SPADE: **S**treaming **PA**RAFAC2 **DE**composition for Large Datasets

Ekta Gujral
UC Riverside
egujr001@ucr.edu

Georgios Theocharous
Adobe Inc.
theochar@adobe.com

Evangelos E. Papalexakis
UC Riverside
epapalex@cs.ucr.edu

Abstract

In tensor mining, PARAFAC2 is a powerful and a multi-modal factor analysis method that is ideally suited for modeling for batch processing of data which forms "irregular" tensors, e.g., user movie viewing profiles, where each user's timeline does not necessarily align with other users. However, these days data is dynamically changing which hinders the use of this model for large data. The tracking of the PARAFAC2 decomposition for the dynamic tensors is very pivotal and challenging task due to the variability of incoming data and lack of online efficient algorithm in terms of time and memory.

In this paper, we fill this gap by proposing an efficient method to compute the PARAFAC2 decomposition of streaming large tensor datasets containing millions of entries, called SPADE. In terms of effectiveness, our proposed method shows comparable results with the prior work, PARAFAC2, while being computationally much more efficient. We evaluate SPADE on both synthetic and real datasets, indicatively, our proposed method shows $10-23\times$ speedup and saves $17-150\times$ memory usage over the baseline methods and is also capable of handling larger tensor streams (≈ 7 million users) for which the batch baseline was not able to operate. To the best of our knowledge, SPADE is the first approach to online PARAFAC2 decomposition while not only being able to provide on par accuracy but also provide better performance in terms of scalability and efficiency.

1 Introduction

The PARAFAC1 (CP) decomposition method is used to handle multi-aspect or multi-way data, and the principle is well researched among the data mining community, for example, by Kolda and Bader [14], Bro [5], and Papalexakis et al. [16]. Regardless of recent development on temporal data through classic tensor decomposition approaches [3, 8, 15, 21, 24], there are certain instances [11, 12] wherein time modeling is difficult for the regular tensor factorization methods, due to either data irregularity or time-shifted latent factor appearance as shown in Figure 1. The PARAFAC2 decomposition, proposed by Harshman [10], is another alternative to the PARAFAC1 (CP) model. PARAFAC2 can easily handle sub-matrices of dynamic length as opposed to the PARAFAC1 model which requires fixed data length for which no time-alignment is necessary. The PARAFAC2 model showed a remarkable ability because: a) The actual structure of each slice or sub-matrix is well approximated without any additional parameters. b) As the features (tutorials, movies, etc.), i.e., 2nd mode, of the tensor are uniform across all slices, PARAFAC2 is able to extract the common characteristics by employing such uniformity, which is important in the click-stream problem. c) The PARAFAC2 allows one mode to be irregular (See Figure 1) that is particularly suitable for chromatographic data [2, 19] and electronic health records[18]. d) Similar to the PARAFAC1/CP decomposition method, the PARAFAC2 method provides unique solutions under certain mild assumptions [22, 20], but this model is more loosely constrained and hence, provides more relevant details than PARAFAC1/CP[13].

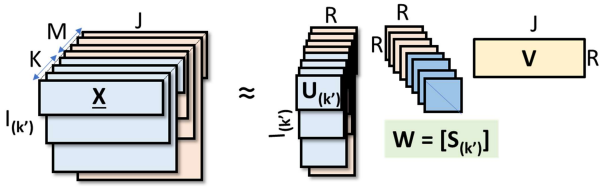

Figure 1: An illustration of the tensor decomposition on streaming PARAFAC2 data.

In the era of information explosion, the data of diverse variety is generated or modified in large volumes. In many cases, data may be added or removed from any of the dimensions with high velocity. When using tensors to represent this dynamically changing data, an instance of the problem is of the form of a "streaming", "incremental", or "online" tensors. Considering an example of Electronic Health Records [18] data as shown in Figure 1, where we have K number of subjects for which we observe J features and we permit each k^{th} subject to have I_k observations. As time grows, a number of subjects M is added with more or fewer observations. Each such subject is a new incoming

slice(s) to the tensor $\underline{\mathbf{X}}_k$ on its N^{th} mode, which is seen as a streaming update. Additionally, the tensor may be growing in all of its N-modes, especially in complex and evolving environments such as online social networks. As shown in Figure 1, PARAFAC2 approximates entire data (old + new) as: $\underline{\mathbf{X}}_k \approx \mathbf{U}_{k'}\mathbf{S}_{k'}\mathbf{V}^T$, where $k^{'} \in [1,(K+M)]$, $\mathbf{U}_{k'} \in \mathbb{R}^{I_{k'} \times R}$, $\mathbf{S}_{k'}$ is a diagonal $R \times R$, $\mathbf{V} \in \mathbb{R}^{J \times R}$ and R is the target rank of the decomposition.

Streaming PARAFAC2 decomposition is a challenging task due to the following reasons. First, to fit the PARAFAC2 model, alternating least squares (ALS) is commonly used. For 3-mode tensor, the estimations of all the three modes are done alternatively and iteratively until no significant changes are observed or local minimum solution is achieved and thus its main drawback is very slow convergence for large datasets that is often observed due to expensive recalculations for the entire tensor. Second, for PARAFAC2 tensor data, any pre-processing to accumulate across any mode may lose significant information. Third, maintaining high-accuracy (competitive to decomposing the full tensor) using significantly fewer computations than the full decomposition calls for innovative and, ideally, sub-linear approaches. Lastly, operating on the full ambient data space, as the tensor is being updated online, leads to an increase in time and space complexity, rendering such approaches is hard to scale, and thus calls for efficient methods that work on memory spaces which are significantly smaller than the original ambient data dimensions.

To handle the above challenges, in this paper, we propose a method to decompose online or incremental tensors based on PARAFAC2 decomposition. Our goal is, given an already computed PARAFAC2 decomposition, to *track* the PARAFAC2 decomposition of an online tensor, as it receives streaming updates, 1) *efficiently*, being much faster than re-computing the entire decomposition from scratch after every update, and utilizing smaller amount of memory, and 2) *accurately*, incurring an approximation error that is as close as possible to the decomposition of the full tensor. Answering the above questions, we propose SPADE (**S**treaming **PA**RAFAC2 **DE**compistion) framework. Our SPADE achieves the best of both worlds in terms of speed and memory efficiency: a) it is faster than a highly-optimized baseline in all cases considered for both real (Figure 3) and synthetic (Table 3, 4, 5) datasets, achieving up to $10-23\times$ performance gain; b) at the same time, SPADE is more scalable, in that it can execute in reasonable time for large problem instances when the baseline fails due to excessive memory consumption (Figure 2). For exposition purposes, we focus on the streaming scenario, where a 3-mode tensor grows on the third mode, however, our work extends to cases where more than one modes are online.

To the best of our knowledge, no work has assessed streaming or online PARAFAC2 for large-scale dense/sparse data, as well as the challenges arising by doing so. Our **contributions** are summarized as follows:

- **Novel Scalable Online Algorithm**: We introduce SPADE, a scalable and effective algorithm for tracking the PARAFAC2 decompositions of online tensors that admits an efficient parallel implementation. We do not limit to 3-mode tensors, our algorithm can easily handle higher-order tensor decompositions. We make our Matlab implementation publicly available on the link[1].
- **Extensive Evaluation** We evaluate the scalability of SPADE using datasets originating from two different application domains, namely a sequential user viewing patterns dataset by Adobe and a time-evolving movie ratings dataset, which is publicly available. Additionally, we perform extensive synthetic data experiments.
- **Real-world case study** We performed a case study of applying SPADE on the dataset by Adobe which consists of a sequence of tutorials watched by $\approx 7Million$ users . The communities discovered were evaluated by an expert from Adobe.

2 Background

In this Section, we provide the necessary background for notations and tensor operations. Then, we briefly discuss the related work regarding the classical and sparse method for PARAFAC2 for tensor factorization. Table 1 contains the symbols used throughout the paper.

Symbols	Definition
$\underline{\mathbf{X}}, \mathbf{X}, \mathbf{x}, x$	Tensor, Matrix, Column vector, Scalar
$\mathbf{X}^T, \mathbf{X}^{-1}, \mathbf{X}^\dagger$	Transpose, Inverse, Pseudo-inverse
$diag(\mathbf{X})$	Extract diagonal of matrix $\mathbf{X}$
$\underline{\mathbf{X}}_k$	shorthand for $\underline{\mathbf{X}}(:,:,k)$ (k-th frontal slice of $\underline{\mathbf{X}}$)
$\underline{\mathbf{X}}^{(n)}$, $\mathbf{X}^{(n)}$	mode-n matricization of $\underline{\mathbf{X}}$, matrix $\mathbf{X}$ at mode-n
$\|\mathbf{A}\|_F$, $\|\mathbf{a}\|_2$	Frobenius norm, ℓ_2 norm
$\circ$, $\circledast$, $\otimes$, $\odot$	Outer, Hadmard, Kronecker and Khatri-Rao product
OoM	Out of Memory
MTTKRP	Matricized tensor times Khatri-Rao product[14]

Table 1: Table of symbols and their description

2.1 PARAFAC1 (CP) Decomposition

Tensor decomposition is a general tool for tensor analysis. Using tensor decomposition, we find latent factor or relations within data. One of the most popular and extensively used tensor decompositions is the Canoni-

[1] http://www.cs.ucr.edu/~egujr001/ucr/madlab/src/SPADE.zip

cal Polyadic (CP) or CANDECOMP/ PARAFAC1 decomposition [6, 5, 4, 14] referred as CP decomposition. Given a N-mode tensor $\underline{\mathbf{X}}$ of dimension $\mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$, its CP decomposition can be written as $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R}$, where $n = (1, 2, \ldots N)$ and R represents the number of latent factors or upper bound rank on tensor $\underline{\mathbf{X}}$. The N-mode CP decomposition of tensor $\underline{\mathbf{X}}$ can be written as

(2.1)
$$\begin{aligned}\mathcal{L} =& \text{argmin} \frac{1}{2} ||\underline{\mathbf{X}}^{(n)} - [\mathbf{U}^{(n)}(\mathbf{U}^{(N)} \cdots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \mathbf{U}^{(1)})^T||_F^2 \\ =& \text{argmin} \frac{1}{2} ||\underline{\mathbf{X}}^{(n)} - [\mathbf{U}^{(n)}(\odot_{i \neq n}^{N} \mathbf{U}^{(i)})^T||_F^2 \quad \forall i \in [1, N]\end{aligned}$$

The direct fitting of Eq. (2.1) is difficult. The most common fitting for CP decomposition is by using ALS (Alternating Least Squares). The main idea behind ALS is the following: when we fix all factor matrices except for one, the problem reduces to a linear least squares problem which can be solved optimally. We refer the interested reader to several well-known surveys that provide more details on tensor decompositions and its applications [14, 16].

2.2 Classic PARAFAC2 Decomposition The PARAFAC2 model was first developed by Harshman [10] to handle the situation where the number of observations (row dimension) in each $\underline{\mathbf{X}}_k$ may vary e.g study of phonetics. In his work, Harshman described a way to factorize multiple matrices simultaneously given that one factor was not exactly the same in all those matrices. This can be solved by imposing orthogonality constraints on a linear transformation as a coupling relationship between the similar factors to ensure identifiability. Hence, the PARAFAC2 model for 3-mode tensor $\underline{\mathbf{X}}_k \in \mathbb{R}^{I_k \times J}$ is given by:

$$\begin{aligned}\mathcal{L} =& \text{argmin}_{\mathbf{U},\mathbf{S},\mathbf{V}} \frac{1}{2} ||\underline{\mathbf{X}}_k - \mathbf{U}_k \mathbf{S}_k \mathbf{V}^T||_2^F \quad \forall k \\ & \text{subject to} \quad \mathbf{U}_k = \mathbf{Q}_k * \mathbf{H} \text{ and } \mathbf{Q}_k \mathbf{Q}_k^T = \mathbf{I}_r\end{aligned} \tag{2.2}$$

where $\mathbf{U}_k \in \mathbb{R}^{I_k \times R}$ are coupled matrices, $\mathbf{H} \in \mathbb{R}^{R \times R}$ is coefficients matrix, $\mathbf{Q}_k \in \mathbb{R}^{I_k \times R}$ are left-orthogonal coupling matrices to ensure uniqueness of factors and $\mathbf{W} = \mathbf{S}_k \in \mathbb{R}^{R \times R}$ is set of diagonal matrix. The Equ (2.2) in form of orthogonal form can be re-written as :

$$\begin{aligned}\mathcal{L} =& \text{argmin}_{\mathbf{Q}} \frac{1}{2} ||\underline{\mathbf{X}}_k - \mathbf{Q}_k \mathbf{H} \mathbf{W} \mathbf{V}^T||_2^F \quad \forall k \\ & \text{subject to} \quad \mathbf{Q}_k \mathbf{Q}_k^T = \mathbf{I}_r\end{aligned} \tag{2.3}$$

To solve Eq (2.3), most common method is Alternating Least Square (ALS) that updates $\mathbf{Q}_k$ by fixing other factor matrices i.e $\mathbf{H}, \mathbf{W}$, and $\mathbf{V}$. The orthogonal coupling matrix $\mathbf{Q}_k$ can be obtained by Singular Value decomposition (SVD) of $(\mathbf{H}\mathbf{W}\mathbf{V}^T \underline{\mathbf{X}}_k^T) = [\mathbf{P}_n, \Sigma_n, \mathbf{Z}_n^T]$. With $\mathbf{Q}_k^T = \mathbf{P}_n \mathbf{Z}_n^T$ fixed, the rest of factors can be obtained as:

(2.4)
$$\begin{aligned}\mathcal{L} =& \text{argmin}_{\mathbf{H},\mathbf{W},\mathbf{V}} \frac{1}{2} ||\mathbf{Q}_k \underline{\mathbf{X}}_k - \mathbf{H}\mathbf{W}\mathbf{V}^T||_2^F s.t. \mathbf{Q}_k \mathbf{Q}_k^T = \mathbf{I}_r \\ & \text{argmin}_{\mathbf{H},\mathbf{W},\mathbf{V}} \frac{1}{2} ||\underline{\mathbf{Y}} - \mathbf{H}\mathbf{W}\mathbf{V}^T||_2^F\end{aligned}$$

The Eq. (2.3) is equivalent of solving CP decomposition of $\underline{\mathbf{Y}}$ using ALS method. The classic method of PARAFAC2 is limited to small sized dense data and require large amount of resources (time and space) to process big data. The author [18] proposed method namely SPARTAN (Scalable PARAFAC2) for large and sparse tensors. The speed up of the process is obtained by modifying core computational kernel.The author [1] proposed constrained version of Scalable PARAFAC2. But these methods are limited to static data. In this era, data is growing very fast and a recipe for handling the limitations is to adapt existing approaches using online techniques. To our best knowledge, there is no work in the literature that deals dynamic PARAFAC2 tensor decomposition. To fill the gap, we propose a scalable and efficient (time and space) method to find the PARAFAC2 decomposition for streaming large-scale high-order PARAFAC2 data and maintain comparable accuracy.

3 Proposed Method: SPADE

In this section, we introduce our proposed method for tracking the PARAFAC2 decomposition of data in an incremental setting. For presentation purposes, initially a 3-mode irregular tensor case will be discussed. Then, we further present our proposed method to handle higher mode tensors. Here, we assume that only last mode of a tensor is increasing over time and other modes remain unchanged over time. Formally, the problem that we solve is the following:

PROBLEM 1. **Given** *(a) an existing set of PARAFAC2 decomposition i.e.* $\mathbf{U}_{old}, \mathbf{V}_{old}$ *and* $\mathbf{W}_{old}$ *factor matrices, having R latent components, that approximate tensor* $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I_k \times J \times K}$ *at time t , (b) new incoming slice (s) in form of tensor* $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_n \times J \times N}$ *at any time* Δt*,*
Find *updates of* $\mathbf{U}_{new}, \mathbf{V}_{new}$ *and* $\mathbf{W}_{new}$ **incrementally** *to approximate PARAFAC2 tensor* $\underline{\mathbf{X}} \in \mathbb{R}^{I_{(k+n)} \times J \times (K+N)}$ *after appending new slice(s) at* $t = t_1 + \Delta t$ *in last mode while maintaining a comparable accuracy with running the full PARAFAC2 decompositon on the entire updated tensor* $\underline{\mathbf{X}}$*.*

3.1 The Principle of SPADE To address the online PARAFAC2 problem, SPADE follows the same alternating update schema as ALS, such that only one factor matrix is updated at one time by fixing all others.

Assumptions:

- The factor matrices $\mathbf{U}_{old}, \mathbf{V}_{old}$ and $\mathbf{W}_{old}$ for old data ($\underline{\mathbf{X}}_{old}$) at time stamp t_1 is available. $\mathbf{U}_{old}$ is obtained using product of $\mathbf{Q}_k$ and $\mathbf{H}_{old}$.
- We have pre-existing supporting matrices $\mathbf{L}_{old}$ and $\mathbf{M}_{old}$ from old data.
- There is no rank or concept drift[17] in the data.

3.1.1 CP slice-wise tensor $\underline{\mathbf{Y}}$ formulation Initiate $\mathbf{W}_{rand}$ random linear combinations of columns of the existing $\mathbf{W}_{old}$ factor matrix. To formulate CP tensor $\underline{\mathbf{Y}}$, we obtain SVD of existing factors and new incoming data as follows:

$$[\mathbf{P}_n, \Sigma_n, \mathbf{Z}_n] = SVD[\mathbf{H}_{old} \times diag(\mathbf{W}_{rand}(n,:)) \times (\underline{\mathbf{X}}_{new_n} \times \mathbf{V}_{old})^T] \tag{3.5}$$

Given the SVD of above equation, the minimum of Eq. (2.3) over left-orthonormal $\mathbf{Q}_n$ is given by $\mathbf{Q}_n = \mathbf{Z}_n\mathbf{P}_n^T$. This equivalence implies that minimizing the objective Equ. (2.4) is achieved by executing the decomposition on a tensor $\underline{\mathbf{Y}} = \mathbf{Q}_n^T\underline{\mathbf{X}}_{new}(n) \in \mathbb{R}^{R\times J\times N}$ with frontal slices as :

$$\mathcal{LS} = \text{argmin}\frac{1}{2}||\underline{\mathbf{Y}} - [\![\mathbf{H}_{new}; \mathbf{V}_{new}; \mathbf{W}_{new}]\!]||_F^2 \tag{3.6}$$

Note that our loading or factor matrices for CP tensor decomposition in Eq. 3.6 are: $\mathbf{H}_{new} \in \mathbb{R}^{R\times R}$, $\mathbf{V}_{new} \in \mathbb{R}^{J\times R}$ and $\mathbf{W}_{new} \in \mathbb{R}^{N\times R}$.

3.1.2 Initialization of Supporting Matrices M can be initialized as MTTKRP [14, 18] w.r.t 1^{st} and 2^{nd} mode of tensor. We use the notation $\mathbf{M}^{(i)}$ to denote the MTTKRP corresponding to the i^{th} tensor mode. For example, for mode-1, it is computed as $\mathbf{M}^{(1)} = \underline{\mathbf{Y}}^{(1)} * (\mathbf{V} \odot \mathbf{W})^T$ and so on. Similarly, supporting matrix $\mathbf{L}$ for mode 'n' can be computed as Hadmard product of all factor matrices expect the n-mode factor matrix. For example, for mode-1 , it can be written as $\mathbf{L}^{(1)} = (\mathbf{W}^T\mathbf{W} * \mathbf{V}^T\mathbf{V})$ and so on.

3.1.3 Update temporal factor Consider first the update of factor $\mathbf{W}^{'}$ obtained after fixing $\mathbf{V}_{old}$ and $\mathbf{H}_{old}$, and solving the corresponding minimization in Equ 3.7.

$$\text{argmin}_{\mathbf{W}}\frac{1}{2}||\underline{\mathbf{Y}}^{(3)}_{new} - \mathbf{W}_{rand}(\mathbf{V}_{old} \odot \mathbf{H}_{old})^T)||_2^F \tag{3.7}$$

where $\mathbf{W}_{rand} \in \mathbb{R}^{N\times R}$ is random linear combinations of columns of the existing $\mathbf{W}_{old}$ matrix. The $\widetilde{\mathbf{W}}$ can be obtained after minimizing above equation as :

$$\begin{aligned}\widetilde{\mathbf{W}} &= \underline{\mathbf{Y}}^{(3)}_{new} * ((\mathbf{V}_{old} \odot \mathbf{H}_{old})^T)^\dagger \\ &= \underline{\mathbf{Y}}^{(3)}_{new} * (\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old})^\dagger * (\mathbf{V}_{old} \odot \mathbf{H}_{old})\end{aligned} \tag{3.8}$$

As the term $(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old})$ is invertible, so its pseudo-inverse is its inverse and Equ (3.9) can be written as:

$$\widetilde{\mathbf{W}} = \frac{\underline{\mathbf{Y}}^{(3)}_{new} * (\mathbf{V}_{old} \odot \mathbf{H}_{old})}{(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old})} \tag{3.9}$$

The existing MTTKRP is expensive process [18, 23]. Thus the *accelerated MTTKRP* [18] regarding the mode-3 could be written as the inner product between the corresponding r^{th} columns of $\mathbf{H}_{old}$ and $[\underline{\mathbf{Y}}^{(3)}_{new}\ \mathbf{V}_{old}]$, respectively. Thus, in order to retrieve a row $\mathbf{M}^{(3)}(n,:)$, we can simply operate as:

$$\mathbf{M}^{(3)}(n,:) = dot(\mathbf{H}_{old}, \underline{\mathbf{Y}}^{(3)}_{new}\mathbf{V}_{old}) \tag{3.10}$$

The arising sub-problem, after manipulation can be rewritten as:

$$\widetilde{\mathbf{W}} = \frac{\mathbf{M}^{(3)}_{new}}{(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old}))} \tag{3.11}$$

$\mathbf{W}_{new}$ is updated by appending the projection $\mathbf{W}_{old}$ of previous time stamp, to $\widetilde{\mathbf{W}}$ of new time stamp, i.e.,

$$\mathbf{W}_{new} = \begin{bmatrix} \mathbf{W}_{old} \\ \frac{\mathbf{M}^{(3)}_{new}}{(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old}))} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{old} \\ \widetilde{\mathbf{W}} \end{bmatrix} \tag{3.12}$$

where the MTTKRP, $\mathbf{M}^{(3)}_{new}$ is parallelizable and is efficiently calculated in linear complexity to the number of non-zeros in $\underline{\mathbf{Y}}$.

3.1.4 Update factor non-temporal factors We update $\mathbf{H}_{new}$ by fixing $\mathbf{V}_{old}$ and $\mathbf{W}_{new}$. We set derivative of the loss $\mathcal{LS}$ w.r.t. $\mathbf{H}$ to zero to find local minima as :

$$\frac{\delta([\underline{\mathbf{Y}}^{(1)}_{new} - \mathbf{H}_{new}(\mathbf{W}_{new} \odot \mathbf{V}_{old})^T]}{\delta H_{new}} = 0 \tag{3.13}$$

By solving above equation, we obtain:

$$\begin{aligned}\mathbf{H}_{new} &= \frac{\underline{\mathbf{Y}}^{(1)} * (\mathbf{W}_{new} \odot \mathbf{V}_{old})^T}{(\mathbf{W}_{new} \odot \mathbf{V}_{old})^T(\mathbf{W}_{new} \odot \mathbf{V}_{old})} \\ &= \frac{\underline{\mathbf{Y}}^{(1)}_{old} * (\mathbf{V}_{old} \odot \mathbf{W}_{old})^T + \underline{\mathbf{Y}}^{(1)}_{new} * (\widetilde{\mathbf{W}} \odot \mathbf{V}_{old})^T}{(\mathbf{W}_{old}^T\mathbf{W}_{old} * \mathbf{V}_{old}^T\mathbf{V}_{old}) + (\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}} * \mathbf{V}_{old}^T\mathbf{V}_{old})} \\ &= \frac{\mathbf{M}^{(1)}_{old} + \underline{\mathbf{Y}}^{(1)}_{new} * (\widetilde{\mathbf{W}} \odot \mathbf{V}_{old})^T}{\mathbf{L}^{(1)}_{old} + (\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}} * \mathbf{V}_{old}^T\mathbf{V}_{old})}\end{aligned} \tag{3.14}$$

Algorithm 1: SPADE Update Framework

Input: $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_k \times J_2 \times \cdots \times J_{N-1}} \quad \forall k = [1, K]$, old data factors $(\mathbf{U}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)})$, supporting matrices $[\mathbf{L}_{old}, \mathbf{M}_{old}]$, Rank R.

Output: Updated factor matrices $(\mathbf{U}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)})$,

1: Initialize $\mathbf{A}^{(N)}_{rand} \in \mathbb{R}^{K \times R}$ from $\mathbf{A}^{(N)}_{old}$
2: **for** $k \leftarrow 1$ to K **do**
3: $[\mathbf{P}_k, \Sigma_k, \mathbf{Z}_k] \leftarrow \text{SVD}(\mathbf{A}^{(1)}\mathbf{A}^{(N)}_{rand}(k)\underline{\mathbf{X}}^T_{new_k} \odot_{i=2}^{N-1} \mathbf{A}^{(i)})$ with Rank R.
4: $\mathbf{Q}_k = \mathbf{Z}_k\mathbf{P}_k^T$
5: $\underline{\mathbf{Y}}_k = \mathbf{Q}_k^T\underline{\mathbf{X}}_{new_k}$
6: **end for**
7: Update temporal modes of CP tensor $\underline{\mathbf{Y}}$

$$\mathbf{A}^{(N)} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{A}^{(N)}_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \frac{\mathbf{M}^{(N)}}{\circledast_{i=1}^{N-1}\mathbf{A}^{(i)}} \end{bmatrix} \quad \forall i \in [1, N]$$

8: **for** $n \leftarrow 1$ to $N-1$ **do**
9: Update other modes of CP tensor $\underline{\mathbf{Y}}$

$$\mathbf{A}^{(i)} = \frac{\mathbf{M}^{(i)}_{old} + \odot_{i\neq n}^{N}\mathbf{A}^{(i)}}{\mathbf{L}^{(i)}_{old} + \circledast_{i\neq n}^{N}\mathbf{A}^{(i)}} \quad \forall i \in [1, N]$$

10: **end for**
11: Update first mode of PARAFAC2 tensor $\underline{\mathbf{X}}_{new}$

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{old} \\ \mathbf{Q}_n * \mathbf{A}^{(1)} \end{bmatrix}$$

12: **return** Updated $(\mathbf{U}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)})$

As classic MTTKRP $\underline{\mathbf{Y}}^{(1)}_{new} * (\widetilde{\mathbf{W}} \odot \mathbf{V}_{old})^T$ is expensive, therefore modified and accelerated MTTKRP can be expressed as a summation of block matrix multiplications:

$$\mathbf{M}^{(1)}_{new} = \sum_{n=0}^{N} \underline{\mathbf{Y}}_{new}\mathbf{T}_n = (\underline{\mathbf{Y}}_{new}\mathbf{V}_{old}) * \mathbf{W}^{'} \tag{3.15}$$

where $\mathbf{T}_n$ is n^{th} vertical block of the Khatri Rao Product $(\mathbf{V}_{old} \odot \mathbf{W}^{'})$. The above computation can be easily parallelized over N independent sub-problems and summing the partial results. Other efficient way is by computing the slice wise matrix product $\underline{\mathbf{Y}}_{new}\mathbf{V}_{old}$ and for each row of the intermediate result of size $\mathbb{R}^{R\times R}$, we compute the Hadamard product with $\mathbf{W}^{'}(n,:)$ as described in [18]. Hence $\mathbf{H}_{new}$ can be updated as :

$$\mathbf{H}_{new} = \frac{\mathbf{M}^{(1)}_{new}}{\mathbf{L}^{(1)}_{new}} = \frac{\mathbf{M}^{(1)}_{old} + (\underline{\mathbf{Y}}_{new}\mathbf{V}_{old}) * \mathbf{W}^{'}}{\mathbf{L}^{(1)}_{old} + (\mathbf{W}^{'T}\mathbf{W}^{'} * \mathbf{V}^T_{old}\mathbf{V}_{old})} \tag{3.16}$$

In this way, the factor update equation and supporting matrices update consist of two parts: the historical part; and the new data part that makes computation fast.

Similarly, $\mathbf{V}_{new}$ can be updated with accelerated MTTKRP for mode-2 as $\mathbf{M}^{(2)} = \sum_{n=0}^{N} \underline{\mathbf{Y}}^T_{new}\mathbf{T}_n$ as :

$$\mathbf{V}_{new} = \frac{\mathbf{M}^{(2)}_{new}}{\mathbf{L}^{(2)}_{new}} = \frac{\mathbf{M}^{(2)}_{old} + (\underline{\mathbf{Y}}^T_{new}\mathbf{H}_{new}) * \mathbf{W}^{'}}{\mathbf{L}^{(2)}_{old} + (\mathbf{H}_{new}\mathbf{H}_{new} * \mathbf{W}^{'T}\mathbf{W}^{'})} \tag{3.17}$$

3.1.5 Update factor U Finally, we update mode-1 factor of PARAFAC2 tensor $\underline{\mathbf{X}}_{new}$ by appending the projection $\mathbf{U}_{old}$ of previous time step, to $\widetilde{\mathbf{U}}$ obtained from factor matrix $\mathbf{H}_{new}$ and $\mathbf{Q}_n$ as given below:

$$\mathbf{U}_{new} = \begin{bmatrix} \mathbf{U}_{old} \\ \mathbf{Q}_n * \mathbf{H}_{new} \end{bmatrix} \tag{3.18}$$

Summary: Our proposed algorithm, SPADE, consist of three parts: First, it obtains slice wise CP tensor $\underline{\mathbf{Y}}$ from incoming tensor data $\underline{\mathbf{X}}_{new}$ using existing non-temporal factor or loading matrices i.e $\mathbf{H}_{old}$ and $\mathbf{V}_{old}$ and matrix created from random linear combinations of columns of the existing temporal factor matrix $\mathbf{W}_{old}$, Second, we initialize two small set of supporting matrices $\mathbf{L}_{old}$ and $\mathbf{M}_{old}$ with the old tensor and its factors by using *accelerated MTTKRP*[18]. In last step, the new incoming tensor data $\underline{\mathbf{X}}_{new}$ is processed by our proposed effective, parallel and fast incremental update method presented in Algorithm 1.

3.2 Extending to Higher-Order Tensors

We now show how our approach is extended to higher-order cases. Consider N-mode tensor $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I_m \times J_2 \times \cdots \times J_{N-1}}$. The factor matrices are $(\mathbf{U}_{old}, \mathbf{A}^{(1)}_{old}, \mathbf{A}^{(2)}_{old}, \ldots, \mathbf{A}^{(N-1)}_{old}, \mathbf{A}^{(T_1)}_{old})$ for PARAFC2 decomposition with N^{th} mode as new incoming data. A new tensor $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_k \times J_2 \times \cdots \times J_{N-1}}$ is added to $\underline{\mathbf{X}}_{old}$ to form new tensor of $\mathbb{R}^{I_t \times J_2 \times \cdots \times J_{N-1}}$ where $t = k + m$. In addition, supporting matrices $\mathbf{L}^{(1)}, ..., \mathbf{L}^{(N-1)}$ and $\mathbf{M}^{(1)}, ..., \mathbf{M}^{(N-1)}$ are stored, where $\mathbf{M}^{(n)}$ and $\mathbf{L}^{(n)}$, $n \in [1, N-1]$ are the supporting matrices for mode n. Additionally, the Khatri-Rao and Hadamard products of a sequence of N matrices are denoted by $\odot_{i\neq n}^{N}\mathbf{A}^{(i)}$ and $\circledast_{i=1}^{N-1}\mathbf{A}^{(i)}$, respectively. The subscript $i \neq n$ indicated the n^{th} matrix is not included in the operation.

The Temporal mode can be updated as :

$$\mathbf{A}^{(N)} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{A}^{(N)}_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \frac{\mathbf{M}^{(N)}}{\circledast_{i=1}^{N-1}\mathbf{A}^{(i)}} \end{bmatrix} \tag{3.19}$$

The Non-Temporal modes can be updated as:

$$\mathbf{A}^{(i)} = \frac{\mathbf{M}^{(n)}_{old} + \odot_{i\neq n}^{N}\mathbf{A}^{(i)}}{\mathbf{L}^{(n)}_{old} + \circledast_{i\neq n}^{N}\mathbf{A}^{(i)}} \tag{3.20}$$

where $i \in [1, N-1]$.

The dynamic or first mode of PARAFAC2 tensor can be updated as:

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{old} \\ \mathbf{Q}_n * \mathbf{A}^{(1)} \end{bmatrix} \tag{3.21}$$

We obtain the general version of update rule of our SPADE for N-mode tensor, as presented in Algorithm 1. **NOTE**: Line 3-5 can be executed in parallel.

Dataset	Statistics (K: Thousands M: Millions)				
	I_{max}	**J**	***K***	***Batch***	***#nnz***
I	$1K$	$3K$	$10K$	500	$3M$
II	$2K$	$6K$	$50K$	500	$60M$
III	$5K$	$8K$	$100K$	450	$133M$
IV	$5K$	$10K$	$500K$	50	$239M$
V	$10K$	$12K$	$1M$	10	$507M$
ML	21	$28K$	$139K$	$1K$	$20M$
Adobe	$2K$	$17K$	$6.8M$	$10K$	$35M$

Table 2: Details for the datasets. ***K*** is the number of subjects, **J** is the number of features, I_{max} is the number of observations for the k-th subject and #nnz corresponds to the total number of non-zeros.The rank of tensors is $\boldsymbol{R = 15}$. Density is between $[10^{-3}, 10^{-5}]$. **ML: MovieLens**

SYN	PARAFAC2	SPARTan	SPADE
I	**1649.2 ± 0.1**	1649.9 ± 0.1	1659.7 ± 0.1
II	**16601.3 ± 9.4**	16611.3 ± 7.3	16652.2 ± 2.2
III	4453.1 ± 2.7	**4443.9 ± 2.5**	4454.1 ± 3.7
IV	[OoM]	3107.6 ± 3.3	**3099.5 ± 3.2**
V	[OoM]	[OoM]	**1777.6 ± 9.5**

Table 3: Mean LOSS over complete tensor data. The boldface means the best results.

4 EXPERIMENTAL EVALUATION

In this section we extensively evaluate the performance of SPADE on five synthetic and two real datasets, and compare its performance with state-of-the-art approaches. Note that all comparisons were carried out over 5 iterations each, and each number reported is an average with its standard deviation attached to it.

4.1 Experimental Setup

4.1.1 Synthetic Data Generation : The specifications of each synthetic dataset are given in Table 2. For all synthetic data we use rank $R = 15$. The entries of loading matrix **V** and **W** are Gaussian with unit variance, and orthogonality is imposed on factors **H** and **Q**, then a few entries are clipped to zero randomly to create a sparse PARAFAC2 tensor. The MATLAB script is provided on *link*[1].

4.1.2 Real Data Description : We evaluate the performance of the proposed method SPADE against the state-of-art methods for the real datasets as well. In our experiments, we include **MovieLens - 20M**[9] and **Adobe dataset**. MovieLens-20M dataset is widely used in recent literature. For this dataset, we created tensor as year-by-movie-by-user i.e each year of ratings corresponds to a certain observation for each user's activity. Adobe dataset is sequential data and it consists of tutorial sequence of anonymous 7 million users. The data is structured as sequence-by-tutorial-by-user. We have semi synthetic ground truth values for each user based on tutorial watched.

SYN	PARAFAC2	SPARTan	SPADE
I	7.4 ± 1.5	4.47 ± 2.5	**1.3± 0.3**
II	725.7 ± 9.2	290.8 ± 2.3	**21.9 ± 2.7**
III	1158.7 ± 10.7	330.5 ± 4.6	**22.3 ± 1.4**
IV	[OoM]	672.4 ± 7.5	**36.7± 1.3**
V	[OoM]	[OoM]	**144.7 ± 1.8**

Table 4: Mean CPU TIME (mins) over all batches of third-order datasets. The boldface means the best results.

SYN	PARAFAC2	SPARTan	SPADE
I	1490.3 ± 0.1	393.9 ± 0.2	**133.3 ± 0.3**
II	21978.5 ± 0.1	16003.5 ± 0.3	**1737.7± 0.2**
III	12813.7 ± 0.1	8835.3 ± 0.1	**739.2± 0.4**
IV	[OoM]	10785.4 ± 0.1	**622.1± 0.1**
V	[OoM]	[OoM]	**545.4± 0.1**

Table 5: Average MEMORY USAGE (MBytes) for decomposition. The boldface means the best results.

4.1.3 Evaluation Measures We evaluate SPADE and the baselines using three criteria: **approximation loss**, **CPU time** in minutes and **memory usage** in Megabytes. These measures provide a quantitative way to compare the performance of our method. For all criterias, lower is better.

4.1.4 Baselines In this experiment, two baselines have been used as the competitors to evaluate the performance.

- **PARAFAC2** [13] : an implementation of standard fitting algorithm PARAFAC2 with random initialization.
- **SPARTan** [18] : a scalable PARAFAC2 fitting algorithm was proposed for large and sparse data.

Note: Our proposed method is natural extension of scalable PARAFAC2[18].

4.2 SPADE is fast and memory-efficient

4.2.1 Synthetic data results First, we remark that SPADE is both more memory-efficient and faster than the state-of-art methods. In particular, the state-of-art methods fail to execute in the two largest problems i.e. SYN-VI and SYN-V for given target rank due to out of memory problems during the formation of the slice wise CP tensor $\underline{\mathbf{Y}}$. This improvement stems from the fact that state-of-art methods attempt to decompose in full tensor, whereas SPADE can do the same in streaming mode, thus having higher immunity to large data volumes in short time interval and small memory space with comparable accuracy.

From the results shown in Table (3), it can be concluded that, the SPADE best performance is on par to the state-of-the-art (i.e. classic PARAFAC2 as

	Algorithm	MovieLens		Adobe	
		R=10	R=50	R=10	R=50
Time	SPARTan	257.8 ± 8.3	5673.4 ± 4.1	[OoM]	[OoM]
	SPADE	51.9 ± 3.3	609.6 ± 5.6	80.1 ± 4.6	824.9 ± 9.1
Mem.	SPARTan	66474.8 ± 0.1	250212 ± 0.1	[OoM]	[OoM]
	SPADE	2092.7 ± 34.9	10346.3 ± 23.7	690.7 ± 37.1	3409.3 ± 56.8

Table 6: The average and standard deviation of memory usage and time metric comparison on MovieLens and Adobe using two different target for five random initialization.

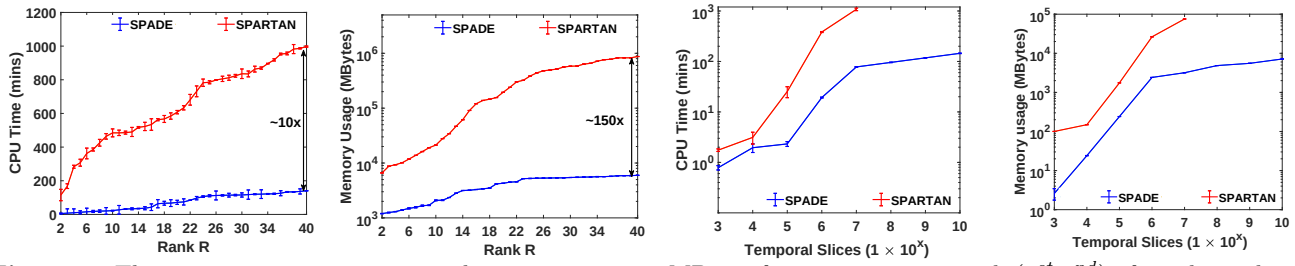

Figure 2: The average time in minutes and memory usage in MBytes for varying target rank (1^{st},2^{nd}) of synthetic data of size ($1000 \times 1000 \times 10^5$) and varying number of subjects(K) (3^{rd}, 4^{th}) for synthetic data of size ($100 \times 100 \times [10^3, 10^{10}]$).

well as scalable PARAFAC2 (SPARTAN)), algorithm best performance, in terms of approximation loss and SPADE performances are **significantly better** in CPU time and memory usage as shown in Table (4, 5). Most importantly, however, SPADE performed very well on SYN-IV and SYN-V datasets, arguably the hardest of the five synthetic datasets we examined *where none of the baselines was able to run efficiently (under 10 hours)*. Overall, it is clear that the state-of-the-art approaches cannot fully handle the large data of size $15 - by - 12K - by - 1Mil$ as it requires $> 1TB$ memory and surpasses the available storage capacity of our system. On the contrary, SPADE properly performs for all the datasets considered in a reasonable amount of time, since it only operates directly on the incoming tensor slice(s). In particular when tested on SYN-IV dataset, for $R = 40$, SPADE is up to 13× faster than the state-of-the-art method. Even for a lower target rank of $R = 5$, SPADE obtains up to 20× faster computation. These results show that SPADE is able to handle large dimensions in reasonable time and memory.

4.2.2 Real data results We evaluate the performance of the proposed SPADE approach against the baseline method for the real datasets as well. The empirical results in terms of CPU time and Memory usage is given in Table (6). There is no significant difference is observed in their effectiveness in terms of approx.loss. The baseline method has $1 - 2\%$ less approx.loss as compared to our proposed method. However, the time and memory saving with SPADE is significant in case of Movielens data. For Adobe dataset, baseline is unable to create intermediate slice wise CP tensor of size $10 \times 17K \times 6.8Mil$. Our proposed method took only < 14 hours for computing factors for it. Our proposed algorithm, SPADE, shows very promising results in speed and space utilization and showing that it is less sensitive to the size of the data, and thus, having better performance.

4.2.3 Scalability Evaluation We also valuate the scalability of our algorithm on synthetic and real dataset. Firstly, a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{1000 \times 1000 \times 10^5}$ is decomposed with increasing target rank.The baseline approach consumes more time as we increase the target rank as shown in Figure 2 (1^{st}, 2^{nd}). On the contrary, the time needed by SPADE increases very moderately. Overall, our method achieves up to 10× gain regarding the time required and 150× gain over memory usage. Second, we create a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{100 \times 100 \times 10^{10}}$ of small slice size but long 3^{rd} dimension. We decomposed it using fixed target rank $R = 5$. The baseline method runs upto 10^7 slices and runs out of memory for further data. However, our proposed method, successfully decomposed the tensor in reasonable time as shown in Figure 2 (3^{rd}, 4^{th}).

We also evaluate scalability using both real dataset. The SPADE saved up to 23× memory used and achieved 17× speedup over the baseline approach for the MovieLens dataset as shown in Figure 3. The baselines unable to run for Adobe dataset because of high computation requirements for intermediate CP tensor creation. However, our proposed SPADE successfully able to decompose the tensor (Figure 3) and shows effectiveness over large irregular datasets. In terms of batch size, it is observed that the time consumed by our method is linearly increasing as the batch size grows. However, their slopes vary with different rank used. The analysis is not

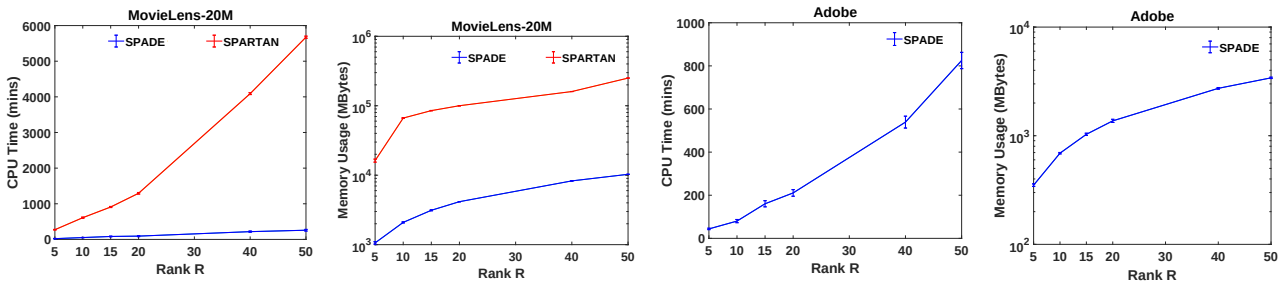

Figure 3: The average time in minutes and memory usage in MBytes for varying target rank for both the real datasets. For Adobe dataset baseline method runs out of memory.

included due to limitation of space here.

5 Community discovery on Adobe data

5.1 Motivation: Here, we discuss the usefulness of PARAFAC2 towards extracting meaningful communities or clusters of users from Adobe data. The challenge in community detection of such large data ($\approx$ 7 million users) is to capture behaviour regarding the sequential click of the tutorials for each user. In this dataset, there is no real alignment in the "time" mode, since every view of a tutorial can happen anywhere in time, therefore, we care about the sequence followed by each user and it cannot be modeled as a regular online tensor decomposition methods. Therefore, there is a serious need to learn richer and more useful representation. Below, we describe how SPADE can be used to successfully handle this challenge.

5.2 Model Interpretation: the model interpretation towards the target challenge:

- **Incremental factor**: Each column of factor or loading matrix $\mathbf{W}$ represents a community and each row represents the importance of community membership for a user to each one of the R communities. Therefore, an entry $\mathbf{W}$ (i, j) represents the membership of user i to the j^{th} community. We consider non-overlapping communities based on type of tutorial watched by user. So we normalize the matrix w.r.t row between [0,1] and highest value indicates the community membership to corresponding user.
- **Irregular dimension factor**: Each $\mathbf{U}_k$ loading matrix gives the sequential signature of each user i.e. each r^{th} column of $\mathbf{U}_k$ reflects the evolution of the community r for all I_k tutorial watched sequences for user k.
- **Non-incremental factor**: The factor matrix $\mathbf{V}$ reflects the community definition based on tutorials and each row indicates a tutorial features.

5.3 Qualitative Analysis: For this case study, we decompose tensor in batch of $50K$ users to extract communities. For this experiment, we set the Rank $R = 24$ based on semi-synthetic ground truth labels. We compute *Kull-back Leibler divergence* or simply KL-div [7] to evaluate the communities quality.

In order to present the use of SPADE towards community detection, we focus our analysis on a subset of tutorials(s) watched by each community in Adobe dataset. Figure 4(a) shows the top 5 (based on number of users) community's most frequent tutorial(s) sequence watched. Conceptually, those users share similar interest in terms of learning, domain knowledge or interests. As a result, it becomes a very important challenge to accurately cluster the users. Nevertheless, SPADE achieves significantly good performance in terms of $KL - div$ i.e ≈ 0.553. In Figure4(b), we show top 5 communities of the dataset as clustered by SPADE with $R = 24$. Qualitatively, we see that the methods output concurs with the communities that appear to be strong on the spy-plots. These communities are connected strongly within the group and have very few connections outside the group. As any of the baseline is unable to execute the entire data, our proposed method gives advantage to decompose the data in streaming fashion in reasonable time.

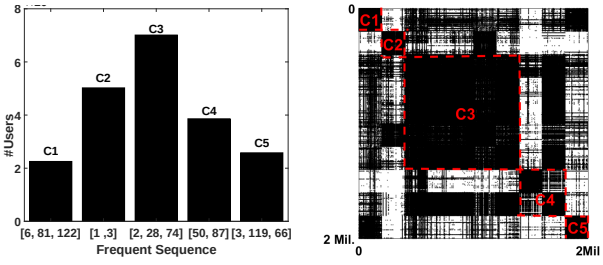

Figure 4: For top 5 communities (a) frequent sequence of tutorials watched (b) spy-plot of user-user view.

6 Conclusion

We propose SPADE, a novel online PARAFAC2 decomposition method. We demonstrate its efficiency

and scalability over synthetic and real datasets. The SPADE provides comparable approximation quality to baselines and it is both fast $(10-23\times)$ and memory-efficient $(17-150\times)$ than the baseline approaches. Extensive experiments with Adobe dataset have demonstrated that the proposed method is capable of handling larger dataset in incremental fashion for which none of the baseline performs due to lack of memory. Future directions include, but are not limited to: a) extension of the proposed method for multi-aspect-streaming tensors, b) incorporate various constraints e.g smoothness, sparsity and non-negativity for more applications.

7 Acknowledgements

Research was supported by an Adobe Data Science Research Faculty Award, the Department of the Navy, Naval Engineering Education Consortium under award no. N00174-17-1-0005, and the National Science Foundation Grant no. 1901379. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] A. Afshar, I. Perros, E. E. Papalexakis, E. Searles, J. Ho, and J. Sun. Copa: Constrained parafac2 for sparse & large datasets. In *Proceedings of the 27th ACM Int. Conf. on Information and Knowledge Management*, pages 793–802. ACM, 2018.

[2] J. M. Amigo, M. J. Popielarz, R. M. Callejón, M. L. Morales, A. M. Troncoso, M. A. Petersen, and T. B. Toldam-Andersen. Comprehensive analysis of chromatographic data by using parafac2 and principal components analysis. *Journal of Chromatography a*, 1217:4422–4429, 2010.

[3] W. Austin, G. Ballard, and T. G. Kolda. Parallel tensor compression for large-scale scientific data. In *PDPS, 2016 IEEE Int.*, pages 912–922. IEEE, 2016.

[4] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. available online.(february 2015). *URL http://www. sandia. gov/˜ tgkolda/TensorToolbox/index-2.6. html*, 2015.

[5] R. Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38, 1997.

[6] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35:283–319, 1970.

[7] A. Gorovits, E. Gujral, E. E. Papalexakis, and P. Bogdanov. Larc: Learning activity-regularized overlapping communities across time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1465–1474. ACM, 2018.

[8] E. Gujral, R. Pasricha, and E. E. Papalexakis. Sambaten: Sampling-based batch incremental tensor decomposition. In *Proceedings of the 2018 SIAM Int. Conf on Data Mining*, pages 387–395. SIAM, 2018.

[9] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5:19, 2016.

[10] R. A. Harshman. Parafac2: Mathematical and technical notes. *UCLA working papers in phonetics*, 22:122215, 1972.

[11] J. C. Ho, J. Ghosh, S. R. Steinhubl, W. F. Stewart, J. C. Denny, B. A. Malin, and J. Sun. Limestone: High-throughput candidate phenotype generation via tensor factorization. *Journal of biomedical informatics*, 52:199–211, 2014.

[12] J. C. Ho, J. Ghosh, and J. Sun. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proc. of the 20th ACM SIGKDD Int. Conf on Knowledge discovery and data mining*. ACM, 2014.

[13] H. A. Kiers, J. M. Ten Berge, and R. Bro. Parafac2part i. a direct fitting algorithm for the parafac2 model. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 13:275–294, 1999.

[14] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51:455–500, 2009.

[15] D. Nion and N. D. Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *IEEE Transactions on Signal Processing*, 57:2299–2310, 2009.

[16] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *TIST*, 2016.

[17] R. Pasricha, E. Gujral, and E. E. Papalexakis. Identifying and alleviating concept drift in streaming tensor decomposition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 327–343. Springer, 2018.

[18] I. Perros, E. E. Papalexakis, F. Wang, R. Vuduc, E. Searles, M. Thompson, and J. Sun. Spartan: Scalable parafac2 for large & sparse data. In *Proceedings of the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 375–384. ACM, 2017.

[19] T. Skov, D. Ballabio, and R. Bro. Multiblock variance partitioning: A new approach for comparing variation in multiple data blocks. *Analytica chimica acta*, 615:18–29, 2008.

[20] A. Stegeman and T. T. Lam. Multi-set factor analysis by means of p arafac2. *British Journal of Mathematical and Statistical Psychology*, 69:1–19, 2016.

[21] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis:theory and applications. *ACM Trans. Knowl. Discov. Data*, 2008.

[22] J. M. ten Berge and H. A. Kiers. Some uniqueness results for parafac2. *Psychometrika*, pages 123–132, 1996.

[23] S. Zhou, S. Erfani, and J. Bailey. Online cp decomposition for sparse tensors. In *2018 IEEE Int. Conf. on Data Mining (ICDM)*, pages 1458–1463. IEEE, 2018.

[24] S. Zhou, N. Vinh, J. Bailey, Y. Jia, and I. Davidson. Accelerating online cp decompositions for higher order tensors. In *22nd ACM SIGKDD*. ACM, 2016.