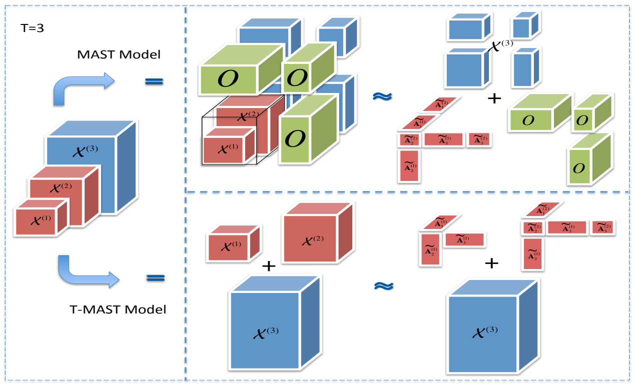


Figure 4: An illustration of T-MAST comparing with MAST.

$T-1$. We use $\mathrm{A}_n^{(T)}, n = 1, \dots, N$ to represent the incremental part of the decomposition matrix on mode-n at time T and use $\mathrm{A}_n^{(t)}$ $(1 \le t \le T-1)$ to represent the updated result of $\widetilde{\mathrm{A}}_n^{(t)}$ at current timestamp T. At each time step T, the problem is converted to how to recover $\mathfrak{X}^{(T)}$ based on the decompositions of former recovered slices $\{\mathfrak{X}^{(t)}\}_{(1\le t\le T-1)}$ and get the updated CP decomposition set $\{[\![\mathrm{A}_1^{(t)}, \begin{bmatrix}\mathrm{A}_2^{(1)}\\ \vdots \\ \mathrm{A}_2^{(t)}\end{bmatrix}, \dots, \begin{bmatrix}\mathrm{A}_N^{(1)}\\ \vdots \\ \mathrm{A}_N^{(t)}\end{bmatrix}]\!]\}_{(1\le t\le T)}$.

Comparing to MAST, which requires zero-paddings to redefine a multi-aspect sequence shown in Figure 4, for T-MAST, rather than reconstructing it as a whole tensor, we directly treat this ladder type object as coupled tensor slices and use the decomposition of each slice to substitute it. Since newly added zero-padding tensors and the block operation of $\mathfrak{X}^{(T)}$ has no influence on the current update, the difference between MAST and T-MAST models is how we substitute the former existed slices. Owing to the slice-based substitution in T-MAST model, the substitution of the former zero-padding parts can be omitted compared with the tensor-based substitution used in MAST model. Thus, it successfully reduces the proportion of missing data substitution as described before. The modified loss function is defined as follows:

$$\mathcal{L} = \left\| \mathfrak{X}^{(T)} - [\![\mathrm{A}_1^{(T)}, \mathrm{A}_2, \mathrm{A}_3, \dots, \mathrm{A}_N]\!] \right\|_{\mathrm{F}}^2 + \sum_{n=1}^{N} \alpha_n \left\| \mathrm{A}_n \right\|_* + \sum_{t=1}^{T-1} \mu_t \left\| [\![\widetilde{\mathrm{A}}_1^{(t)}, \begin{bmatrix}\widetilde{\mathrm{A}}_2^{(1)}\\ \vdots \\ \widetilde{\mathrm{A}}_2^{(t)}\end{bmatrix}, \dots, \begin{bmatrix}\widetilde{\mathrm{A}}_N^{(1)}\\ \vdots \\ \widetilde{\mathrm{A}}_N^{(t)}\end{bmatrix}]\!] - [\![\mathrm{A}_1^{(t)}, \begin{bmatrix}\mathrm{A}_2^{(1)}\\ \vdots \\ \mathrm{A}_2^{(t)}\end{bmatrix}, \dots, \begin{bmatrix}\mathrm{A}_N^{(1)}\\ \vdots \\ \mathrm{A}_N^{(t)}\end{bmatrix}]\!] \right\|_{\mathrm{F}}^2, \tag{17}$$

where $\mathrm{A}_n = \begin{bmatrix}\mathrm{A}_n^{(1)}\\ \vdots \\ \mathrm{A}_n^{(T)}\end{bmatrix}$, $n = 1, \dots, N$.

3.3.3 Optimization Scheme. By comparing Equation (17) with Equation (11), we can easily find that their first and second terms are correspondingly equivalent when dealing with the temporal-mode-involved scenario. The different is the last term. Hence, we can still use ADMM algorithm to solve the above optimization problem and the only difference lies in the optimization of $\{\mathrm{A}_n\}_n$. For the sake of brevity, we omit other repetitive derivation and only focus on the update of matrices $\{\mathrm{A}_n^{(t)}\}_{t,n}$. For each non-temporal mode $n =$

$2, \ldots, N$, we calculate the partial derivatives of matrices $\{\mathbf{A}_n^{(t)}\}$, $t = 1, \ldots, T$, and define intermediate recursive matrix sequences $\{\mathbf{B}_n^{(t)}\}$, $\{\mathbf{D}_n^{(t)}\}$, $\{\mathbf{P}_n^{(t)}\}$, and $\{\mathbf{Q}_n^{(t)}\}$ to update them in each iteration. Assume $\widetilde{\mathbf{A}_k^{(T)}} = 0$, $\mathbf{P}_k^{(T)} = \sum_{j=1}^{T} \widetilde{\mathbf{A}_k^{(j)}}^\top \mathbf{A}_k^{(j)}$, and $\mathbf{Q}_k^{(T)} = \sum_{j=1}^{T} \mathbf{A}_k^{(j)\top} \mathbf{A}_k^{(j)}$, $k = 2, \ldots, N$. For any $k = 2 \ldots, N$, we have:

$$\mathbf{P}_k^{(t)} = \mathbf{P}_k^{(t+1)} - \widetilde{\mathbf{A}_k^{(t)}}^\top \mathbf{A}_k^{(t)}; \quad \mathbf{Q}_k^{(t)} = \mathbf{Q}_k^{(t+1)} - \mathbf{A}_k^{(t)\top} \mathbf{A}_k^{(t)}, \ t = T-1, \ldots, 1.$$

Based on $\{\mathbf{P}_n^{(t)}\}$ and $\{\mathbf{Q}_n^{(t)}\}$, if we define $\mathbf{D}_n^{(T+1)} = 0$, $\mu_T = 1$, and $\mathbf{B}_n^{(T+1)} = 0$, we have:

$$\begin{cases} \mathbf{D}_n^{(t)} = \mathbf{D}_n^{(t+1)} + \mu_t (\widetilde{\mathbf{A}_1^{(t)}}^\top \mathbf{A}_1^{(t)}) \circledast (\mathbf{P}_k^{(t)})^{\circledast k \neq 1, n}, \\ \mathbf{B}_n^{(t)} = \mathbf{B}_n^{(t+1)} + \mu_t (\mathbf{A}_1^{(t)\top} \mathbf{A}_1^{(t)}) \circledast (\mathbf{Q}_k^{(t)})^{\circledast k \neq 1, n}, \ t = T, \ldots, 1. \end{cases}$$

Then we can get the update rules for $\{\mathbf{A}_n^{(t)}\}$ as follows:

$$\mathbf{A}_n^{(t)} \leftarrow \frac{\mathbf{C}_n^{(t)} + \widetilde{\mathbf{A}_n^{(t)}} \mathbf{D}_n^{(t)} + \eta \mathbf{Z}_n^{(t)} + \mathbf{Y}_n^{(t)}}{\mathbf{B}_n^{(t)} + \eta \mathbf{I}_R}, \ t = T, \ldots, 1, \ n = 2, \ldots, N, \tag{18}$$

where $\{\mathbf{Y}_n^{(t)}\}_{t,n}$ are the Lagrange matrix multipliers. $\{\mathbf{C}_n^{(t)}\}_t$ are the block sub-matrices of $\mathbf{C}_n \triangleq \mathbf{X}_{(n)}^{(T)} \left[\mathbf{A}_1^{(T)} \odot \left(\mathbf{A}_k \right)^{\odot k \neq 1, n} \right]$ based on the size of $\{\mathbf{A}_n^{(t)}\}_t$.

For each mode n, only $2(N-1)R^2$ extra space is needed comparing to the MAST model if we follow the updating order below:

$$\begin{aligned} \mathbf{C}_n \ \rightarrow \ \{\mathbf{P}_k^{(T)}\}_{k \in [2,N]} \ \rightarrow \ \{\mathbf{Q}_k^{(T)}\}_{k \in [2,N]} \ \rightarrow \ \mathbf{D}_n^{(T)} \ \rightarrow \ \mathbf{B}_n^{(T)} \ \rightarrow \mathbf{A}_n^{(T)} \\ \rightarrow \{\mathbf{P}_k^{(T-1)}\}_{k \in [2,N]} \rightarrow \{\mathbf{Q}_k^{(T-1)}\}_{k \in [2,N]} \rightarrow \mathbf{D}_n^{(T-1)} \rightarrow \mathbf{B}_n^{(T-1)} \rightarrow \mathbf{A}_n^{(T-1)} \\ \rightarrow \ldots \rightarrow \{\mathbf{P}_k^{(1)}\}_{k \in [2,N]} \ \rightarrow \ \{\mathbf{Q}_k^{(1)}\}_{k \in [2,N]} \ \rightarrow \ \mathbf{D}_n^{(1)} \ \rightarrow \ \mathbf{B}_n^{(1)} \ \rightarrow \mathbf{A}_n^{(1)}. \end{aligned}$$

The temporal mode ($n = 1$) could be updated by:

$$\mathbf{A}_1^{(t)} \leftarrow \frac{\mu_t \mathbf{C}_n^{(t)} + \eta \mathbf{Z}_1^{(t)} + \mathbf{Y}_1^{(t)}}{\mu_t (\mathbf{Q}_k^{(t)})^{\circledast k \neq 1} + \eta \mathbf{I}_R}, \ t = T, \ldots, 1, \ \mu_T = 1, \tag{19}$$

where $\mathbf{C}_n^{(T)} = \mathbf{X}_{(1)}^{(T)} \left(\mathbf{A}_k \right)^{\odot k \neq 1}$, $\mathbf{C}_n^{(t)} = (\mathbf{P}_k^{(t)})^{\circledast k \neq 1}$. The rest matrices $\{\mathbf{Y}_n^{(t)}\}$, $\{\mathbf{Z}_n^{(t)}\}$ and tensor $\mathcal{X}^{(T)}$ are updated similarly as MAST.

From model perspective, although T-MAST can not handle the general multi-aspect streaming situation as MAST, both of them can be applied to the traditional streaming case. Not surprisingly, comparing Equations (11) and (17), using the traditional streaming setting, these two models are equivalent to each other.

4 EXPERIMENTS

In this section, we empirically evaluate the performance of the proposed framework MAST. Three major aspects are analyzed.

Q1: How effective is MAST compared with static and dynamic CP completion methods on different real-world datasets with different dynamic variation patterns?

Q2: How efficient is MAST compared with the state-of-the-art methods on datasets with different length of time segments?

Q3: What is the influence of the forgetting factor μ on the general model MAST?

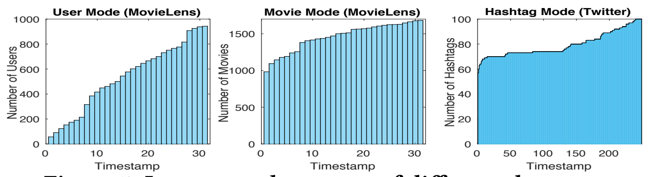


Figure 5: Incremental patterns of different datasets.

	Total Size	Initial Size	Increased Modes
Twitter Topic	$500 \times 500 \times 20$	$50 \times 50 \times 20$	1,2
Youtube	$1066 \times 1066 \times 5$	$100 \times 100 \times 5$	1,2
MovieLens	$943 \times 1682 \times 31$	$57 \times 983 \times 1$	1,2,3
Twitter Hashtag	$100 \times 1000 \times 249$	$57 \times 1000 \times 1$	1,3

Table 2: Characteristics of the four datasets.

4.1 Datasets and Tasks

To evaluate the validity of MAST, we apply it to four datasets with different applications. Their basic information is shown in Table 2.

Twitter Topic [10] (Recommendation): It is a third-order tensor with binary entries of size $500(user) \times 500(expert) \times 20(topic)$. Experts represent the high-quality content producers of 20 topics. It starts from $50 \times 50 \times 20$ and increases 2% of total users and experts at each time step. The task is to recommend the personalized expert of each topic to each user.

Youtube[2] **[33] (Link Prediction)**: It includes 1,066 users which have the most interactions among original 15,088 users sharing five interactions, including contact, co-contact, co-subscription, co-subscribed, and favorite networks. It starts from $100(user) \times 100(user) \times 5(interaction)$ and increases 2% of total users in each timestamp. The task is to predict missing links at each time step.

MovieLens[3] (Recommendation): It is a benchmark dataset for movie recommendation structured as a temporal multi-aspect streaming tensor sequence of size $943(user) \times 1{,}682(item) \times 31(week)$. Instead of predicting concrete movie rates, the task is to predict what movies a user may rate.

Twitter Hashtag [11] (Information Diffusion): This is a spatial-temporal dataset structured as a temporal multi-aspect streaming tensor sequence of size $100(hashtag) \times 1000(city) \times 249(day)$. Each binary entry indicates whether the corresponding hashtag emerges at the corresponding city on a specific day or not. Hashtag mode is increased as time goes. The task is to predict the emergence of previously appeared hashtags in each city at each time step.

The MovieLens and Twitter Hashtag follow the natural evolvement of a ladder-type increment. To give a better understanding of the incremental process, we depict the sectional views of the user and item modes of MovieLens, as well as the hashtag mode of Twitter Hashtag in Figure 5. The incremental pattern of Twitter Hashtag dataset is relatively gentle comparing with MovieLens.

4.2 Baseline Methods

Three state-of-the-art CP completion methods are employed including two categories, i.e., static and dynamic algorithms.

- Static CP-ALS [5]: It is a traditional CP-based method also called EM-ALS [1]. To enhance the learning process, the former-step decomposition is utilized as the initialization for the current step. We implement it with the help of [3].

[2]http://socialcomputing.asu.edu/datasets/YouTube
[3]http://movielens.umn.edu

Tensor Type		General Multi-Aspect Streaming Tensor						Temporal Multi-Aspect Streaming Tensor					
Dataset		Twitter Topic			Youtube			MovieLens			Twitter Hashtag		
Missing Percentage		20%	50%	80%	20%	50%	80%	20%	50%	80%	20%	50%	80%
Static	CP-ALS	0.8394	0.7810	0.6238	**0.9390**	**0.9209**	0.8609	0.9304	0.9057	0.8605	0.9257	0.9236	0.9133
	TNCP	**0.8443**	**0.7936**	**0.6661**	0.9384	0.9199	**0.8659**	**0.9344**	**0.9139**	**0.8790**	**0.9258**	**0.9238**	**0.9137**
Dynamic	OLSTEC	0.6670	0.5035	0.5028	0.8601	0.7315	0.7225	0.7048	0.6550	0.6227	0.7449	0.6716	0.6314
	MAST	**0.8392**	**0.7714**	**0.6493**	**0.9326**	**0.9141**	**0.8618**	0.8393	0.8354	0.8099	0.9160	0.9019	0.8641
	T-MAST	*N.A.*	*N.A.*	*N.A.*	*N.A.*	*N.A.*	*N.A.*	**0.9294**	**0.9023**	**0.8586**	**0.9233**	**0.9153**	**0.8881**

Table 3: Performance in terms of RA-AUC metric. (Results of the best dynamic and static methods are highlighted.)

- Static TNCP [18]: It is one of the state-of-the-art static completion methods. It employs trace norm constraints optimized by ADMM algorithm. Similar initialization approach is employed to accelerate the convergence.
- OLSTEC [15]: It is a state-of-the-art streaming tensor completion method. We extend it for multi-aspect streaming tensor completion by splitting the multi-aspect incremental process into several streaming ones and update them in random order in each iteration until it converges.
- T-MAST: This is a variation of MAST tailored towards temporal multi-aspect streaming tensor sequence. It cannot be applied to the Twitter Topic and Youtube.

4.3 Experimental Setup

Following widely used settings [15, 20], for all datasets, we randomly cover a fixed percentage of data and consider the remaining entries as observed information. The hidden data is used as ground truth. To further study the impact of the missing ratio, we vary it as {20%,50%,80%}. To evaluate the performance, a widely used metric is employed for all tasks, i.e., running-average Area Under Curve (RA-AUC). For the first two datasets, we focus on the incremental part $\mathcal{X}^{(t)} \backslash \mathcal{X}^{(t-1)}$ at time step t to avoid double counting and calculate the Area Under Curve (AUC) score of each topic slice or interaction slice based on the prediction. The average AUC at time t among topics or interaction slices is denoted as AUC_t. For the last two datasets which are two multi-aspect streaming tensor sequences, AUC_t is calculated using the prediction of missing data in slice $\mathcal{X}^{(t)}$. The running-average AUC score is defined as: $\text{RA-AUC} = \frac{1}{T}\sum_{t=1}^{T} \text{AUC}_t$, where T is the total number of time steps. The efficiency is measured by average running time defined as $\frac{1}{T}\sum_{t=1}^{T} \text{RT}_t$, where RT_t is the running time at time step t.

Parameter Setting: We set the tolerance rate to 10^{-5}, the maximum number of iteration to 500 for all the algorithms. By testing the performance of static methods on whole four datasets using 10 different ranks varying from 5 to 50, we set $R = 10$ in all the experiment considering of both accuracy and speed for fair comparison. In the implementation of our proposed method, default parameters are set as $\alpha_n = \frac{1}{10N}$ $n = 1, \ldots, N$, $\eta = 10^{-4}$, $\rho = 1.05$ and $\eta_{\max} = 10^6$. The forgetting factors are fine-tuned according to the missing ratio f (usually $\mu = \mu_t = 1 - f$ ($\forall\, t > 0$)). For baselines, we follow the suggestions of original papers to set parameters. The initial completion and warm start matrices are calculated using TNCP method for two reasons: (1) MAST framework degenerates to static TNCP when $T = 1$. (2) For fair comparison, we use the same best warm start at initial time for all methods. All experimental results are the arithmetic average of five runs and are ran on a Dell OptiPlex 9030 i7-16GB desktop with MATLAB R2016b.

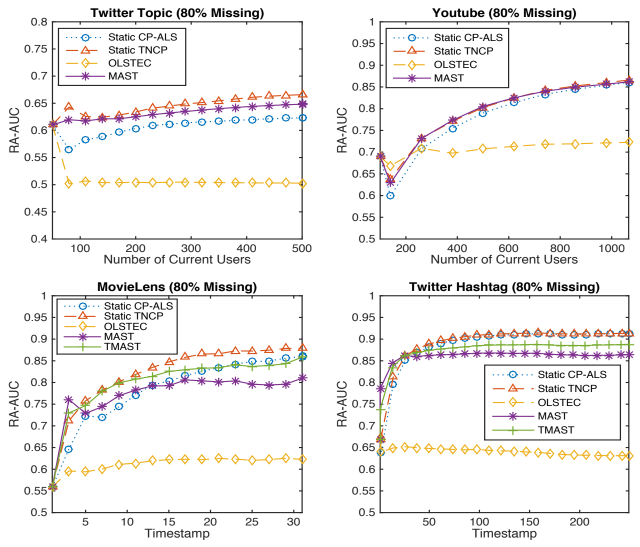


Figure 6: Performance of different methods on the four datasets with different timestamps.

4.4 Evaluation of Effectiveness

Table 3 shows the performance of MAST and baselines on different datasets in terms of RA-AUC. Three main conclusions are observed.

First, MAST has commensurate performance comparing with the two static models and higher performance than dynamic baseline method OLSTEC with different percentages of random missing entries on the first two datasets. To illustrate the performance of different models over time, we display the RA-AUC variation curve of MAST compared with baselines on four datasets with 80% missing entries in Figure 6. The results show that, with time goes, the increase of users leads to a fluctuant increasing RU-AUC for all the methods except OLSTEC. MAST has comparable performance to the static baselines and has higher accuracy than OLSTEC.

Second, on both MovieLens and Twitter Hashtag, which have different dynamic patterns shown in Figure 5, the variation model T-MAST outperforms MAST. This result empirically validates our analysis in Section 3.3 that the ladder-type structure of temporal multi-aspect streaming pattern could help to reduce the quantity of substitution on missing entries thereby improving general MAST framework. Besides, as can be seen from the results, the difference between T-MAST and MAST on MovieLens is larger than the one on Twitter Hashtag. This is because MovieLens dataset has fewer snapshots and sharper changes on the size of tensor slices than Twitter Hashtag as shown in Figure 5. The dramatically increased ladder-type structure leads to a larger ratio of missing data substitution at each time step. This further results in a hysteresis effect for capturing characteristics of the new data during the update.

Dataset		Twitter Topic			Youtube			MovieLens			Twitter Hashtag		
Missing Percentage		20%	50%	80%	20%	50%	80%	20%	50%	80%	20%	50%	80%
Static	CP-ALS	305.20	307.42	302.85	504.77	502.73	500.72	1189.9	1223.3	1170.2	257.16	468.13	582.38
	TNCP	110.73	118.69	117.92	371.33	368.56	365.67	509.87	511.34	456.59	172.01	194.78	193.59
Dynamic	OLSTEC	14.042	14.150	13.903	29.592	27.272	27.216	144.18	143.59	140.81	9.2601	9.6796	9.3625
	MAST	**8.7937**	**8.3523**	**7.6398**	**28.270**	**26.443**	**25.626**	118.13	113.74	110.72	**8.9186**	**8.5487**	**7.0420**
	T-MAST	$N.A.$	$N.A.$	$N.A.$	$N.A.$	$N.A.$	$N.A.$	**113.65**	**105.29**	**103.56**	19.1785	18.6397	16.3826

Table 4: Average running time of different methods on the four datasets.

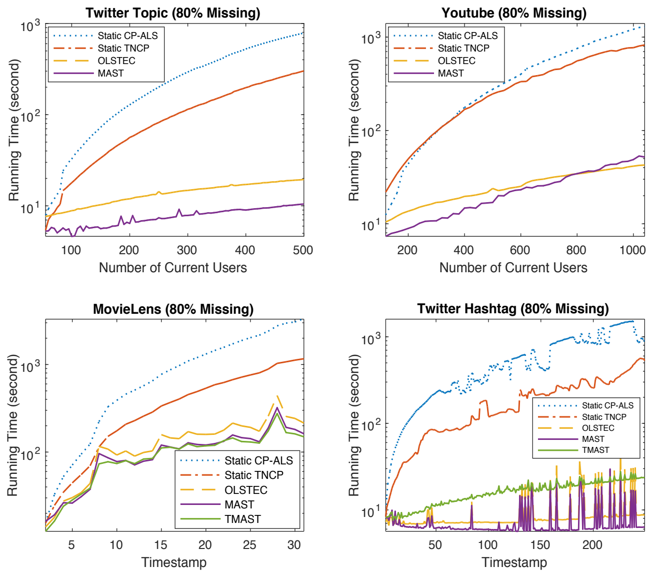


Figure 7: Running time (in logarithmic scale) of all methods.

Third, MAST shows strong stability on different datasets of different dynamic patterns. Furthermore, for different ratios of missing data and different length of time steps, our proposed framework retains comparable performance with static models. Theses results demonstrate the capability of our model in capturing low-rank subspace of dynamically changed tensor objects.

4.5 Evaluation of Efficiency

To study the efficiency of the proposed framework, we compare the average running time of all five models shown in Table 4. For better visualization, Figure 7 displays the computation time (RT_t) in logarithmic scale as a function of timestamps or user numbers on four datasets with 80% missing. From these results, we can come to the conclusion that (1) For general multi-aspect streaming tensor, MAST model takes much less running time than static models and outperforms OLSTEC. (2) T-MAST model shows comparable efficiency with MAST on short-term datasets. Although it sacrifices the time complexity to some degree on long-term datasets for the sake of higher effectiveness, it is still significantly faster than static models. If we assume $R \ll I_n$, $n = 1, \ldots, N$, and define $S^t = \prod_{n=1}^{N} I_n^t$, where I_n^t is size of the N^{th} mode of slice $\mathcal{X}^{(t)}$ in a temporal multi-aspect streaming tensor sequence, the complexities per iteration of all three dynamic models at time t would be $O((N+1)R(S^t - S^{(t-1)}))$ comparing to $O((N+1)RS^t)$ of two static models. Moreover, a dramatic change in the size of the tensor will cause a violent fluctuation on the completion time. For instance, the structures of Youtube and Twitter Topic datasets changed more steadily than MovieLens and Twitter Hashtag datasets resulting in less violent fluctuations on the running-time curves. If we focus on one dataset such as Twitter Hashtag, the time curve fluctuates more acutely when the size of the newly slice increases sharply.

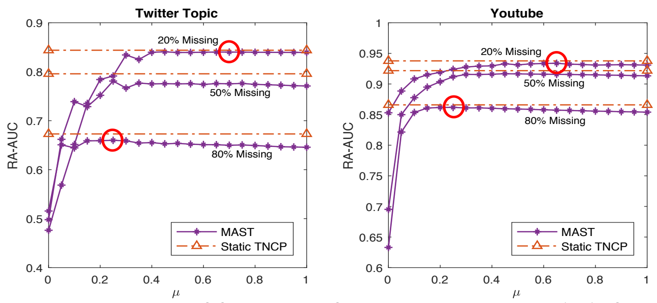


Figure 8: Impact of forgetting factor μ on MAST with different missing percentages.

4.6 Influence of Forgetting Factor μ

To investigate the effect of forgetting factor μ on our proposed framework, we vary it from 0 to 1 with a step size 0.05 and compare the performance of MAST model on Twitter Topic and Youtube datasets. Results shown in Figure 8 demonstrate that with the decreasing of forgetting factor, the performance of MAST increases at first and then decreases. With the increase of the missing ratio, the turning point becomes closer to 0, which inspires us to choose a smaller μ to alleviate the substitution error. In sum, these observations illustrate that: (1) A suitable shrinkage on the approximately substituted tensor could alleviate the previous fitting error and result in better completion effectiveness. (2) The higher the missing ratio is, the smaller the forgetting factor we should choose to alleviate the hysteresis effect of our framework in capturing the low-rank subspace for multi-aspect streaming tensor completion.

5 RELATED WORK

The related work can be categorized into two main topics as follows:

Dynamic Tensor Factorization. Tensor factorization methods have received widespread concerns and achievements under static setting [16, 23]. Increasingly massive amount of real-world dynamic data nowadays requires an extensive concern on the problem of dynamic tensor factorization [15]. Nion and Sidiropoulos [22] proposed two adaptive PARAFAC algorithms adopting the recursive least square and simultaneous diagonalization tracking methods to solve the online third-order tensor factorization. Phan et al. [24] partitioned a large-scale tensor into small grids and proposed a grid-based scalable tensor factorization method which could also be used in dynamic tensor factorization. Zhou et al. [38] proposed an accelerated online algorithm that can track the CP decompositions of incremental N^{th}-order tensors. However, all of them are not directly applicable to multi-aspect streaming situations and the completion task. Kasai [15] substituted the stochastic gradient descent method with recursive least square method and improved the algorithm proposed by Mardani et al. [20] focusing on the problem of subspace learning and imputation for streaming tensors. Besides CP decomposition, dynamic tucker decomposition methods were also proposed [31, 32, 37]. Some of them were not only focusing on one-mode increasing condition [13, 30], but also giving possible solutions for all-mode incremental update using matrix-based online methods such as incremental SVD [19] without considering about missing entries. Finally, it is worth to mention that though not a factorization method, the histogram-based approach [8] conducted on multi-aspect streaming tensor analysis can be treated as one of the pioneer researches on the multi-aspect streaming pattern.

Low-Rank Tensor Completion. Since the real-world multidimensional datasets are oftentimes raw and incomplete because of missing at random and limited permissions [11, 18], low-rank tensor completion problem has been attractive to researchers and practitioners in data mining, online learning, computer vision, signal processing, etc. Generalized from matrix cases, a wide range of approaches have been proposed such as trace-norm based methods [9, 17, 29, 35], factorization-based approaches [1, 5, 34], tensor completion with auxiliary information [2, 11, 21], and online tensor imputation [15, 20]. Although both theoretical analysis and various practical applications have been considered, to our best knowledge, no existing work has been conducted on the low-rank tensor completion with general multi-aspect streaming patterns.

6 CONCLUSION AND FUTURE WORK

In this paper, we focus on the multi-aspect streaming tensor completion problem and propose an updatable CP completion framework MAST. The proposed framework can effectively capture the low-rank subspace of multi-aspect streaming tensor sequences so as to achieve the completion purpose. To further enhance the effectiveness, we also tailor the general framework toward a special case where time is one mode of the multi-aspect streaming tensors. By conducting experiments on various real-world applications, we empirically validate the effectiveness and efficiency of our proposed framework. Future work will center on investigating dynamic Tucker-based tensor completion methods and incorporating scalable tensor mining techniques [27, 28] into our framework.

ACKNOWLEDGMENTS

This work is, in part, supported by DARPA (#W911NF-16-1-0565) and NSF (#IIS-1657196). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

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

[2] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. 2011. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv* (2011).

[3] Brett W Bader, Tamara G Kolda, and others. 2015. MATLAB Tensor Toolbox Version 2.6, Available online,. (2015).

[4] Alex Beutel, Partha P Talukdar, Abhimanu Kumar, Christos Faloutsos, Evangelos E Papalexakis, and Eric P Xing. 2014. Flexifact: scalable flexible factorization of coupled tensors on hadoop. *ICDM* (2014).

[5] Rasmus Bro. 1998. Multi-way analysis in the food industry: models, algorithms, and applications. *MRI, EPG and EMA," Proc ICSLP* (1998).

[6] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. 2010. A singular value thresholding algorithm for matrix completion. *SIOPT* (2010).

[7] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. 2011. Temporal link prediction using matrix and tensor factorizations. *TKDD* (2011).

[8] Hadi Fanaee-T and João Gama. 2015. Multi-aspect-streaming tensor analysis. *Knowledge-Based Systems* (2015).

[9] Silvia Gandy, Benjamin Recht, and Isao Yamada. 2011. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems* (2011).

[10] Hancheng Ge, James Caverlee, and Haokai Lu. 2016. Taper: a contextual tensor-based approach for personalized expert recommendation. *Proc. of RecSys* (2016).

[11] Hancheng Ge, James Caverlee, Nan Zhang, and Anna Squicciarini. 2016. Uncovering the spatio-temporal dynamics of memes in the presence of incomplete information. *CIKM* (2016).

[12] Johan Håstad. 1990. Tensor rank is NP-complete. *Journal of Algorithms* (1990).

[13] Weiming Hu, Xi Li, Xiaoqin Zhang, Xinchu Shi, Stephen Maybank, and Zhongfei Zhang. 2011. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *IJCV* (2011).

[14] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. *SDM* (2017).

[15] Hiroyuki Kasai. 2016. Online low-rank tensor subspace tracking from incomplete data by CP decomposition using recursive least squares. *ICASSP* (2016).

[16] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* (2009).

[17] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2013. Tensor completion for estimating missing values in visual data. *TPAMI* (2013).

[18] Yuanyuan Liu, Fanhua Shang, Licheng Jiao, James Cheng, and Hong Cheng. 2015. Trace norm regularized CANDECOMP/PARAFAC decomposition with missing data. *IEEE Transactions on Cybernetics* (2015).

[19] Xiang Ma, Dan Schonfeld, and Ashfaq Khokhar. 2009. Dynamic updating and downdating matrix SVD and tensor HOSVD for adaptive indexing and retrieval of motion trajectories. *ICASSP* (2009).

[20] Morteza Mardani, Gonzalo Mateos, and Georgios B Giannakis. 2015. Subspace learning and imputation for streaming big data matrices and tensors. *IEEE TSP* (2015).

[21] Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. 2011. Tensor factorization using auxiliary information. *ECML PKDD* (2011).

[22] Dimitri Nion and Nicholas D Sidiropoulos. 2009. Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. *IEEE TSP* (2009).

[23] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 2016. Tensors for data mining and data fusion: models, applications, and scalable Algorithms. *TIST* (2016).

[24] Anh Huy Phan and Andrzej Cichocki. 2011. PARAFAC algorithms for large-scale problems. *Neurocomputing* (2011).

[25] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. *KDD* (2009).

[26] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. *WSDM* (2010).

[27] Lee Sael, Inah Jeon, and U Kang. 2015. Scalable tensor mining. *Big Data Research* (2015).

[28] Kijung Shin and U Kang. 2014. Distributed methods for high-dimensional and large-scale tensor factorization. *ICDM* (2014).

[29] Marco Signoretto, Lieven De Lathauwer, and Johan AK Suykens. 2010. Nuclear norms for tensors and their use for convex multilinear estimation. *Linear Algebra and Its Applications* (2010).

[30] Andrews Sobral, Christopher G Baker, Thierry Bouwmans, and El-hadi Zahzah. 2014. Incremental and multi-feature tensor subspace learning applied for background modeling and subtraction. *ICIAR* (2014).

[31] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. 2006. Beyond streams and graphs: dynamic tensor analysis. *KDD* (2006).

[32] Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S Yu, and Christos Faloutsos. 2008. Incremental tensor analysis: theory and applications. *TKDD* (2008).

[33] Lei Tang, Xufei Wang, and Huan Liu. 2009. Uncoverning groups via heterogeneous interaction analysis. *ICDM* (2009).

[34] Giorgio Tomasi and Rasmus Bro. 2005. PARAFAC and missing values. *Chemometrics and Intelligent Laboratory Systems* (2005).

[35] Ryota Tomioka, Kohei Hayashi, and Hisashi Kashima. 2010. Estimation of low-rank tensors via convex optimization. *arXiv* (2010).

[36] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel Kho, You Chen, Bradley A Malin, and Jimeng Sun. 2015. Rubik: Knowledge guided tensor factorization and completion for health data analytics. *KDD* (2015).

[37] Rose Yu, Dehua Cheng, and Yan Liu. 2015. Accelerated online low-rank tensor learning for multivariate spatio-temporal streams. *ICML* (2015).

[38] Shuo Zhou, Nguyen X Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. 2016. Accelerating online CP decompositions for higher order tensors. *KDD* (2016).