Inductive Framework for Multi-Aspect Streaming Tensor Completion with Side Information

Madhav Nimishakavi
Indian Institute of Science
Bangalore, India
madhav@iisc.ac.in

Bamdev Mishra
Microsoft
Hyderabad, India
bamdevm@microsoft.com

Manish Gupta
Microsoft
Hyderabad, India
gmanish@microsoft.com

Partha Talukdar
Indian Institute of Science
Bangalore, India
ppt@iisc.ac.in

ABSTRACT

Low rank tensor completion is a well studied problem and has applications in various fields. However, in many real world applications the data is dynamic, i.e., new data arrives at different time intervals. As a result, the tensors used to represent the data grow in size. Besides the tensors, in many real world scenarios, side information is also available in the form of matrices which also grow in size with time. The problem of predicting missing values in the dynamically growing tensor is called dynamic tensor completion. Most of the previous work in dynamic tensor completion make an assumption that the tensor grows only in one mode. To the best of our Knowledge, there is no previous work which incorporates side information with dynamic tensor completion. We bridge this gap in this paper by proposing a dynamic tensor completion framework called Side Information infused Incremental Tensor Analysis (SIITA), which incorporates side information and works for general incremental tensors. We also show how non-negative constraints can be incorporated with SIITA, which is essential for mining interpretable latent clusters. We carry out extensive experiments on multiple real world datasets to demonstrate the effectiveness of SIITA in various different settings.

CCS CONCEPTS

• **Computing methodologies** → **Online learning settings**; **Factorization methods**;

KEYWORDS

Tensor Decomposition, Online Learning, Side Information, Tucker Decomposition, Nonnegative Decomposition

ACM Reference Format:
Madhav Nimishakavi, Bamdev Mishra, Manish Gupta, and Partha Talukdar. 2018. Inductive Framework for Multi-Aspect Streaming Tensor Completion with Side Information. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271713

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271713

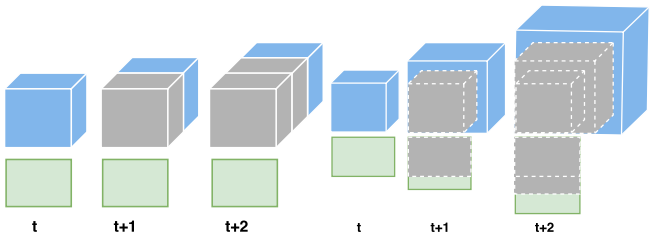
1 INTRODUCTION

Low rank tensor completion is a well-studied problem and has various applications in the fields of recommendation systems [30], link-prediction [4], compressed sensing [3], to name a few. Majority of the previous works focus on solving the problem in a static setting [6, 8, 14]. However, most of the real world data is dynamic, for example in an online movie recommendation system the number of users and movies increase with time. It is prohibitively expensive to use the static algorithms for dynamic data. Therefore, there has been an increasing interest in developing algorithms for dynamic low-rank tensor completion [13, 18, 27].

Usually in many real world scenarios, besides the tensor data, additional side information is also available, e.g., in the form of matrices. In the dynamic scenarios, the side information grows with time as well. For instance, movie-genre information in the movie recommendation etc. There has been considerable amount of work in incorporating side information into tensor completion [7, 21]. However, the previous works on incorporating side information deal with the static setting. In this paper, we propose a dynamic low-rank tensor completion model that incorporates side information growing with time.

Most of the current dynamic tensor completion algorithms work in the streaming scenario, i.e., the case where the tensor grows only in one mode, which is usually the time mode. In this case, the side information is a static matrix. Multi-aspect streaming scenario [5, 27], on the other hand, is a more general framework, where the tensor grows in all the modes of the tensor. In this setting, the side information matrices also grow. Figure 1 illustrates the difference between streaming and multi-aspect streaming scenarios with side information.

Besides side information, incorporating nonnegative constraints into tensor decomposition is desirable in an unsupervised setting. Nonnegativity is essential for discovering interpretable clusters [10, 20]. Nonnegative tensor learning is explored for applications in computer vision [15, 25], unsupervised induction of relation schemas [23], to name a few. Several algorithms for online Nonnegative Matrix Factorization (NMF) exist in the literature [17, 34],



(a) Streaming tensor sequence with side information.

(b) Multi-aspect streaming tensor sequence with side information.

Figure 1: Illustration of streaming and multi-aspect streaming sequences with side information. The blue block represents the tensor at time step and the green block represents the side information. The blocks in grey represent the data at previous time steps. For easy understanding, we show side information along only one mode.

but algorithms for nonnegative online tensor decomposition with side information are not explored to the best of our knowledge. We also fill this gap by showing how nonnegative constraints can be enforced on the decomposition learned by our proposed framework SIITA.

In this paper, we work with the more general multi-aspect streaming scenario and make the following contributions:

- Formally define the problem of multi-aspect streaming tensor completion with side information.
- Propose a Tucker based framework Side Information infused Incremental Tensor Analysis(SIITA) for the problem of multi-aspect streaming tensor completion with side information. We employ a stochastic gradient descent (SGD) based algorithm for solving the optimization problem.
- Incorporate nonnegative constraints with SIITA for discovering the underlying clusters in unsupervised setting.
- Demonstrate the effectiveness of SIITA using extensive experimental analysis on multiple real-world datasets in all the settings.

The organization of the paper is as follows. In Section 3, we introduce the definition of multi-aspect streaming tensor sequence with side information and discuss our proposed framework SIITA in Section 4. We also discuss how nonnegative constraints can be incorporated into SIITA in Section 4. The experiments are shown in Section 5, where SIITA performs effectively in various settings. All our codes are implemented in Matlab, and can be found at https://madhavcsa.github.io/.

2 RELATED WORK

Dynamic Tensor Completion: Sun et al. [28, 29] propose multiple Higher order SVD based algorithms, namely Dynamic Tensor Analysis (DTA), Streaming Tensor Analysis (STA) and Window-based Tensor Analysis (WTA) for the streaming scenario. Nion and Sidiropoulos [24] propose two adaptive online algorithms for CP decomposition of 3-order tensors. Yu et al. [33] propose an accelerated online algorithm for Tucker factorization in streaming scenario, while an accelerated online algorithm for CP decomposition is developed in [35].

A significant amount of research work is carried out for dynamic tensor decompositions, but work focusing on the problem of dynamic tensor completion is relatively less explored. Work by Mardani et al. [18] can be considered a pioneering work in dynamic tensor completion. They propose a streaming tensor completion algorithm based on CP decomposition. Recent work by Kasai [13] is an accelerated second order Stochastic Gradient Descent (SGD) algorithm for streaming tensor completion based on CP decomposition. Fanaee-T and Gama [5] introduces the problem of multi-aspect streaming tensor analysis by proposing a histogram based algorithm. Recent work by Song et al. [27] is a more general framework for multi-aspect streaming tensor completion.

Tensor Completion with Auxiliary Information: Acar et al. [1] propose a Coupled Matrix Tensor Factorization (CMTF) approach for incorporating additional side information, similar ideas are also explored in [2] for factorization on hadoop and in [4] for link prediction in heterogeneous data. Narita et al. [21] propose with-in mode and cross-mode regularization methods for incorporating similarity side information matrices into factorization. Based on similar ideas, Ge et al. [7] propose AirCP, a CP-based tensor completion algorithm.

Welling and Weber [31] propose nonnegative tensor decmposiiton by incorporating nonnegative constraints into CP decomposition. Nonnegative CP decomposition is explored for applications in computer vision in [25]. Algorithms for nonnegative Tucker decomposition are proposed in [15] and for sparse nonnegative Tucker decomposition are proposed in [19]. However, to the best our knowledge, nonnegative tensor decomposition algorithms do not exist for dynamic settings, a gap we fill in this paper.

Inductive framework for matrix completion with side information is proposed in [11, 22, 26], which has not been explored for tensor completion to the best of our knowledge. In this paper, we propose an online inductive framework for multi-aspect streaming tensor completion.

Table 1 provides details about the differences between our proposed SIITA and various baseline tensor completion algorithms.

3 PRELIMINARIES

An N^{th}-order or N-mode tensor is an N-way array. We use boldface calligraphic letters to represent tensors (e.g., $\boldsymbol{\mathcal{X}}$), boldface uppercase to represent matrices (e.g., $\mathbf{U}$), and boldface lowercase to represent vectors (e.g., $\mathbf{v}$). $\boldsymbol{\mathcal{X}}[i_1, \cdots, i_N]$ represents the entry of $\boldsymbol{\mathcal{X}}$ indexed by $[i_1, \cdots, i_N]$.

Definition 1 (Coupled Tensor and Matrix) [27]: A matrix and a tensor are called coupled if they share a mode. For example, a $user \times movie \times time$ tensor and a $movie \times genre$ matrix are coupled along the movie mode.

Definition 2 (Tensor Sequence) [27]: A sequence of N^{th}-order tensors $\boldsymbol{\mathcal{X}}^{(1)}, \ldots, \boldsymbol{\mathcal{X}}^{(t)}, \ldots$ is called a tensor sequence denoted as $\{\boldsymbol{\mathcal{X}}^{(t)}\}$, where each $\boldsymbol{\mathcal{X}}^{(t)} \in \mathbb{R}^{I_1^t \times I_2^t \times \ldots \times I_N^t}$ at time instance t.

Definition 3 (Multi-aspect streaming Tensor Sequence) [27]: A tensor sequence of N^{th}-order tensors $\{\boldsymbol{\mathcal{X}}^{(t)}\}$ is called a multi-aspect streaming tensor sequence if for any $t \in \mathbb{Z}^+$, $\boldsymbol{\mathcal{X}}^{(t-1)} \in$

Table 1: Summary of different tensor streaming algorithms.

Property	TeCPSGD[18]	OLSTEC [13]	MAST [27]	AirCP [7]	SIITA (this paper)
Streaming	✓	✓	✓		✓
Multi-Aspect Streaming			✓		✓
Side Information				✓	✓
Sparse Solution					✓

$\mathbb{R}^{I_1^{t-1} \times I_2^{t-1} \times \ldots \times I_N^{t-1}}$ is the sub-tensor of $\boldsymbol{\mathcal{X}}^{(t)} \in \mathbb{R}^{I_1^t \times I_2^t \times \ldots \times I_N^t}$, i.e.,

$$\boldsymbol{\mathcal{X}}^{(t-1)} \subseteq \boldsymbol{\mathcal{X}}^{(t)}, \text{ where } I_i^{t-1} \leq I_i^t, \forall 1 \leq i \leq N.$$

Here, t increases with time, and $\boldsymbol{\mathcal{X}}^{(t)}$ is the snapshot tensor of this sequence at time t.

Definition 4 (Multi-aspect streaming Tensor Sequence with Side Information) : Given a time instance t, let $\mathbf{A}_i^{(t)} \in \mathbb{R}^{I_i^t \times M_i}$ be a side information (SI) matrix corresponding to the i^{th} mode of $\boldsymbol{\mathcal{X}}^{(t)}$ (i.e., rows of $\mathbf{A}_i^{(t)}$ are coupled along i^{th} mode of $\boldsymbol{\mathcal{X}}^{(t)}$). While the number of rows in the SI matrices along a particular mode i may increase over time, the number of columns remain the same, i.e., M_i is not dependent on time. In particular, we have,

$$\mathbf{A}_i^{(t)} = \begin{bmatrix} \mathbf{A}_i^{(t-1)} \\ \Delta_i^{(t)} \end{bmatrix}, \text{ where } \Delta_i^{(t)} \in \mathbb{R}^{[I_i^{(t)} - I_i^{(t-1)}] \times M_i}.$$

Putting side information matrices of all the modes together, we get the side information set $\mathcal{A}^{(t)}$,

$$\mathcal{A}^{(t)} = \{\mathbf{A}_1^{(t)}, \ldots, \mathbf{A}_N^{(t)}\}.$$

Given an N^{th}-order multi-aspect streaming tensor sequence $\{\boldsymbol{\mathcal{X}}^{(t)}\}$, we define a multi-aspect streaming tensor sequence with side information as $\{(\boldsymbol{\mathcal{X}}^{(t)}, \mathcal{A}^{(t)})\}$.

We note that all modes may not have side information available. In such cases, an identity matrix of appropriate size may be used as $\mathbf{A}_i^{(t)}$, i.e., $\mathbf{A}_i^{(t)} = \mathbf{I}^{I_i^t \times I_i^t}$, where $M_i = I_i^t$.

The problem of multi-aspect streaming tensor completion with side information is formally defined as follows:

Problem Definition: Given a multi-aspect streaming tensor sequence with side information $\{(\boldsymbol{\mathcal{X}}^{(t)}, \mathcal{A}^{(t)})\}$, the goal is to predict the missing values in $\boldsymbol{\mathcal{X}}^{(t)}$ by utilizing only entries in the relative complement $\boldsymbol{\mathcal{X}}^{(t)} \setminus \boldsymbol{\mathcal{X}}^{(t-1)}$ and the available side information $\mathcal{A}^{(t)}$.

4 PROPOSED FRAMEWORK SIITA

In this section, we discuss the proposed framework SIITA for the problem of multi-aspect streaming tensor completion with side information. Let $\{(\boldsymbol{\mathcal{X}}^{(t)}, \mathcal{A}^{(t)})\}$ be an N^{th}-order multi-aspect streaming tensor sequence with side information. Assuming that, at every time step, $\boldsymbol{\mathcal{X}}^{(t)}[i_1, i_2, \cdots, i_N]$ are only observed for some indices $[i_1, i_2, \cdots, i_N] \in \Omega$, where Ω is a subset of the complete set of indices $[i_1, i_2, \cdots, i_N]$. Let the sparsity operator $\mathcal{P}_\Omega$ be defined as:

$$\mathcal{P}_\Omega[i_1, i_2, \cdots, i_N] = \begin{cases} \boldsymbol{\mathcal{X}}[i_1, \cdots, i_N], & \text{if } [i_1, \cdots, i_N] \in \Omega \\ 0, & \text{otherwise.} \end{cases}$$

Tucker tensor decomposition [16], is a form of higher-order PCA for tensors. It decomposes an N^{th}-order tensor $\boldsymbol{\mathcal{X}}$ into a core tensor multiplied by a matrix along each mode as follows

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \mathbf{U}_N,$$

where, $\mathbf{U}_i \in \mathbb{R}^{I_i \times r_i}, i = 1 : N$ are the factor matrices and can be thought of as principal components in each mode. The tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots r_N}$ is called the *core tensor*, which shows the interaction between different components. $(r_1, r_2, \cdots, r_N)$ is the (multilinear) rank of the tensor. The i-mode matrix product of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_N}$ with a matrix $\mathbf{P} \in \mathbb{R}^{r \times I_i}$ is denoted by $\boldsymbol{\mathcal{X}} \times_i \mathbf{P}$, more details can be found in [16]. The standard approach of incorporating side information while learning factor matrices in Tucker decomposition is by using an additive term as a regularizer [21]. However, in an online setting the additive side information term poses challenges as the side information matrices are also dynamic. Therefore, we propose the following fixed-rank *inductive framework* for recovering missing values in $\boldsymbol{\mathcal{X}}^{(t)}$, at every time step t:

$$\min_{\substack{\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times \ldots \times r_N} \\ \mathbf{U}_i \in \mathbb{R}^{M_i \times r_i}, i=1:N}} F(\boldsymbol{\mathcal{X}}^{(t)}, \mathcal{A}^{(t)}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}_i\}_{i=1:N}), \tag{1}$$

where

$$F(\boldsymbol{\mathcal{X}}^{(t)}, \mathcal{A}^{(t)}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}_n\}_{i=1:N}) = \left\| \mathcal{P}_\Omega(\boldsymbol{\mathcal{X}}^{(t)}) - \mathcal{P}_\Omega(\boldsymbol{\mathcal{G}} \times_1 \mathbf{A}_1^{(t)} \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{A}_N^{(t)} \mathbf{U}_N) \right\|_F^2 + \lambda_g \|\boldsymbol{\mathcal{G}}\|_F^2 + \sum_{i=1}^{N} \lambda_i \|\mathbf{U}_i\|_F^2. \tag{2}$$

$\|\cdot\|_F$ is the Frobenius norm, $\lambda_g > 0$ and $\lambda_i > 0, i = 1 : N$ are the regularization weights. Conceptually, the inductive framework models the ratings of the tensor as a weighted scalar product of the side information matrices. Note that (1) is a generalization of the inductive matrix completion framework [11, 22, 26], which has been effective in many applications.

The inductive tensor framework has two-fold benefits over the typical approach of incorporating side information as an additive term. The use of $\mathbf{A}_i \mathbf{U}_i$ terms in the factorization reduces the dimensionality of variables from $\mathbf{U}_i \in \mathbb{R}^{I_i \times r_i}$ to $\mathbf{U}_i \in \mathbb{R}^{M_i \times r_i}$ and typically $M_i \ll I_i$. As a result, computational time required for computing the gradients and updating the variables decreases remarkably. Similar to [15], we define

$$\mathbf{U}_i^{(\backslash n)} = \Big[\mathbf{A}_{i-1}^{(t)} \mathbf{U}_{i-1} \otimes \ldots \otimes \mathbf{A}_1^{(t)} \mathbf{U}_1 \otimes \ldots \otimes \mathbf{A}_N^{(t)} \mathbf{U}_N \otimes \ldots \otimes \mathbf{A}_{i+1}^{(t)} \mathbf{U}_{i+1}\Big],$$

which collects Kronecker products of mode matrices except for $\mathbf{A}_i \mathbf{U}_i$ in a backward cyclic manner.

The gradients for (1) wrt $\mathbf{U}_i$ for $i = 1:N$ and $\boldsymbol{\mathcal{G}}$ can be computed as following:

$$\begin{aligned}\frac{\partial F}{\partial \mathbf{U}_i} &= -(\mathbf{A}_i^{(t)})^\top \boldsymbol{\mathcal{R}}_{(i)}^{(t)} \mathbf{U}_i^{(\backslash n)} \boldsymbol{\mathcal{G}}_{(i)}^\top + 2\lambda_i \mathbf{U}_i \\ \frac{\partial F}{\partial \boldsymbol{\mathcal{G}}} &= -\boldsymbol{\mathcal{R}}^{(t)} \times_1 (\mathbf{A}_1^{(t)}\mathbf{U}_1)^\top \times_2 \ldots \times_N (\mathbf{A}_N^{(t)}\mathbf{U}_N)^\top \\ &\quad + 2\lambda_g \boldsymbol{\mathcal{G}},\end{aligned} \tag{3}$$

where

$$\boldsymbol{\mathcal{R}}^{(t)} = \boldsymbol{\mathcal{X}}^{(t)} - \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}_1^{(t)}\mathbf{U}_1 \times_2 \ldots \times_N \mathbf{A}_N^{(t)}\mathbf{U}_N.$$

By updating the variables using gradients given in (3), we can recover the missing entries in $\boldsymbol{\mathcal{X}}^{(t)}$ at every time step t, however that is equivalent to performing a static tensor completion at every time step. Therefore, we need an incremental scheme for updating the variables. Let $\mathbf{U}_i^{(t)}$ and $\boldsymbol{\mathcal{G}}^{(t)}$ represent the variables at time step t, then

$$\begin{aligned}&F(\boldsymbol{\mathcal{X}}^{(t)}, \boldsymbol{\mathcal{A}}^{(t)}, \boldsymbol{\mathcal{G}}^{(t-1)}, \{\mathbf{U}_i^{(t-1)}\}_{i=1:N}) = \\ &\quad F(\boldsymbol{\mathcal{X}}^{(t-1)}, \boldsymbol{\mathcal{A}}^{(t-1)}, \boldsymbol{\mathcal{G}}^{(t-1)}, \{\mathbf{U}_i^{(t-1)}\}_{i=1:N}) \; + \\ &\quad F(\boldsymbol{\mathcal{X}}^{(\Delta t)}, \boldsymbol{\mathcal{A}}^{(\Delta t)}, \boldsymbol{\mathcal{G}}^{(t-1)}, \{\mathbf{U}_i^{(t-1)}\}_{i=1:N}),\end{aligned} \tag{4}$$

since $\boldsymbol{\mathcal{X}}^{(t-1)}$ is recovered at the time step t-1, the problem is equivalent to using only $F^{(\Delta t)} = F(\boldsymbol{\mathcal{X}}^{(\Delta t)}, \boldsymbol{\mathcal{A}}^{(\Delta t)}, \boldsymbol{\mathcal{G}}^{(t-1)}, \{\mathbf{U}_i^{(t-1)}\}_{i=1:N})$ for updating the variables at time step t.

We propose to use the following approach to update the variables at every time step t, i.e.,

$$\begin{aligned}\mathbf{U}_i^{(t)} &= \mathbf{U}_i^{(t-1)} - \gamma \frac{\partial F^{(\Delta t)}}{\partial \mathbf{U}_i^{(t-1)}}, i = 1:N \\ \boldsymbol{\mathcal{G}}^{(t)} &= \boldsymbol{\mathcal{G}}^{(t-1)} - \gamma \frac{\partial F^{(\Delta t)}}{\partial \boldsymbol{\mathcal{G}}^{(t-1)}},\end{aligned} \tag{5}$$

where γ is the step size for the gradients. $\boldsymbol{\mathcal{R}}^{(\Delta t)}$, needed for computing the gradients of $F^{(\Delta t)}$, is given by

$$\begin{aligned}\boldsymbol{\mathcal{R}}^{(\Delta t)} = \boldsymbol{\mathcal{X}}^{(\Delta t)} - \boldsymbol{\mathcal{G}}^{(t-1)} \times_1 \mathbf{A}_1^{(\Delta t)}\mathbf{U}_1^{(t-1)} \times_2 \ldots \\ \times_N \mathbf{A}_N^{(\Delta t)}\mathbf{U}_N^{(t-1)}.\end{aligned} \tag{6}$$

Algorithm 1 summarizes the procedure described above. The computational cost of implementing Algorithm 1 depends on the update of the variables (5) and the computations in (6). The cost of computing $\boldsymbol{\mathcal{R}}^{(\Delta t)}$ is $O(\sum_i I_i M_i r_i + |\Omega| r_1 \ldots r_N)$. The cost of performing the updates (5) is $O(|\Omega| r_1 \ldots r_N + \sum_i M_i r_i)$. Overall, at every time step, the computational cost of Algorithm 1 is $O(K(\sum_i I_i M_i r_i + |\Omega| r_1 \ldots r_N))$.

Extension to the nonnegative case: NN-SIITA

We now discuss how nonnegative constraints can be incorporated into the decomposition learned by SIITA. Nonnegative constraints allow the factor of the tensor to be interpretable.

We denote SIITA with nonnegative constraints with NN-SIITA. At every time step t in the multi-aspect streaming setting, we seek to learn the following decomposition:

$$\min_{\substack{\boldsymbol{\mathcal{G}} \in \mathbb{R}_+^{r_1 \times \ldots \times r_N} \\ \mathbf{U}_i \in \mathbb{R}_+^{M_i \times r_i}, i=1:N}} F(\boldsymbol{\mathcal{X}}^{(t)}, \boldsymbol{\mathcal{A}}^{(t)}, \boldsymbol{\mathcal{G}}, \{\mathbf{U}_i\}_{i=1:N}), \tag{7}$$

where $F(\cdot)$ is as given in (2).

Algorithm 1: Proposed SIITA Algorithm

Input : $\{\boldsymbol{\mathcal{X}}^{(t)}, \boldsymbol{\mathcal{A}}^{(t)}\}, \lambda_i, i = 1:N, (r_1, \ldots, r_N)$

Randomly initialize $\mathbf{U}_i^{(0)} \in \mathbb{R}^{M_i \times r_i}$, $i = 1:N$ and $\boldsymbol{\mathcal{G}}^{(0)} \in \mathbb{R}^{r_i \times \ldots \times r_N}$;

for *t = 1, 2, …* **do**

  $\mathbf{U}_i^{(t)0} := \mathbf{U}_i^{(t-1)}$, $i = 1:N$;

  $\boldsymbol{\mathcal{G}}^{(t)0} := \boldsymbol{\mathcal{G}}^{(t-1)}$;

  for *k = 1:K* **do**

    Compute $\boldsymbol{\mathcal{R}}^{(\Delta t)}$ from (6) using $\mathbf{U}_i^{(t)k-1}$, $i = 1:N$ and $\boldsymbol{\mathcal{G}}^{(t)k-1}$;

    Compute $\frac{\partial F^{(\Delta t)}}{\partial \mathbf{U}_i^{(t)k-1}}$ for $i = 1:N$ from (3);

    Update $\mathbf{U}_i^{(t)k}$ using $\frac{\partial F^{(\Delta t)}}{\partial \mathbf{U}_i^{(t)k-1}}$ and $\mathbf{U}_i^{(t)k-1}$ in (5) ;

    Compute $\frac{\partial F^{(\Delta t)}}{\partial \boldsymbol{\mathcal{G}}^{(t)k-1}}$ from (3);

    Update $\boldsymbol{\mathcal{G}}^{(t)k}$ using $\boldsymbol{\mathcal{G}}^{(t)k-1}$ and $\frac{\partial F^{(\Delta t)}}{\partial \boldsymbol{\mathcal{G}}^{(t)k-1}}$ in (5);

  end

  $\mathbf{U}_i^{(t)} := \mathbf{U}_i^{(t)K}$;

  $\boldsymbol{\mathcal{G}}^{(t)} := \boldsymbol{\mathcal{G}}^{(t)K}$;

end

Return: $\mathbf{U}_i^{(t)}$, $i = 1:N$, $\boldsymbol{\mathcal{G}}^{(t)}$.

We employ a projected gradient descent based algorithm for solving the optimization problem in (7). We follow the same incremental update scheme discussed in Algorithm 1, however we use a projection operator defined below for updating the variables. For NN-SIITA, (5) is replaced with

$$\begin{aligned}\mathbf{U}_i^{(t)} &= \Pi_+[\mathbf{U}_i^{(t-1)} - \gamma \frac{\partial F^{(\Delta t)}}{\partial \mathbf{U}_i^{(t-1)}}], i = 1:N \\ \boldsymbol{\mathcal{G}}^{(t)} &= \Pi_+[\boldsymbol{\mathcal{G}}^{(t-1)} - \gamma \frac{\partial F^{(\Delta t)}}{\partial \boldsymbol{\mathcal{G}}^{(t-1)}}],\end{aligned}$$

where Π_+ is the element-wise projection operator defined as

$$\Pi_+[x_i] = \begin{cases} x_i, & \text{if } x_i > 0 \\ 0, & \text{otherwise.} \end{cases}$$

The projection operator maps a point back to the feasible region ensuring that the factor matrices and the core tensor are always nonnegative with iterations.

5 EXPERIMENTS

We evaluate SIITA against other state-of-the-art baselines in two dynamic settings viz., (1) multi-aspect streaming setting (Section 5.1), and (2) traditional streaming setting (Section 5.2). We then evaluate effectiveness of SIITA in the non-streaming batch setting (Section 5.3). We analyze the effect of different types of side information in Section 5.4. Finally, we evaluate the performance of NN-SIITA in the unsupervised setting in Section 5.5.

Datasets: Datasets used in the experiments are summarized in Table 2. **MovieLens 100K** [9] is a standard movie recommendation dataset. **YELP** is a downsampled version of the YELP(Full) dataset [12]. The YELP(Full) review dataset consists of 70K (user) × 15K (business) × 108 (year-month) tensor, and a side information matrix of size 15K (business) × 68 (city). We select a subset of this dataset for comparisons as the considered baselines algorithms cannot scale to the full dataset. We note that SIITA, our proposed method,

Table 2: Summary of datasets used in the paper. The starting size and increment size given in the table are for Multi-Aspect Streaming setting. For Streaming setting, the tensor grows in the third dimension, one slice at every time step.

	MovieLens 100K	YELP
Modes	user × movie × week	user × business × year-month
Tensor Size	943×1682×31	1000×992×93
Starting size	19×34×2	20×20×2
Increment step	19, 34, 1	20, 20, 2
Sideinfo matrix	1682 (movie) × 19 (genre)	992 (business) × 56 (city)

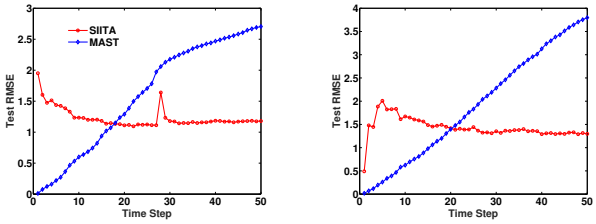
Table 3: Test RMSE (lower is better) averaged across all the time steps in the multi-aspect streaming tensor sequence setting (Definition 4) for MAST and SIITA. SIITA, the proposed method, outperforms MAST for all the datasets. Section 5.1 provides more details.

Dataset	Missing%	Rank	MAST	SIITA
MovieLens 100K	20%	3	1.60	**1.23**
		5	1.53	1.29
		10	1.48	2.49
	50%	3	1.74	**1.28**
		5	1.75	1.29
		10	1.64	2.55
	80%	3	2.03	**1.59**
		5	1.98	1.61
		10	2.02	2.96
YELP	20%	3	1.90	**1.43**
		5	1.92	1.54
		10	1.93	4.03
	50%	3	1.94	**1.51**
		5	1.94	1.67
		10	1.96	4.04
	80%	3	1.97	1.71
		5	1.97	**1.61**
		10	1.97	3.49

doesn't have such scalability concerns. In Section 5.4, we show that SIITA scales to datasets of much larger sizes. In order to create YELP out of YELP(Full), we select the top frequent 1000 users and top 1000 frequent businesses and create the corresponding tensor and side information matrix. After the sampling, we obtain a tensor of size 1000 (user) × 992 (business) × 93 (year-month) and a side information matrix of dimensions 992 (business) × 56 (city).

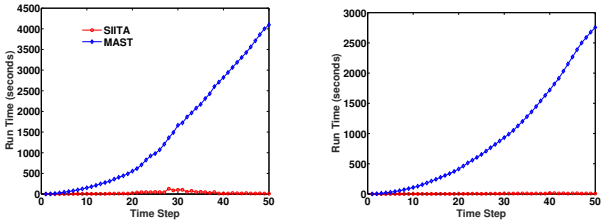
5.1 Multi-Aspect Streaming Setting

We first analyze the model in the multi-aspect streaming setting, for which we consider MAST [27] as a state-of-the-art baseline.



(a) MovieLens 100K (20% Missing) (b) YELP (20% Missing)

Figure 2: Evolution of test RMSE of MAST and SIITA with each time step. For both the datasets, SIITA attains a stable performance after a few time steps, while the performance of MAST degrades with every time step. Refer to Section 5.1.



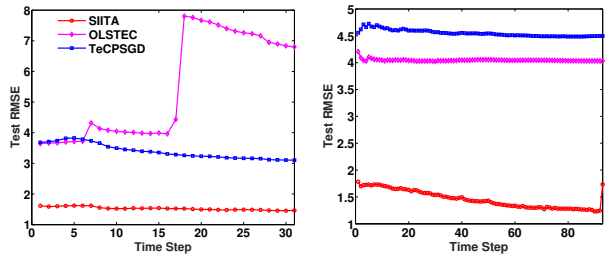
(a) MovieLens 100K (20% Missing) (b) YELP (20% Missing)

Figure 3: Runtime comparison between MAST and SIITA at every time step. SIITA is significantly faster than MAST. Refer to Section 5.1.

MAST [27]: MAST is a dynamic low-rank tensor completion algorithm, which enforces nuclear norm regularization on the decomposition matrices of CP. A tensor-based Alternating Direction Method of Multipliers is used for solving the optimization problem.

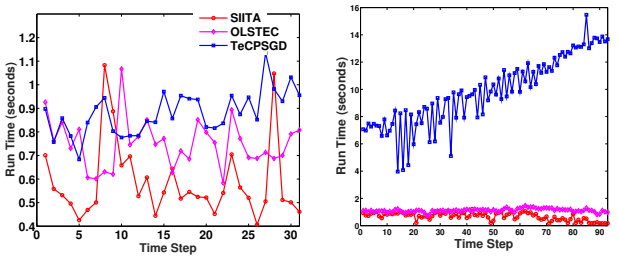
We experiment with the MovieLens 100K and YELP datasets. Since the third mode is time in both the datasets, i.e., (week) in MovieLens 100K and (year-month) in YELP, one way to simulate the multi-aspect streaming sequence (Definition 3) is by considering every slice in third-mode as one time step in the sequence, and letting the tensor grow along other two modes with every time step, similar to the ladder structure given in [27, Section 3.3]. Note that this is different from the traditional streaming setting, where the tensor only grows in time mode while the other two modes remain fixed. In contrast, in the multi-aspect setting here, there can be new users joining the system within the same month but on different days or different movies getting released on different days in the same week etc. Therefore in our simulations, we consider the third mode as any normal mode and generate a more general multi-aspect streaming tensor sequence, the details are given in Table 2. The parameters for MAST are set based on the guidelines provided in [27, Section 4.3].

We compute the root mean square error on test data (test RMSE; lower is better) at every time step and report the test RMSE averaged across all the time steps in Table 3. We perform experiments on



(b) MovieLens 100K (20% Missing) (a) YELP (20% Missing)

Figure 4: Evolution of Test RMSE of TeCPSGD, OLSTEC and SIITA with each time step. In both datasets, SIITA performs significantly better than the baseline algorithms in the pure streaming setting. Refer to Section 5.2 for more details.



(b) MovieLens 100K (20% Missing) (a) YELP (20% Missing)

Figure 5: Runtime comparison between TeCPSGD, OLSTEC and SIITA. SIITA is able to exploit sparsity in the data and is much faster. Refer to Section 5.2 for more details.

multiple train-test splits for each dataset. We vary the test percentage, denoted by *Missing%* in Table 3, and the rank of decomposition, denoted by *Rank* for both the datasets. For every (Missing%, Rank) combination, we run both models on ten random train-test splits and report the average. For SIITA, Rank = r in Table 3 represents the Tucker-rank (r, r, r).

In Table 3, the proposed SIITA achieves better results than MAST. Figure 2 shows the plots for test RMSE at every time step. Since SIITA handles the sparsity in the data effectively, as a result SIITA is significantly faster than MAST, which can be seen from Figure 3. Overall, we find that SIITA, the proposed method, is more effective and faster compared to MAST in the multi-aspect streaming setting.

5.2 Streaming Setting

In this section, we simulate the pure streaming setting by letting the tensor grow only in the third mode at every time step. The number of time steps for each dataset in this setting is the dimension of the third mode, i.e., 31 for MovieLens 100K and 93 for YELP. We compare the performance of SIITA with TeCPSGD and OLSTEC algorithms in the streaming setting.

TeCPSGD [18]: TeCPSGD is an online Stochastic Gradient Descent based algorithm for recovering missing data in streaming tensors.

Table 4: Test RMSE averaged across all the time steps in the streaming setting for TeCPSGD, OLSTEC, a state-of-the-art streaming tensor completion algorithm, and SIITA. SIITA outperforms the baseline algorithms significantly. See Section 5.2 for more details.

Dataset	Missing%	Rank	TeCPSGD	OLSTEC	SIITA
MovieLens 100K	20%	3	3.39	5.46	**1.53**
		5	3.35	4.65	1.54
		10	3.19	4.96	1.71
	50%	3	3.55	8.39	**1.63**
		5	3.40	6.73	1.64
		10	3.23	3.66	1.73
	80%	3	3.78	3.82	1.79
		5	3.77	3.80	**1.75**
		10	3.84	4.34	2.47
YELP	20%	3	4.55	4.04	**1.45**
		5	4.79	4.04	1.59
		10	5.17	4.03	2.85
	50%	3	4.67	4.03	**1.55**
		5	5.03	4.03	1.67
		10	5.25	4.03	2.69
	80%	3	4.99	4.02	**1.73**
		5	5.17	4.02	1.78
		10	5.31	4.01	2.62

This algorithm is based on PARAFAC decomposition. TeCPSGD is the first proper tensor completion algorithm in the dynamic setting.

OLSTEC [13]: OLSTEC is an online tensor tracking algorithm for partially observed data streams corrupted by noise. OLSTEC is a second order stochastic gradient descent algorithm based on CP decomposition exploiting recursive least squares. OLSTEC is the state-of-the-art for streaming tensor completion.

We report test RMSE, averaged across all time steps, for both MovieLens 100K and YELP datasets. Similar to the multi-aspect streaming setting, we run all the algorithms for multiple train-test splits. For each split, we run all the algorithms with different ranks. For every (Missing%, Rank) combination, we run all the algorithms on ten random train-test splits and report the average. SIITA significantly outperforms all the baselines in this setting, as shown in Table 4. Figure 4 shows the average test RMSE of every algorithm at every time step. From Figure 5 it can be seen that SIITA takes much less time compared to other algorithms. The spikes in the plots suggest that the particular slices are relatively less sparse.

5.3 Batch Setting

Even though our primary focus is on proposing an algorithm for the multi-aspect streaming setting, SIITA can be run as a tensor completion algorithm with side information in the batch (i.e., non streaming) setting. To run in batch setting, we set $K = 1$ in Algorithm 1 and run for multiple passes over the data. In this setting, AirCP [7] is the current state-of-the-art algorithm which is also capable of handling side information. We consider AirCP as the baseline in this section. The main focus of this setting is to demonstrate that SIITA incorporates the side information effectively.

Table 5: Mean Test RMSE across multiple train-test splits in the Batch setting. SIITA achieves lower test RMSE on both the datasets compared to AirCP, a state-of-the-art algorithm for this setting. Refer to Section 5.3 for details.

Dataset	Missing%	Rank	AirCP	SIITA
MovieLens 100K	20%	3	3.351	**1.534**
		5	3.687	1.678
		10	3.797	2.791
	50%	3	3.303	**1.580**
		5	3.711	1.585
		10	3.894	2.449
	80%	3	3.883	**1.554**
		5	3.997	1.654
		10	3.791	3.979
YELP	20%	3	1.094	**1.052**
		5	1.086	1.056
		10	1.077	1.181
	50%	3	1.096	1.097
		5	1.095	**1.059**
		10	1.719	1.599
	80%	3	1.219	1.199
		5	**1.118**	1.156
		10	2.210	2.153

AirCP [7]: AirCP is a CP based tensor completion algorithm proposed for recovering the spatio-temporal dynamics of online memes. This algorithm incorporates auxiliary information from memes, locations and times. An alternative direction method of multipliers (ADMM) based algorithm is employed for solving the optimization. AirCP expects the side information matrices to be similarity matrices and takes input the Laplacian of the similarity matrices. However, in the datasets we experiment with, the side information is available as feature matrices. Therefore, we consider the covariance matrices $\mathbf{A}_i\mathbf{A}_i^\top$ as similarity matrices.

We run both algorithms till convergence and report test RMSE. For each dataset, we experiment with different levels of test set sizes, and for each such level, we run our experiments on 10 random splits. We report the mean test RMSE per train-test percentage split. We run our experiments with multiple ranks of factorization. Results are shown in Table 5, where we observe that SIITA achieves better results. Note that the rank for SIITA is the Tucker rank, i.e., rank = 3. This implies a factorization rank of (3, 3, 3) for SIITA.

Remark: Since all the baselines considered for various settings are CP based, we only compare for CP tensor rank. From Tables 3, 4 and 5 it can be seen that the performance suffers for rank = 10. However, when we run SIITA with a rank = (10, 10, 2) we achieve a lower test RMSE.

5.4 Analyzing Merits of Side Information

Our goal in this paper is to propose a flexible framework using which side information may be easily incorporated during incremental tensor completion, especially in the multi-aspect streaming setting. Our proposed method, SIITA, is motivated by this need. In order to evaluate merits of different types of side information on SIITA, we report several experiments where performances of SIITA with and without various types of side information are compared.

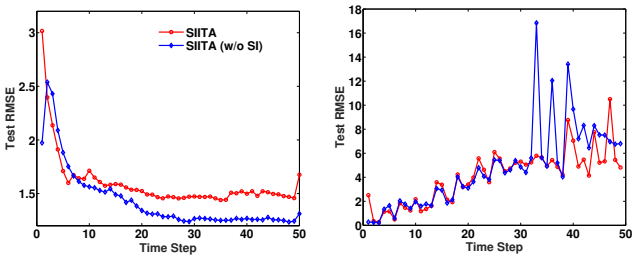
Table 6: Test RMSE averaged across multiple train-test splits in the Multi-Aspect Streaming setting, analyzing the merits of side information. See Section 5.4 for more details.

Dataset	Missing%	Rank	SIITA (w/o SI)	SIITA
MovieLens 100K	20%	3	**1.19**	1.23
		5	**1.19**	1.29
		10	2.69	2.49
	50%	3	**1.25**	1.28
		5	**1.25**	1.29
		10	3.28	2.55
	80%	3	1.45	1.59
		5	**1.42**	1.61
		10	2.11	2.96
YELP	20%	3	1.44	**1.43**
		5	1.48	1.54
		10	3.90	4.03
	50%	3	1.57	**1.51**
		5	1.62	1.67
		10	5.48	4.04
	80%	3	1.75	1.71
		5	1.67	**1.61**
		10	5.28	3.49

Table 7: Test RMSE averaged across multiple train-test splits in the streaming setting, analyzing the merits of side information. See Section 5.4 for more details.

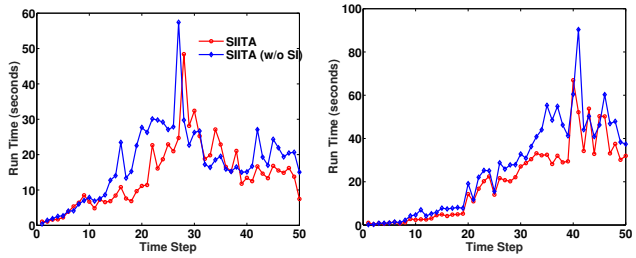
Dataset	Missing%	Rank	SIITA (w/o SI)	SIITA
MovieLens 100K	20%	3	**1.46**	1.53
		5	1.53	1.54
		10	1.55	1.71
	50%	3	1.58	1.63
		5	1.67	1.64
		10	**1.56**	1.73
	80%	3	1.76	1.79
		5	**1.74**	1.75
		10	2.31	2.47
YELP	20%	3	1.46	**1.45**
		5	1.62	1.59
		10	2.82	2.85
	50%	3	1.57	**1.55**
		5	1.69	1.67
		10	2.54	2.67
	80%	3	1.76	**1.73**
		5	1.80	1.78
		10	2.25	2.62

Single Side Information: In the first experiment, we compare SIITA with and without side information (by setting side information to identity; see Section 3). We run the experiments in both multi-aspect streaming and streaming settings. Table 6 reports the mean test RMSE of SIITA and SIITA (w/o SI), which stands for running SIITA without side information, for both datasets in multi-aspect streaming setting. For MovieLens 100K, SIITA achieves better



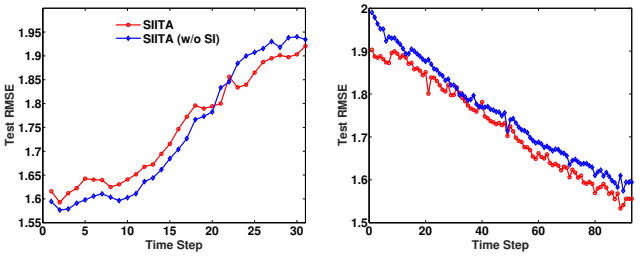
(b) MovieLens 100K (80% Missing) (a) YELP (80% Missing)

Figure 6: Evolution of test RMSE with every time step in the multi-aspect streaming setting for SIITA and SIITA (w/o SI). See Section 5.4 for more details.



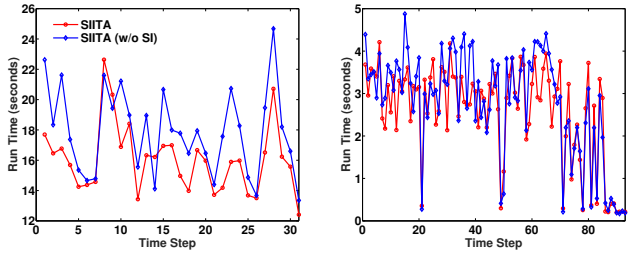
(b) MovieLens 100K (80% Missing) (a) YELP (80% Missing)

Figure 7: Runtime comparison between SIITA and SIITA (w/o SI) in the multi-aspect streaming setting. See Section 5.4 for more details.



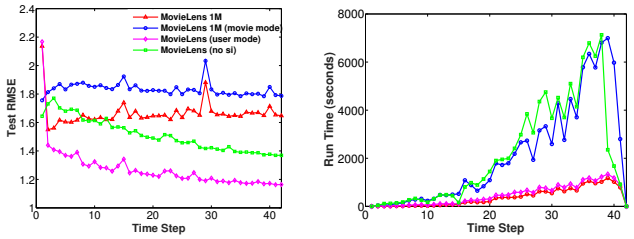
(b) MovieLens 100K (80% Missing) (a) YELP (80% Missing)

Figure 8: Evolution of test RMSE with every time step in the streaming setting for SIITA and SIITA(w/o SI). See Section 5.4 for more details.



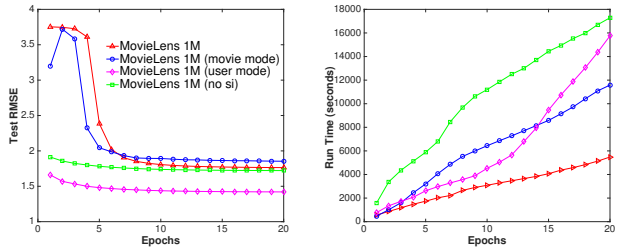
(b) MovieLens 100K (80% Missing) (a) YELP (80% Missing)

Figure 9: Runtime comparison between SIITA and SIITA (w/o SI) in the Streaming setting. See Section 5.4 for more details.



(a) Test RMSE at every time step (b) Runtime at every time step

Figure 10: Investigating the merits of side information for MovieLens 1M dataset in the multi-aspect streaming setting. Side information along the user mode is the most useful for tensor completion. See Section 5.4 for more details.

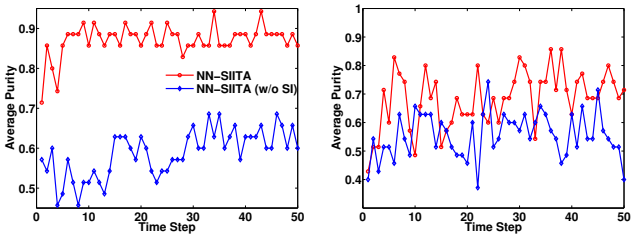


(a) Evolution of Test RMSE. (b) Time elapsed with every epoch.

Figure 11: Investigating the merits of side information for the MovieLens 1M dataset in the batch setting. Side information along the user mode is the most useful for tensor completion. See Section 5.4 for more details.

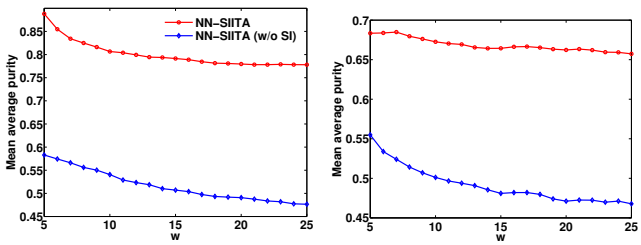
performance without side information. Whereas for YELP, SIITA performs better with side information. Figure 6 shows the evolution of test RMSE at every time step. Figure 7 shows the runtime of SIITA when run with and without side information. SIITA runs faster in the presence of side information. Table 7 reports the mean test RMSE for both the datasets in the streaming setting. Similar to the multi-aspect streaming setting, SIITA achieves better performance without side information for MovieLens 100K dataset and with side information for YELP dataset. Figure 8 shows the test RMSE of SIITA against time steps, with and without side information. Figure 9 shows the runtime at every time step.

Multi Side Information: In all the datasets and experiments considered so far, side information along only one mode is available to SIITA. In this next experiment, we consider the setting where side information along multiple modes are available. For this experiment,



(a) MovieLens 100K (b) YELP

Figure 12: Average-Purity of clusters learned by NN-SIITA and NN-SIITA (w/o SI) in the unsupervised setting. For both datasets, side information helps in learning purer clusters. See Section 5.5 for more details.



(a) MovieLens 100K (b) YELP

Figure 13: Evolution of mean average-purity with w for NN-SIITA and NN-SIITA (w/o SI) for both MovieLens 100K and YELP datasets. See Section 5.5 for more details.

we consider the **MovieLens 1M** [9] dataset, a standard dataset of 1 million movie ratings. This dataset consists of a 6040 (user) × 3952 (movie) × 149 (week) tensor, along with two side information matrices: a 6040 (user) × 21 (occupation) matrix, and a 3952 (movie) × 18 (genre) matrix.

Note that among all the methods considered in the paper, SIITA is the only method which scales to the size of MovieLens 1M datasets.

We create four variants of the dataset. The first one with the tensor and all the side information matrices denoted by MovieLens 1M, the second one with the tensor and only the side information along the movie mode denoted by MovieLens 1M (movie mode). Similarly, MovieLens (user mode) with only user mode side information, and finally MovieLens 1M (no si) with only the tensor and no side information.

We run SIITA in multi-aspect streaming and batch modes for all the four variants. Test RMSE at every time step in the multi-aspect streaming setting is shown in Figure 10(a). Evolution of Test RMSE (lower is better) against epochs are shown in Figure 11(a) in batch mode. From Figures 10(a) and 11(a), it is evident that the variant MovieLens 1M (user mode) achieves best overall performance, implying that the side information along the user mode is more useful for tensor completion in this dataset. However, MovieLens 1M (movie mode) achieves poorer performance than other variants implying that movie-mode side information is not useful for tensor completion in this case. This is also the only side information mode available to SIITA during the MovieLens 100K experiments in Tables 6 and 7. This sub-optimal side information may be a reason for SIITA's diminished performance when using side information for MovieLens100K dataset. From the runtime comparisons in Figures 11 (b) and 10(b), we observe that MovieLens 1M (where both types of side information are available) takes the least time, while the variant MovieLens 1M (no si) takes the most time to run. This is a benefit we derive from the inductive framework, where in the presence of useful side information, SIITA not only helps in achieving better performance but also runs faster.

5.5 Unsupervised Setting

In this section, we consider an unsupervised setting with the aim to discover underlying clusters of the items, like movies in the MovieLens 100K dataset and businesses in the YELP dataset, from a sequence of sparse tensors. It is desirable to mine clusters such that similar items are grouped together. Nonnegative constraints are essential for mining interpretable clusters [10, 20]. For this set of experiments, we consider the nonnegative version of SIITA denoted by NN-SIITA. We investigate whether side information helps in discovering more coherent clusters of items in both datasets.

We run our experiments in the multi-aspect streaming setting. At every time step, we compute *Purity* of clusters and report average-Purity. Purity of a cluster is defined as the percentage of the cluster that is coherent. For example, in MovieLens 100K, a cluster of movies is 100% pure if all the movies belong to the same genre and 50% pure if only half of the cluster belong to the same genre. Formally, let clusters of items along mode-i are desired, let r_i be the rank of factorization along mode-i. Every column of the matrix $\mathbf{A}_i\mathbf{U}_i$ is considered a distribution of the items, the top-w items of the distribution represent a cluster. For p-th cluster, i.e., cluster representing column p of the matrix $\mathbf{A}_i\mathbf{U}_i$, let w_p items among the top-w items belong to the same category, Purity and average-Purity are defined as follows:

$$\text{Purity}(p) = w_p/w,$$

$$\text{average-Purity} = \frac{1}{r_i}\sum_{p=1}^{r_i}\text{Purity}(p).$$

Note that Purity is computed per cluster, while average-Purity is computed for a set of clusters. Higher average-Purity indicates a better clustering.

We report average-Purity at every time step for both the datasets. We run NN-SIITA with and without side information. Figure 12 shows average-Purity at every time step for MovieLens 100K and YELP datasets. It is clear from Figure 12 that for both the datasets side information helps in discovering better clusters. We compute the Purity for MovieLens 100K dataset based on the genre information of the movies and for the YELP dataset we compute Purity based on the geographic locations of the businesses. Table 8 shows some example clusters learned by NN-SIITA. For MovieLens 100K dataset, each movie can belong to multiple genres. For computing the Purity, we consider the most common genre for all the movies in a cluster. Results shown in Figure 12 are for $w = 5$. However, we also vary w between 5 and 25 and report the *mean* average-Purity, which is obtained by computing the mean across all the time steps

Table 8: Example clusters learned by NN-SIITA for MovieLens 100K and YELP datasets. The first column is an example of a pure cluster and the second column is an example of noisy cluster. See Section 5.5 for more details.

	Cluster (Action, Adventure, Sci-Fi)		Cluster (Noisy)	
	Movie	Genres	Movie	Genres
MovieLens100K	The Empire Strikes Back (1980)	Action, Adventure, Sci-Fi, Drama, Romance	Toy Story (1995)	Animation, Children's, Comedy
	Heavy Metal (1981)	Action, Adventure, Sci-Fi, Animation, Horror	From Dusk Till Dawn (1996)	Action, Comedy, Crime, Horror, Thriller
	Star Wars (1977)	Action, Adventure, Sci-Fi, Romance, War	Mighty Aphrodite (1995)	Comedy
	Return of the Jedi (1983)	Action, Adventure, Sci-Fi, Romance, War	Apollo 13 (1995)	Action, Drama, Thriller
	Men in Black (1997)	Action, Adventure, Sci-Fi, Comedy	Crimson Tide (1995)	Drama, Thriller, War
	Cluster (Phoenix)		Cluster (Noisy)	
	Business	Location	Business	Location
YELP	Hana Japanese Eatery	Phoenix	The Wigman	Litchfield Park
	Herberger Theater Center	Phoenix	Hitching Post 2	Gold Canyon
	Scramble A Breakfast Joint	Phoenix	Freddys Frozen Custard & Steakburgers	Glendale
	The Arrogant Butcher	Phoenix	Costco	Avondale
	FEZ	Phoenix	Hana Japanese Eatery	Phoenix

in the multi-aspect streaming setting. As can be seen from Figure 13, having side information helps in learning better clusters for all the values of w. For MovieLens 100K, the results reported are with a factorization rank of $(3, 7, 3)$ and for YELP, the rank of factorization is $(5, 7, 3)$. Since this is an unsupervised setting, note that we use the entire data for factorization, i.e., there is no train-test split.

6 CONCLUSION

We propose an inductive framework for incorporating side information for tensor completion in standard and multi-aspect streaming settings. The proposed framework can also be used in the batch setting. Given a completely new dataset with side information along multiple modes, SIITA can be used to analyze the merits of different side information for tensor completion. Besides performing better, SIITA is also significantly faster than state-of-the-art algorithms. We also propose NN-SIITA for handling nonnegative constraints and show how it can be used for mining interpretable clusters. Our experiments confirm the effectiveness of SIITA in many instances. In future, we plan to extend our proposed framework to handle missing side information problem instances [32].

Acknowledgement

This work is supported in part by the Ministry of Human Resource Development (MHRD), Government of India.

REFERENCES

[1] E. Acar, T. Kolda, and D. Dunlavy. 2011. All-at-once Optimization for Coupled Matrix and Tensor Factorizations. In *MLG*. arXiv:1105.3422
[2] A. Beutel, P. Talukdar, A. Kumar, C. Faloutsos, E. Papalexakis, and E. Xing. 2014. Flexifact: Scalable flexible factorization of coupled tensors on hadoop. In *SDM*.
[3] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. Phan. 2015. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine* 32, 2 (2015), 145–163.
[4] B. Ermiş, E. Acar, and A. Cemgil. 2015. Link prediction in heterogeneous data via generalized coupled tensor factorization. In *KDD*.
[5] H. Fanaee-T and J. Gama. 2015. Multi-aspect-streaming Tensor Analysis. *Know.-Based Syst.* 89 (2015), 332–345.
[6] M. Filipović and A. Jukić. 2015. Tucker factorization with missing data with application to low-n-rank tensor completion. *Multidimens Syst Signal Process* (2015).
[7] H. Ge, J. Caverlee, N. Zhang, and A. Squicciarini. 2016. Uncovering the Spatio-Temporal Dynamics of Memes in the Presence of Incomplete Information *(CIKM)*. 10.
[8] X. Guo, Q. Yao, and J. Kwok. 2017. Efficient sparse low-rank tensor completion using the Frank-Wolfe algorithm. In *AAAI*.
[9] F. Harper and J. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, Article 19 (Dec. 2015), 19 pages.
[10] S. Hyvönen, P. Miettinen, and E. Terzi. 2008. Interpretable nonnegative matrix decompositions. In *KDD*. ACM, 345–353.
[11] P. Jain and I. Dhillon. 2013. Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626* (2013).
[12] B. Jeon, I. Jeon, L. Sael, and U. Kang. 2016. Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries. In *ICDE*.
[13] H. Kasai. 2016. Online Low-Rank Tensor Subspace Tracking from Incomplete Data by CP Decomposition using Recursive Least Squares. In *ICASSP*.
[14] H. Kasai and B. Mishra. 2016. Low-rank tensor completion: a Riemannian manifold preconditioning approach. In *ICML*.
[15] Y. Kim and S. Choi. 2007. Nonnegative tucker decomposition. In *CVPR*.
[16] T. Kolda and B. Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
[17] A. Lefevre, F. Bach, and C. Févotte. 2011. Online algorithms for nonnegative matrix factorization with the Itakura-Saito divergence. In *WASPAA*. IEEE.
[18] M. Mardani, G. Mateos, and G. Giannakis. 2015. Subspace learning and imputation for streaming big data matrices and tensors. *IEEE Transactions on Signal Processing* (2015).
[19] M. Mørup, L. Hansen, and S. Arnfred. 2008. Algorithms for sparse nonnegative Tucker decompositions. *Neural computation* 20, 8 (2008), 2112–2131.
[20] B. Murphy, P. Talukdar, and T. Mitchell. 2012. Learning Effective and Interpretable Semantic Models using Non-Negative Sparse Embedding. In *COLING*.
[21] A. Narita, K. Hayashi, Ryota R. Tomioka, and H. Kashima. 2011. Tensor Factorization Using Auxiliary Information. In *ECML PKDD*.
[22] N. Natarajan and I. Dhillon. 2014. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics* 30, 12 (2014), i60–i68.
[23] M. Nimishakavi, U. Saini, and P. Talukdar. 2016. Relation Schema Induction using Tensor Factorization with Side Information. In *EMNLP*.
[24] D. Nion and N. Sidiropoulos. 2009. Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. *IEEE Transactions on Signal Processing* (2009).
[25] A. Shashua and T. Hazan. 2005. Non-negative Tensor Factorization with Applications to Statistics and Computer Vision. In *ICML*.
[26] S. Si, K. Chiang, C. Hsieh, N. Rao, and I. Dhillon. 2016. Goal-directed inductive matrix completion. In *KDD*.
[27] Q. Song, X. Huang, H. Ge, J. Caverlee, and X. Hu. 2017. Multi-Aspect Streaming Tensor Completion. In *KDD*.
[28] J. Sun, D. Tao, and C. Faloutsos. 2006. Beyond streams and graphs: dynamic tensor analysis. In *KDD*.
[29] J. Sun, D. Tao, S. Papadimitriou, P. Yu, and C. Faloutsos. 2008. Incremental Tensor Analysis: Theory and Applications. *ACM Trans. Knowl. Discov. Data* 2, 3, Article 11 (2008), 37 pages.
[30] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. 2008. Tag Recommendations Based on Tensor Dimensionality Reduction. In *RecSys*.
[31] M. Welling and M. Weber. 2001. Positive tensor factorization. *Pattern Recognition Letters* 22, 12 (2001), 1255–1261.
[32] K. Wimalawarne, M. Yamada, and H. Mamitsuka. 2017. Convex Coupled Matrix and Tensor Completion. *arXiv preprint arXiv:1705.05197* (2017).
[33] R. Yu, D. Cheng, and Y. Liu. 2015. Accelerated Online Low-rank Tensor Learning for Multivariate Spatio-temporal Streams. In *ICML*.
[34] R. Zhao, V. Tan, and H. Xu. 2017. Online Nonnegative Matrix Factorization with General Divergences. In *AISTATS*.
[35] S. Zhou, N. Vinh, J. Bailey, Y. Jia, and I. Davidson. 2016. Accelerating online cp decompositions for higher order tensors. In *KDD*.