UNIVERSITY OF CALIFORNIA
RIVERSIDE

Modeling and Mining Multi-Aspect Graphs With Scalable Streaming Tensor Decomposition

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Ekta Gujral

June 2021

Dissertation Committee:

Dr. Evangelos E. Papalexakis, Chairperson
Dr. Eamonn Keogh
Dr. Jiasi Chen
Dr. Christos Faloutsos

The Dissertation of Ekta Gujral is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am extremely grateful to my advisor, Evangelos E. (Vagelis) Papalexakis, without whose help, I would not have been here. Throughout my entire PhD journey, Vagelis has been a great role model as a mentor, a friend and a human being. As a mentor, he patiently listened to my challenges, understood them, shared his genuine advice and motivated me to push harder in life. It is because of this, I have become responsible, dedicated and hopefully, successful, in my career. Vagelis's chrisma is such that you will instantly love and never forget once you meet him. He is the fun loving advisor and one of the best and smart people I know.

I would also like to thank all my outstanding committee members, Christos Faloutsos., Jiasi Chen, and Eamonn Keogh: It was Eamonn who taught me about Artificial Intelligence. His research work always inspired me and I felt lucky when he accepted to be part of the committee member. I met Jiasi when she started WinC in the computer science department and I was immediately impressed by her capabilities. Later, we were part of Riverside Unified School District where we presented our work to the high school girls together. Her enthusiasm and love for teaching is contagious. Christos gave me extremely valuable industrial perspective on real-world data mining problems. Multiple conversations with Christos also helped me tremendously towards the end of my PhD studies.

I thank all my collaborators, Petko Bogdanov, Alex Gorovits, Saba Al-Sayouri, Danai Koutra, Tianxiong Yang, Neil Shah, Georgios Theocharous, Leonardo Neves, Brian J Gallagher, Ming Jiang and Anup Rao. Without their invaluable discussion and feedback, my journey would not have been completed. I would also like to thanks to my vanpool mem-

bers, Arthur Jia, Xinping Cui, Jacob Greenstein, Amanda Pagul, Ian Marcus, Haripriya Vasireddy, and Daniel Cicala, who always waited for me whenever I was late.

Thanks to all the people who shared my life in Riverside: Ravdeep Pasricha, Sara Abdali, Rutuja Gurav, Pravallika Devineni, Yorgos Tsitsikas, Negin Entezari, Uday Singh Saini, William Shiao, Sunita Kopper, Kamalika Poddar, Pratheek Chindodi Rajashekar, Jiahuan Liu, Harini Venkatesan, Mario Salazar, Abhishek Srivastava, Umar Farooq, Dipankar Ranjan, Dipan Shaw, and Payas Rajan. A good support system is always required to surviving and staying sane in grad school. Finally, my special gratitude to the Computer Science and Engineering department administrative members Vanda Yamaguchi, Madie Heersink, Sara Galloway, and Heidi Nam (International office) for making my Ph.D. journey smooth. I feel extremely fortunate to have everyone in my life.

I would also like to thank Vandana Jagdish, Gurbir Singh, Tejinderpal Singh, Raghav Kohli, Nazdeep Behl, Balkarn Singh, Nutan Pant, Jyoti Rai, Anirban Ghosh, Arti Ahuja, Aarti Joshi, Divya Singh, Amanjot Singh Duggal, Karandeep Singh, Navdeep Thind, Patricia, Vanessa, Nour, Erum, all my life long friends who always supported me in my ups and downs during this journey and have had a significant impact on my life.

Most of all, I am grateful to my parents, Ajay Kumar Gujral and Sunita Gujral, and my in-laws Anil Kumar Chhabra and Sunita Chhabra, my lovely brother, Kushal Gujral. A huge thank you to my husband, Ravi Chhabra for giving me the freedom to pursue opportunities and filling my life with love and optimism. Ohh, am I forgetting someone here? Yes, its my son, Kiyan Chhabra, who just turned 2 and cheers me up whenever I need it. I am extremely lucky to have you in my life.

To my husband, Ravi Chhabra and son, Kiyan Chhabra who brings me to joy.

To my parents and in-laws, who enthusiastically support me everyday.

ABSTRACT OF THE DISSERTATION

Modeling and Mining Multi-Aspect Graphs With Scalable Streaming Tensor Decomposition

by

Ekta Gujral

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2021
Dr. Evangelos E. Papalexakis, Chairperson

Graphs emerge in almost every real-world application domain, ranging from online social networks all the way to health data and movie viewership patterns. Typically, such real-world graphs are big and dynamic, in the sense that they evolve over time. Furthermore, graphs usually contain multi-aspect information i.e. in a social network, we can have the "means of communication" between nodes, such as who messages whom, who calls whom, and who comments on whose timeline and so on.

How can we model and mine useful patterns, such as communities of nodes in that graph, from such multi-aspect graphs? How can we identify dynamic patterns in those graphs, and how can we deal with streaming data, when the volume of data to be processed is very large? In order to answer those questions, in this thesis, we propose novel tensor-based methods for mining static and dynamic multi-aspect graphs. In general, a tensor is a higher-order generalization of a matrix that can represent high-dimensional multi-aspect data such as time-evolving networks, collaboration networks, and spatio-temporal data like Electroencephalography (EEG) brain measurements.

The thesis is organized in two synergistic thrusts: First, we focus on static multi-aspect graphs, where the goal is to identify coherent communities and patterns between nodes by leveraging the tensor structure in the data. Second, as our graphs evolve dynamically, we focus on handling such streaming updates in the data without having to re-compute the decomposition, but incrementally update the existing results. In more detail, the following two thrusts are as follows:

- **Mining Graphs and Networks**: Firstly, we focus on static multi-aspect data, in which, static network information is available. We detail our proposed algorithms spanning the topics of community detection in a semi-supervised matrix-tensor coupling settings and structurally dynamic multi-aspect graph summarization through tensor analysis. Our SMACD algorithm based on the CP decomposition is the first to incorporate multi-aspect graph information and semi-supervision while being able to discover overlapping and non-overlapping communities in social networks. Moving away from the restrictive assumptions of CP decomposition, we propose RichCom, where we leverage the concept of block term decomposition in order to extract rich and interpretable structure from general multi-aspect data. Next, we propose POPLAR, where we introduce the concept of Laplacian regularization on the PARAFAC2 decomposition, which improves community detection in time-evolving social networks, by leveraging graph-based auxiliary information. Can community detection benefit by having more auxiliary graphs? To answer this question, we proposed CAPTION that effectively decomposes multi-view auxiliary information and PARAFAC2 data jointly into interpretable latent factors using a variety of objective functions. In ad-

dition to the above, we pose and study the niche detection problem, which imposes an explainable lens on the classical problem of co-clustering interactions. We design an end-to-end framework, NED, which discovers co-clusters of user behaviors based on interaction densities and explaining them using attributes of involved nodes.

- **Mining Streaming Tensors**: Next, we expand our scope to incremental multi-aspect data, in which we leverage network information over time. In a wide array of modern real-world applications, the data is far from being static. As the volume and velocity of data grow, the need for time and space-efficient online tensor decomposition is imperative. This is a challenging task primarily because of following reasons a) high-accuracy (competitive to decomposing the full tensor) using significantly fewer computations than the full decomposition calls for innovation, b) the velocity of incoming data is very high and require real-time enforcement, c) operating on the full ambient space of data leads to the increase in space complexity, rendering such approaches hard to scale, d) for irregular tensor data, any pre-processing to accumulate across any mode may lose significant information, and e) last but not least, there are certain instances wherein rank-1 decomposition (CP or Tucker) can not be useful (e.g. EEG/ECG data signals) and require online methods that go beyond rank-1. Motivated by the above challenges, we investigate how to adaptively monitor various decompositions of a tensor that is dynamically changing over time without having to compute from scratch, provided that the previous decomposition is available. We propose fast, scalable and efficient methods i.e. SamBaTen, OCTen that tackle streaming CP decomposition, SPADE to tackle irregular streaming tensors (PARAFAC2)

and OnlineBTD to handle beyond rank-1 streaming tensor decomposition. In a given tensor, static or dynamic, the number of useful patterns corresponds to the low-rank of the tensor. Unfortunately, this is an NP-Hard problem, and especially in the streaming case, these challenges make it more complex and non-trivial. This is the reason that various tensor mining researchers, understandably, set the number of components manually for static as well as streaming tensor decompositions. To fill the gap, we propose an effective and efficient method Aptera to estimate the rank of irregular tensor data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graphs are effective way to represent a large variety of data like computer networks, biological networks, etc. and relations between data entities. In most real applications, however, the information available usually goes beyond a plain graph that captures relations between different nodes. For instance, in an online social network such as Facebook, relations and interactions between users are represented by a set of edge types rather than a single type of edge. Such different edge-types can be "who messages whom", "who pokes whom", "who-comments on whose timeline" and so on. We refer these graphs as multi-aspect graphs or tensors. Considering all the aspects of interaction or communication yields high clustering accuracy. Tensor decompositions are invaluable tools in analyzing multi-aspect graphs or multi-modal datasets.

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 [202] data, where we have K number of subjects for which we observe features and we permit each subject to have multiple observations. As time grows, a number of subjects are added with more or fewer observations. Each such subject is a new incoming slice(s) to the tensor, 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 network where new interactions occur every second and new friendships are formed at a similar pace. Given such a time-evolving data, it is very predictive that we cannot keep using the old decomposition obtained from previous set of data at certain timestamp as it is already outdated when new interaction formed. Therefore, it is necessary to have fast and efficient methods to handle the latest decomposition when new data is arrived. However, state-of-the-art tensor decomposition methods are developed for static data and they are not able to handle streaming data due to their poor performance, in terms of both time and space.

In most of the tensor mining algorithms and applications, it is assumed that rank of the tensor is known. The tensor rank calculation has been proven to be NP-Hard [116] and in various cases NP-Complete [110]. This is the reason that various tensor mining papers [6, 135], understandably, set the number of components manually. In literature, there are various algorithms [31, 193, 228, 252, 282, 129] available to find the rank of CP tensor. Unfortunately, state-of-the-art methods are not able to tackle irregular data like PARAFAC2.

1.1 Research Questions

In this thesis, we want to answer the following questions, all of that are fundamental to understand growing multi-aspect or temporal data. Given multi-aspect graphs or time evolving data:

- **Q1 Mining Graphs and Networks**: How can we identify useful repetitive sub-graphs i.e. communities and its structural properties that might affect various applications?
- **Q2 Streaming Tensor Mining**: How can we maintain a valid and accurate tensor decomposition of a dynamically evolving data, without having to re-compute the entire decomposition after every single update?
- **Q3 Automatic Tensor Mining**: How can we automatically find how many PARAFAC2 components are required in a data driven and unsupervised way?

1.1.1 Mining Graphs and Networks

For mining graphs and network analysis, we develop a semi-supervised clustering algorithm [89, 90] that simultaneously consider multi-view clustering and semi-supervised learning to not only use multiple views of data but also improve the clustering accuracy by utilizing partially available labels of nodes. For accuracy, our algorithms exploit non-negativity and sparsity constraints [89] to find sub-groups or patterns in multi-aspect graphs. Moreover, in [93], we discover meaningful rich structure of communities in multi-aspect graphs. Specifically, our algorithm exploits the Block Term Decomposition to extract higher than rank-1 but still interpretable structure from a multi-aspect dataset and uses AO-

ADMM to speed up decomposition. We also explore less known but effective graph laplacian constraints for irregular tensor decomposition [97] that have potential to offer more accurate results. Additionally, we develop efficient community detection [92, 10, 80, 81] methods for temporal data.

1.1.2 Streaming Tensor Mining

For online tensor decomposition, we develop various decomposition algorithms that incrementally update a large data efficiently and effectively in dynamic graphs. In [94], we propose SamBaTen, a sampling-based tensor decomposition and OcTen [95], a compression-based tensor decomposition. Specifically, within a limited memory budget, our algorithms are fast, scalable and achieved comparable accuracy. Our algorithms exploit random sampling, compression and utilize computational resources distributed across multiple machines. Regardless of recent development on temporal data through CP tensor decomposition approaches, there are certain instances wherein time modeling is difficult for the regular tensor factorization methods, due to either data irregularity or time-shifted latent factor appearance. To handle this problem, we propose a SPADE [98] to track the updates of online PARAFAC2 decomposition. Specifically, our algorithm avoids the expensive computations of MTTKRP, which classic algorithms suffer from. Additionally, to handle data like streaming Electroencephalography (EEG) brain measurements where we need decomposition beyond rank-1, we develop a fast, efficient and scalable method namely OnlineBTD [91].

1.1.3 Automatic Tensor Mining

In data mining, PARAFAC2 is a powerful tensor decomposition method that is ideally suited for unsupervised modeling of "irregular" tensor data, e.g., patient's diagnostic profiles, where each patient's recovery timeline does not necessarily align with other patients. In real-world applications, where no ground truth is available for this data, how can we automatically choose how many components to analyze? Although extremely trivial, finding the number of components is very hard. So far, under traditional settings, to determine a reasonable number of components, when using PARAFAC2 data, is to compute decompositions with a different number of components and then analyze the outcome manually. This is an inefficient and time-consuming path, first, due to large data volume and second, the human evaluation makes the selection biased. To handle this problem, we propose a APTERA for automatic PARAFAC2 tensor mining that is based on locating the L-curve corner.

1.2 Thesis Outline and Contributions

The thesis addresses a number of important questions regarding the community detection and summarization of multi-aspect graphs by leveraging the tensor structure of the data. Also, the dissertation focuses on incremental and efficient decomposition of streaming tensor data that enables many real applications in diverse domains. More specifically, in this thesis we investigate the problem of community detection, incremental tensor decomposition

Part I Mining Graphs and Networks	Chapter 3: Semi-supervised Community Detection [89], [90] Chapter 4: Discover Rich Community Structure [93] Chapter 5 and 6: Coupled PARAFAC2 [97] [87] Chapter 7: Niche Detection
Part II Streaming Tensor Mining	Chapter 8 and 9: Online CP Tensor Decomposition [94] , [95] Chapter 10: Online Block Term Decomposition [91] Chapter 11: Online PARAFAC2 Tensor Decomposition [98] Chapter 12: Automatic PARAFAC2 Mining

Table 1.1: Overview of the thesis with references to chapters.

and automatic tensor mining. Table 1.1 gives the overall structure of our research with the mapping to the chapters of this thesis.

Chapter 2 provides the background to graph and tensor-related notations, preliminaries definitions and a review on the existing related algorithms. It also builds a foundation for better understanding the proceeding chapters.

In the Part I, **Mining graphs and Networks**, we focus on static multi-aspect graphs, where the goal is to identify and summarize coherent communities between nodes by leveraging the tensor structure. We exploit semi-supervision, tensor decompositions, and graph mining techniques to address the following problems: community detection and summarization in the multi-aspect graphs, joint analysis of static and dynamic graph and explainability of the communities. We address following questions:

- How to find communities or clusters in large multi-aspect graphs? Can we leverage semi-supervision to improve accuracy?

- How are communities in real multi-aspect graphs structured? How we can effectively and concisely summarize and explore those communities in a high-dimensional, multi-aspect graph without losing important information?
- Can we jointly model temporal and static information from PARAFAC2 data to extract meaningful insights? Does the static information added in temporal data improve predictive performance?
- Given a rich interaction graph, and nodal attributes, how can we explain and make sense of it?

Impact

We develop a collection of novel methods to detect communities in a multi-aspect graphs. We are among the first to propose following methods that detect communities in semi-supervised way, summarize the structures in multi-aspect graphs and jointly analyze the PARAFAC2 data with static information. These methods serve as an ensemble that can be employed under different or changing conditions.

- In Chapter 3, we study the problem of finding the assignment of each node into one (or more) of R community labels by effectively integrates and leverages both (a) the multi-view nature of real graphs, and (b) partial supervision in the form of community labels for a small number of the nodes to improve the quality of community detection. The sparsity and non-negativity constraints are exploited also when we devise sparse, and ALS based optimization algorithms to fit our model to multi-aspect data. Our proposed method SMACD is two step process i.e. Decomposition and Assignment. In the decomposition step, we compute a sparse and non-negative latent

components using CP decomposition on coupled data and partial labels. We further design a procedure to automatically find the best sparsity constrained parameter. As validated by multiple synthetic and real dataset, that SMACD through combining semi-supervision and multi-aspect edge information, outperforms the state-of-the-arts.

- In Chapter 4, we study the problem of discovering and summarizing community structure from a multi-aspect graph. State-of-the-art studies focused on patterns in single graphs, identifying structures in a single snapshot of a large network or in time evolving graphs and stitch them over time.To fill the gap, we proposed cLL1 to extract rich and interpretable structure from general multi-aspect data and RichCom, a summarization and encoding scheme to discover and explore structures of communities identified by cLL1. We showed empirical results on small and large real datasets that demonstrate performance on par or superior to existing state-of-the-art.
- In chapter 5 and 6, we explore joint analysis of the PARAFAC2 with matrix and tensor data respectively. In chapter 5 we study the impact of laplacian constraints using auxiliary information on PARAFAC2 decomposition and proposed new method called POPLAR. Motivated by the promising outcome of the POPLAR,we provide a scalable method for decomposing coupled CP and PARAFAC2 tensor data through non-negativity-constrained least squares optimization on a variety of objective function. We present results showing the scalability of this novel implementation on a millions of elements as well as demonstrate the high level of interpretability on real world data.

- In chapter 7, we focus on developing features that attract and appeal to customers and are very critical to companies, as it determines their success and revenue. Specifically, in creating content, it is beneficial to understand the attributes that make a particular type of content (e.g food-related videos) attractive to specific markets that might be under-served by the platform (e.g. 18-24 year old women in Australia). Thus, to improve targeting and to create and promote content that will likely better retain, engage, and satisfy target audiences, we propose a method for self-explaining *niche detection* in user content consumption data. Our work characterizes niches as outstanding co-clusters in user-content interaction graph data, imbued with user and content-oriented attributed co-cluster explanations which designate audience and content types.

In the Part II, **Streaming Tensor Mining**, we focus on dynamic multi-aspect graphs, where the goal is to focus on handling streaming updates in the data without having to re-compute the decomposition, but incrementally update the existing results. For many different applications like sensor network monitoring or evolving social network, the data stream is an important model. Streaming decomposition is a challenging task due to the following reasons. First, **accuracy**: high-accuracy (competitive to decomposing the full tensor) using significantly fewer computations than the full decomposition calls for innovation. Second, **speed**: the velocity of incoming data into the system is very high and require real-time execution. Third, **space**: operating on the full ambient space of data, as the tensor is being updated online, leads to increase in space complexity, rendering offline approaches hard to scale, and calling for efficient methods that work on memory spaces which

are significantly smaller than the original ambient data dimensions. Lastly, **beyond rank-1 data** [93]: there are certain instances wherein rank-1 decomposition (CP or Tucker) can not be useful, for example, EEG signals [122] needed to be modeled as a sum of exponentially damped sinusoids and allow the retrieval of peaks by singular value decomposition. The rank-one terms can only model components of data that are proportional along columns and rows, and it may not be realistic to assume this. Alternatively, it can be handled with blocks of decomposition. Based on this, we develop models that can handle dynamic regular real-world graphs. Then, we focus particularly on irregular data and build a model to handle the streaming updates. Next, we focus on automatic mining of irregular data. We address following questions:

- How to summarize various types of high-order data tensor? How to incrementally update those patterns over time?
- How to automatic mine PARAFAC2 in a data driven and unsupervised way?

Impact

- In chapter8 and 9, we investigate the problem of finding the CP decompositions of streaming tensors. Operating on the full ambient data space, as the tensor is being updated online, leads to an increase in time and space complexity, rendering traditional approaches hard to scale, and thus calls for efficient methods that work on memory spaces which are significantly smaller than the original ambient data dimensions. Based on this, we propose a novel sample-based SamBaTen and compression based OCTen framework, respectively. The proposed frameworks effectively identify the low rank latent factors of incoming slice(s) to achieve online tensor decomposi-

tions. To further enhance the capability, we also tailor our general framework towards higher-order online tensors. Through experiments, we empirically validate its effectiveness and accuracy and we demonstrate its memory efficiency and scalability by outperforming state-of-the-art approaches.

- In Chapter 10, we focus on the problem of finding the PARAFAC2 decompositions of streaming irregular tensors. Streaming PARAFAC2 decomposition is a challenging task due to the reason that for PARAFAC2 tensor data, any pre-processing to accumulate across any mode may lose significant information and 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. 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 and memory-efficient 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.
- In chapter 11 discusses the framework to effectively identify the beyond rank-1 latent factors of incoming slice(s) to achieve online block term tensor decompositions. The tracking of the BTD decomposition for the dynamic tensors is a very pivotal and challenging task due to the variability of incoming data and lack of efficient online algorithms in terms of accuracy, time and space. Through experimental evaluation

on multiple datasets, we show that OnlineBTD provides stable decompositions and have significant improvement in terms of run time and memory usage.

- In chapter 11, we work towards an automatic, PARAFAC2 tensor mining algorithm that minimizes human intervention. With a rich variety of applications, PARAFAC2 decomposition [106] is a very effective analytical method and if performed correctly, it can reveal underlying structures of data. However, there are research issues that need to be tackled in the field of data mining in order for PARAFAC2 decompositions to assert their role as an effective tool for practitioners. One challenge, which has received considerable attention, is finding the correct number of components aka rank of PARAFAC2 decomposition. Motivated by this, we propose a effective and efficient method Aptera to estimate the rank of irregular 'PARAFAC2' data that discover the number of components (interchangeably rank) through higher-order singular values.

We made all of the algorithms produced throughout this thesis open source for reproducibility and the benefit of the community. All the codes are available at link[1]

[1] https://sites.google.com/view/gujralekta/research-work/software

Chapter 2

Background

We explore the context of our research in this chapter, review existing literature, and describe the weaknesses and gaps that will be addressed in subsequent chapters. We begin with a summary of various notations used throughout this thesis, tensor-mining related terminologies and operations, then move on to a discussion of various tensor decomposition algorithms and their applications. Next, we discuss multi-aspect static graph mining, various methods that we are particularly interested in the first part of the thesis, into more details. Finally, we introduce the streaming tensor decomposition problem in the last section and review existing works and discuss their limitations.

2.1 Preliminary Definitions and Notation

This section introduces basic notations and operations that are widely used in the field. A summary of symbols that we use through the whole thesis can be found in the Table 2.1.

Symbols	Definition
$\underline{\mathbf{X}}, \mathbf{A}, \mathbf{a}, a$	Tensor, Matrix, Column vector, Scalar
$\mathbb{R}$	Set of Real Numbers
$vec()$	Vectorization operator
$diag(\mathbf{A})$	Diagonal of matrix $\mathbf{A}$
$[\mathbf{A}; \mathbf{B}]$	Vertical stacking of $\mathbf{A}, \mathbf{B}$
$[\mathbf{A} \quad \mathbf{B}]$	Horizontal stacking of $\mathbf{A}, \mathbf{B}$
$\mathbf{A}(i,:)$	i^{th} row of $\mathbf{A}$
$\mathbf{A}(:,j)$	j^{th} column of $\mathbf{A}$
$\mathbf{A}(i,j)$	$(i,j)^{th}$ element of $\mathbf{A}$
$\mathbf{a}(r)$	r^{th} element of $\mathbf{a}$
$\|\mathbf{A}\|$	Frobenius norm
$\mathbf{A}^T$, $\mathbf{A}^{-1}$,$\mathbf{A}^\dagger$	Transpose of $\mathbf{A}$, Inverse of $\mathbf{A}$, Pseudoinverse of $\mathbf{A}$
$\mathbf{A}^n$	n^{th} matrix from a set of matrices $\{\mathbf{A}\}$
$\circ$	Outer product
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$\circledast$	Hadamard product
$\oslash$	Element-wise division
x_n	n-mode product
$\circledast_{i=1}^{N} \mathbf{A}^{(i)}$	$\mathbf{A}^{(N)} \circledast \cdots \circledast \mathbf{A}^{(i)} \circledast \cdots \circledast \mathbf{A}^{(1)}$
$\otimes_{i=1}^{N} \mathbf{A}^{(i)}$	$\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(i)} \otimes \cdots \otimes \mathbf{A}^{(1)}$
$\odot i = 1^N \mathbf{A}^{(i)}$	$\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(i)} \odot \cdots \odot \mathbf{A}^{(1)}$
$\mathbf{X}_{(n)}$	n-mode matricization of $\underline{\mathbf{X}}$
reshape( )	Rearrange the elements of a given matrix or tensor to a given set of dimensions
MTTKRP	Matricized Tensor Times Khatri-Rao Product
OoM	Out of Memeory

Table 2.1: Table of symbols and their description.

2.1.1 Introduction to Tensors

A multi-view graph with K views is a collection of K matrices $\mathbf{X}_1, \cdots \mathbf{X}_K$ with dimensions $I \times J$ (where I, J are the number of nodes). This collection of matrices is naturally represented as a tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$. In order to avoid overloading the

term "dimension", we call an $I \times J \times K$ tensor a three "mode" tensor, where "modes" are the numbers of indices used to index the tensor.

Definition 1 ***Rank-1 Tensor****[141, 197]: A tensor is a higher order generalization of a matrix. For example, vectors are* 1^{st}*-mode and matrices are* 2^{nd}*-mode tensors and a tensor* $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ *is an* 3^{rd}*-order tensor as shown in Figure 2.1 (adopted from [141]). An* N*-mode*[1] *tensor* $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ *is the outer product of* N *vectors, as given in equation 2.1.*

$$\underline{\mathbf{X}} = \mathbf{a}_1 \circ \mathbf{a}_2 \cdots \circ \mathbf{a}_N \tag{2.1}$$

It can essentially indexed by N *variables i.e.* $(\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N)$*. The outer product 3-mode tensor* $\underline{\mathbf{X}}$ *of vectors* $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ *can be written as* $x_{ijk} = a_i b_j c_k$ *for all values of the indices.*

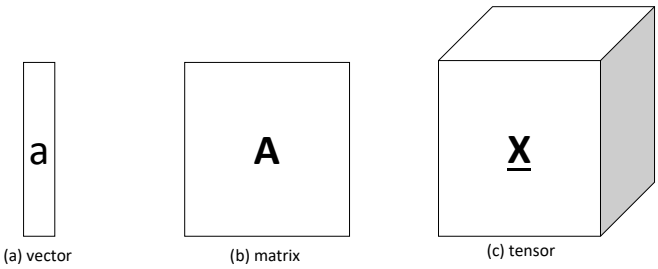


Figure 2.1: Tensor generalization: (a) 1-order tensor (i.e., vector) (b) 2-order tensor (i.e., matrix) (c) 3-order tensor $\underline{\mathbf{X}}$

Tensor Indexing: In order to index matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$, we denote its (i, j)-th element by a_{ij}, i^{th} row vector by a_i, and j^{th} column vector by a_j. Similarly, the (i, j, k)-th element of a 3rd-mode tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$ is denoted by x_{ijk} and the $(i1, i2, ..., iN)$-th

[1]Notice that the literature (and thereby this paper) uses the above terms as well as "order" interchangeably.

element of an Nth-order $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ is denoted by $y_{i_1 i_2 \ldots i_N}$. We refer to tensors with more than 3 modes as higher-order ones.

Definition 2 ***Fibers****: The fibers are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $x_{:jk}, x_{i:k}$, and $x_{ij:}$, respectively; see Figure 2.2 (adopted from [141]). When extracted from the tensor, fibers are always assumed to be oriented as column vectors*

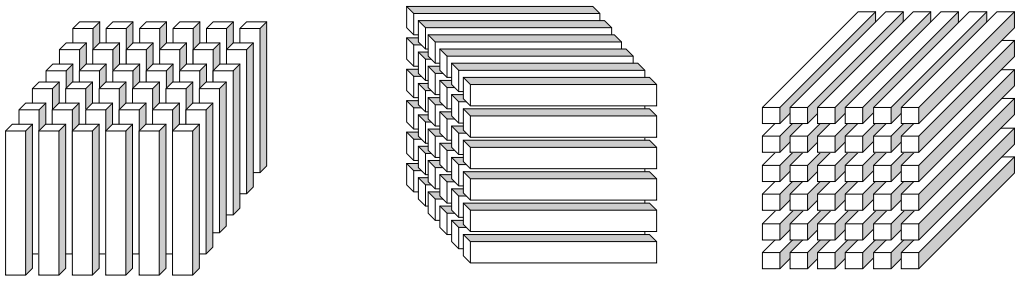


Figure 2.2: Fibers of a 3rd-order tensor (a) Mode-1 or column fibers, (b) Mode-2 or row fibers (c) Mode-3 or tube fibers

Definition 3 ***Slice*** *: A slice is a (m-1)-dimension partition of tensor where an index is varied in one mode and the indices fixed in the other modes. There are three categories of slices : horizontal ($\underline{\mathbf{X}}(i,:,:)$) , lateral ($\underline{\mathbf{X}}(:,j,:)$), and frontal ($\underline{\mathbf{X}}(:,:,k)$) for third-order tensor X as shown in Figure 2.3.*

Definition 4 Rank-1 Tensor *A N-mode tensor is of rank-1 if it can be strictly decomposed into the outer product of N vectors. Therefore, we add different scaling of a sub-tensor as*

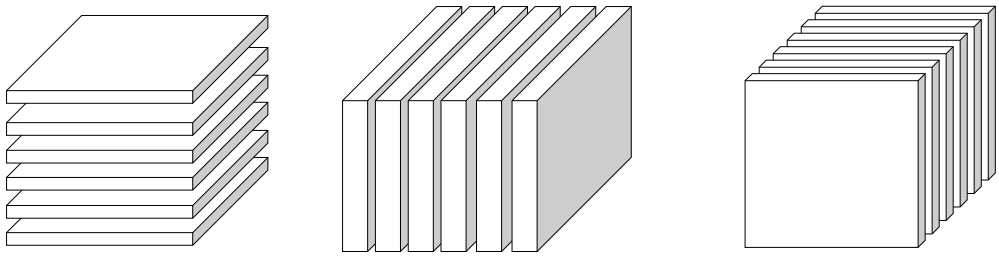


Figure 2.3: Slices of 3-order tensor (a) horizontal $\underline{\mathbf{X}}(i,:,:)$ (b) lateral $\underline{\mathbf{X}}(:,j,:)$, (c) frontal $\underline{\mathbf{X}}(:,:,k)$.

*we introduce more modes when reconstructing the full tensor. A rank-*1 3*-mode tensor can be written as* $\underline{\mathbf{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$*. A pictorial view of the rank-*1 *concept is shown in Figure (2.4).*

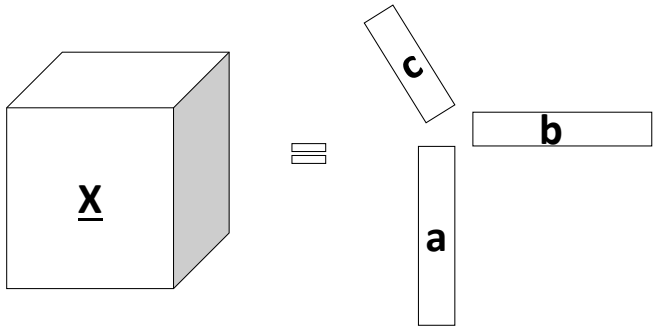


Figure 2.4: A rank-1 mode-3 tensor.

Definition 5 Tensor Rank *The rank of a tensor rank(*$\underline{\mathbf{X}}$*) =* R *is defined as the minimum number of rank-1 tensors that are required to produce* $\underline{\mathbf{X}}$ *as their sum.*

Definition 6 Outer product *of two vectors* **a** *and* **b** *of dimensions* I *and* J*, respectively is defined as:*

$$a \circ b = ab^T \tag{2.2}$$

and their outer product is an $I \times J$ *matrix.*

Definition 7 Inner product *of two equal-sized tensors* $\underline{\mathbf{X}}, \underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *is the sum of the products of their elements.*

$$\langle \underline{\mathbf{X}}, \underline{\mathbf{Y}} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_1=1}^{I_1} \cdots \sum_{i_n=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N} \tag{2.3}$$

Definition 8 Frobenius Norm *of a matrix* **A** *and* $\underline{\mathbf{X}}$ *is computed as the square root of the sum of the squares of all its elements as given in equation 2.4 and 2.5 respectively.*

$$||\mathbf{A}||_F^2 = \sqrt{\sum_{i=1}^{I} \sum_{j=1}^{J} a_{ij}^2} \tag{2.4}$$

For tensors:

$$||\underline{\mathbf{X}}||_F^2 = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_1=1}^{I_1} \cdots \sum_{i_n=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2} \tag{2.5}$$

2.1.2 Tensor Reordering

Definition 9 N-mode Matricization *It is the method that reorder the tensor into a matrix [197]. This is also know as flattening or unfolding. Given a tensor* $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$*, the folding is denoted as* $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times \Pi_{i \neq n}^{N} I_i}$ *and the concept is easier to understand using a brief example of the matricization of a third-order tensor as shown in Figure 2.5.*

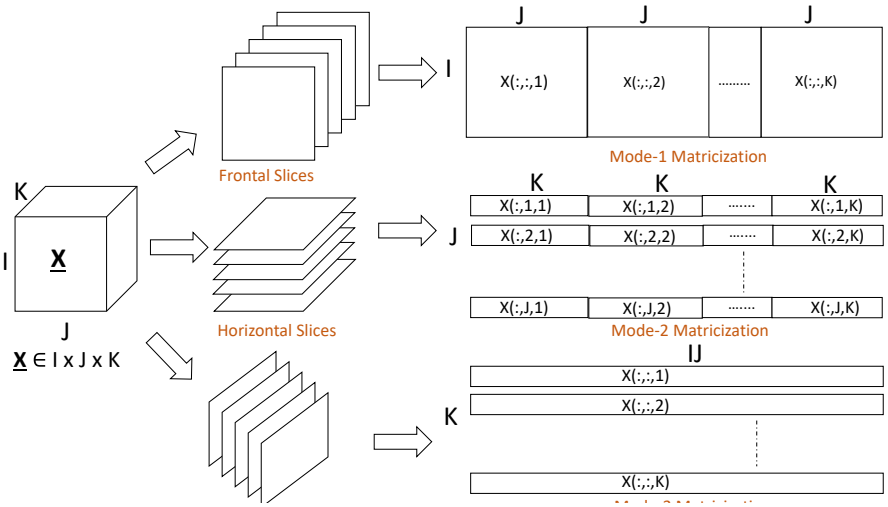


Figure 2.5: Mode-3 tensor matricization

Definition 10 ***Vectorization****: The vectorization of a tensor can be computed by vertically stacking the columns of N-mode unfolded tensor as:*

$$vec(\underline{\mathbf{X}}) = \begin{bmatrix} x_{111} \\ x_{112} \\ \vdots \\ x_{ijk} \end{bmatrix}$$

2.1.3 Matrix/Tensor Products

Definition 11 Kronecker product*[197] is denoted by symbol* $\otimes$ *and the Kronecker product of two matrices* $\mathbf{A} \in \mathbb{R}^{I \times L}$ *and* $\mathbf{B} \in \mathbb{R}^{J \times L}$ *results in matrix size of* $(IJ \times L^2)$ *and it is*

defined as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1L}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2L}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \dots & a_{IL}\mathbf{B} \end{bmatrix} \tag{2.6}$$

Definition 12 Column-wise Khatri-Rao product *[197] : It is denoted by symbol* $\odot_c$ *and the column-wise Khatri-Rao product of two matrices* $\mathbf{A} \in \mathbb{R}^{I \times L}$ *and* $\mathbf{B} \in \mathbb{R}^{J \times L}$, $\mathbf{A} \odot_c \mathbf{B} \in \mathbb{R}^{IJ \times L}$ *is defined as:*

$$\mathbf{A} \odot_c \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \dots \mathbf{a}_L \otimes \mathbf{b}_L \end{bmatrix} \tag{2.7}$$

In case of a and b are in vector form, then the Kronecker and column-wise Khatri-Rao products are same, i.e., $\mathbf{a} \otimes \mathbf{b} = \mathbf{a} \odot_c \mathbf{b}$.

Definition 13 Partition-wise Kronecker product *[157, 56] : Let* $\mathbf{A} = [\mathbf{A}_1 \ \ \mathbf{A}_2 \dots \mathbf{A}_R] \in \mathbb{R}^{I \times LR}$ *and* $\mathbf{B} = [\mathbf{B}_1 \ \ \mathbf{B}_2 \dots \mathbf{B}_R] \in \mathbb{R}^{I \times LR}$ *are two partitioned matrices. The partition-wise Kronecker product is defined by:*

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}_1 \otimes \mathbf{B}_1 & \mathbf{A}_2 \otimes \mathbf{B}_2 & \dots & \mathbf{A}_R \otimes \mathbf{B}_R \end{bmatrix} \tag{2.8}$$

Definition 14 Hadamard product *of two same size matrices* $\mathbf{A}$ *and* $\mathbf{B}$ *is their element-wise product.*

$$\mathbf{A} \circledast \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2J}b_{2J} \\ \vdots & \vdots & & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \dots & a_{IJ}b_{IJ} \end{bmatrix} \tag{2.9}$$

Definition 15 N-way product *[157, 56] : Given an N-mode tensor* $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *and a matrix* $\mathbf{A} \in \mathbb{R}^{I_n \times R}$*, the n-mode product is computed as* $\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_n \mathbf{A}$

$$\underline{\mathbf{Y}}(i_1, \ldots, i_{n-1}, r, i_{n+1} \ldots, i_n) = \sum_{j=1}^{I_n} \underline{\mathbf{X}}(i_1, \ldots, i_{n-1}, j, i_{n+1} \ldots, i_n)\mathbf{A}(j, r) \tag{2.10}$$

2.2 Tensor Decompositions and Applications

In this section, we introduce three popular tensor decompositions, namely the CANDECOMP/PARAFAC (CP) decomposition, PARAFAC2 decomposition and Block term decomposition. We refer the interested reader to several well-known surveys that provide more details on other tensor decompositions and their applications [141, 197].

2.2.1 Canonical Polyadic Decomposition

The most popular and widely used tensor decompositions are the CANonical DECOMPosition (CANDECOMP) and the PARAllel FACtors (PARAFAC) decomposition [36, 104, 21]. Both are originated from different knowledge domains and evolved independently over times, but they both boil down to the same principles, henceforth referred as canonical polyadic (CP) decomposition. The CP decomposition of a N-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_1 \cdots \times I_N}$ is defined as the sum of outer product rank-1 components given in Equ. 2.11:

$$\begin{aligned} \mathcal{L} &= \underset{\mathbf{a}_{1_r}, \mathbf{a}_{2_r}, \ldots, \mathbf{a}_{N_r}}{\arg\min} \; ||\underline{\mathbf{X}} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_{1_r} \circ \mathbf{a}_{2_r} \cdots \circ \mathbf{a}_{N_r}||_F^2 \\ &= \underset{\mathbf{a}_{1_r}, \mathbf{a}_{2_r}, \ldots, \mathbf{a}_{N_r}}{\arg\min} \; ||\underline{\mathbf{X}} - [\lambda; \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]||_F^2 \end{aligned} \tag{2.11}$$

The factor matrices $(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N)$ are the combination of the vectors from the rank-1 components:

$$\mathbf{A}_i = [\mathbf{a}_i^{(1)} \quad \mathbf{a}_i^{(2)} \ldots \mathbf{a}_i^{(R)}] \tag{2.12}$$

We can formalize the CP decomposition of 3-mode tensor as follows:

$$\text{argmin} \, ||\underline{\mathbf{X}} - \underline{\hat{\mathbf{X}}}|| \quad \text{where} \quad \underline{\hat{\mathbf{X}}} = \underset{\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r}{\arg\min} \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{2.13}$$

where $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$ and $\mathbf{c} \in \mathbb{R}^K$. R is the number of components or latent factors and also known as rank of the tensor. A pictorial view of the CP decomposition of 3-mode tensor is given in Figure 2.6. Due to its ease of interpretation, CP decomposition has became one of the most popular tensor decomposition techniques that has been extensively and widely applied in different fields. The exact case when $||\underline{\mathbf{X}} - \underline{\hat{\mathbf{X}}}|| = 0$, we refer to $\underline{\hat{\mathbf{X}}}$ being a low rank approximation of the original tensor $\underline{\mathbf{X}}$ and can be written as matricized form as:

$$\begin{aligned} \hat{\mathbf{X}}_{(1)} &= (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T \\ \hat{\mathbf{X}}_{(2)} &= (\mathbf{C} \odot \mathbf{A})\mathbf{B}^T \\ \hat{\mathbf{X}}_{(3)} &= (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T \end{aligned} \tag{2.14}$$

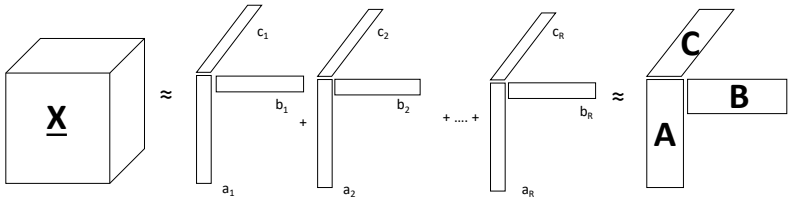


Figure 2.6: CP Decomposition of a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$.

Alternating Least Squares (ALS) Algorithm

The objective function in Equ.2.13 is highly non-convex and thus hard to directly optimize. However, we use Alternating Least Squares (ALS), a form of Block Coordinate Descent (BCD) optimization algorithm, in order to solve the problem. The main idea behind ALS is the following: when fixing all optimization variables except for one, the problem essentially boils down to linear least squares problem which can be solved optimally. Thus, ALS cycles over all the optimization variables and updates them iteratively until the value of the objective function stops changing between consecutive iterations. For the 3-way tensor case, the ALS algorithm would perform the following steps repeatedly until convergence.

$$\begin{aligned}
\mathbf{A} &\leftarrow \arg\min_{A} ||\mathbf{X}_{(1)} - (\mathbf{C} \odot \mathbf{B})\mathbf{A}^T||_F^2 \\
\mathbf{B} &\leftarrow \arg\min_{B} ||\mathbf{X}_{(2)} - (\mathbf{C} \odot \mathbf{A})\mathbf{B}^T||_F^2 \\
\mathbf{C} &\leftarrow \arg\min_{C} ||\mathbf{X}_{(3)} - (\mathbf{B} \odot \mathbf{A})\mathbf{C}^T||_F^2
\end{aligned} \tag{2.15}$$

To this minimization problem the optimal solution is given by:

$$\begin{aligned}
\mathbf{A} &= \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})^\dagger \\
\mathbf{B} &= \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A})^\dagger \\
\mathbf{C} &= \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A})^\dagger
\end{aligned} \tag{2.16}$$

The ALS algorithm 1 is simple to understand and implement but it is possible that it might take lot of iterations to converge and it is also possible that it might not converge to a global optimum. The performance is highly depends on the initialization.

Algorithm 1: Classic CP-ALS

Data: $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ and target rank R

Result: Factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$,$\mathbf{B} \in \mathbb{R}^{J \times R}$, $\mathbf{C} \in \mathbb{R}^{J \times R}$ and $\lambda \in \mathbb{R}^{R}$

Initialize $\mathbf{B}, \mathbf{C}$

while *convergence criterion is not met* **do**

  $\mathbf{A} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})$

  for *r=1,...,R* **do**

    $\lambda_r = ||\mathbf{A}(:, r)||$, $\mathbf{A}(:, r) = \mathbf{A}(:, r)/\lambda_r$

  end

  $\mathbf{B} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A})$

  for *r=1,...,R* **do**

    $\lambda_r = ||\mathbf{B}(:, r)||$, $\mathbf{B}(:, r) = \mathbf{B}(:, r)/\lambda_r$

  end

  $\mathbf{C} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A})$

  for *r=1,...,R* **do**

    $\lambda_r = ||\mathbf{C}(:, r)||$, $\mathbf{C}(:, r) = \mathbf{C}(:, r)/\lambda_r$

  end

end

Uniqueness

When there is only one possible combination of rank-1 tensors that sums to $\underline{\mathbf{X}}$, with the exception of scaling indeterminacy and permutation of vectors, the solution is refers as unique solution. The scaling indeterminacy (Equ. 2.17) is refers as the certainty that we can scale the individual vectors and permutation (Equ. 2.18) refers as the rank-1 component of the tensor can be arbitrarily reordered.

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} (\alpha_r \mathbf{a}_r) \circ (\beta_r \mathbf{b}_r) \circ (\gamma_r \mathbf{c}_r) \quad \text{s.t.} \quad \alpha_r \beta_r \gamma_r = 1 \tag{2.17}$$

$$\underline{\mathbf{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = [\![\mathbf{A}\Pi, \mathbf{B}\Pi, \mathbf{C}]\!]\Pi], \quad \Pi \in \mathbb{R}^{R \times R} \tag{2.18}$$

It is observed that rank decomposition are not generally unique for the matrices and similar is observed for higher order tensors. To analyze it further, we need to understand the concept of k-rank.

Definition 16 ***Kruskal rank****: The k-rank of a matrix* $\mathbf{A}$*, denoted by* $k_{\mathbf{A}}$*, is defined as the maximum number k such that any k columns are linearly independent [145].*

The sufficient condition of the uniqueness of the CP decomposition of N-mode tensor is in Equ. 2.19:

$$\sum_{n=1}^{N} k_{\mathbf{A}_n} \geq 2R + (N - 1) \tag{2.19}$$

For example, for 3-mode tensor sufficient condition of the uniqueness is in Equ. 2.20:

$$k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} \geq 2R + 2 \tag{2.20}$$

The above equation provide only sufficient condition for CP's uniqueness. Berge el at [248] proved that the sufficient condition is not enough for the tensor with $R > 3$. Sidiropoulos and Liu [160] provided the necessary conditions for CP's uniqueness for N-mode tensor as:

$$\min_{1,\ldots,N} \left(\Pi_{m=1,m\neq n}^{N} \text{rank}(\mathbf{A}_m) \right) \geq R \tag{2.21}$$

Applications of CP

CP decomposition is widely poplar tensor decomposition since 1970. Carroll and Chang [36] uses CP decomposition in psychometrics to analyze multiple similarity or dissimilarity matrices collected for a variety of subjects. The main concept was that by simply averaging the data across all subjects, various points of view on the data would be eliminated. They used the method on two sets of data: one set of auditory tones from Bell Labs and another set of country comparisons. In [107], used CP to obtain explainable factors to reduce the PCA ambiguity. CP decomposition has been shown to be useful in the fluorescence data modeling in chemometrics, and its application can be found in [14, 13, 234]. For neuroscientists (fMRI data), CP decomposition is a common analyzing method [175]. Similar to fMRI [51], EEG data is also a typical type of data being analyzed by CP decomposition to find the source of various seizure [1, 178, 177, 179].

From last decade, data mining researchers showed wide interest in the CP decomposition. In [3], author constructed tensor data from online chat-room and used CP decomposition to show its effectiveness. Bader, Berry, and Browne [19] used CP decomposition to automatically detect email conversations from enron data. In [194, 167], CP

is used for community detection or clustering from multi-aspect data and network traffic modeling for time-evolving networks [15, 65]. Various applications focused on outlier detection on the temporal factor [196]. [8] explored the differences and various similarities between search engines using CP decomposition. Other application domains includes healthcare [118, 274, 278], signal processing [187, 231, 232] and recommendation system [148, 208, 200]. Overall, tensor is a strong data modeling method for a wide range of real-world data, and CP decomposition has a lot of potential for analysis.

2.2.2 PARAFAC2 Decomposition

PARAFAC2 model [106] differs from CP/PARAFAC [21, 36, 104] where a low-rank trilinear model is not required. The CP decomposition applies the same factors across all the different modes, whereas PARAFAC2 allows for non-linearities such that variation across the values and/or the size of one mode as shown in Fig 5.1. PARAFAC2 with factors $\mathbf{U}, \mathbf{V}$ and $\mathbf{W}$ can be written w.r.t. the frontal slices of the tensor $\underline{\mathbf{X}}$ as:

$$\underline{\mathbf{X}}_k = \mathbf{U}_k \mathbf{S}_k \mathbf{V}^T \tag{2.22}$$

where $k = 1, \ldots, K$, $\mathbf{U}_k \in \mathbb{R}^{I_k \times R}$, $\mathbf{S}_k = diag(W(k,:)) \in \mathbb{R}^{R \times R}$ is diagonal matrix, and $\mathbf{V} \in \mathbb{R}^{J \times R}$. To preserve the uniqueness of the solution, the paper [106] proposed the constraint that the cross product $\mathbf{U}_k^T \mathbf{U}_k$ is invariant regardless of the subject k is considered. For this constraint to hold, $\mathbf{U}_k$ can be computed as given in Equ. 2.23:

$$\mathbf{U}_k \approx \mathbf{Q}_k \mathbf{H} \tag{2.23}$$

where $\mathbf{Q}_k$ is orthogonal matrix across columns and of size $\mathbb{R}^{I_k \times R}$. The matrix $\mathbf{H}$ is a positive definite matrix independent of k and of size $\mathbb{R}^{R \times R}$. Therefore,

$$\mathbf{U}_k^T \mathbf{U}_k = \mathbf{H}^T \mathbf{Q}_k^T \mathbf{Q}_k \mathbf{H} = \mathbf{H}^T \mathbf{I}_k \mathbf{H} = \Phi \tag{2.24}$$

Given the above modeling, the standard algorithm to solve PARAFAC2 for data $\underline{\mathbf{X}}$ tackles the following optimization problem:

$$\min_{\{\mathbf{U_k}\},\{\mathbf{S_k}\},\mathbf{V}} \sum_{k=1}^{K} \|\underline{\mathbf{X}}_k - \mathbf{U}_k \mathbf{S}_k \mathbf{V}^T\|_F^2 \tag{2.25}$$

subject to $\mathbf{U}_k = \mathbf{Q}_k \mathbf{H}$, $\mathbf{Q}_k^T \mathbf{Q}_k = \mathbf{I}$, and $\mathbf{S}_k$ is diagonal matrix. The $\mathbf{U}_k$ decomposed into two matrices, $\mathbf{Q}_k$ that has orthonormal columns and $\mathbf{H}$ which is invariant regardless of k.

To solve Eq (2.25), 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 $(\mathbf{HCB}^T \underline{\mathbf{X}}_k^T) = [\mathbf{P}_k, \Sigma_k, \mathbf{Z}_k^T]$. With $\mathbf{Q}_k^T = \mathbf{Z}_k \mathbf{P}_k^T$ fixed, the rest of factors can be obtained as:

$$\begin{aligned} \mathcal{L} &= \min_{\mathbf{H},\{\mathbf{S}_k\},\mathbf{V}} \frac{1}{2} ||\mathbf{Q}_k \underline{\mathbf{X}}_k - \mathbf{H}\mathbf{S}_k \mathbf{V}^T||_2^F \, s.t. \mathbf{Q}_k \mathbf{Q}_k^T = \mathbf{I}_r \\ &\min_{\mathbf{H},\{\mathbf{S}_k\},\mathbf{V}} \frac{1}{2} ||\underline{\mathbf{Y}} - \mathbf{H}\mathbf{S}_k \mathbf{V}^T||_2^F \end{aligned} \tag{2.26}$$

A pictorial view of the PARAFAC2 decomposition is shown in Figure (2.7)

The constrained version of Equ. (2.26) can be written as:

$$\mathcal{L} = \min_{\mathbf{H},\{\mathbf{S}_k\},\mathbf{V}} \frac{1}{2} ||\underline{\mathbf{Y}} - \mathbf{H}\mathbf{S}_k \mathbf{V}^T||_2^F + \frac{\alpha}{2} (\mathcal{R}(\underline{\mathbf{Y}}; \mathbf{H}, \mathbf{S}_k, \mathbf{V})) \tag{2.27}$$

PARAFAC2 is unique under certain conditions pertaining to the number of matrices (K), full column rank of $\mathbf{U}$, the positive definiteness of Φ, and non-singularity of $\mathbf{S}_k$ [108, 247]. There have been several published work regarding the uniqueness property of

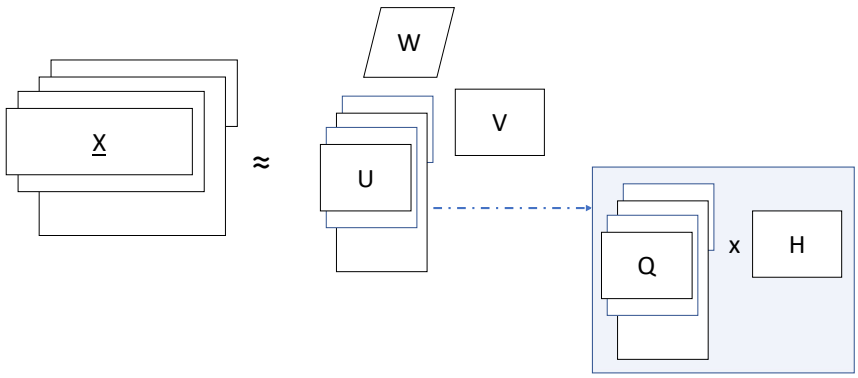


Figure 2.7: PARAFAC2 Decomposition of a third-order tensor $\mathbf{X}_k \in \mathbb{R}^{I_k \times J} \quad k \in [1, \ldots, K]$.

PARAFAC2 [108, 141, 247]. However the most relevant towards the large-scale data is that PARAFAC2 is unique for $K \geq 4$ [239].The classic method of PARAFAC2 (Algorithm 2)is limited to dense data and require large amount of resources (time and space) to process big data.The author [202] 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 [6] 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 in chapter 10.

Algorithm 2: Classic PARAFAC2-ALS

Data: $\{\mathbf{X}_k \in \mathbb{R}^{I_k \times J}\}$ for k = 1,...,K and target rank R

Result: Factor matrices $\{\mathbf{U}_k \in \mathbb{R}^{I_k \times R}\}$, $\{\mathbf{S}_k \in \mathbb{R}^{R \times R \times K}\}$ $\forall$ k, $\mathbf{V} \in \mathbb{R}^{J \times R}$ and $\mathbf{H} \in \mathbb{R}^{R \times R}$

Initialize $\mathbf{H}$, $\mathbf{V}$ and $\mathbf{S}_k$

while *convergence criteria* **do**

  for $k = 1,\ldots,K$ **do**

    $[\mathbf{P}_k, \Sigma_k, \mathbf{Z}_k^T] = \text{SVD}(\mathbf{H}\mathbf{S}_k\mathbf{V}^T\mathbf{X}_k^T)$

    $\mathbf{Q}_k = \mathbf{Z}_k\mathbf{P}_k^T$

  end

  for $k = 1,\ldots,K$ **do**

    $\mathbf{Y}_k = \mathbf{Q}_k^T\mathbf{X}_k$

  end

  $[\mathbf{H}, \mathbf{V}, \mathbf{W}] = \text{CP-ALS}\ (\mathbf{Y}_k)$

  for $k = 1,\ldots,K$ **do**

    $\mathbf{S}_k = diag(\mathbf{W}(k,:))$

  end

  for $k = 1,\ldots,K$ **do**

    $\mathbf{U}_k = \mathbf{Q}_k\mathbf{H}$

  end

end

PARAFAC2 Applications

Bro et al [30] used PARAFAC2 to handle time shifts in resolving chromatographic data with spectral detection. In this application, the irregular mode corresponds to elution time. The PARAFAC2 model did not assume parallel proportional elution profiles but in the application the elution profiles matrix preserves its "inner-product structure" across samples. Wise et al [267] applied PARAFAC2 to the problem of fault detection in a semi-conductor etch process. Chew et al. [39] used PARAFAC2 for clustering documents across multiple languages

PARAFAC2, a linear decomposition method, is well suited for the data and yields robust, valid, and automated models that allow for the detection of erroneous measurements. Most recently, Perros et al [202], Afshar et al [6] and Yin et al [277] used PARAFAC2 for the mining of temporally-evolving phenotypes on data taken from medically complex pediatric patients. The idea is to extend classic method to large scaled sparse data and incorporate constraints such as temporal smoothness, sparsity, and non-negativity in the resulting factors. The resulting PARAFAC2 decomposition is such that factors reveal concise phenotypes and meaningful temporal profiles of patients.

Augustijn et al [17] used for the isothermal chemical denaturation (ICD) data and yields robust, valid, and automated models that allow for the detection of erroneous measurements. The idea is to model the data entirely by PARAFAC2 method and the, detect outliers using a cutoff based on their contribution to the residual tensor. The outliers are removed, and another PARAFAC2 model is obtained and validated. Finally, outliers can then predicted in the denaturant dependent mode.

2.2.3 Block Term Decomposition

De Lathauwer et al. [56, 52, 53] introduced a new type of tensor decomposition that unifies the Tucker and the CP decomposition and referred as Block Term Decomposition (BTD). The BTD of a 3-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, shown in figure 2.8, is a sum of rank-(L, M, N) terms is a represented as:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^{R} \mathcal{G}_r \times_1 \mathbf{A}_r \times_2 \mathbf{B}_r \times_3 \mathbf{C}_r \tag{2.28}$$

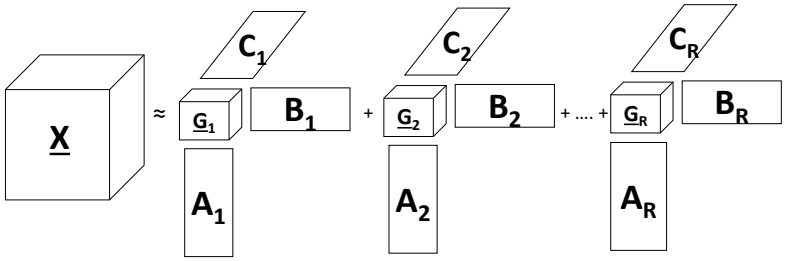


Figure 2.8: BTD -(L, M, N) for a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$.

The factor matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ are defined as $\mathbf{A} = [\mathbf{A}_1 \;\; \mathbf{A}_2 \; \ldots \mathbf{A}_R] \in \mathbb{R}^{I \times LR}$, $\mathbf{B} = [\mathbf{B}_1 \;\; \mathbf{B}_2 \ldots \mathbf{B}_R] \in \mathbb{R}^{J \times MR}$ and $\mathbf{C} = [\mathbf{C}_1 \;\; \mathbf{C}_2 \ldots \mathbf{C}_R] \in \mathbb{R}^{K \times NR}$. The small core tensors $\mathcal{G}_r \in \mathbb{R}^{L \times M \times N}$ are full rank-(L, M, N). If R=1, then Block-term and Tucker decompositions are same. In terms of the standard matrix representations of $\underline{\mathbf{X}}$, (2.28) can be written as:

$$\begin{aligned}
\mathbf{X}^{(1)}_{I \times JK} &\approx \mathbf{A}.(blockdiag(\mathcal{G}^{(1)}_1 \ldots \mathcal{G}^{(1)}_R)).(\mathbf{C} \odot \mathbf{B})^T \\
\mathbf{X}^{(2)}_{J \times IK} &\approx \mathbf{B}.(blockdiag(\mathcal{G}^{(2)}_1 \ldots \mathcal{G}^{(2)}_R)).(\mathbf{C} \odot \mathbf{A})^T \\
\mathbf{X}^{(3)}_{K \times IJ} &\approx \mathbf{C}.(blockdiag(\mathcal{G}^{(3)}_1 \ldots \mathcal{G}^{(3)}_R)).(\mathbf{B} \odot \mathbf{A})^T
\end{aligned} \tag{2.29}$$

In terms of the $(IJK \times 1)$ vector representation of $\underline{\mathbf{X}}$, the decomposition can be written as:

$$\mathbf{x}_{IJK} \approx ((\mathbf{C} \odot \mathbf{B}) \odot \mathbf{A}) \begin{bmatrix} (\mathcal{G}_1)_{LMN} \\ \cdots \\ (\mathcal{G}_R)_{LMN} \end{bmatrix} \tag{2.30}$$

Algorithm: The direct fitting of Equ. (2.30) is difficult. A various types of fitting algorithms [258, 141] have been derived and discussed in the literature for tensor decompositions. The most common fitting for tensor decomposition is by using ALS (Alternating Least Squares) and approximation loss can be written as:

$$\mathcal{LS}(\underline{\mathbf{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \underline{\mathcal{G}}) = \underset{\mathbf{A},\mathbf{B},\mathbf{C},\underline{\mathcal{G}}}{\arg\min} ||\underline{\mathbf{X}} - \sum_{r=1}^{R} [\![\mathbf{A}_r, \mathbf{B}_r, \mathbf{C}_r, \underline{\mathcal{G}}_r]\!]||_F^2 \tag{2.31}$$

The main idea behind alternating least squares (ALS) is to fix all the factor matrices except for one, then the problem reduces to a linear least squares which can be solved optimally. The ALS fitting of BTD Decomposition [53] and to promote reproducibility, we provide clean implementation of the method as described in Algorithm 3.

Uniqueness

It was provided in [52] that the BTD is essentially unique up to scaling, permutation and the simultaneous post multiplication of $\mathbf{A}_r$ by a non-singular matrix $\mathbf{F}_r \in \mathbb{R}^{L \times L}$, $\mathbf{A}_r$ by a non-singular matrix $\mathbf{G}_r \in \mathbb{R}^{M \times M}$, and $\mathbf{C}_r$ by a non-singular matrix $\mathbf{H}_r \in \mathbb{R}^{N \times N}$, provided that $\mathcal{G}_r$ is replaced by $\mathcal{G}_r \bullet_1 \mathbf{F}_r^{-1} \bullet_2 \mathbf{G}_r^{-1} \bullet_3 \mathbf{H}_r^{-1}$ and the matrices $[\mathbf{A}_1 \quad \mathbf{A}_2 \ldots \mathbf{A}_R]$ and $[\mathbf{B}_1 \quad \mathbf{B}_2 \ldots \mathbf{B}_R]$ are full column rank.

Algorithm 3: ALS algorithm for BTD Decomposition

Data: $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, Rank (L_r, M_r, N_r)

Result: Factor matrices $\mathbf{A}, \mathbf{B}$, $\mathbf{C}$ and core tensor $\underline{\mathbf{D}}$

Initialize R blocks of $\mathbf{B} \in \mathbb{R}^{J \times M}$, $\mathbf{C} \in \mathbb{R}^{K \times N}$ and $\underline{\mathcal{G}} \in \mathbb{R}^{L \times M \times N}$

while *convergence criterion is not met* **do**

Update $\mathbf{A} \leftarrow \mathbf{X}^{(1)}.(\mathcal{G}_1^{(1)}.(\mathbf{C}_1 \otimes \mathbf{B}_1)^T \dots \mathcal{G}_R^{(1)}.(\mathbf{C}_R \otimes \mathbf{B}_R)^T)^\dagger$

Update $\mathbf{B} \leftarrow \mathbf{X}^{(2)}.(\mathcal{G}_1^{(2)}.(\mathbf{C}_1 \otimes \mathbf{A}_1)^T \dots \mathcal{G}_R^{(2)}.(\mathbf{C}_R \otimes \mathbf{A}_R)^T)^\dagger$

Update $\mathbf{C} \leftarrow \mathbf{X}^{(3)}.(\mathcal{G}_1^{(3)}.(\mathbf{B}_1 \otimes \mathbf{A}_1)^T \dots \mathcal{G}_R^{(3)}.(\mathbf{B}_R \otimes \mathbf{A}_R)^T)^\dagger$

Update $\underline{\mathbf{D}} \leftarrow \begin{bmatrix} (\mathcal{G}_1)_{LMN} \\ \vdots \\ (\mathcal{G}_R)_{LMN} \end{bmatrix} \leftarrow ((\mathbf{C}_1 \otimes \mathbf{B}_1 \otimes \mathbf{A}_1) \dots (\mathbf{C}_R \otimes \mathbf{B}_R \otimes \mathbf{A}_R))^\dagger \mathbf{x}_{IJK}$

end

Proof of Uniqueness[52, 53, 57]: Suppose that the conditions a) $N > L + M - 2$ and b) $k'_{\mathbf{A}} + k'_{\mathbf{B}} + k'_{\mathbf{C}} \geq 2R + 2$, hold and that we have an alternative decomposition of $\underline{\mathbf{X}}$, represented by $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\underline{\mathbf{D}}})$, with $k_{\overline{\mathbf{A}}'}$ and $k_{\overline{\mathbf{B}}'}$ maximal under the given dimensionality constraints. $\overline{\mathbf{A}} = \mathbf{A}\Pi_a\Delta_a$ and $\overline{\mathbf{B}} = \mathbf{B}\Pi_b\Delta_b$, in which Π is a block permutation matrix and Δ is a square non-singular block-diagonal matrix, compatible with the structure of factor matrix.

It suffices to prove it for $\mathbf{A}$. The result for $\mathbf{B}$ and $\mathbf{C}$ can be obtained by switching modes. Let $\omega(\mathbf{x})$ denote the number of nonzero entries of a vector $\mathbf{x}$.

From [52], Lemma 5.2 (i), Upper-bound on $\omega'(\mathbf{x}^{\mathbf{T}}\overline{\mathbf{A}})$: The constraint on $k_{\overline{\mathbf{A}}'}$ implies that $k'_{\overline{\mathbf{A}}} \geq k'_{\mathbf{A}}$. Hence, if $\omega'(\mathbf{x}^{\mathbf{T}}\overline{\mathbf{A}}) \leq R - k'_{\overline{\mathbf{A}}} + 1$ then

$$\omega'(\mathbf{x}^{\mathbf{T}}\overline{\mathbf{A}}) \leq R - k'_{\overline{\mathbf{A}}} + 1 \leq R - k'_{\mathbf{A}} + 1 \leq k'_{\mathbf{B}} + k'_{\mathbf{C}} - (R + 1)$$

where the last inequality corresponds to condition (a).

From [52], Lemma 5.2 (ii), Lower-bound on $\omega(\mathbf{x}^{\mathbf{T}}\overline{\mathbf{A}})$: After columns are sampled in the column space of the corresponding sub-matrix of $\mathbf{B}$ and $\mathbf{C}$, lower bound is

$$\omega'(\mathbf{x}^{\mathbf{T}}\overline{\mathbf{A}}) \geq min(\gamma, k'_{\mathbf{B}}) + min(\gamma, k'_{\mathbf{C}}) - \gamma$$

From [52], Lemma 5.2 (iii), Combination of the two bounds.

$$min(\gamma, k'_{\mathbf{B}}) + min(\gamma, k'_{\mathbf{C}}) - \gamma \leq \omega'(\mathbf{x}^{\mathbf{T}}\overline{\mathbf{A}}) \leq k'_{\mathbf{B}} + k'_{\mathbf{C}} - (R + 1)$$

If matrix $\mathbf{A}$ or $\mathbf{B}$ or $\mathbf{C}$ is tall and full column rank, then its essential uniqueness implies essential uniqueness of the overall tensor decomposition. We call the decomposition essentially unique when it is subject only to these trivial indeterminacies i.e $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \underline{\mathbf{D}})$ and $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{C}}, \overline{\underline{\mathbf{D}}})$ are equal.

BTD-(L, L, 1)

The LL1-decomposition [57] of a 3-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ with tensor rank R is a sum of blocks with rank-$(L_r, L_r, 1)$, is a represented as:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^{R} (\mathbf{A}_r \cdot \mathbf{B}_r^T) \circ \mathbf{c}_r \tag{2.32}$$

where factor blocks $A_r \in \mathbb{R}^{I \times L_r}$ and the matrix $B_r \in \mathbb{R}^{J \times L_r}$ are both rank-L_r, $1 \leq r \leq R$. Here, the factor matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is of dimension $\mathbf{A} = [\mathbf{A}_1 \;\; \mathbf{A}_2 \; \ldots \mathbf{A}_R] \in \mathbb{R}^{I \times LR}$, $\mathbf{B} = [\mathbf{B}_1 \;\; \mathbf{B}_2 \ldots \mathbf{B}_R] \in \mathbb{R}^{J \times LR}$ and $\mathbf{C} = [\mathbf{c}_1 \;\; \mathbf{c}_2 \ldots \mathbf{c}_R] \in \mathbb{R}^{K \times R}$. The $(L \times L)$ identity matrix is represented by $I_{L \times L}$. 1_L is a column vector of all ones of length L.

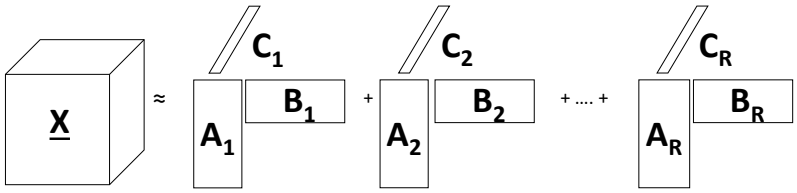


Figure 2.9: BTD -(L, L, 1) for a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$.

BTD Applications

The BTD framework offers a coherent viewpoint on how to generalize the basic concept of rank from matrices to tensors. The author [122] presented an application of BTD where epileptic seizures pattern was non-stationary, such a trilinear signal model is insufficient. The epilepsy patients suffer from recurring unprovoked seizures, which is a cause and a symptom of abrupt upsurges. They showed the robustness of BTD against

these sudden upsurges with various model parameter settings. The author [41] used a higher-order BTD for the first time in fMRI analysis. Through extensive simulation, they demonstrated its effectiveness in handling strong instances of noise. A deterministic block term tensor decomposition (BTD) - based Blind Source Separation [54, 209] method was proposed and offered promising results in analyzing the atrial activity (AA) in short fixed segments of an AF ECG signals. The paper [58] extends the work [54] for better temporal stability. The paper [180] proposed doubly constrained block-term tensor decomposition to extract fetal signals from maternal abdominal signals.

2.2.4 Tucker Decomposition

In 1963, the Tucker decomposition as shown in Figure 2.10 was proposed by Tucker [256, 253] and later, in 19 clarified by Levin [156] and Tucker [253, 255, 254]. A decomposition of a 3-mode tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ with Rank P, Q, and R is defined as the sum of outer product rank-1 components and one small core tensor $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$:

$$\begin{aligned} \mathcal{L} &= \arg\min ||\underline{\mathbf{X}} - \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r||_F^2 \\ &\arg\min ||\underline{\mathbf{X}} - \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \bullet_3 \mathbf{C}||_F^2 \\ &\arg\min ||\underline{\mathbf{X}} - [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}||_F^2]\!] \end{aligned} \tag{2.33}$$

Most fitting algorithms consider columnwise orthonormality of the factor matrices, but it is not necessary. Actually, CP decomposition is considered as special case of tucker decomposition when $P = Q = R$ and block term decomposition is also a special case of tucker when $R = 1$.

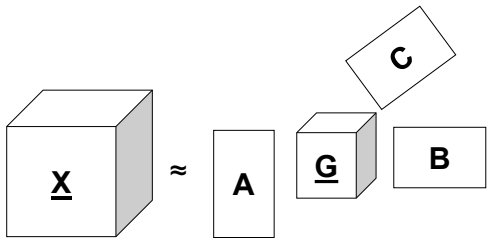


Figure 2.10: Tucker decomposition for a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$.

2.3 Streaming Tensor Decomposition

This section gives the background of the second research problem we focus in this thesis: streaming tensor decomposition. We start with an introduction to streaming tensors. Next, we provide representative researches that have been proposed for various types of streaming tensors and a brief review of the more recent and advanced approaches.

2.3.1 Streaming Tensors

In many real-world applications, data grow dynamically and may do so in many modes. For example, given a dynamic tensor in a movie-based recommendation system, organized as *users* $\times$ *movie* $\times$ *rating* $\times$ *hours*,the number of registered users, movies watched or rated, and hours may all increase over time. Another example is network monitoring sensor data where tons of information like source and target IP address, users, ports etc., is collected every second. In the Social Networks example, we observe user interactions every few seconds, which can be translated to new tensor slices, after fixing the temporal granularity. Such type of data that evolve over time is referred as streaming or online tensor

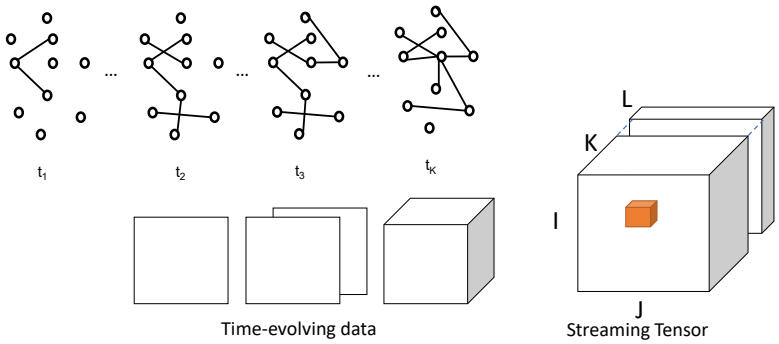


Figure 2.11: Illustration of streaming tensors

data. The Figure 2.11 illustrates tensor example where the orange region indicates a single element in the tensor such as number of interactions between two friends at any timestamp or it could be a number of packets sent from a source IP to a destination IP through a certain port by specific user.

In such conditions, the update needs to process the new data very quickly, which makes non-incremental methods to fall short because they need to recompute the decomposition for the full dataset.

2.3.2 Streaming CP/PARAFAC Decomposition

There is very limited study on online CP decomposition methods.

Phan *el at.* [203] had purposed a theoretic method namely GridTF to large-scale tensors decomposition based on CP's basic mathematical theory to get sub-tensors and join the output of all decompositions to achieve final factor matrices.

Sidiropoulos *el at.*[186], proposed algorithm that focus on CP decomposition namely RLST (Recursive Least Squares Tracking), which recursively update the factors by minimizing the mean squared error.

In 2014, Sidiropoulos *el at.* [230] , proposed a parallel algorithm for low-rank tensor decomposition that is suitable for large tensors.

The paper by Zhou, *el at.* [284] describes an online CP decomposition method, where the latent components are updated for incoming data. These state-of-the-art techniques focus on only fast computation but not effective memory usage.

Besides CP decomposition, Tucker decomposition methods[243, 192] were also introduced. Online Tucker decomposition was first proposed by Sun *el at.*[243] as ITA (Incremental Tensor Analysis). In there research, they described the three variants of Incremental Tensor Analysis. First, they proposed DTA i.e. Dynamic tensor analysis which is based on calculation of co-variance of matrices in traditional higher-order singular value decomposition in an incremental fashion. Second, with the help of the SPIRIT algorithm, they found approximation of DTA named as Stream Tensor Analysis (STA). Third, they proposed window-based tensor analysis (WTA). To improve the efficiency of DTA, it uses a sliding window strategy. Liu *el at.* [192] proposed an efficient method to diagonalize the core tensor to overcome this problem. Other approaches replace SVD with incremental SVD to improve the efficiency. Hadi *el at.* [70] proposed multi-aspect-streaming tensor analysis (MASTA) method that relaxes constraint and allows the tensor to concurrently evolve through all modes. These methods were not only able to handle data increasing in one-mode, but also have solution for multiple-mode updates using methods such as incre-

mental SVD. The latest line of work is introduced in [18] i.e TuckerMPI to find inherent low-dimensional multi-linear structure, achieving high compression ratios. Tucker is mostly focused on recovering subspaces of the tensor, rather than latent factors, whereas our focus is on the CP/PARAFAC decomposition which is more suitable for exploratory analysis.

Another line of work is incremental tensor completion. The main difference between completion and decomposition techniques is that in completion "zero" values are considered "missing" and are not part of the model, and the goal is to impute those missing values accurately, rather than extracting latent factors from the observed data. The earliest work on incremental tensor completion traces back to [168], and recently, Qingquan *el at.*[238], proposed streaming tensor completion based on block partitioning.

2.3.3 Streaming PARAFAC2 Decomposition

The PARAFAC2 model was first developed by Harshman [106] 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. The classic method of PARAFAC2 [106] is limited to small sized dense data and require large amount of resources (time and space) to process big data. The author [202] 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 [6] 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 in Chapter 11.

2.3.4 Streaming Block Term Decomposition

The Block Term Decomposition (BTD) unifies the CP and Tucker Decomposition. The BTD framework offers a coherent viewpoint on how to generalize the basic concept of rank from matrices to tensors. The author [122] presented an application of BTD where epileptic seizures pattern was non-stationary, such a trilinear signal model is insufficient. The epilepsy patients suffer from recurring unprovoked seizures, which is a cause and a symptom of abrupt upsurges. They showed the robustness of BTD against these sudden upsurges with various model parameter settings. The author [41] used a higher-order BTD for the first time in fMRI analysis. Through extensive simulation, they demonstrated its effectiveness in handling strong instances of noise. A deterministic block term tensor decomposition (BTD) - based Blind Source Separation [54, 209] method was proposed and offered promising results in analyzing the atrial activity (AA) in short fixed segments of an AF ECG signals. The paper [58] extends the work [54] for better temporal stability. The paper [180] proposed doubly constrained block-term tensor decomposition to extract fetal signals from maternal abdominal signals. Recently, the paper [93] proposed ADMM based constrained BTD method to find structures of communities within social network data. However, these classic methods and applications of BTD are limited to small-sized static

dense data ($1K \times 1K \times 100$) and require a large amount of resources (time and space) to process big 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 the best of our knowledge, our proposed method **OnlineBTD** (Chapter 10) is the first approach to track streaming block term decomposition while not only being able to provide stable decompositions but also provides better performance in terms of efficiency and scalability.

2.4 Conclusion

In this Chapter we presented the necessary notation used throughout this thesis, as well as algorithms and applications of CP, PARAFAC2 and Block term decomposition, which are the decompositions used in this thesis. Furthermore, we also discussed the streaming tensor and various incremental existing decomposition methods to built a solid foundation for better understanding.

Part I

Mining Graphs and Networks

Chapter 3

Semi-supervised Multi-Aspect Community Detection

"How to find communities or clusters in large multi-aspect graphs? Can we leverage semi-supervision to improve accuracy?"

Community detection in real-world graphs has been shown to benefit from using multi-aspect information, e.g., in the form of "means of communication" between nodes in the network. An orthogonal line of work, broadly construed as semi-supervised learning, approaches the problem by introducing a small percentage of node assignments to communities and propagates that knowledge throughout the graph. In this chapter we introduce SMACD, a novel semi-supervised multi-aspect community detection method that effectively integrates and leverages both (a) the multi-view nature of real graphs, and (b) partial supervision in the form of community labels for a small number of the nodes along with an automated parameter tuning algorithm which essentially renders SMACD parameter-

free. To the best of our knowledge, SMACD is the first approach to incorporate multi-aspect graph information and semi-supervision, while being able to discover overlapping and non-overlapping communities. Our results on real and synthetic data demonstrate that SMACD, through combining semi-supervision and multi-aspect edge information, outperforms the baselines and yields high clustering accuracy. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, and Evangelos E. Papalexakis. "Smacd: Semi-supervised multi-aspect community detection." In Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 702-710. Society for Industrial and Applied Mathematics, 2018.

3.1 Introduction

Community detection in real graphs is a widely pervasive problem with applications in social network analysis and collaboration networks, to name a few. There have been continuing research efforts in order to solve this problem. Traditionally, research has focused plain graphs where the only piece of information present is the nodes and the edges [154].

In most real applications, however, the information available usually goes beyond a plain graph that captures relations between different nodes. For instance, in an online social network such as Facebook, relations and interactions between users are inherently *multi-aspect* or *multi-view*, i.e., they are naturally represented by a set of edge types rather than a single type of edge. Such different edge-types can be "who messages whom", "who pokes whom", "who-comments on whose timeline" and so on. There exists a significant

body of work that uses this multi-view nature of real graphs for community detection. Indicatively,[245, 43, 194, 63] proposed algorithm combines multiple views of a graph in order to detect communities more accurately.

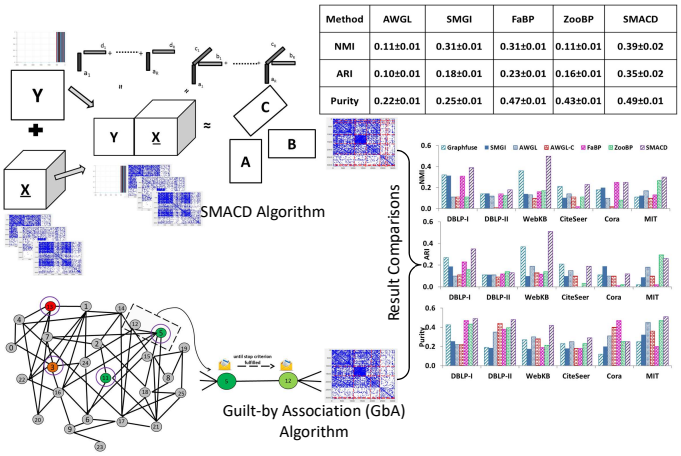


Figure 3.1: SMACD vs state-of-art techniques: Our proposed method SMACD successfully combines multi-view graph information and semi-supervision and outperforms state-of-the-art techniques.

Another line of work leverages partial ground truth information that may be available to us. Such partial ground truth information manifests as a small percentage of nodes for which we know the community where they belong. These partial node labels may be obtained via questionnaires or by leveraging domain expert opinion, however, since the process of obtaining those labels may be costly and time-consuming, we assume that they represent a small percentage of the nodes in our graph. The most popular school of thought that takes such partial ground truth into account are the so called "Guilt-by-Association" or label propagation techniques where the main idea is that affinity between nodes implies affiliation

with the same community and those techniques iteratively propagate the known node labels throughout the graph estimating the unknown labels. Belief propagation [144, 275] is one of the widely used "Guilt-by-Association" method and has been very successful in various real life scenarios including community detection. In view of that method which only gives non zero weight to every graph Karsuyama et al. [131] proposed another multiple graph learning method (SMGI), where weight can be sparse. Auto-weighted Multiple-Graph Learning (AWGL) [184] framework learn the set of weights automatically for all graphs and classify graphs into different classes.

Our contributions are as follows:

- **Novel Approach**: We introduce SMACD as shown in Fig. (3.1), a semi-supervised multi-aspect[1] community [2] detection algorithm. To the best of our knowledge, this is the first principled method for leveraging multiple views of a graph and an existing (small) percentage of node labels for community detection and is able to handle *overlapping* communities.
- **Algorithm**: Under the hood of SMACD runs our proposed algorithm for Non-Negative Sparse Coupled Matrix-Tensor Factorization (NNSCMTF) which jointly decomposes a tensor that represents a multi-view graph, and a matrix which contains partial node label information. NNSCMTF introduces latent sparsity and non-negativity constraints to the Coupled Matrix-Tensor Factorization model [4], which are well suited for community detection.

[1]Note that in the paper we use the terms multi-view and multi-aspect interchangeably
[2]Note that in the paper we use the terms community and cluster interchangeably

- **Automated Parameter Turning**: Sparsity introduced by NNSCMTF is controlled by a parameter which if chosen arbitrarily may not yield the best possible performance. We introduce SelSPF, an automated parameter tuning algorithm that does not rely on the partial node labels and selects a value for the sparsity parameter which yields performance in terms of community detection accuracy which is on par with the one obtained when doing an exhaustive search for that parameter based on all the ground truth available to us.
- **Evaluation on Real Data**: We conduct extensive experiments in order to evaluate SMACD's performance in comparison to state-of-the-art methods.

Reproducibility: We encourage reproducibility and extension of our results by making our Matlab implementation and the synthetic data we used available at link [3]. Note also that all the datasets we use for evaluation are publicly available.

The rest of the chapter is organized as follows. Section 3.2 discusses related work. In Section 3.3, we formally introduce our problem and outlined our proposed method SMACD, and in Section 3.5 we present our experimental evaluation. Finally, in Section 3.6 we conclude with a few remarks for future extensions of this work.

3.2 Related work

We provide review of work related to our problem here.

Multi-vew Clustering/Community Detection: Real data usually exhibit different cross-domain relations and can be represented as multi-view graphs.In [25] the authors

[3] http://www.cs.ucr.edu/~egujr001/ucr/madlab/src/SHOCD.zip

introduce a graph theoretic based community detection algorithm over multi-view graphs and relationships between nodes represented by various types of edges. In [78] the authors proposed two algorithms namely Weighted Simultaneous Symmetric Non-negative Matrix Trifactorization (WSSNMTF) and Natural Gradient Weighted Simultaneous Symmetric Non-negative Matrix Trifactorization (NG-WSSNMTF); these algorithms work on binary as well as weighted graphs but are limited to symmetric adjacency matrices. In [194] the authors introduce a robust algorithm for community detection on multi-view graphs based on tensor decomposition which uses a regularized CP model with sparsity penalties. In addition to different graph views, "time" is also a multi-aspect feature of a graph. In [246] the authors propose a method for identifying and tracking dynamic communities in time-evolving networks. Finally, most recently in [227] the authors introduce a CP-based community detection framework for egonets.

Heterogeneous Information Networks (HIN): Heterogeneous Information Networks are versatile representations of networks that involve multiple typed objects (or nodes) and multiple typed links denoting different relations (or edges). There is a fairly rich body of work in the literature working on related problems to ours [127, 261], however, we were unable to find an implementation directly applicable to the problem at hand for experimental comparison.

Guilt-by-Association techniques: Guilt-by-Association is a general framework of techniques which propagate partial knowledge in the graph and make inferences pertaining to the nodes of that graph or multi-view graphs. Belief Propagation [275] is one of the most widely used techniques for multi-view graphs, which has been successfully used in com-

munity detection in collaboration networks [144]. Closely related to Belief Propagation is Random Walk with Restarts (RWR) and related techniques [251] . In [144, 75] the authors unify different Guilt-by-Association techniques into a very efficient framework. In [68], author approximates Belief Propagation in undirected heterogeneous graph (i.e., a graph that consists of different types of nodes and edges) to speed-up the process.

Semi-supervised approaches: Semi-supervised learning is generally the learning framework where only a small portion of labels is present and the vast majority of data points are unlabeled. The author of [286] provides a concise overview of different semi-supervised techniques. An example of semi-supervised multi-view graph classification can be found in [125, 184] where the authors introduce a graph regularize and a small set of node labels in order to predict the class of all the nodes in a heterogeneous graph.

Tensor and Coupled Models: For a detailed overview of different tensor models and coupled matrix-tensor models we refer the reader to two concise survey papers [141, 197]. Coupled Matrix-Tensor Factorization (CMTF) has received an increasing amount of attention in the recent years and a detailed overview of the publication history of CMTF can be found in [197]. Most relevant to our proposed NNSCMTF model are different tensors models with sparsity constraints, such as [77, 34] which is the first CP decomposition with latent factor sparsity, and [69] which introduces a Tucker decomposition with a sparse core, as well as constrained CMTF models, such as [5, 265] where the authors introduce scalar weights on each component which are regularized for sparsity, thereby resulting in a decomposition which is flexible and contains individual and shared components between the tensor and the matrix.

To the best of our knowledge the NNSCMTF model has not been previously proposed. Most relevant to our proposed framework, Cao *et al.* [34], propose a semi-supervised learning framework, based on matrix-tensor coupling. We were unable to directly compare the method of [34] as released because the focus of [34] is 4-mode tensors.

3.3 Problem Formulation

Graphs can represent a large variety of data and relations between data entities. Each entity represented by node or vertex (V) and relation between entities are defined by weighted or unweighted edges (E_i, w_i). In this paper we focus on multi-view or multi-aspect graphs, i.e., a collection of graphs for the same set of nodes and different set of edges per view or layer. In the remainder of the paper we use the terms "view", "aspect", and "layer" interchangeably. Each graph can be represented using an adjacency matrix, a square node-by-node matrix that indicates an edge (and a potential weight associated to it) between two nodes. A multi-view graph can be, thus, represented as a collection of adjacency matrices.

The goal of our work is to identify communities in that multi-view graph, which essentially boils down to assigning each node into one of R community labels. In order to simplify our problem definition, we assume that R is given to us. (there exist, however, heuristics in tensor literature [199] that can deal with an unknown R).

The problem that we solve is the following:

Given (a) a multi-view or multi-aspect graph, and (b) a p% of node labels to R communities, **find** an assignment of all nodes of the graph to one (or more) of the R communities.

3.4 Proposed Method: SMACD

As [194] has demonstrated, using higher-order information for the edges of a graph, such as the "means of communication", results in more accurate community detection. What if we additionally have semi-supervision in the form of community labels for a small subset of the individuals? In this section we introduce SMACD which formulates this problem as a matrix-tensor couple as shown in Figure 3.2, where the matrix contains the community labels for the small subset of users that are known, and missing values for the rest of its entries. The key rationale behind SMACD is the following: Using the coupled matrix that contains partial label information for each node will provide a *soft guide* to the tensor decomposition with respect to the community structure that it seeks to identify. Thus, using this side information we essentially guide the decomposition to compute a solution which bears a community structure as close to the partial labels as possible (in the least squares sense).

In [34] the authors propose semiBAT, where they follow a different approach of incorporating semi-supervision in the context of matrix-tensor coupling: instead of a bilinear decomposition for $\mathbf{Y}$ (the partial label matrix) which provides soft guidance to the structure discovery, semiBAT explicitly uses a classification loss in the objective function. In [34] the goal classification of brain states, rather than discovering community structure, thus explicitly using the classification loss instead of taking a low-rank factorization of the label matrix seems more appropriate.

At a high level, SMACD takes as input a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times I \times K}$ which contains the multi-view graph, a matrix $\mathbf{Y} \in \mathbb{R}^{I \times R}$ containing the node assignments to communities,

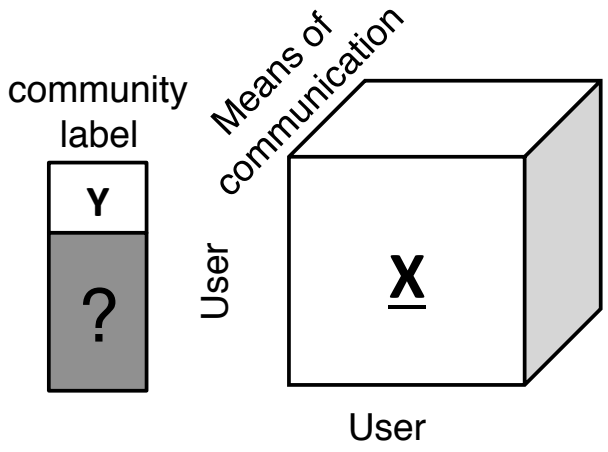


Figure 3.2: SMACD: Semi-supervised community detection via coupling

and the number of communities R (which is given implicitly through matrix $\mathbf{Y}$. SMACD consists of the following two steps.

Step 1: Decomposition Given $\underline{\mathbf{X}}, \mathbf{Y}$ compute an $R-1$ component Sparse and Non-negative MTF (as shown below in Section 3.4.1). The columns of $\mathbf{A}$ and $\mathbf{B}$ contain soft assignments of each node to one of $R-1$ communities. Both matrices contain similar information (which in practice ends up being almost identical, especially in cases where we have symmetric tensors in the first two modes).

Step 2: Hard Assignment In this step we assign each node to a single community by finding the community with maximum membership. This translates to finding the maximum column index for each row (which corresponds to each node). In the previous step we have computed a sparse decomposition which causes a number of the nodes to have all-zero rows in $\mathbf{A}$, i.e., they have no assignment to any of the $R-1$ communities. We assign those nodes to the R-th community which essentially is meant for capturing all remaining variation that our CP model in the CMTF decomposition was unable to capture. Step 2 is necessary

only in the case where we have *non-overlapping* communities. However, SMACD works for overlapping communities as well, simply by eliminating Step 2 and computing Step 1 for R communities instead of $R-1$ as we show in Section 3.4.2.

3.4.1 Nonnegative Sparse Coupled MatrixTensor Factorization

In this section we describe our model along with an Alternating Least Squares algorithm that computes a locally optimal solution. We propose two constraints on top of the CMTF model , motivated by community detection:

Non-negativity Constraint: SMACD uses the factor matrices $\mathbf{A}, \mathbf{B}$ as community assignments. Such assignments are inherently non-negative numbers (a negative assignment to a community is hard to interpret and is not natural). Thus, in NNSCMTF we impose element-wise non-negativity constraints (denoted as $\mathbf{A} \geq 0$) to all factor matrices. In addition to interpretability, non-negativity constraints have recently been shown to promote uniqueness in matrix decompositions [121] (note that the CP decomposition is already unique [197]) which, in turn, improves the quality of our results.

Latent Sparsity Constraint: In order to (a) further enhance interpretability and (b) suppress noise, we impose latent sparsity to the factors of the model. Intuitively, we would like the coefficients of the factor matrices to be non-zero only when a node belongs to a particular community, thus eliminating the need for adhoc thresholding. To that end we introduce ℓ_1 norm regularization for all factors which promotes a sparse solution.

The proposed model is:

$$\min_{\mathbf{A}\geq\mathbf{0},\mathbf{B}\geq\mathbf{0},\mathbf{C}\geq\mathbf{0},\mathbf{D}\geq\mathbf{0}} \|\underline{\mathbf{X}} - \sum_r \mathbf{A}(:,r)\circ\mathbf{B}(:,r)\circ\mathbf{C}(:,r)\|_F^2 + \|\mathbf{Y} - \mathbf{A}\mathbf{D}^T\|_F^2 \\ + \lambda\sum_{i,r}|\mathbf{A}(i,r)| + \lambda\sum_{j,r}|\mathbf{B}(j,r)| + \lambda\sum_{k,r}|\mathbf{C}(k,r)| + \lambda_d\sum_{l,r}|\mathbf{D}(l,r)| \tag{3.1}$$

where λ is the sparsity regularizer penalty. The above objective function is highly non-convex and thus hard to directly optimize. However, we use Alternating Least Squares (ALS), a form of Block Coordinate Descent (BCD) optimization algorithm, in order to solve the problem of Eq. 3.1. The reason why we choose ALS over other existing approaches, such as Gradient Descent [4], , is the fact that ALS offers ease of implementation and flexibility of adding constraints and regularizers, does not introduce any additional parameters that may influence convergence, and as a family of algorithms has been very extensively studied and used in the context of tensor decompositions. The main idea behind ALS is the following: when fixing all optimization variables except for one, the problem essentially boils down to a constrained and regularized linear least squares problem which can be solved optimally. Thus, ALS cycles over all the optimization variables and updates them iteratively until the value of the objective function stops changing between consecutive iterations. In ALS/BCD approaches, such as the one proposed here, when every step of the algorithm is solved optimally, then the algorithm decreases the objective function monotonically.

In the following lines we demonstrate the derivation of one of the ALS steps. Let us denote $\mathbf{X}_{(i)}$ the i-th mode matricization or unfolding of $\underline{\mathbf{X}}$, i.e., the unfolding of all slabs of $\underline{\mathbf{X}}$ into an $I \times JK$ matrix (we refer the interested reader to [141] for a discussion on matricization), then because of properties of the CP/PARAFAC model [141], fixing $\mathbf{B}, \mathbf{C}, \mathbf{D}$ we have

$$\min_{\mathbf{A}\geq\mathbf{0}} \|\mathbf{X}_{(1)} - \mathbf{A}[(\mathbf{B}\odot\mathbf{C})^T\|_F^2 + \|\mathbf{Y} - \mathbf{A}\mathbf{D}^T\|_F^2 + \lambda\sum_{i,r}|\mathbf{A}(i,r)|$$

$$\Rightarrow \min_{\mathbf{A}\geq 0}\|[\ \mathbf{X}_{(1)}\ ;\mathbf{Y}] - \mathbf{A}[(\mathbf{B}\odot\mathbf{C})^T\ \ \mathbf{D}^T]\|_F^2 + \lambda\sum_{i,r}|\mathbf{A}(i,r)| \tag{3.2}$$

$$\Rightarrow \min_{\mathbf{A}\geq 0}\|[\mathbf{L} - \mathbf{A}\mathbf{M}]\|_F^2 + \lambda\sum_{i,r}|\mathbf{A}(i,r)|$$

where $\mathbf{L} = [\ \mathbf{X}_{(1)}\ ;\mathbf{Y}]$, and $\mathbf{M} = [(\mathbf{B}\odot\mathbf{C})^T\ \ \mathbf{D}^T]$. This problem is essentially a Lasso regression on the columns of $\mathbf{A}$ [249] and we use coordinate descent to solve it optimally . The update formulas for $\mathbf{B}, \mathbf{C}, \mathbf{D}$ follow the same derivation after fixing all but the matrix that is being updated. Algorithm 4 outlines all the update steps of the ALS algorithm for NNSCMTF. In our implementation we set the stopping criterion to be that the absolute relative error between two consecutive iterations is smaller than 10^{-8}, and the maximum number of iterations is set to 10^3.

Algorithm 4: SMACD ALS using NNSCMTF

Data: Tensor $\underline{\mathbf{X}}$ of size $I \times J \times K$, Shared matrix $\mathbf{Y}$ of size $I \times R$, number of clusters R, λ.

Result: Factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of size $I \times R$, $J \times R$ and $K \times R$.

1 $\mathbf{X}_{(1)}$ = [X(1,:,:) X(2,:,:) ..... X(I,:,:)] : mode-1 fiber

2 $\mathbf{X}_{(2)}$ = [X(:,1,:) X(:,2,:) ..... X(:,J,:)] : mode-2 fiber

3 $\mathbf{X}_{(3)}$ = [X(:,:,1) X(:,:,2) ..... X(:,:,K)] : mode-3 fiber

4 **while** *not converged* or *max Iterations* **do**

9 **end**

3.4.2 Overlapping Communities

Our goal is to design an algorithm which consumes tensor $\underline{\mathbf{X}} = \{X_1, X_2....., X_N\}$ along with small amount of labels $\mathbf{Y}$ and it outputs the set of collection of subsets of Nodes V which we considered as overlapping clusters. Thus, we will refer to nodes with multiple classes as overlapping nodes. In real-world networks, these nodes represent bridges between different communities. For this reason, the ability to identify these bridges or overlapping nodes, although often neglected, is necessary for evaluating the accuracy of any community detection algorithms. Given $\underline{\mathbf{X}}$ and $\mathbf{Y}$, CP decomposition is used to learn latent factors which detect community structure.

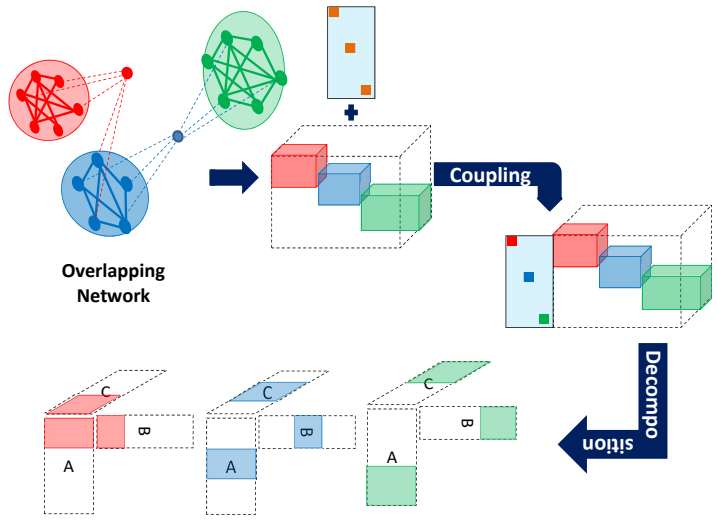


Figure 3.3: SMACD successfully combines multi-view graph information and semi supervision.

Multi-view connectivity of tensor and coupling with label matrix as shown in Fig. (3.3) can increase the robustness of community detection in the case of highly-mixed communities. Overlapping communities amounts to soft clustering over the nodes, as opposed to hard clustering which forces each node to belong to a unique community. The advocated approach only requires slight modifications in Step 2 to yield soft community assignments, that is, by treating A_{nk} as the normalized affiliation of node n to community k, and requiring $\mathbf{A}_{nk} \geq t$ per node n. The tensor factorization with regularization can now be written as follows:

$$\min_{\mathbf{A}\geq\mathbf{0},\mathbf{B}\geq\mathbf{0},\mathbf{C}\geq\mathbf{0},\mathbf{D}\geq\mathbf{0}} \|\underline{\mathbf{X}} - \sum_r \mathbf{A}(:,r)\circ\mathbf{B}(:,r)\circ\mathbf{C}(:,r)\|_F^2 + \|\mathbf{Y} - \mathbf{A}\mathbf{D}^T\|_F^2 +$$

$$\lambda\sum_{i,r}|\mathbf{A}(i,r)| + \lambda\sum_{j,r}|\mathbf{B}(j,r)| + \lambda\sum_{k,r}|\mathbf{C}(k,r)| + \lambda_d\sum_{l,r}|\mathbf{D}(l,r)| \tag{3.3}$$

$$s.t. \quad ||A_n||_1 \geq t \quad \forall n = 1\ldots K$$

Every step of the algorithm [4] is solved optimally, then the algorithm decreases the objective function monotonically. Once the algorithm returns the solution for NNSCMTF, the rows of factor matrix $\mathbf{A}$ provides the community association in networks with overlapping communities where a node can be associated with more than one community. To evaluate SMACD's result with ground truth communities, we compared resultant $A_{i,j}$ with threshold t and node is assigned with community 'r' if $\mathbf{A}(i,j) \geq t$. Each node's predicted label (or labels) is ordered incrementally based on corresponding value of $\mathbf{A}(i,j)$.

$$\text{Predicted Label(s)\{i\}} = \begin{cases} indices(A(i,j)), & \text{if } \mathbf{A}(i,j) \geq t \\ R+1, & \text{otherwise} \end{cases} \tag{3.4}$$

3.4.3 SelSPF: Automated Selection of the Sparsity Penalty

The SMACD model contains the λ sparsity penalty, which if not chosen correctly may have an impact on the final result. Traditionally, such parameters are chosen via trial-and-error, and in fact, all the baseline methods that we compare against in Section 3.5 follow this empirical approach for their parameter tuning. On the other hand, as part of SMACD we introduce SelSPF (based on principle of Armijo-Goldstein's rule [4] for selecting step size in backtracking line search methods), an automated algorithm that selects a "good" value of λ which achieves accuracy which is on-par with a brute force selection of λ based on community detection accuracy, which obviously entails knowing *all* community labels.

The intuition behind SelSPF is simple: Start with a very high λ which gives all-zero community assignments. Start decreasing λ on a logarithmic scale until a solution is reached for which all communities have at least one node assigned to them. Subsequently focus the search on a grid that starts from the previous stopping point and increase λ to the last point before at least one of the communities is empty again. SelSPF is based on the fact that λ and sparsity levels in the latent factors are directly related. We provide a

[4] https://en.wikipedia.org/wiki/Backtracking_line_search

detailed outline of SELSPF in Algorithm 5. Essentially with the introduction of SELSPF, there is no need for hand-tuning SMACD via trial-and-error.

Algorithm 5: SELSPF for automated selection of λ

Data: Tensor $\underline{\mathbf{X}}$, Shared matrix $\mathbf{Y}$, R, initial λ_{high}, τ = Step Size.

Result: Best λ value for our SMACD.

1 Set $\lambda = \lambda_{high}$ and iteration counter j=0.

2 Until $Rank\{f(\underline{\mathbf{X}}, \mathbf{Y}, \lambda_j)\} \geq R$ is satisfied, increment j and set $\lambda_j = \tau\, \lambda_{j-1}$

3 Split (λ_j,λ_{j-1}) into a grid of $\frac{1}{\tau}$ values and repeat step 2 with the λ_{j-1}.

4 Return λ as the solution.

3.4.4 Analysis of Algorithm

As we demonstrate in Section 3.5.4, this automatic selection of λ yields a very close solution in terms of accuracy to the brute force selection. Let us consider that the upper limit for λ is λ_{high} =10^8 and lower limit of λ is λ_{low} =10^{-8}. and step size varies in powers of 10. If we were to do a brute force search, we would need to run an experiment on every λ that falls between $\lambda_{low} \leq \lambda \leq \lambda_{high}$. For example λ = 10 to λ=100, we have testcases for λ={10, 20, 30....., 100} with step size of 10. Thus, in the worst case , we would need to run NNSCMTF for {(Upper-power of 10,L_h)-(Lower-power of 10,L_l)* number of testcases (T)} i.e (8-(-8))*10=160 times. Worst case running time will be $O((L_h - L_l) * T)$ for the brute force selection of λ. On the other hand, SELSPF requires only $O(L_h - L_l + T)$iterations to find suitable value of λ (26 in our example).

3.4.5 Deciding the Number of Communities

As it is currently described, SMACD takes the number of communities R as an input. A natural question that may arise is "what if the number of communities is unknown?"; this in fact can also happen even in the semi-supervised case, where we may have *partial* knowledge about the number of communities, i.e., we know that there are at least R communities in the graph. In this case, we are essentially interested in estimating the number of latent factors in a tensor $\underline{\mathbf{X}}$, which is generally a very hard problem, however, there exist efficient heuristic approaches that provide good estimates [199]. Therefore, SMACD is also extensible under partial knowledge of the number of communities.

3.4.6 Missing Values

SMACD does not recover the missing data in multi-view graph networks.This might result in overfitting or high biasing. Traditionally, we can handle missing data problem in different ways. Firstly by using weighted ALS in each step but it results in slower algorithm. Secondly, using Gradient Descent with a weights [4], it may give us more local minima than ALS. Finally, by imposing Stochastic Gradient Descent [273] which handles only non missing data, but there is no guarantee of convergence because community detection in multi-view graph networks is highly non-convex problem. To overcome this issue, we use ℓ_1 regularization by constraining the factor matrices. The results of our SMACD outperform baselines and indicates that ℓ_1 regularization takes care of overfitting and biasing.

3.4.7 Convergence

Here we demonstrate the convergence of Algorithm 3.4.1 for NNSCMTF on all real datasets that we use for evaluation. Figure 3.4 summarizes the convergence of the algorithm, showing the approximation error as a function of the number of iterations. It is clear that the algorithm converges to a very good approximation error (in the order of 10^{-5}) within 10-20 iterations. For each dataset, computation cost was average 12 sec/iteration.

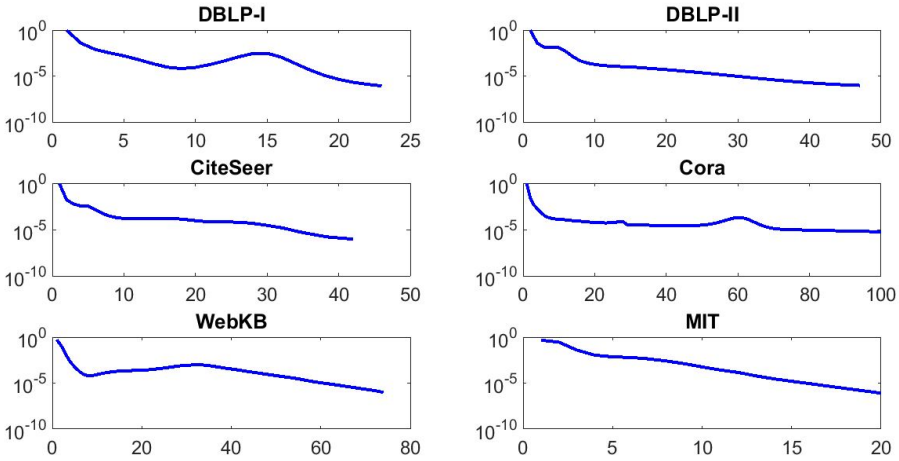


Figure 3.4: Approximation error vs. number of iterations. NNSCMTF converges very quickly to error as low as 10^{-5}.

3.5 Experiments

In this section we extensively evaluate the performance of SMACD on two synthetic and eight real datasets, and compare its performance with state-of-the-art approaches which either use multi-view graphs or semi-supervision (but not both) for community de-

tection. We implemented SMACD in Matlab using the functionality of the Tensor Toolbox for Matlab [20] which supports efficient computations for sparse tensors.

3.5.1 Data-set description

Synthetic Data Description

In order to fully control the community structure in our experiments we generate synthetic multi-view graphs with different cluster density. We generally follow the synthetic data creation of [194]. We partition the adjacency matrices corresponding to different graph views into different blocks, each one corresponding to a community. We then assign different nodes to each block with a probability which is a function of the density of the block (i.e., community) we desire; if this probability is not close to 1, then there will be a considerable amount of nodes falling outside of those blocks, effectively acting as noise. We further corrupt those datasets with random Gaussian noise with variance 0.05. We construct two synthetic datasets: Synthetic-1 has 5 views and 5 communities and has very few "cross-edges", whereas Synthetic-2 has 3 views and higher number of "cross-edges", making it a harder dataset. We include those synthetic datasets in our code package.

Real Data Description

In order to truly evaluate the effectiveness of SMACD, we test its performance against six real datasets that have been used in the literature. Those datasets are: DBLP-I, DBLP-II, Cora, CiteSeer, WebKB, and MIT reality mining dataset. DBLP-I and DBLP-II datasets are collected from the DBLP online database and were used in [194]. In DBLP-I and DBLP-II, the first graph view represents citations of one author to another. The second

view represents co-authorship relations. Finally the third view relates two authors if they share any three terms in a title or in abstract of their publication. DBLP-I contains authors who published in SIGIR, TODS, STOC+FOCS. DBLP-II contains who published in venues, PODS, ICDE, CACM and TKDE.These publication venues constitute the communities.

The Cora dataset [223] was collected from the LINQS online database, and consists of 2708 machine learning publications and citations. This network consists of 5429 edges and 7 different communities. The groups are categorized in 7 different classes i.e. Neural Networks, Rule Learning, Case Based, Probabilistic methods, Reinforcement based and Theory. CiteSeer dataset [223]consists of 3312 publications related to AI, DB, IR, ML and HCI research categories. The first view connects papers based on their word vectors, and second view connects the paper based on citations.

The WebKB dataset [223] is small dataset of 878 web pages of Washington universities which belong to 5 categories, namely courses, facilities, student, project and staff. We consider these categories as ground truth classes for WebKB. We considered these categories as ground truth classes. Finally, the MIT reality mining [62], collected by researchers at MIT, consists of 87 mobile users information collected on campus. Ground truth is the self-reported affiliation of the users.

Overlapping communities: The "Insight Resources (IR) Repository" (a.k.a IRR) consists of five multi-view datasets with manual annotation of user stances (e.g., political or sports). Our interest is in Rugby Union dataset[83]. It is a collection of 854 international Rugby Union players, clubs , and organizations active on Twitter. The ground truth consists of communities corresponding to 15 countries. The communities are overlap-

ping, as players can be assigned to both their home nation and the nation in which they play club rugby. SNOW2014G dataset is first introduced in [213] and author extracted largest connected component and retweet social interactions to form the graph edges from the tweet collection. It consists of top 10992 users, 3 views and clustered them into 90 classes.

3.5.2 Evaluation Measures

We evaluate the community detection performance in terms of three different quality measures: Normalized Mutual Information (NMI), Adjacent Random Index (ARI) and Purity. These measures provide a quantitative way to compare the obtained communities $\Omega = w_1, w_2, ..., w_r$ to ground truth classes C= $c_1, c_2, ..., c_r$.

- **Normalized Mutual Information (NMI)**: Mathematically NMI is defined as:

$$\text{NMI}(\Omega, C) = \frac{I(\Omega, C)}{[H(\Omega) + H(C)]} \tag{3.5}$$

  where I(Ω,C) is mutual information between cluster Ω and C, H(Ω) and H(C) are entropy of cluster and classes. Next, Purity is defined as the ratio of number of nodes correctly extracted to total number of nodes.

- **Purity**: Mathematically Purity is defined as:

$$\text{Purity}(\Omega, C) = \frac{1}{N} \sum_{k=0}^{N} max|w_k \cap c_k| \tag{3.6}$$

  where w_k and c_k are the number of objects in a community and a class respectively. $|w_k \cap c_k|$ is the interaction of objects of w_k and c_k

- **Adjacent Random Index (ARI)**: Finally when interpreting communities as binary decisions of each object pair, Adjacent Random Index(ARI) is defined as:

$$ARI(\Omega, c) = \frac{tp + tn}{tp + fp + fn + tn}, \quad \omega(c_1, c_2) = \frac{\omega_u(c_1, c_2) - \omega_e(c_1, c_2)}{1 - \omega_e(c_1, c_2)} \tag{3.7}$$

  where tp, tn, fp and fn are true positive, true negative, false positive and false negative, respectively.

- **Omega index (ω)**: Omega index (ω) is the overlapping version of the Adjusted Random Index (ARI). Omega index considers the number of nodes pairs belong together in no clusters, how many are belong together in exactly single cluster or exactly two clusters, and so on.

NMI, Purity and ARI (or Omega index for overlapping community) are defined on the scale $[0, 1]$ and the higher the score, the better the community quality is.

3.5.3 Baselines for Comparison

Here we briefly present the state-of-the-art baselines. For each baseline we use the *reported parameters* that yielded the best performance in the respective publications. For fairness, we also compare against the parameter configuration for SMACD that yielded the best performance in terms of NMI. However, moving one step further, we also evaluate SelSPF and demonstrate that an unsupervised selection of parameters yields qualitatively the same performance for SMACD as the brute force selection. All comparisons were carried out over 50 iterations each, and each number reported is an average with a standard deviation attached to it.

- **GraphFuse** [194]: GraphFuse is a tensor decomposition based approach which can be seen as a special case of SMACD when there is no semi-supervision. The sparsity penalty factor λ for DBLP-I, DBLP-II, CiteSeer, Cora, WebKB and MIT is set for λ= 0.000001, 0.0001, 0.000001, 0.1, 0.00005 and 0.00001, respectively and a maximum of 150 iterations was used for convergence.
- **WSSNMTF and NG-WSSNMTF** [78]: The details for the methods are described in [78]. We used the SVD matrix initialization. The sparsity penalty parameter η for WSSNMTF and NG-WSSNMTF are chosen as DBLP-I and DBLP-II ($\eta_1 = \eta_2 =$ 0.01), for CiteSeer ($\eta_1 = \eta_2 = 1$) , Cora ($\eta_1 = 0.01$, $\eta_2 = 10$) ,WebKB ($\eta_1 = \eta_2 = 0.01$) and MIT ($\eta_1 = 1$, $\eta_2 = 1000$). These η values are chosen to lead to best clustering performance and max 100 iterations are used for reaching the convergence.
- **Fast Belief Propagation (FaBP)** [144]: FaBP is a fast, iterative Guilt-by-Association technique, in particular conducting Belief Propagation. A belief in our case is a community label for each node. We used one-vs-all technique for multi-clustering.
- **ZooBP** [68]: ZooBP works on any undirected heterogeneous graph with multiple edge types. As in FaBP, a belief here is a community label for each node.
- **SMGI** [131]: Sparse Multiple Graph Integration method is another method of integrating multiple graphs for label propagation, which introduces sparse graph weights which eliminate the irrelevant views in the multi-view graph.
- **AWGL** [184]: Parameter-Free Auto-Weighted Multiple Graph Learning is the latest auto-weighted multiple graph learning framework, which can be applied to multi-view unsupervised (AWGL-C) as well as semi-supervised (AWGL) clustering task.

- **Parameter Tuning** In order to be on-par with the baselines, we tuned SMACD's parameter λ so that we obtain the maximum performance. We provided $\leq$10% labels in matrix and rest of labels are empty. The maximum number of iterations for SMACD is set to 10^3. We perform experiments with various values of λ ranging from 10^{-8} to 10^6 on all real multi-view networks to explore the behaviour of our algorithm. λ is chosen to give best clustering results in terms of NMI, for DBLP-I, DBLP-II, CiteSeer, Cora, WebKB and MIT values for λ= 0.3,0.09, 0.0001,1, 0.9 and 600, respectively. For both the synthetic data, penalty factor is set to 1. For overlapping communities, we use $\mathbf{t}$=0.1 for both datasets.

3.5.4 Experimental Results

Below we extensively evaluate SMACD and compare it against baseline methods.

Comparison with Baselines

For all datasets we compute Normalized Mutual Information, Purity and Adjacent Random Index. For SMACD, AWGL , SMGI, ZooBP and FaBP we use labels for 10% of the nodes in each dataset.

The results for the Synthetic data are shown in Figure 3.5, with each bar-plot corresponding to a different method. We observe that SMACD performed better than other approaches when applied on SYN-I and SYN-II. SYN-I is designed with high cluster density in layer 2 and 3, and noisy links, and has high number of cross-community edges

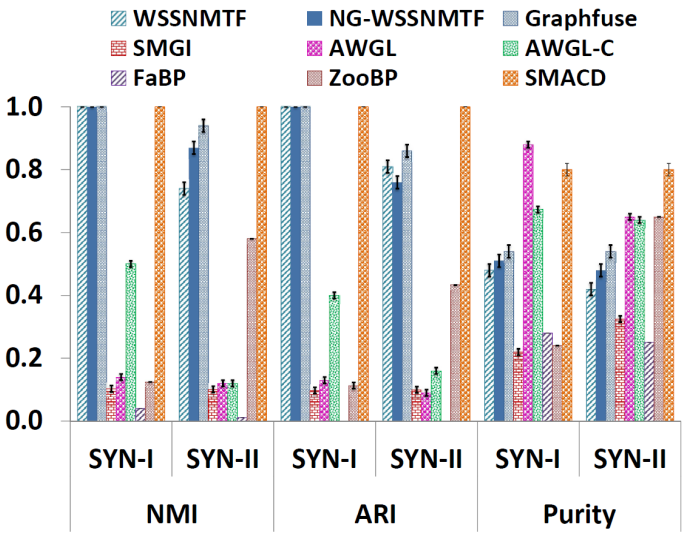


Figure 3.5: Experimental results of SYN-I and SYN-II dataset: SMACD outperforms the baselines.

between nodes. Given that, we found that SMACD achieved the highest NMI, ARI and Purity. We omit the figure of the results due to space restrictions.

The most interesting comparison, however, is on the real datasets, since they present more challenging cases than the synthetic ones. SMACD outperforms the other state-of-the-art approaches in most of the real multi-view networks, with the exception of Cora. In the cases of DBLP-I and DBLP-II, SMACD gave better results compared to the baselines , specifically in terms of NMI and Purity. For Citeseer, SMACD has comparable behavior with the baselines in terms of NMI. Most importantly, however, SMACD achieves the highest NMI, ARI, and Purity for WebKB and MIT, arguably the hardest of the six real datasets we examined and have been analyzed in the literature.

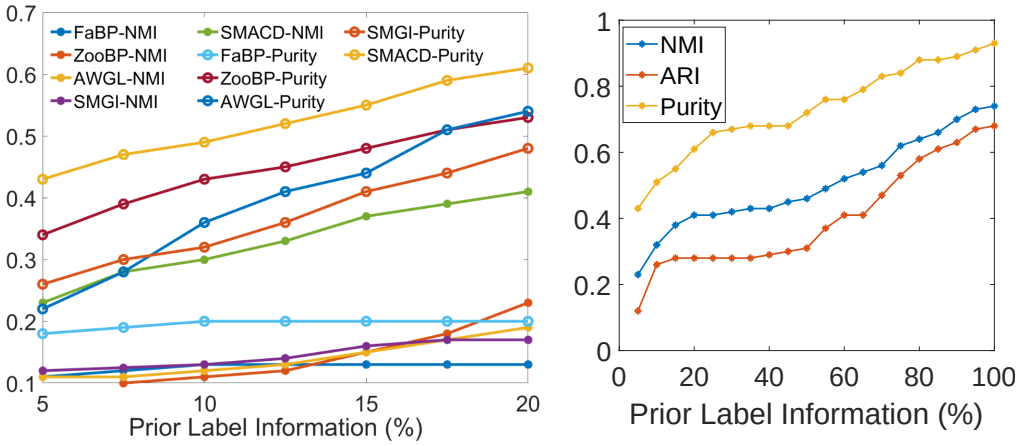


Figure 3.6: (a) SMACD vs. Guilt-by-Association(FaBP and ZooBP), AWGL and SMGI for different degrees of semi-supervision for DBLP-I. (b) Performance of SMACD as a function of the number of labels. These results confirm the intuition, since performance improves as the number of labels increases.

SMACD Performance on overlapping communities

We report the accuracy of our method for real world Rugby and SNOW2014G datasets with overlapping communities. Figure 3.7 shows the results. SMACD outpeforms all other state-of-art methods in accuracy (or purity) measures. In particular our method has always the highest purity score and in all but one case it has the best NMI score with small (i.e. 2.5%) number of prior label information. This shows that coupled tensor and matrix allows the overlapping community structure to be more easily and accurately detectable. We ran SMACD and other state-of-art methods for 20 times using 2.5-30% prior label information. In Figure 3.7 we present our clustering accuracy on both data-sets.

For SNOW2014G dataset our algorithm outperformed by predicting cluster with ≈ **3x, 4x, 2x and 5x** more accuracy than AWGL, SMGI, ZooBP and FaBP respectively.

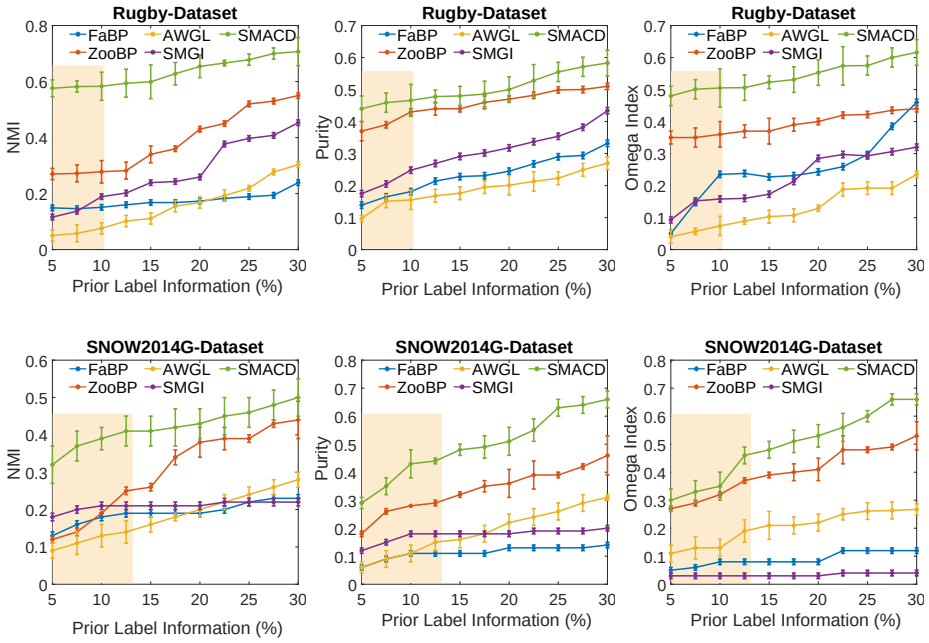


Figure 3.7: Experimental results for NMI, Purity and Omega index . SMACD consistently outperforms the baseline with an upward trend as the number of available labels increases and works better in *small amounts of labels*.

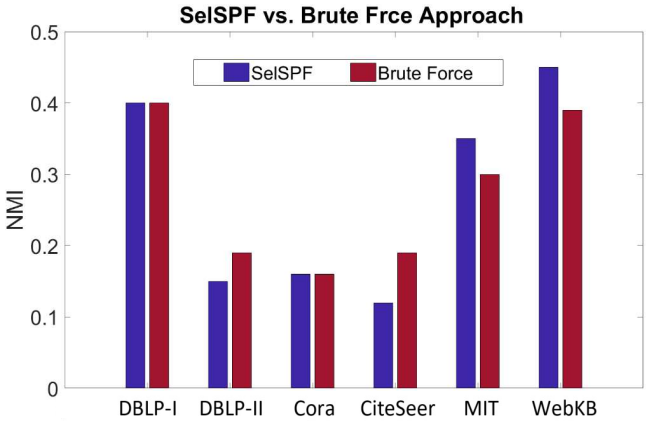
Performance vs. Degree of Semi-supervision

Next, we evaluate the performance of SMACD compared to Guilt-by-Association as a function of the degree of semi-supervision, i.e., the percentage of available labels. We performed experiments for the DBLP-I dataset for 5%, 10% and 20% labeled nodes and we show the results in Figure 3.6(a) showing a consistent trend between the two methods. We further measure the performance of SMACD as the number of labels grows, and summarize

the results in Figure 3.6(b) where we can see that what we would expect intuitively holds true: the more labels we have the higher the community accuracy. Due to limited space, we show the trend only for DBLP-I but we observe similar behavior for the rest of the datasets.

Evaluation of SelSPF

We evaluate the effectiveness of SelSPF in choosing a λ that yields good community quality. We compare SMACD's performance with respect to the λ chosen using a brute force evaluation (on 50 iterations per λ) of the performance according to NMI (using all the labels), against the selection made by SelSPF. Figure 3.8 demonstrates that, in terms of NMI, both parameter selections in fact yield very comparable (if not identical) performance. This result indicates that SMACD can be used by practitioners as a black box, without the need for specialized and tedious trial-and-error tuning.



λ	DBLP-I	DBLP-II	Cora	Citeseer	MIT	WebKB
SelSPF	0.3	0.09	1	0.001	600	0.9
Brute force	0.5	0.001	1	0.0001	100	0. 1

Figure 3.8: λ selection using SELSPF vs. brute force approach. SELSPF is able to choose a value for λ which yields similar accuracy as the expensive and grossly impractical brute force approach, effectively rendering SMACD parameter-free.

Why SMACD?

The ability to effectively leverage the multi-view nature of a graph stems from the model that SMACD uses under the hood. The underlying CP model has well-studied uniqueness properties [141] which have implications about the quality of the decomposition, and hence the community assignments. In short, CP is unique under mild conditions, which essentially guarantees that the computed decomposition is the only combination of factors (thus, community assignments) which can reconstruct the data, and not any rotated version thereof. On the other hand, matrix-based approaches, such as [78], typically suffer from rotational ambiguity (this is easy to see since, for a bilinear model, $\mathbf{X} \approx \mathbf{A}\mathbf{B}^T = \mathbf{A}\mathbf{Q}\mathbf{Q}^{-1}\mathbf{B}^T = \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T$ for any invertible $\mathbf{Q}$) and fail to guarantee that the

computed community assignments are the best possible, and not any rotation thereof. Finally, coupling with the matrix containing partial community labels is a "soft" manner of imposing semi-supervision. Instead of making hard assignments of the nodes for which we have labels, SMACD is using the underlying structure of the $\mathbf{Y}$ label matrix in order to "guide" the low-rank structure discovered by the CP decomposition on the tensor $\underline{\mathbf{X}}$. Thus, in combination with CP's uniqueness, soft semi-supervision of $\mathbf{Y}$ guides the decomposition to a set of unique community assignments, as close as possible to the partially observed community assignments.

3.6 Conclusion

We introduce SMACD, a novel approach on semi-supervised multi-aspect community detection based on a novel coupled matrix-tensor model. We propose an automated parameter tuning algorithm, which effectively renders SMACD *parameter-free*. We extensively evaluate SMACD's effectiveness over the state-of-the-art, in a wide variety of real and synthetic datasets, demonstrating the merit of leveraging semi-supervision and higher-order edge information towards high quality overlapping and non-overlapping community detection.

Chapter based on material published in SDM 2018 [89] and HetroNam 2018 [90].

Chapter 4

Beyond Rank-1: Discovering Rich Community Structure in Multi-Aspect Graphs

"How are communities in real multi-aspect or multi-view graphs structured? How we can effectively and concisely summarize and explore those communities in a high-dimensional, multi-aspect graph without losing important information?"

State-of-the-art studies focused on patterns in single graphs, identifying structures in a single snapshot of a large network or in time evolving graphs and stitch them over time. However, to the best of our knowledge, there is no method that discovers and summarizes community structure from a multi-aspect graph, by *jointly* leveraging information from all aspects. State-of-the-art in multi-aspect/tensor community extraction is limited to discov-

ering clique structure in the extracted communities, or even worse, imposing clique structure where it does not exist.

In this chapter we bridge that gap by empowering tensor-based methods to extract rich community structure from multi-aspect graphs. In particular, we introduce cLL1, a novel constrained Block Term Tensor Decomposition, that is generally capable of extracting higher than rank-1 but still interpretable structure from a multi-aspect dataset. Subsequently, we propose RichCom, a community structure extraction and summarization algorithm that leverages cLL1 to identify rich community structure (e.g., cliques, stars, chains, etc) while leveraging higher-order correlations between the different aspects of the graph.

Our contributions are four-fold: (a) **Novel algorithm**: we develop cLL1, an efficient framework to extract rich and interpretable structure from general multi-aspect data; (b) **Graph summarization and exploration**: we provide RichCom, a summarization and encoding scheme to discover and explore structures of communities identified by cLL1; (c) **Multi-aspect graph generator**: we provide a simple and effective synthetic multi-aspect graph generator, and (d) **Real-world utility**: we present empirical results on small and large real datasets that demonstrate performance on par or superior to existing state-of-the-art. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, Ravdeep Pasricha, and Evangelos Papalexakis. "Beyond rank-1: Discovering rich community structure in multi-aspect graphs." In Proceedings of The Web Conference 2020, pp. 452-462. 2020.

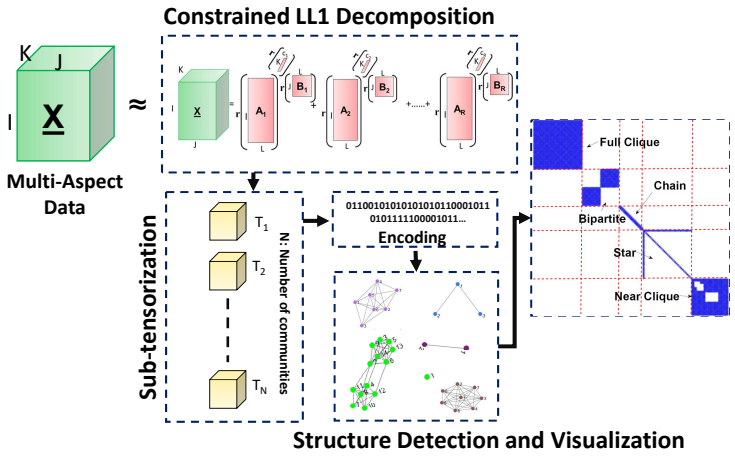


Figure 4.1: A toy example of RichCom: It decomposes the multi-aspect data and identifies non-overlapping as well as overlapping sets of nodes, that form sub-tensors and resultant structures like cliques, bi-bipartite, chains, stars etc. are encoded and visualized.

4.1 Introduction

Multi-aspect graphs emerge in various applications, as diverse as biomedical imaging [126], social networks [152], computer vision [264], recommender systems [150], and communication networks [153], and are generally shaped using high-order tensors [141]. A simple example of such multi-aspect graph could be a three-mode tensor, where each different aspect, represented by a frontal slice of the tensor, is the adjacency matrix of a social network under a different means of communication. Each such aspect of the data is an impression of the same underlying phenomenon e.g, the formation of friendship in social networks or evolution of communities over time. Even though there has been a significant focus on graph mining and community detection for single graphs [260, 154, 72], incorporating different aspects of the graph has only recently received attention [25, 194, 89, 80].

In fact, taking into account all aspects of a graph has been shown to lead to better results compared to "two-dimensional" counterparts [194, 89, 80]. The problem we address in this chapter is the following: Given a multi-aspect graph, say, an air traffic networks of airlines like European-ATN [138], how can we efficiently describe and summarize its *community structure*? The heart of this chapter is finding and visualizing the structures (e.g., clique, chain, star etc.) of communities in multi-aspect/layer graphs via tensor decomposition approach, in order to get a better insight of their properties.

To the best of our knowledge, the state-of-the-art [143] focuses on a single graph and provide vocabulary (e.g clique, star) of graph primitives using the Minimum Description Length (MDL) principle. The paper [224] is further extension of [143] for dynamic graphs and stitching the temporal patterns in the time-evolving scenario. These methods [224, 143], even though they are offering very valuable insights, but they are either focus on a single graph or find community structure in single graph and then track them over time.

We propose a tensor-based method. A simple but interpretable CANDECOMP/-PARAFAC tensor decomposition [36, 104, 21], aka CP (see Def. 2.2.1) method is well known and studied in the literature. The fundamental *road-block* in traditional exploratory tensor decomposition (CP [36, 104, 21] and Tucker [256]) analysis, is that the only form of latent structure it can discover is *rank*-1. Consider a time-evolving or a multi-aspect graph [194, 89], both of which can be expressed as a set of adjacency matrices, forming a third-order tensor $\underline{\mathbf{X}}$. These decompositions can only discover rank-1 structures in the graph, which translate into dense cliques; even if the real underlying structure in the graph is not a clique but, say, a star, CP is unable to extract it, and in the best case it will approximate

it as a near-clique, or in the worst case it will fail to identify it. In the signal processing literature there exists the Block Term Decomposition (BTD) [52, 53] which shows promise; a form of BTD is the $(L, L, 1)$ decomposition[57]: $\underline{\mathbf{X}} \approx \sum_{r=1}^{R} (\mathbf{A}_r \cdot \mathbf{B}_r^T) \circ \mathbf{c}_r$ where $\mathbf{A}_r$ and $\mathbf{B}_r$ have L columns, and which essentially extracts rank-L structures in the first two dimensions. In this chapter, we adapt and extend this less well-known tensor model, the Block Term Decomposition-$(L, L, 1)$ (see Def. 8), to that end. This model holds potential for "general" multi-aspect graph exploration, where structure is richer than *rank-1* but we still wish to have interpretable results. Note that chapter does not focus on the community detection in data and it is out of scope of work. Here, after block term tensor decomposition, each latent component is considered as community and we focus on detecting structures of those communities.

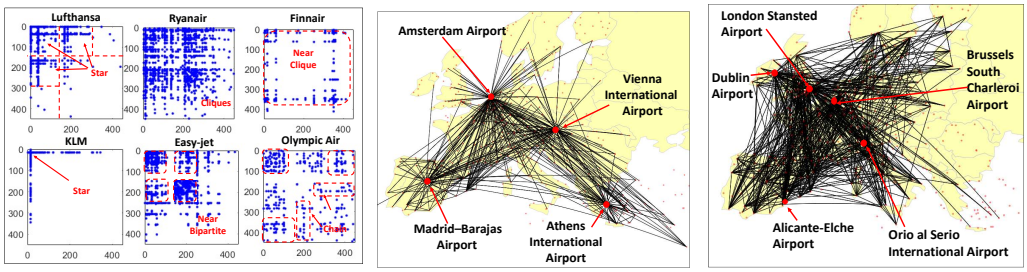


Figure 4.2: RichCom finds meaningful structures in EU-Airline dataset as (a) we show the adjacency matrix created from factors over multiple views showing RichCom able to find stable decompositions and detect various structures, (b) An example of ATN network of a major airline's (e.g. KLM) operating airport forming star structures and, (c) the network of a low-fare (low-cost) airline's (e.g. Ryanair) air traffic forming clique structure. In each aspect, the airports with the highest degree (hubs) are highlighted.

Motivation : The motivation behind RichCom is that exploring a high-dimensional, multi-aspect graph is impossible to do manually. Thus, extraction and visualization of the main communities within such a graph is an important tool towards enabling multi-aspect graph exploration and a handful of simple structures could be easily understood, and often meaningful. Figure 4.2 is an illustrating example of RichCom, where the most 'important' sub-tensor that provides *EU-Air Traffic Network*'s summary is semantically interesting. Here, importance is computed based on MDL approach explained in Section (4.4.2). Alleviating the limitations of existing approaches, this chapter presents cLL1 a novel factorization model using constrained Block Term Decomposition-rank $(L, L, 1)$ to account for the multi-aspect graph's structures. Using cLL1, factors are estimated via a novel algorithm based on the alternating optimization and alternating method of multipliers (AO-ADMM) and RichCom is used for encoding cost to discover community structures entries in the data and our contributions can be summarized as:

- **Novel Problem Formulation** : We formulate the exploration and discovery of rich structures in multi-aspect graphs using a novel tensor modeling as shown in Figure 4.1.
- **Novel and Effective Framework**: We introduce novel constrained LL1-tensor decomposition cLL1 and alternating optimization and alternating direction method of multipliers (AO-ADMM) is also developed that recovers the non-negative and sparse factors.
- **Real-World Utility** : Finally, the proposed decomposition approach enables discovery of community structures via RichCom on tensor by using the recovered factors.

We provide qualitative analysis of RichCom on synthetic and real, public multi-aspect datasets consisting up to thousands of edges. Experiments testify RichCom spots interesting structures like 'stars' and 'cliques' in the European Air Traffic Network (ATN) data (See Fig. 4.2).

Reproducibility: We make our Python and MATLAB implementation publicly available Link[1]. Furthermore, graph and tensor generator along with the small size dataset we use for evaluation are also available at the same link.

The rest of this chapter is organized as overview of related work (Sec 4.2), problem formulation (Sec 4.3), proposed method (Sec 4.4), qualitative analysis (Sec 4.5) on six real datasets. Finally, Sec. 4.6 summarizes some closing remarks.

4.2 Related work

In this section, we provide review of the work related to our algorithm. At large, in the literature this work can be categorized into three main categories as described below:

Multi-aspect Community Detection: Real data usual exhibit different cross-domain relations and can be represented as multi-aspect graphs. In [25] the authors introduce a graph theoretic based community detection algorithm over multi-aspect graphs and relationships between nodes represented by various types of edges. In [194, 89] the authors introduce a robust algorithm for community detection on multi-view graphs based on tensor decomposition which uses a regularized CP model with sparsity penalties. In addition to different graph views, "time" is also a multi-aspect feature of a graph. Finally,

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

most recently in [80] the authors propose a method for identifying and tracking dynamic communities in time-evolving networks. None of these works summarize in terms of local structures; our work focuses on interpretable community structures.

Static/Dynamic Graph Summarization: Graph based method like [250] and [271] uses structural equivalence and minimum DFS code, respectively, to simplify single graph representation to obtain a compressed smaller graph. VoG [143] uses minimum descriptive length to label sub-graphs in terms of a vocabulary including cliques, stars, chains and bipartite cores on static graphs. Further, TimeCrunch [224] extends [143] for dynamic graphs and it uses MDL to label and stitch the dynamic sub-graphs. Here, compression is not our aim. Our work proposes AO-ADMM based constrained Block Term-(L, L, 1) decomposition for multi-aspect graphs to label community structures and provides an effective and stable algorithm for discovering them.

Tensor Decomposition and Optimization: Besides well known CP and Tucker decomposition, the Block Term Decomposition aka BTD was presented in the 3-segment papers [52, 53, 57]. BTD provides a tensor decomposition in a sum of Tucker terms. The paper [41] use BTD in modeling process for better exploitation of the spatial dimension of fMRI images. In literature, well-suited optimization framework namely Alternating Method of Multipliers, that has shown promise in other, simpler tensor models [120, 235, 6] was introduced to speed up tensor decomposition process.

4.3 Problem Formulation

Consider a multi-aspect graph, where each frontal slice of a tensor is a snapshot of the adjacency matrix at a given time point or a different aspect of user relation (e.g., "calls", "text", etc). A typical application here is the extraction of communities in the graph and their evolution over time. Traditionally, one would take a CP decomposition of the tensor, where each rank-one component would map to each community; furthermore, the values on the a_r and b_r vectors (corresponding to the two dimensions of the nodes in our graph) would give the membership of each node to each community. This method has been shown to be very accurate in extracting community memberships [194, 33, 89], however, it is not able to identify the shape of the sub-graph that defines the community. As we argue in the introduction (Sec. 4.1) of this chapter, the reconstructed adjacency matrix of the $r-th$ community will be $a_r b_r^T$ which is a rank-1 block, corresponding to a clique. Therefore, CP imposes clique structure to all communities, which need not be true, and may result in false conclusions. On the other hand, a BTD-(L, L, 1) is able to extract higher rank-1 components in the first two modes, providing flexibility in extracting richer structure in the multi-aspect graph.

The problem that we solve is the following:

Given: a multi-aspect graph given as a higher-order tensor $\underline{\mathbf{X}}$ and $\mathcal{S}$ as structure vocabulary:

Extract: latent components of $\underline{\mathbf{X}}$ which allow for higher

than rank-1 in the first two modes,

Find: a set of possibly overlapping sub-tensors $\{\underline{\mathbf{T}}_1, \underline{\mathbf{T}}_2 \dots \underline{\mathbf{T}}_R\}$ with minimal encoded

length (average bits) i.e. $\mathcal{B}(\mathcal{S}_i) + \mathcal{B}(Err)$ to **briefly describe** the given multi-aspect graph communities structure in a efficient and scalable fashion.

4.4 Proposed Method: RICHCOM

Given $\underline{\mathbf{X}}$, this section proposes the constrained LL1 decomposition in order to factorize the multi-aspect graph or tensor into its constituent community-revealing factors and provide community structure's encoding formulation. We focus on a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ for our problem and its loss function formulation is given by:

$$\mathcal{LS}(\underline{\mathbf{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C}) = \underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\operatorname{argmin}} ||\underline{\mathbf{X}} - \sum_{r=1}^{R} (\mathbf{A}_r \cdot \mathbf{B}_r^T) \circ \mathbf{c}_r||_F^2 \tag{4.1}$$

The above Eq. (4.1) in its non-convex optimization form can be solved as:

$$\{\mathbf{A}, \mathbf{B}, \mathbf{C}\} \leftarrow \underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \mathcal{LS}(\underline{\mathbf{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C}) + r(\mathbf{A}) + r(\mathbf{B}) + r(\mathbf{C}) \tag{4.2}$$

where **r(.)** is a penalty or constrained function. Instead of solving (4.2) for the all three factors at once, we can use least square method by fixing all factor matrices but solve one each at a time. Thus the problem is converted into three coupled linear least-squares sub-problems. A very well-suited optimization framework, that has shown promise in other,

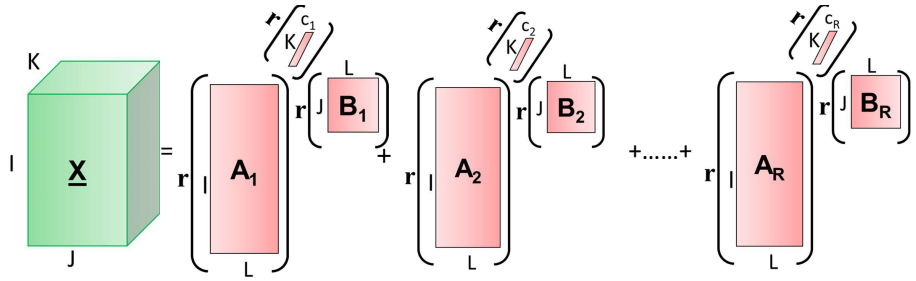


Figure 4.3: Proposed constrained -(L, L, 1) for a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$. Here, r(.) represents constraint or penalty function on each block.

simpler tensor models [120, 235, 6], is the Alternating Method of Multipliers [26], applied in an alternating optimization fashion. In the next subsection we derive and describe our optimization method in detail.

4.4.1 Solving the cLL1

In the proposed framework, each step consists of fixing two factors and minimizing the sub-problem with respect to the third factor. In this section, we provide solver for tackling the problem efficiently.

Factor A update

Consider first the update of factor $\mathbf{A} = [\mathbf{A}_1 \ \mathbf{A}_2 \dots \mathbf{A}_R] \in \mathbb{R}^{I \times LR}$ at iteration k, obtained after fixing $\mathbf{B} = \mathbf{B}^{(k-1)}$ and $\mathbf{C} = \mathbf{C}^{(k-1)}$ and solving the corresponding minimization. The arising sub-problem, after manipulation can be re-written as:

$$\mathbf{A}^{(k)} \leftarrow \underset{\mathbf{A}}{\operatorname{argmin}} [(\mathbf{B}^{(k-1)} \odot \mathbf{C}^{(k-1)})^{\dagger} \cdot \underline{\mathbf{X}}_{(1)}]^T \tag{4.3}$$

where $\underline{\mathbf{X}}_{(1)} = (\mathbf{B} \odot \mathbf{C})\mathbf{A}^T$ is a metricized reshaping of the tensor $\underline{\mathbf{X}}$ in mode-1. Also $\mathbf{B}^{(k-1)} \odot \mathbf{C}^{(k-1)} := [(\mathbf{B}_1^{(k-1)} \otimes \mathbf{c}_1^{(k-1)}) \ \ (\mathbf{B}_2^{(k-1)} \otimes \mathbf{c}_2^{(k-1)}) \ \ \dots \ \ (\mathbf{B}_R^{(k-1)} \otimes \mathbf{c}_R^{(k-1)})]$ is the partition-wise Kronecker product of $\mathbf{B}^{(k-1)}$ and $\mathbf{C}^{(k-1)}$, where $\mathbf{B}_r^{(k-1)}$ denotes partitioned matrix r of $\mathbf{B}^{(k-1)}$ and $\mathbf{c}_r^{(k-1)}$ denotes column r of $\mathbf{C}^{(k-1)}$, and $\otimes$ denotes the Kronecker product operator; see also Def. 13. The regularized version of Equ. 4.3 is given as:

$$\mathbf{A}^{(k)} \leftarrow \underset{\mathbf{A}_{reg}}{\operatorname{argmin}} [(\mathbf{B}^{(k-1)} \odot \mathbf{C}^{(k-1)})^{\dagger} \cdot \underline{\mathbf{X}}_{(1)}]^T + r(\mathbf{A}^{(k-1)}) \tag{4.4}$$

Constraints are implemented in such a way that when the constraints are violated then $\mathbf{r(.)}$ takes the value of infinity (∞), otherwise in normal scenario regularization (e.g. sparsity,

l1 etc.) uses finite values to penalize undesirable but reasonable solutions. Following the steps in [235], the primal, an auxiliary and a dual variables as $\mathbf{H} \in \mathbb{R}^{I \times LR}$, $\widetilde{\mathbf{H}} \in \mathbb{R}^{LR \times I}$ and $\mathbf{U} \in \mathbb{R}^{I \times LR}$, respectively are introduced to account for the augmented Lagrangian of Eq. (4.4). Each iteration optimizes factor $\mathbf{A}$ by means of *AO-ADMM* (given in algorithm 6) as:

$$\begin{aligned} \mathcal{L}_A^{(k)}(\mathbf{H}, \widetilde{\mathbf{H}}, \mathbf{U}) = \underset{\mathbf{H}, \widetilde{\mathbf{H}}}{\operatorname{argmin}} \, & ||(\mathbf{Y}_A^{(k)} \cdot \underline{\mathbf{X}}_{(1)})^T||_F^2 + Trace(\mathbf{H}\mathbf{H}^T) \\ & + r(\mathbf{H}) + (\rho/2)||\mathbf{H} - \widetilde{\mathbf{H}}^T + \mathbf{U}||_F^2 \\ \text{subject to} \quad & \mathbf{H} = \widetilde{\mathbf{H}} \end{aligned} \tag{4.5}$$

where $\mathbf{Y}_A^{(k)} := (\mathbf{B}^{(k-1)} \odot \mathbf{C}^{(k-1)})^\dagger$, and $r(\mathbf{H})$ is the regularizer (e.g. non-negativity, sparsity etc.) constraint on factor $\mathbf{A}$. The Lagrange multiplier ρ is set to minimum value between 10^{-3} and $(||\mathbf{Y}_A||_F^2/LR)$ to yield good performance. The AO-ADMM solver advances further by iteratively updating the variables $\mathbf{H}$, $\widetilde{\mathbf{H}}$,$\mathbf{U}$ until a convergence criterion is met, i.e. whether the prescribed error tolerance is met, i.e., $||\mathbf{A}_{curr}^{(k)} - \mathbf{A}_{prev}^{(k)}||_F / ||\mathbf{A}_{prev.}^{(k)}||_F \leq \epsilon$ or maximum number of iterations are reached.

Factor B update

Update of factor $\mathbf{B} = [\mathbf{B}_1 \quad \mathbf{B}_2 \ldots \mathbf{B}_R] \in \mathbb{R}^{J \times LR}$ can be similarly obtained by solving the sub-problem as:

$$\mathbf{B}^{(k)} \leftarrow \underset{\mathbf{B} \geq 0}{\operatorname{argmin}} [(\mathbf{C}^{(k-1)} \odot \mathbf{A}^{(k)})^\dagger \cdot \underline{\mathbf{X}}_{(2)}]^T + r(\mathbf{B}^{(k-1)}) \tag{4.6}$$

And its *AO-ADMM* solver as :

$$\begin{aligned}\mathcal{L}_B^{(k)}(\mathbf{H},\widetilde{\mathbf{H}},\mathbf{U}) = \underset{\mathbf{H},\widetilde{\mathbf{H}}}{\operatorname{argmin}} \, &||(\mathbf{Y}_B^{(k)} \cdot \underline{\mathbf{X}}_{(2)})^T||_F^2 + Trace(\mathbf{H}\mathbf{H}^T) \\ &+ r(\mathbf{H}) + (\rho/2)||\mathbf{H} - \widetilde{\mathbf{H}}^T + \mathbf{U}||_F^2 \\ \text{subject to} \quad &\mathbf{H} = \widetilde{\mathbf{H}}\end{aligned} \tag{4.7}$$

where $\mathbf{H} \in \mathbb{R}^{J \times LR}$, $\widetilde{\mathbf{H}} \in \mathbb{R}^{LR \times J}$ and $\mathbf{U} \in \mathbb{R}^{J \times LR}$ and $\mathbf{Y}_B^{(k)} := (\mathbf{C}^{(k-1)} \odot \mathbf{A}^{(k)})^\dagger := [(\mathbf{c}_1^{(k-1)} \otimes \mathbf{A}_1^{(k)}) \;\; (\mathbf{c}_2^{(k-1)} \otimes \mathbf{A}_2^{(k)}) \;\; \dots \;\; (\mathbf{c}_R^{(k-1)} \otimes \mathbf{A}_R^{(k)})]^\dagger$, providing a similar optimization problem as in 4.4. Eq. 4.7 formulates the update rule for solving (4.4) and similarly method using a general framework for cLL1 to update factor $\mathbf{B}$.

Proposition 1: If the sequence generated by AO-ADMM in Algorithm 6 is bounded, then the sequence $\{\mathbf{A}(k), \mathbf{B}(k), \mathbf{C}(k)\}$ and AO-ADMM converges to a stationary point of Equ. 4.2.

Proof: The convergence follows from [[120], Theorem 1; [207], Theorem 2].

Factor C update

Update of factor $\mathbf{C} = [\mathbf{c}_1 \;\; \mathbf{c}_2 \dots \mathbf{c}_R] \in \mathbb{R}^{K \times R}$ is obtained by fixing $\mathbf{A}$ and $\mathbf{B}$ at their most recent values, and solving it by :

$$\begin{aligned}\mathbf{C}^{(k)} \leftarrow \underset{\mathbf{C}_{reg}}{\operatorname{argmin}} \, &\left\{[(\mathbf{A}_1^{(k)} \odot_c \mathbf{B}_1^{(k)})1_{L_1} \dots (\mathbf{A}_R^{(k)} \odot_c \mathbf{B}_R^{(k)})1_{L_R}]^\dagger \cdot \underline{\mathbf{X}}_{(3)}\right\}^T \\ &+ r(\mathbf{C}^{(k-1)})\end{aligned} \tag{4.8}$$

where $\underline{\mathbf{X}}_{(3)} := [(\mathbf{A}_1 \odot_c \mathbf{B}_1)1_{L_1} \quad (\mathbf{A}_2 \odot_c \mathbf{B}_2)1_{L_2} \ldots (\mathbf{A}_R \odot_c \mathbf{B}_R)1_{L_R}] \cdot C^T$. Utilizing an *AO-ADMM* approach, the augmented Lagrangian is represented as:

$$\begin{aligned} \mathcal{L}_C^{(k)}(\mathbf{H}, \widetilde{\mathbf{H}}, \mathbf{U}) = \underset{\mathbf{H}, \widetilde{\mathbf{H}}}{\operatorname{argmin}} & ||(\mathbf{Y}_C^{(k)} \cdot \underline{\mathbf{X}}_{(3)})^T||_F^2 + Trace(\mathbf{H}\mathbf{H}^T) \\ & + r(\mathbf{H}) + (\rho/2)||\mathbf{H} - \widetilde{\mathbf{H}}^T + \mathbf{U}||_F^2 \\ \text{subject to} \quad & \mathbf{H} = \widetilde{\mathbf{H}} \end{aligned} \tag{4.9}$$

where $\mathbf{H} \in \mathbb{R}^{K \times R}$, $\widetilde{\mathbf{H}} \in \mathbb{R}^{R \times K}$ and $\mathbf{U} \in \mathbb{R}^{K \times R}$ and $\mathbf{Y}_C^{(k)} := (\mathbf{A}^{(k)} \odot \mathbf{B}^{(k)})^\dagger := [(\mathbf{A}_1^{(k)} \otimes \mathbf{B}_1^{(k)})1_{L_1} \quad (\mathbf{A}_2^{(k)} \otimes \mathbf{B}_2^{(k)})1_{L_2} \quad \ldots \quad (\mathbf{A}_R^{(k)} \otimes \mathbf{B}_R^{(k)})1_{L_R}]^\dagger$. The normalization ($c_r \leftarrow c_r/||c_r||$) is applied to each column of obtained factor matrix $\mathbf{C}^{(k)}$ to avoid any underflow or overflow.

Our proposed method readily extends to higher-order tensors, since the underlying tensor model BTD - rank (L,L,1) mathematically extends naturally as such. In this study, we focus on three-mode tensor only for simplicity.

4.4.2 Community Structure Encoding

Once the proposed solver returns the solution of (4.2), next step is to find communities by extracting weakly connected components [123] from first two dimensions of tensor i.e obtained matrices $D_r = (\mathbf{A}_r \cdot \mathbf{B}_r^T)$ from factors $\mathbf{A}$ and $\mathbf{B}$. Once the communities are formed, we extract the sub-tensors $\underline{\mathbf{T}}_i \in \mathbb{R}^{b \times b \times b}$ from original multi-aspect graph or tensor using nodes falls under each community and we find structure $\mathcal{S}$ such as *FC*: Full Clique; *NC*: Near Clique; *ST*: Star; *CH*: Chain; *CB*: Complete Bipartite; *NB*: Near Bipartite, that best describes its characteristics using below encoding cost (average bits) $\mathcal{B}(\underline{\mathbf{T}})$ with help of J. Rissanen modeling $B_{\mathbb{N}}$ [212]. We observed that these structures appear very frequent, in most of real world data, (e.g in company communication networks, there is possibility of one

way interaction that results in star (hub and spoke) structure and in friendship network, most of friends are connected to each other forming cliques or near cliques etc).

Cliques and Near Cliques

These are the simplest structures, as all the nodes have a similar role. For a full and near cliques, we compute a set of fully-connected or almost fully connected nodes. Consider sub-tensor $\underline{\mathbf{T}} \in \mathbb{R}^{|b|\times|b|\times|b|}$, where $|b|$ is number of nodes fall in the community and $|nz|$ as number of non-zero elements in $\underline{\mathbf{T}}$. Also, consider $|n|$ represents number of nodes having at least two non-zero element in $\underline{\mathbf{T}}$. Now, if $\{|nz| = |b| * |b| * |b|\}$ then sub-tensor is considered as full clique, otherwise between range $\{0.75 * |b| * |b| * |b| \leq |nz| \leq |b| * |b| * |b|\}$, sub-tensor is refereed as near clique and we formalize the average bits to encode the structure as.

$$\begin{aligned}\mathcal{B}_{\underline{\mathbf{T}}}^{(FC,NC)} = B_{\mathbb{N}}(\underline{\mathbf{T}}) + \log_2({}^{|b|}C_{|n|}) - |nz|\log(|nz|) \\ - |z|\log(|z|) + 3b^3\log(b)\end{aligned} \tag{4.10}$$

where $|z|$ are number of elements not present in $\underline{\mathbf{T}}$. The intuition is that the more sparse a near-clique is, encoding will be cheaper.

Star

A star is very special case of the structure because of its highly sparse nature and it consists of a single node we call it *hub* connected to at least other two nodes. Consider sub-tensor $\underline{\mathbf{T}} \in \mathbb{R}^{|b|\times|b|\times|b|}$ and we formulate the average bits to encode the structure as :

$$\mathcal{B}_{\underline{\mathbf{T}}}^{(ST)} = B_{\mathbb{N}}(\underline{\mathbf{T}}) + \log_2({}^{|b|-1}C_{|n-1|}) + n\log(n) \tag{4.11}$$

where $|n|$ represents number of nodes having at least one non-zero element in $\underline{\mathbf{T}}$ and $|b|$ is number of nodes fall in the community.

Chain

In the chain structure, each node is linked to only one of its adjacent next node and forming a super-diagonal sub-tensor that means it has non-zero elements below, above and at the diagonal position only. Consider sub-tensor $\underline{\mathbf{T}} \in \mathbb{R}^{|b|\times|b|\times|b|}$ and we formulate the average bits to encode the structure as :

$$\mathcal{B}_{\underline{\mathbf{T}}}^{(CH)} = B_{\mathbb{N}}(\underline{\mathbf{T}}) + |n| \log_2(|b|) \tag{4.12}$$

where $|n|$ represents number of nodes having at least one non-zero diagonal element in $\underline{\mathbf{T}}$ and $|b|$ is number of nodes fall in the community.

Bipartite and Near Bipartite

A complete bipartite and near-bipartite structure is a sub-tensor whose nodes can be divided into two subsets C_1 and C_2 such that no edge has both endpoints in the same subset. We formulate the average bits to encode the structure as :

$$\begin{aligned}\mathcal{B}_{\underline{\mathbf{T}}}^{(CB,NB)} = {} & B_{\mathbb{N}}(\underline{\mathbf{C_1}}) + B_{\mathbb{N}}(\underline{\mathbf{C_2}}) + \log_2({}^{|b|}C_{|n_2|}) \\ & \log_2({}^{|b|}C_{|n_1|}) + -|nz| \log(|nz|) - |z| \log(|z|) \\ & + 3b^3 \log(b)\end{aligned} \tag{4.13}$$

where $|nz|$ as number of non-zero elements in $\underline{\mathbf{T}}$, $|z|$ are number of elements not present in $\underline{\mathbf{T}}$, $|n_1|$ and $|n_2|$ represents number of nodes having at least two non-zero element in $\underline{\mathbf{C_1}}$ and $\underline{\mathbf{C_2}}$, respectively.

4.4.3 Encoding the Error

We encode the errors made by structure $\mathcal{S}$ with regard to $\underline{\mathbf{X}}$ and store the information in two separate encoding matrix $\mathbf{E}^+$ and $\mathbf{E}^-$. The former refers to the area of $\underline{\mathbf{X}}$ that structure $\mathcal{S}$ include and later refer as the area of $\underline{\mathbf{X}}$ that $\mathcal{S}$ does not include. We formulate the average bits to encode the error as:

$$\begin{aligned}\mathcal{B}(\mathbf{E}^+) &= \log_2(|\mathbf{E}^+|) - ||\mathbf{E}^+|| \log(|nz|) - ||\mathbf{E}^+||' \log(|z|) + |\mathbf{E}| \log(|b|) \\ \mathcal{B}(\mathbf{E}^-) &= \log_2(|\mathbf{E}^-|) - ||\mathbf{E}^-|| \log(|nz|) - ||\mathbf{E}^-||' \log(|z|) + ||\mathbf{E}|| \log(|b|)\end{aligned} \tag{4.14}$$

where $\mathbf{E} = \mathbf{E}^+ + \mathbf{E}^-$. We first encode the number of $1s$ in $\mathbf{E}^+$ and $\mathbf{E}^-$, then followed by sending the actual $1s$ and $0s$ to its optimal prefix codes.

4.4.4 Visualization of Community Structure

Community structure visualization is a powerful tool to convey the content of a community and can highlight patterns, and show connections among nodes. We developed tool ($link$[1]) in MATLAB to visualize each community structure. Figure 4.4 is visualization of a few structures discovered by RichCom that provide summarization with the minimum encoding cost in American college football dataset.

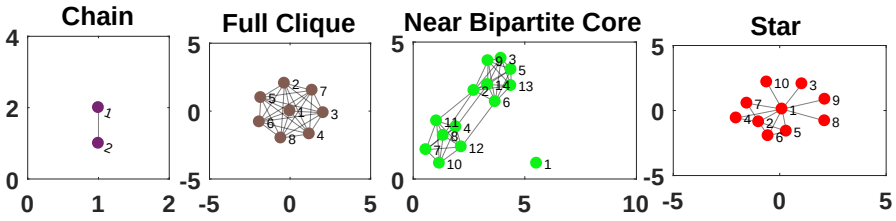


Figure 4.4: Visualization: a few community structures of Football[217] dataset.

Finally, by putting everything together, we obtain the general version of our RichCom and ADMM solver,as presented in Algorithm 6 and 7, respectively.

Algorithm 6: RichCom: Discovering Rich Community Structure

Data: $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$, $L \in \mathbb{R}^{R}$, Max iterations I_{max}.

Result: Factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, Structures $\mathcal{S}$.

$(\mathbf{A}, \mathbf{B}, \mathbf{C}) \leftarrow \text{cLL1}(\underline{\mathbf{X}}, L, I_{max})$

$\mathbf{D}_r \leftarrow (\mathbf{A}_r \cdot \mathbf{B}_r^T) \quad \forall r \in R$

$\{Y_{nodes}, Y_{comm}\} \leftarrow communityDetection(\mathbf{D})$

for *i = 1 : total communities* **do**

  $m \leftarrow Y_{nodes}(find(Y_{comm} == i))$

  $\underline{\mathbf{T}}_i \leftarrow \underline{\mathbf{X}}(m, m, m) \quad \mathcal{S}_i \leftarrow encode(\underline{\mathbf{T}}_i)$ ▷ **using section 4.4.2**

end

return *$(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathcal{S})$*

Function ***cLL1*** *$(\underline{\mathbf{X}}, L, I_{max})$* **is**

  Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly; $s \leftarrow sum(L)$; $R \leftarrow length(L)$

  while *$k < I_{max}$ or not-convergence* **do**

    $\mathbf{G} \leftarrow AA^T$; $\mathbf{Y}_A^{(k)} \leftarrow (\mathbf{B}^{(k-1)} \odot \mathbf{C}^{(k-1)})^\dagger$; $\mathbf{F} \leftarrow \left(\mathbf{Y}_A^{(k)} \cdot \underline{\mathbf{X}}_{(1)}\right)^T$;

      $\rho = min(10^{-3}, (||\mathbf{Y}_A^{(k)}||_F^2/s)$

    $\mathbf{A}^{(k)}, \hat{\mathbf{A}}^{(k)} \leftarrow \mathbf{ADMM}(\mathbf{A}^{(k-1)}, \hat{\mathbf{A}}^{(k-1)}, \mathbf{F}, \mathbf{G}, \rho)$

    $\mathbf{G} \leftarrow BB^T$; $\mathbf{Y}_B^{(k)} \leftarrow (\mathbf{c}^{(k-1)} \odot \mathbf{A}^{(k)})^\dagger$; $\mathbf{F} \leftarrow \left(\mathbf{Y}_B^{(k)} \cdot \underline{\mathbf{X}}_{(2)}\right)^T$;

      $\rho = min(10^{-3}, (||\mathbf{Y}_B^{(k)}||_F^2/s)$

    $\mathbf{B}^{(k)}, \hat{\mathbf{B}}^{(k)} \leftarrow \mathbf{ADMM}(\mathbf{B}^{(k-1)}, \hat{\mathbf{B}}^{(k-1)}, \mathbf{F}, \mathbf{G}, \rho)$

    $\mathbf{G} \leftarrow CC^T$; $\mathbf{Y}_C^{(k)} \leftarrow (\mathbf{A}^{(k)} \odot \mathbf{B}^{(k)})^\dagger = [(\mathbf{A}_1^{(k)} \otimes \mathbf{B}_1^{(k)})1_{L_1}, \ldots, (\mathbf{A}_R^{(k)} \otimes \mathbf{B}_R^{(k)})1_{L_R}]^\dagger$;

      $\mathbf{F} \leftarrow \mathbf{Y}_C^{(k)} \cdot \underline{\mathbf{X}}_{(3)}\big)^T$; $\rho = min(10^{-3}, (||\mathbf{Y}_C^{(k)}||_F^2/s)$

    $\mathbf{C}^{(k)}, \hat{\mathbf{C}}^{(k)} \leftarrow \mathbf{ADMM}(\mathbf{C}^{(k-1)}, \hat{\mathbf{C}}^{(k-1)}, \mathbf{F}, \mathbf{G}, \rho)$

  end

end

4.5 Experiments

We evaluate the quality, scalability, and real-world utility of RichCom on both real and synthetic datasets. We used MatlabR2016b for our implementations, along with functionalities for tensors from the Tensor-Toolbox [21] and Tensorlab [258].

4.5.1 Experimental Setup

We provide the datasets used for evaluation in Table 4.1.

Algorithm 7: ADMM solver for Equ. (4.2).

Data: Residual matrices $\mathbf{R}_H$, $\mathbf{R}_U$, $\mathbf{R}_F$, $\mathbf{R}_G$, and ρ

Result: $\mathbf{R}_H$, $\mathbf{R}_U$

$\mathbf{L} \leftarrow$ Lower Cholesky decomposition$(\mathbf{R}_G + \rho\mathbf{I})$

while *iter* $< I_{ADMM}$ *or* ($r < \epsilon$ *and* $s < \epsilon$) **do**

  $\widetilde{\mathbf{R}}_H \leftarrow (\mathbf{L}^T)^{-1}\mathbf{L}^{-1}(\mathbf{R}_F + \rho(\mathbf{R}_H + \mathbf{R}_U)$

  $\mathbf{R}_H^0 \leftarrow \mathbf{R}_H$

  $\mathbf{R}_H \leftarrow \text{argmin}_{\mathbf{R}_H}\, r(\mathbf{R}_H) + Tr(\mathbf{R}_G) + \frac{\rho}{2}||\mathbf{R}_H - \widetilde{\mathbf{R}}_H^T + \mathbf{R}_U||$

  $\mathbf{R}_U \leftarrow \mathbf{R}_U + \mathbf{R}_H - \widetilde{\mathbf{R}}_H$

  $r \leftarrow ||\mathbf{R}_H - \widetilde{\mathbf{R}}_H^T||_F^2/||\mathbf{R}_H||_F$

  $s \leftarrow ||\mathbf{R}_H - \widetilde{\mathbf{R}}_H^0||_F^2/||\mathbf{R}_U||_F$

end

return $(\mathbf{R}_H, \mathbf{R}_U)$

Synthetic Data Description

In order to fully control and evaluate the community structures in our experiments, we generate synthetic graphs and converted them into multi-aspect graphs or tensor with different cluster density. The code for both generators are available at $link$[1]. Consider graph or network $G = (\mathcal{V},\mathcal{E})$, represented by node or vertex ($\mathcal{V}$) and relation between entities are defined by weighted or unweighted edges $(\mathcal{E}, \mathcal{W}_{ij})$ s.t. $(\mathcal{V}, \mathcal{E}, \mathcal{W}) \in \mathbb{R}^{N\times N}$. Let $\mathcal{W}_{ij}$ denotes the edge weight if $(i,\ j) \in \mathcal{E}$, otherwise $\mathcal{W}_{ij} = 0$. Consider $\underline{\mathbf{X}}_n = (\mathcal{V}, \mathcal{E}^{(n)}, \mathcal{W}^{(n)}) \subset G$ denotes the structure constructed as the sub-tensor by subset of nodes $N \in \mathcal{V}$ and third mode as its single and two-hop neighbors. The tensor $\underline{\mathbf{X}}$ captures dependencies between structures in graph G. In a network, the proposed multi-aspect representation is indeed capable of preserving the inherent "structure" among node adjacency along the 3^{rd} mode. We add Gaussian(μ, σ) white noise where the interactions in $\underline{\mathbf{X}}$ are uniformly distributed. This makes recovering the original communities structure increasingly challenging.

Real Data

Table 4.1 provides a brief description of the real-world datasets from different domains that we use in our experiments.The Football[217] dataset is network of American football games of Fall 2000; European Air Traffic Network(ATN) [138] multi-layer network composed of 37 different layers and each one corresponding to a different airline operating in Europe; Wikipedia[151] consists of users participating in the elections over 7 days and its bipartite in nature; EU-Core[276] consists of e-mail data from a large European research institution over 526 days; Autonomous Systems (AS)-level interconnection[190] consists of

multi-aspect of peering information inferred from (a) Oregon route-views, (b) RIPE RIS BGP, (c) Looking glass data, and (d) Routing registry, all combined and Enron[206] consists of a million emails communication over 899 unique dates.

The baseline method

We compare exploration for methods: (a) VoG[143]: finds meaningful patterns in single-layer graphs. We find each structure of $\underline{\mathbf{X}}$ iteratively with it and remove any duplicate structure found for fair analysis and (b) TimeCrunch[224]: find coherent, temporal patterns in dynamic graphs, and evaluate RichCom: our proposed method to discover rich community structures in multi-aspect graphs. It is also *noted* that RichCom does not necessarily compete against [143] and [224], it is an alternative way of achieving the same goal as those method while incorporating joint information from all aspects. Furthermore, the proposed method has direct implications in advancing our ability to analyze general multi-aspect data, where the underlying latent structure is not necessarily rank-1.

Note: Traditional community detection approaches rely on kernel functions, summary graph statistics (e.g., degrees or clustering coefficients) or designed features to extract structural data from graphs for measuring local structures (cliques and lines). However, these approaches are limited because these hand-engineered features are inflexible and designing these features can be time-consuming and expensive. In this work, we only focus on directly related baselines approaches i.e. VoG[143] and TimeCrunch[224]. Thus, even though the extracted sub-structures can be used to identify members of each community, the scope of this work is to identify the structures themselves, and thus we do not compare against community detection methods since the objectives are different.

Estimating Rank R and L

In this work, we deal with two different kinds of Rank a) the rank 'R' of a tensor $\underline{\mathbf{X}}$ and b) each 'r' block having rank - (L, L, 1). Finding the rank R of a tensor $\underline{\mathbf{X}}$ itself is a extremely hard problem [111] and out of the scope of this chapter. But we refer the interested reader to previous heuristics studies which try to estimate a low-rank estimation [176, 193] for an overview. In our case, it also requires further research on the relationship between rank R and rank L of each block and it is beyond the scope of this chapter. For our work, we set rank R of tensor $\underline{\mathbf{X}}$ as 5 and vary the rank L of block between $10 - 30$, experimental analysis is provided in sub-section 4.5.2.

4.5.2 Experimental Results

Qualitative Analysis

In this section, we provide quantitative evaluation and analysis of our method for structure discovery on synthetic and real-world datasets. Table 4.1 reports the resulting discovered structures.

Synthetic: A network of 50 cliques, 30 stars, 20 chains and 20 bipartite structures is generated and converted to tensor using one-hop neighbor relation with added noise. Discovered structures are presented in Tbl. 4.1 and quality in terms of finding correct structures is given in Tbl. 4.2. RichCom and VoG able to detect almost all of the structures (e.g clique, star and bipartite). VoG successfully detected most of the chains, both RichCom and TimeCrunch not able to detect all chain structures because of added noise to

Dataset	Statistics		RichCom						VoG[143]						TimeCrunch [224]					
	Nodes	#nz	*FC*	*NC*	*ST*	*CH*	*CB*	*NB*	*FC*	*NC*	*ST*	*CH*	CB	*NB*	*FC*	*NC*	*ST*	*CH*	*CB*	*NB*
Synthetic	$5k$	$0.2m$	54	–	42	7	52	5	58	–	30	13	–	–	29	–	22	–	–	–
Football	115	$8k$	11	10	35	1	–	32	8	–	2	–	6	4	35	–	66	–	–	1
EuroATN	450	$10k$	65	12	217	–	24	48	414	–	–	–	–	–	5	–	206	6	11	–
EU-Core	$1k$	$221k$	329	136	1829	360	31	530	731	–	1471	45	41	218	12	–	1996	499	36	192
Wikipedia	$7k$	$0.4m$	55	71	1373	11	30	97	1515	–	536	13	1	53	108	–	939	77	1	115
AS-level	$13k$	$0.7m$	4	26	1246	32	6	237	2	–	626	26	3	170	2442	–	–	3	3	130
Enron	$32k$	$1.9m$	11	57	1800	48	3	127	9	–	1507	30	2	119	1122	–	126	146	–	107

Table 4.1: Summarization of community (having > 1 node) structures found in datasets. The most frequent structures are the 'star' and 'near bipartite'. For each dataset, we provide the frequency of each structure type: *'FC'* for full cliques, *'NC'* for near-cliques, *1ST'* for star, *'CH'* for chains, *'CB'* for complete-bipartite, and *'NB'* for near bipartite.

multi-aspect format of graph. However, for RichCom, we don't observe a significant drop in discovery of any structure after discarding the noise.

EU-Core[276]: This is a temporal higher-order network dataset of emails between members of the research institution during October 2003 to May 2005 (18 months) and messages can be sent to multiple recipients of 42 departments. Agreeing with hunch, EU-Core consists of a large number of clique and near-clique structures corresponding to many instances of discussion within department. However, we also find numerous re-occurred stars (see Tbl. 4.1) which indicate email communication between different departments. Interestingly, figure 4.5 shows researchers for 4 months (corresponding to Oct, 2004 - Jan, 2005) forming continuous near clique structure, consisting of 85 researchers who communicated

each month belonging to same department. Suddenly, some members disappeared during Dec'04 and again in Jan'05 communication resumed back normally. We suspect that this may indicate the days corresponding to Festival of Lights, Christmas celebration week and New Year's Eve holiday time.

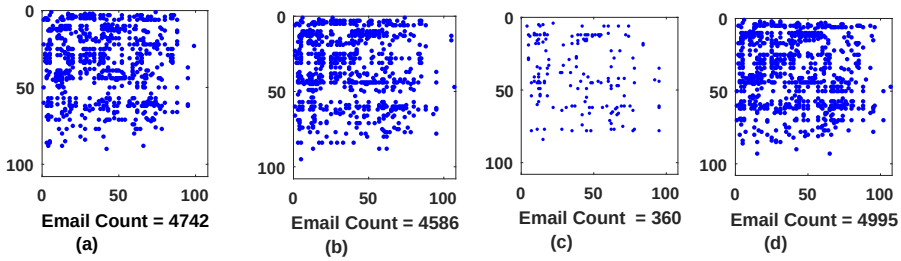


Figure 4.5: RichCom finds 107 research members in EU-core forming a continuous near clique ($\approx$ 43% density) over the observed last 4 months (a-d). The member's interaction drop (see (c)) indicates the festival month (e.g December).

Football[217]: Figure 4.6 provide visualization of Top-10 community structures discovered by RichCom and we plot these nodes using original football graph and mapped them to ground truth communities provided in literature. Football dataset is characterized by multiple cliques and near (cliques) structures. Interestingly, we found 10 conferences forming near cliques (in literature, total 12 conferences are given as ground truth) and few of the conferences teams had games with other conferences groups that result in formation of near bipartite and star relation.

Figure 4.6 provide visualization for the structures found by RichCom and we plot these nodes using original football graph and mapped them to ground truth communities provided in literature.

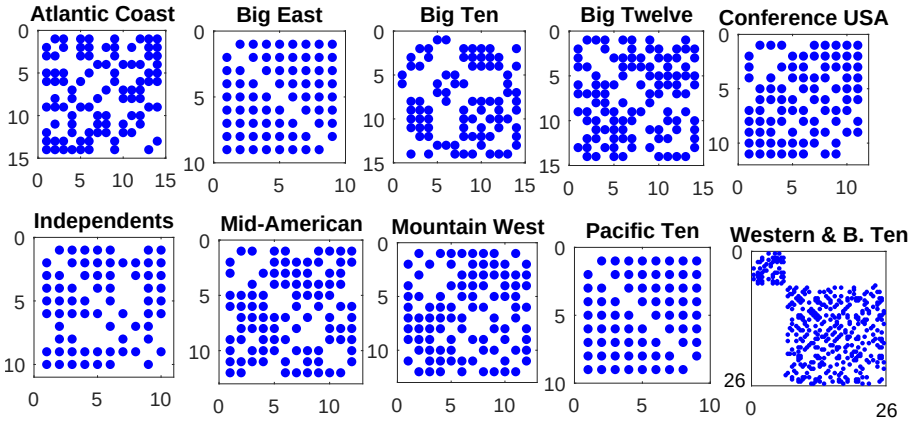


Figure 4.6: Top 10 structures of football teams found by RichCom.

Wikipedia[151]: It is a bipartite graph between users (e.g voter, admins and nominators) participating in the elections. As such, it is characterized by multiple stars and (near) bipartite structures. Many of the voters cast vote to the single user (nominee), as indicated by the presence of 1389 stars and provide the vote as support/neutral/oppose for particular nominee on election week. Interestingly, it is observed that more than half (65%) of these star discovered on the very first day of election (on Sept 14, 2004), indicating the strong support for their favorable nominee. Also, it is observed that about more than half of the votes casted by existing admins and they form near bipartite relation with ordinary Wikipedia users.

Autonomous Systems (AS)[190] : The AS-level dataset is largely comprised of stars and few near clique and bipartite structures. We discover large proportion of stars which occur only at Oregon route-view instance. Further analyzing these results in Fig 4.7, we find that 985 of the 1246 stars (73%) are found on first instance on Oregon route-view and rest were observed in Looking glass and Routing registry instance. Interestingly, for 2

Method	Precision							
	FC	ST	CH	CB	FC	ST	CH	CB
	With Noise				Without Noise			
RichCom	0.93	0.71	0.35	0.71	0.98	0.85	0.75	0.79
VoG[143]	–	–	–	–	0.86	1	0.65	0.0
TimeCrunch [224]	0.58	0.73	0.0	0.0	0.74	0.73	0.0	0.23

Table 4.2: Result based on structures found correctly in synthetic datasets (higher is better).

consecutive weeks, we observed set of routers form (near) clique structure, but later turned into (near) bipartite form indicate operational routers tables changes over time. When a connection between two observed routers on an earlier snapshot disappears from later snapshots, it could be caused either by actual termination or simply by a change in the route server's set of peer routers.

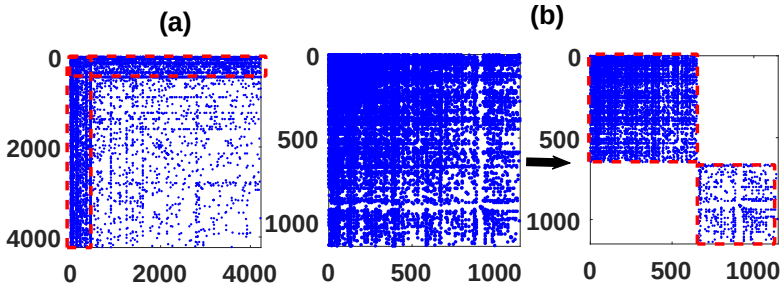


Figure 4.7: AS-level: Adjacency matrix of the (a) top star (4234 nodes) and (b) near bipartite (624 × 522 nodes) structure found by RichCom, corresponding to route-view of "traffic flows".

The RichCom summary for both **Enron**[206] and **European ATN**[138] datasets is very interesting and provided in **Sec 4.5.3**.

Scalability

We also evaluate the scalability of the method. We present the run-time (Fig. 4.8) of RichCom and baseline method TimeCrunch [224] with respect to the number of non-zero elements in the input tensor. For this purpose, we use sub-tensors form of Enron dataset consists of a millions of emails and method is flexible enough to deal over all the layers. Also, we evaluate proposed method on synthetic data with increasing two modes (I and J) of tensor with third mode (K) equivalent to 2% and 20% of I. Fig. 4.8 shows near linear run-time for million edges. For our experiment we used Intel(R) Xeon(R), CPU E5-2680 v3 @ 2.50GHz machine with 48 CPU cores and 378GB RAM. It is worth noting that since it is a AO-ADMM optimization framework, it is possible to parallelize the implementations, which can enable its feasible adoption for analysis of even larger multi-aspect or time evolving tensors.

Parameter Selection L and R

We use synthetic dataset with 20 cliques and 20 stars consisting of 50 nodes in each structure. To evaluate the impact of R, we fixed rank of each block i.e. L. We can see that with higher values of the R, number of cliques and star structure discovered is improved as shown in Figure 4.9 (a). Also, it is observed that after $R \geq 5$, it become

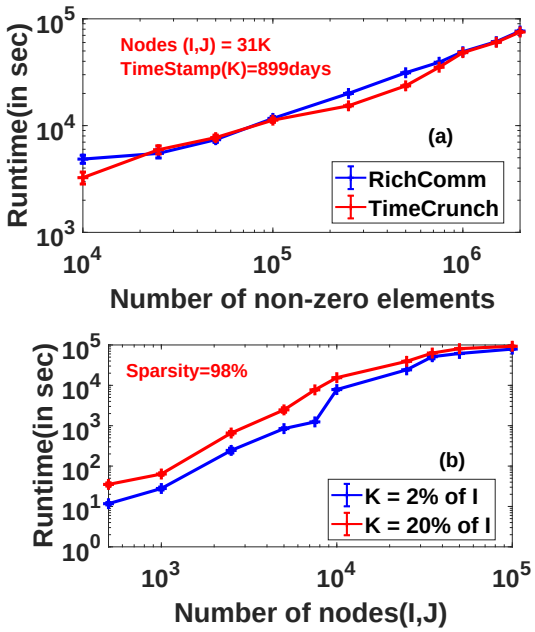


Figure 4.8: RICHCOM scales well on (a) induced aspects on third mode of Enron[155] dataset, up to 2M non-zero elements in size, and (b) on synthetic data varying two modes of tensor (I, J), up to 20M non-zero elements in size.

saturated. Similarly, we fixed rank $R = 5$ and vary L. We found that for $L \geq 10$ all structures discovery become stable.

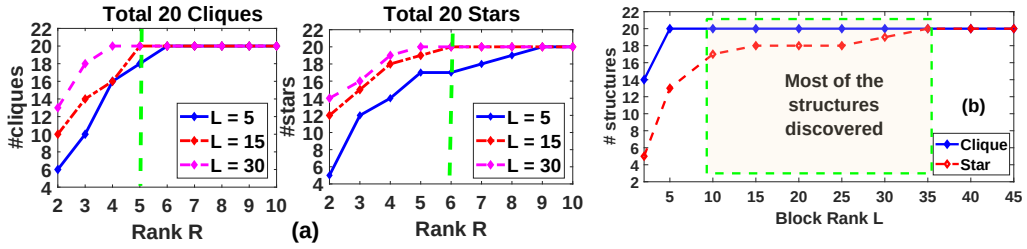


Figure 4.9: RICHCOM performance on synthetic dataset for (a) varying R and constant L, and (b) vary L and constant $R = 5$.

Convergence of RichCom

Here we demonstrate the convergence of Algorithm 6 for cLL1 on three real datasets i.e **Football**[217], **European ATN**[138] and **EU-Core**[276] network that we use for evaluation. The stabilization of fitness is observed after iteration $33, 48$ and 45 for Football, EU-Core and EU-ATN, respectively. Figure 4.10 summarizes the convergence of the algorithm, showing the approximated fitness as a function of the number of iterations. It is clear that the algorithm converges to a very good approximation within $40-50$ iterations. As our method is developed based on accelerated AO-ADMM approach [Razaviyayn et al.

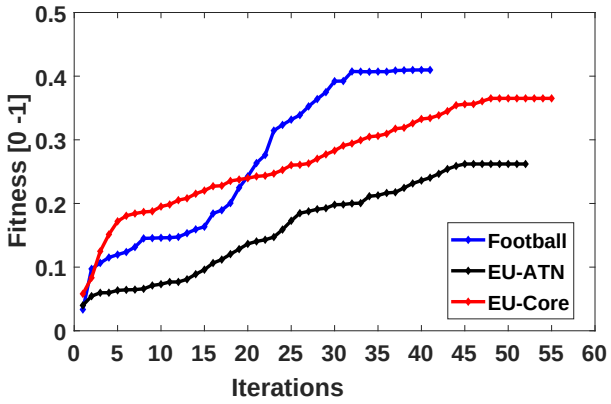


Figure 4.10: Fitness vs. number of iterations. For each dataset, computation cost was average 47 sec/iteration.

[207], Huang et al. 2016 [120] and Smith et al. 2017 [235]], it follows the same convergence rules and the proof follows from that work.

4.5.3 RichCom at Work

Beyond our qualitative analysis of the structure discovery in the six real dataset in Tbl. 4.1, we also consider a sample community structure analysis from the two datasets in Fig. (4.2 , 4.11), and present our findings in this section.

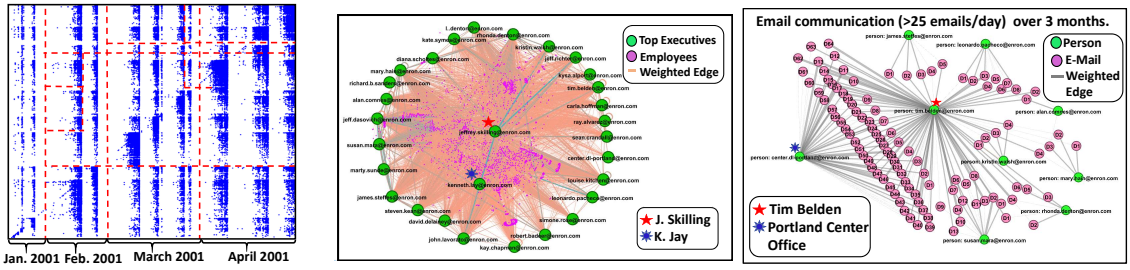


Figure 4.11: (a) Formation of star (include top executives only) structures over period of Jan to April 2001 (b) the most informative 'star' structure describing interactions between CEO and top executives during accounting fraud of Enron. (b) the second most informative star structure - email communication between Tim Belden and Enron's World Trade center office.

Case Study 1: The European air traffic network can be represented as a graph composed of $K = 37$ different layers or aspects each representing a airline (e.g Lufthansa, KLM etc). Each layer k has the same number of nodes, $|I| = 450$, as all European airports (e.g London Heathrow, Zurichi Kloten Airport etc) are represented in each layer. RichCom extracted two structures i.e. 'star' (Fig 4.2(b)) and 'cliques' (Fig 4.2(c)) frequently from this dataset, layers comprising in particular, major national airlines (e.g Lufthansa→ 1, Finnair→ 4 and KLM→ 9), low-cost fares (e.g Ryanair → 2, Easyjets→ 3), regional (e.g Olympic Air → 37, Aegean Airlines → 30) or cargo airlines like Fed-Ex. Figure 4.2(a)

show, instead, the single-layer ATN corresponding to a given major,regional and low-cost airlines reconstructed from decomposed factors using cLL1. These types of airlines have developed based on different commercial and structural constraints. As an example, it is well known that national airlines are designed as star network and ensuring the **hub** and **spoke** structure, to provide an almost full coverage of the airports belonging to a given country and maximize efficiency in terms of national or governmental transportation interests. On other hand low-cost airlines try to avoid this unify structures and, to be more aggressive, generally cover more than one country simultaneously, results in formation of near clique structures. One of the other reasons is that low-cost airlines stay away from busy and expensive hubs. They manage to operate from smaller airports through which ground times and delays are reduced that lead to cost reduction. The result of this study show that RichCom successfully exploit the multi aspect nature of data to discover the various useful structures.

Case Study 2: We use four years (1999-2002) of Enron email communications. In each view, the nodes represent email addresses and edges depict sent/received/cc relations. This network contains a total of $\approx 32k$ unique email addresses used only for the communication inside organization; and we analyze the data over 899 days including weekends also. The RichCom captures structures formed and changed during the major events in the company's history, such as revenue losses, CEO changes, etc. The interesting case is the constant increase in star structures between Jan and April 2001, this abnormal increase was an indicator of cover up of accounting fraud happens at that time; (see Fig. 4.11(a)). Jeffrey Skilling (CEO) and Kenneth Lay (Chairman) were conducting regular meeting with their

top executives (e.g. Sally Beck (Chief Operating Officer), Vincent Kaminski (Quantitative Modeling Group Head), Darren Farmer (Logistics Manager), Michelle Lokay (Administrative Assistant), Louise Kitchen (Enron-online President),Williams III (Senior Analyst), Tim Belden (Trading Head) and Richard Sanders (Assistant General Counsel)), forming star structure (see Figure 4.11(b)), in order to find new ways to handle Enron's liability. Second observation from structure is shown in Figure 4.11(c) where Tim Belden's email address used to send emails to the Enron's World Trade center Office: 'center.dl-portland@enron.com' during time period of Oct - Dec 2001. But he also send a few emails to other employees inside the company as well. This would ensure that RichCom discovers most relevant people as central hub and further analysis could reveal more interesting patterns.

What sets* RichCom *apart: none of the state-of art method meet all the following specifications (which RichCom does): (a) consider multi-aspect graphs or tensors, (b) discover efficient communities using tensor decomposition approach and, (c) provide summarization of obtained community's structure by leveraging higher-order correlations between different aspects (either over time or over different views/aspects) to inform the extraction.

4.6 Conclusion

We proposed RichCom and cLL1, a novel constrained (L, L, 1) based framework to learn and encode the community structures. The performance of the proposed method is assessed via experiments on synthetic as well as six real-world networks.

Take-home points:

- cLL1 is novel constrained LL1-tensor decomposition method, and alternating optimization with alternating direction method of multipliers (AO-ADMM) is developed to recover the non-negative and sparse factors. The utility of cLL1 extends beyond multi-aspect graphs to general multi-aspect data mining, where the underlying latent structure is richer than rank-1.
- Through experimental evaluation on multiple datasets, we show that cLL1 provides stable decompositions and offering high quality structure via RichCom within reasonable run time, in the presence of overlapping and non-overlapping communities.
- Utility: we provide a simple and effective multi-aspect graph generator.

There are several items that can be considered for future work. First, as a natural extension, one can generalize this to account for higher order (> 3 modes) data. Second, AO-ADMM admits parallel extensions, which can enable the exploration of billion-scale multi-aspect graphs .

Chapter based on material published in TheWebConf 2020 [93].

Chapter 5

PARAFAC2 decomposition using auxiliary information

"Can we jointly models temporal and static information from PARAFAC2 data to extract meaningful insights? Does the static information added in temporal data improve predictive performance?"

PARAFAC2 is a powerful method for analyzing multi-modal data consisting of irregular frontal slices. In this work, we propose POPLAR method that imposes graph Laplacians constraints induced by the similarity symmetric tensor as auxiliary information to force decomposition factors to behave similarly and the method is developed using AO-ADMM for 3-way PARAFAC2 tensor decomposition. To the best of our knowledge, POPLAR is the first approach to incorporate graph Laplacians constraints using auxiliary information. We extensively evaluate POPLAR's performance in comparison to state-of-the-art approaches across synthetic and real dataset, and POPLAR clearly exhibits better

performance with respect to the Fitness (better 3-8%), and F1 score (better 5-20%) among the state-of-the-art factorization method. Furthermore, the running time for the method is comparable to the state-of-art method. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, Georgios Theocharous, and Evangelos E. Papalexakis. "POPLAR: Parafac2 decOmPosition using auxiLiAry infoRmation." In 2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM), pp. 1-5. IEEE, 2020.

5.1 Introduction

In real world applications, we often encounter multi-aspect relationships in data. For example, in social network analysis, interactions among various users and their interactions types are the main focus of interest. Tensors (or multi-way arrays) are highly suitable representation for such relationships. Tensor analysis methods have been extensively studied and applied by researchers [141, 197, 28] to many real-world problems. Regardless of recent development of traditional tensor decomposition approaches, there are certain instances [118] 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 5.1. The PARAFAC2 decomposition, proposed by Harshman [106], is another alternative to the traditional model. The PARAFAC2 model is appropriate for 3-mode data that do not follow a perfect trilinear structure and allows one of the modes to vary. It has been extensively used in chemometrics [12, 233], electronic health record[202], natural language processing [46], and social sciences [113].

In many practical cases, we have multi-aspect information represented as tensors (PARAFAC or PARAFAC2), and we can compute information on the objects forming the relationships based on various similarities. For example, in the (user, item, time)-relationships, each user has location/calls/connectivity information, and each item has its product/service information. Therefore, we can consider that we have similarity measures that corresponds to the sets of matrices for non-variable modes of data. Even if recently, a constraint PARAFAC2 (COPA) fitting algorithm was proposed for large and sparse data [6], it cannot incorporate meaningful auxiliary information on the model factors such as: a) user-user interactions, which facilitates model interpretability and understanding, b) product/service similarity that provides relational information among objects.

To handle the these challenges and inspired by the work by Narita et al. [182] which incorporates object similarity into PARAFAC1 tensor factorization, we exploit the auxiliary information given as similarity tensor in a regularization framework for PARAFAC2 tensor factorization. We propose the Parafac2 decOmPosition using auxiLiAry infoRmation method (POPLAR), which introduces graph laplacian constraints/regularization in PARAFAC2 modeling that enables to improve the prediction accuracy of tensor decomposition.

Our contributions are summarized as follows:

- **Novel Algorithm** We propose POPLAR, a method equipping the PARAFAC2 modeling with Graph Laplacian constraints. While POPLAR incorporates a additional auxiliary tensor for Laplacian constraints, it provides more accuracy than baselines and it is comparable in terms of scalability.

- **Experimental Evaluation** : we show experimental results on both synthetic and real datasets.

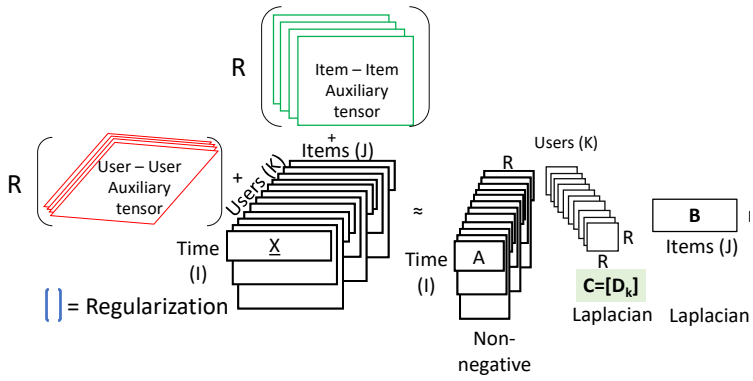


Figure 5.1: An illustration of the laplacian constraints imposed by POPLAR on PARAFAC2 tensor decomposition.

Table [5.1] provides our contributions in the terms of existing works.

Property	PARAFAC2	COPA	POPLAR
Sparsity	–	✓	✓
Graph Laplacian	–	–	✓
Scalability	–	✓	✓
Handle irregular tensors	✓	✓	✓

Table 5.1: Comparison of models.

5.2 Problem Formulation

We consider exploiting auxiliary information for improving the prediction accuracy by PARAFAC2 decomposition, especially for sparse observations. The problem that we focus on in this paper is summarized as follows.

Given: A third-order PARAFAC2 tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_k \times J}$ and symmetric similarity tensors $\underline{\mathbf{M}} \in \mathbb{R}^{J \times J \times M}$ and $\underline{\mathbf{N}} \in \mathbb{R}^{K \times K \times N}$, each corresponding to one of the regular modes of $\underline{\mathbf{X}}$.

Find: A decomposition $\underline{\mathbf{X}}$ defined by Eq.(2.25) and Eq (2.27).

5.3 Proposed Method: POPLAR

We propose Graph Laplacians regularization for PARAFAC2 tensors induced by the similarity tensor as auxiliary information to force factors to behave similarly. This a natural extension of the method proposed by Narita et al. [182] for tensor factorization using matrix as an auxiliary information. Consider 3-mode PARAFAC2 tensor with auxiliary tensors $\underline{\mathbf{M}}$ and $\underline{\mathbf{N}}$ on its fixed (2^{nd} and 3^{rd}) modes. The simple way to obtain these tensors is using various standard similarity methods like cosine similarity, Euclidean distance, Jaccard and ABC similarity etc. The regularization term we propose regularizes factor matrices of PARAFAC2 for its static modes using the similarity matrices (e.g user-user or item-item matrix). For simplicity, we explain process for 3^{rd} mode only. Thus, regularization term for 3^{rd} mode $\mathbf{C}$ is defined as:

$$\mathcal{R}(\mathbf{C}) = \sum_{n=1}^{N} Tr(\mathbf{C}^T \mathbf{L}_n \mathbf{C}) \tag{5.1}$$

where $\mathbf{L}$ is the Laplacian Matrix induced from the similarity tensor $\underline{\mathbf{M}}$ for the factor $\mathbf{C}$. Thus the objective function can be written as:

$$\mathcal{L} = min_{\{\mathbf{A_k}\},\{\mathbf{D_k}\},\mathbf{B}} \sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}_k\mathbf{D}_k\mathbf{B}\|_F^2 + \frac{\alpha}{2}(\mathcal{R}(\underline{\mathbf{M}}) + \mathcal{R}(\underline{\mathbf{N}})) \tag{5.2}$$

The regularization is imposed by using the Graph Laplacians (GL) method on the similarity tensors ($\underline{\mathbf{M}}$ and $\underline{\mathbf{N}}$) that helps to direct two similar objects in 2^{nd} and 3^{rd} mode to behave similarly. Mathematically, Equ. (5.2) can be re-written as :

$$\begin{aligned}\mathcal{L} = \min_{\{\mathbf{A_k}\},\{\mathbf{D_k}\},\mathbf{B}} \sum_{k=1}^{K} \|\mathbf{X}_k - \mathbf{A}_k\mathbf{D}_k\mathbf{B}\|_F^2 + \\ \frac{\alpha}{2}(Tr(\sum_{m=1}^{M} \mathbf{B}^T\mathbf{L}_m\mathbf{B} + \sum_{n=1}^{N} \mathbf{C}^T\mathbf{L}_n\mathbf{C})\end{aligned} \tag{5.3}$$

where $\mathbf{L}_m$ and $\mathbf{L}_n$ are the Graph Laplacian matrices obtained from the slices of similarity tensors $\underline{\mathbf{M}}$ and $\underline{\mathbf{N}}$, respectively. The Graph Laplacian matrix can be computed as follows:

$$\mathbf{L}_n = \mathbf{Deg}_n - \mathbf{N}_n$$

where $\mathbf{Deg}$ is degree matrix and it i^{th} diagonal element is the sum of all of the elements in the i^{th} row similarity matrix and computed as:

$$\mathbf{Deg}_{n_{i,j}} = \begin{cases} \sum_{j=1}^{N} \mathbf{N}_n(i,j), & \text{i = j.} \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

The regularization term can be simply interpreted as:

$$Tr(\sum_{n=1}^{N} \mathbf{C}^T\mathbf{L}_n\mathbf{C}) = \sum_{n=1}^{N}(\sum_{i,j=1}^{K} \mathbf{N}_{n_{i,j}} \sum_{r=1}^{R}(\mathbf{C}_{i,r} - \mathbf{C}_{j,r})^2) \tag{5.5}$$

This term implies that, if two objects are similar to each other, the corresponding factor vectors should be similar to each other. Thus, when using AO-ADMM the update rule for

$\mathbf{C}$ is:

$$\mathbf{C}^T := ((\mathbf{H}^T\mathbf{H} * \mathbf{B}^T\mathbf{B}) + \rho\mathbf{I})^{-1}(\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{H}) + \rho(\overline{\mathbf{C}} + \mathbf{M}_{\mathbf{C}^T}))^T$$

$$\overline{\mathbf{C}} := \min_{\overline{\mathbf{C}}} \sum_{n=1}^{N} Tr(\overline{\mathbf{C}}^T \mathcal{L}_n \overline{\mathbf{C}}) + \rho||\overline{\mathbf{C}} - \mathbf{C}^T + \mathbf{M}_{\mathbf{C}^T}||_F^2$$

$$\mathbf{M}_{\mathbf{C}^T} := \mathbf{M}_{\mathbf{C}^T} - \mathbf{C}^T + \overline{\mathbf{C}} \text{ s.t. } \overline{\mathbf{C}} = \mathbf{C}$$

where $\mathbf{M}_{\mathbf{C}^T}$ is a dual variable and the Lagrange multiplier ρ is a step size related to $\mathbf{C}^T$ factor matrix and set to minimum value between 10^{-3} and $Tr(\mathbf{H}^T\mathbf{H} * \mathbf{B}^T\mathbf{B})/R$ to yield good performance.

Adapting AO-ADMM to solve PARAFAC2 with Laplacians constraints has following benefits:

- AO-ADMM is more general than other methods in the sense that the loss function doesn't need to be differentiable.
- Simple to implement, parallelize and easy to incorporate a wide variety of constraints that can obtained using simple element-wise operations.
- Computational savings gained by using the Cholesky decomposition.
- The splitting scheme can be applied to large-scale data. Data can be distributed across different machines and optimize objective locally with communication on the primal, auxiliary and dual variables between the machines.

5.4 Experiments

5.4.1 Data Set Description

Auxiliary Tensor Creation: We created Auxiliary tensors using ABC similarity[219], Pearson correlation[219] , K-NN similarity [10], Jacard similarity and Edit distance.

Synthetic Data: Table 5.2 provides summary statistics regarding all datasets used. For all synthetic data we use rank $R = 10$. The entries of loading matrix $\mathbf{B}$ and $\mathbf{C}$ are Gaussian with unit variance, and orthogonality is imposed on factors $\mathbf{H}$ and $\mathbf{Q}$, then a few entries are clipped to zero randomly to create a sparse PARAFAC2 tensor. The labels are created by selecting highest value index for each entry in matrix $\mathbf{C}$.

Dataset	K	J	$\max(I_k)$	$\#nnz$	R
SYN-I	25k	5k	1k	2.1 Mil.	10
SYN-II	50k	10k	2k	5.4 Mil.	10
ADOBE	80k	1.7k	17k	3 Mil.	10, 40

Table 5.2: Summary statistics for the datasets of our experiments. K is the number of users, J is the number of items, I_k is the number of time observations for the k^{th} subject, $\#nnz$ corresponds to the total number of non-zeros in tensor $\underline{\mathbf{X}}$ and R is rank of tensor $\underline{\mathbf{X}}$.

Real Data: Adobe dataset is sequential data and it consists of tutorial sequence 7 million anonymized users. The data is structured as sequence-by-tutorial-by-user. We selected users who watched at least unique 10 tutorials. Thus, the dataset is of dimension [max 17k $\times$ 1.7k $\times$ 80k]. We compute user-user tensor similarity tensors. We have semi

synthetic ground truth values for this dataset and we assigned each user to class based on the type of tutorial watched.

5.4.2 Baselines

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

- **PARAFAC2** [135] : an implementation of standard fitting algorithm PARAFAC2 with random initialization.
- **COPA** [6] : a scalable constrained PARAFAC2 fitting algorithm was proposed for large and sparse data.

5.4.3 Evaluation Measures

We evaluate POPLAR and the baselines using three quantitative criteria: Fitness [0-1], F1-score [0-1] and CPU-Time (in seconds). These measures provide a quantitative way to compare the performance of our method. For Fitness and F1-score, higher is better.

5.4.4 Results

Fitness and Accuracy

We run each method for 3 different random initialization and provide the average and standard deviation of FIT as shown in Figure 5.2. This Figure illustrates the impact of proposed constraint on the FIT values across both synthetic for fixed rank R = 10 and

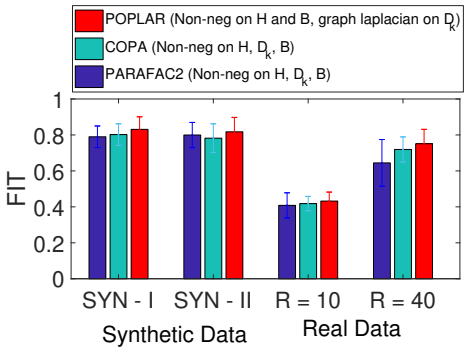


Figure 5.2: Comparison of FIT for different approaches on synthetic data with rank R = 10 and two target ranks R = 10 and R = 40 on real world dataset. Overall, POPLAR shows comparable fitness to baseline while supporting additional graph laplacian constraint.

real datasets for two different target ranks (R=10, 40). Overall,POPLAR achieves average $3-8\%$ improvement.

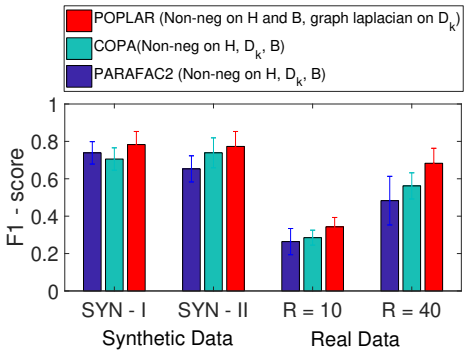


Figure 5.3: Comparison of F1-score for different approaches on synthetic data with rank R = 10 and two target ranks R = 15 and R = 40 on real world dataset. Overall, POPLAR achieves better F1 score to baseline.

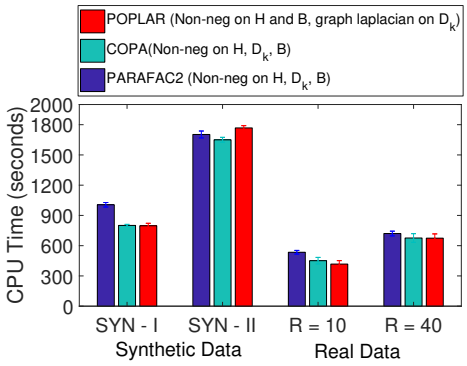


Figure 5.4: The CPU Time comparison (average and standard deviation) in seconds for non-negative version of PARAFAC2 & COPA and POPLAR for 3 different random initialization on synthetic data with rank R = 10 and two target ranks R = 10 and R = 40 on real world dataset. Note that even with additional processing of auxiliary tensor POPLAR performs just slightly slower than COPA for SYN-II, which does not support such graph laplacian constraints.

Similarly, we evaluate accuracy in terms of predicting correct labels using F1-score. We provide the average and standard deviation of F1 score as shown in Figure 5.3. Overall, POPLAR achieves significant improvement $5 - 20\%$ over baselines.

Computation time

Finally, we briefly discuss the computation time of our method. Although using auxiliary tensor as constraints slightly increases the time complexity of the PARAFAC2 decomposition method, the actual computation time is almost as same as that for baseline methods as shown in Figure 5.4. This is partially because 1) POPLAR converges (tolerance $= 10^{-7}$) faster than baselines 2) we used medium sized datasets in the experiments, and further investigation with larger datasets ($K > 10^5$) should be made in future work.

Scalability

We also evaluate the scalability of our algorithm on synthetic dataset. A PARAFAC2 tensors $\underline{\mathbf{X}} \in \mathbb{R}^{max1000 \times 1000 \times [5k-100k]}$ are decomposed with increasing target rank. The time needed by POPLAR increases very moderately. Our proposed method, successfully decomposed the tensor in reasonable time as shown in Figure 5.5.

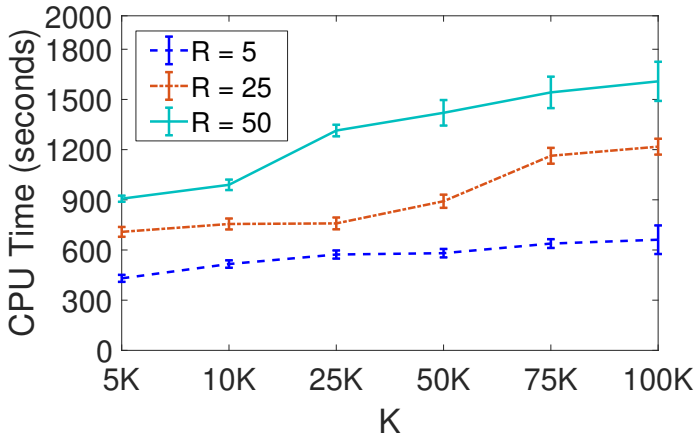


Figure 5.5: The average time in seconds for varying target rank.

5.5 Conclusion

This paper outlined our vision on exploring the graph laplacian regularization on PARAFAC2 tensor decomposition using auxiliary information to improve accuracy of factorization. We propose POPLAR, a AO-ADMM-based framework that is able to offer interpretable results, and we provide a experimental analysis on synthetic as well as real world dataset. By inspecting Figure 5.2-5.5, PARAFAC2 along with auxiliary information as laplacian constraints clearly exhibits the better performance with respect to the Fitness, and F1 score among the state-of-the-art factorization method. Furthermore, the running

time for method is comparable to state-of-art method. Furthermore, this paper outlines a set of interesting future research directions:

- How can we couple one of auxiliary tensor with PARAFAC2 tensor to obtain better approximation?
- What other constraints, other than graph laplacian or non-negative, for the PARAFAC2 decomposition are well suited for various application and have potential to offer more accurate results?
- How can we incorporate cross mode graph laplacian regularization for PARAFAC2 decomposition?

Chapter based on material published in SAM 2020 [97].

Chapter 6

Constraint Coupled CP and PARAFAC2 Tensor Decomposition

"Given data from a variety of sources that share a number of dimensions, how can we effectively decompose them jointly into interpretable latent factors?"

The coupled tensor decomposition framework captures this idea by jointly supporting the decomposition of several CP tensors. However, coupling tends to suffer when one dimension of data is irregular, i.e., one of the dimensions of the tensor is uneven, such as in the case of PARAFAC2. In this work, we provide a scalable method for decomposing coupled CP and PARAFAC2 tensor datasets through non-negativity-constrained least squares optimization on a variety of objective functions. We offer the following contributions: (1) Our algorithm can perform coupled factorization with an active-set, block principal pivoting and least square optimization method including the Frobenius norm induced non-negative factorization. (2) CAPTION scales to billions of non-zero elements in both the data and

model. Comprehensive experiments on large data confirmed that CAPTION is up to 5× faster and $70-80\%$ accurate than several baselines. We present results showing the scalability of this novel implementation on a billion elements as well as demonstrate the high level of interpretability in the latent factors produced, implying that coupling is indeed a promising framework for large-scale, unsupervised pattern exploration and cluster discovery. The content of this chapter is adapted from the following published paper:

Guiral, Ekta, Georgios Theocharous, and Evangelos E. Papalexakis. "C^{3}APTION: Constrainted Coupled CP And PARAFAC2 Tensor Decomposition." In 2020 IEEE/ ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 401-408. IEEE, 2020.

6.1 Introduction

With the opportunity to handle large volumes and velocity of data as a result of recent technical developments, such as mobile connectivity [189], digital tools [166], biomedical technology [24] and modern medical testing techniques [47], we face multi-source and multi-view data [89, 93] sets. Suppose, for example, that we are given a health care record data, such as Centers for Medicare and Medicaid (CMS) [47], and we have information about patient who visited hospital, or who got what kind of diagnosis in which visit, and when. This data may be formulated as a three mode PARAFAC2 tensor. Suppose now that we also have some static features information pertaining to the patient, e.g. multi-aspect relation based on demographic information. This data may be formulated as a three mode CP tensor. This problem can be formulated as an example of a coupled factorization,

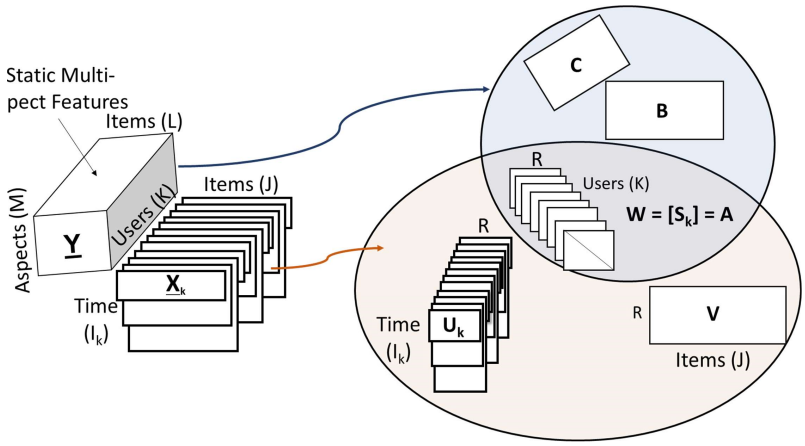


Figure 6.1: Illustration of CAPTION decomposition. Each slice of $\underline{\mathbf{X}}_k$ represents the different clinical visits for patient k. CP tensor $\underline{\mathbf{Y}}$ includes the similarity CP tensor based on demographic information of patients. CAPTION decomposes $\underline{\mathbf{X}}_k$ into three parts: $\mathbf{X}_k$, $\mathbf{W} = \mathbf{S}_k$, and $\mathbf{V}$. CP tensor $\underline{\mathbf{Y}}$ is decomposed into $\mathbf{W} = \mathbf{S}_k$, $\mathbf{B}$ and $\mathbf{C}$. Note that latent factor $\mathbf{W}$ is shared between both tensors.

where the two tensors of a 3-mode (visits, diagnosis, patients) PARAFAC2 and a 3-mode (patients, patients, aspect) CP tensor share a common dimension.

In many practical cases, we have multi-aspect information represented as tensors. Despite its attractiveness, individual tensor factorization suffers from robustness issues. Applying coupled tensors and matrices to heterogeneous datasets from multiple sources has been a topic of interest in many areas. Coupled tensor decomposition gives an equivalent representation of multi-way data by a set of factors and parts of the factors are shared for coupled data. In literature, fusion and coupled methods [4, 5, 7, 40, 76] reported so far ignore the underlying irregular nature of the data in at least one of the modalities

among the data like health care data. Acar et al. proposed an all-at-once coupled gradients based optimization approach, called CMTF-OPT [4]. The advanced version of CMTF-OPT, ACMTF-OPT [5], places additional constraints on the model to force good performance when distinguishing between shared and unshared data latent factors. Many researchers have subsequently made improvements [198, 2, 124] to CMTF for large-scale data. In [40], paper proposed fusion or soft coupling of both EEG and fMRI PARAFAC2 data and provides insights on presence of shifts in the ERPs per subject. Similarly, [76], proposed robust coupling of two CP tensors via measuring distance between factors. Recently, Afshar et al. [7] proposed method based on block principle pivot, namely TASTE, for coupling between PARAFAC2 tensor and matrix and this method provides valuable insights for phenotyping of electronic health records. But, these prior work has either focused on a specific type of coupled factorization (two CP tensors or two PARAFAC2 tensors or tensor-matrix) or a specific objective function, thus having a limited range of potential applications where two different format of data is required.

To handle the these limitations and inspired by the work by Afshar et al. [7], we proposed a scalable method namely CAPTION that couple CP and PARAFAC2 tensor which incorporates non-negative constraints with multiple update settings of latent factors as shown in figure 6.1. We demonstrate, with synthetic and real data, the advantage of the proposed method over baseline methods in terms of accuracy and computation time. A preliminary version of this work appeared in [66] as a short paper. In this paper, we extend those preliminary results by (i) providing a detailed description of all proposed methods to handle limitations of previous work, (ii) provide thorough experimentation on synthetic

and real data, (iii) provide detailed case studies in real data using our proposed method, and (iv) we conduct a scalability analysis, demonstrating that our proposed method can scale up to billions of non-zero elements in data and 5× faster and 70-80% accurate than any state-of-art method. Our contributions are summarized as follows:

- **Novel and Scalable Algorithm**: We propose CAPTION, a method of coupling the CP and PARAFAC2 tensor with various optimization update rules with non-negative constraint. Our proposed method is efficient, scalable and provides stable decompositions than baselines.
- **Fast and Accurate Algorithm**: Our proposed fitting algorithm is up to 5× faster than the state of the art baseline. At the same time, CAPTION preserves model accuracy better than baselines while maintaining interpretability.
- **Experimental Evaluation**: we show experimental results on both synthetic and real datasets.

To promote reproducibility, we make our MATLAB implementation publicly available at link[1].

6.2 Proposed Method: CAPTION

A generalized CP and PARAFAC2 approach is appealing from several perspectives including the ability to use different aspect or information of data, improved interpretability of decomposed factors, and more reliable and robust results. We propose CAPTION, a scalable and coupled CP and PARAFAC2 model, to impose non-negativity constraints on

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

the factors. We consider exploiting coupling for improving the prediction accuracy by PARAFAC2 decomposition, especially for sparse observations. The problem that we focus on in this paper is summarized as follows.

Given: A third-order PARAFAC2 tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_k \times J}$ and CP tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{K \times L \times M}$ corresponding to 3^{rd} mode of $\underline{\mathbf{X}}$.

Find: A decomposition defined by Eq.(6.3).

6.2.1 General Framework for CAPTION

Given PARAFAC2 tensor $\underline{\mathbf{X}}$ and CP tensor $\underline{\mathbf{Y}}$ coupled in its 3^{rd} mode, this section proposes three different settings of the coupled tensor decomposition CAPTION in order to factorize the multi-aspect graph or tensor into its constituent community-revealing factors. We focus on a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_k \times J}$ $(\forall k \in [1, K])$ and $\underline{\mathbf{Y}} \in \mathbb{R}^{K \times L \times M}$ for our problem and its loss function formulation is given by:

$$\begin{aligned} \mathcal{LS} = & \underset{\mathbf{Q}_k, \mathbf{U}_k, \mathbf{H}, \mathbf{V}, \mathbf{S}_k, \mathbf{B}, \mathbf{C}}{\arg\min} \frac{1}{2} ||\mathbf{X}_k - \mathbf{U}_k \mathbf{S}_k \mathbf{V}^T||_F^2 + \\ & \frac{\lambda}{2} ||\underline{\mathbf{Y}} - \mathbf{W}(\mathbf{C} \odot \mathbf{B})^T]||_F^2 + \sum_{k=1}^{K} \left(\frac{\mu_k}{2} ||\mathbf{U}_k - \mathbf{Q}_k \mathbf{H}||_F^2 \right) \\ & \textbf{s. t. } \mathbf{Q}_k^T \mathbf{Q}_k = \mathbf{I}, \mathbf{U}_k \geq 0, \mathbf{S}_k \geq 0, \\ & \mathbf{W}(k, 0) = diag(\mathbf{S}_k), \mathbf{B} \geq 0, \mathbf{C} \geq 0 \quad \forall k \in [1, K] \end{aligned} \tag{6.1}$$

For PARAFAC2, we re-write the first part $\mathcal{LS}_1$ of minimization of $\mathcal{LS}$ function in terms of $\mathbf{Q}_k$ as $Trace(\underline{\mathbf{X}}_k^T \underline{\mathbf{X}}_k) + Trace(\mathbf{V}\mathbf{S}_k \mathbf{H}^T \mathbf{Q}_k^T \mathbf{Q}_k \mathbf{H} \mathbf{S}_k \mathbf{V}^T) - 2 * Trace(\underline{\mathbf{X}}_k^T \mathbf{Q}_k \mathbf{H} \mathbf{S}_k \mathbf{V}^T)$. The first and second terms are constant and by rearranging the rest, it is equivalent to

$$\begin{aligned} \mathcal{LS}_1 = & \underset{\mathbf{Q}_k, \mathbf{U}_k, \mathbf{H}, \mathbf{V}, \mathbf{S}_k}{\arg\min} \frac{1}{2} ||\mathbf{Q}_k^T \mathbf{X}_k - \mathbf{H} \mathbf{S}_k \mathbf{V}^T||_F^2 \\ & \textbf{s. t. } \mathbf{Q}_k^T \mathbf{Q}_k = \mathbf{I}, \mathbf{U}_k \geq 0, \mathbf{S}_k \geq 0, \quad \forall k \in [1, K] \end{aligned} \tag{6.2}$$

Thus, the constrained coupled tensor decomposition objective $\mathcal{LS}$ is of the form:

$$\begin{aligned}\mathcal{LS} = \underset{\mathbf{Q}_k,\mathbf{U}_k,\mathbf{H},\mathbf{V},\mathbf{S}_k,\mathbf{B},\mathbf{C}}{\arg\min} & \frac{1}{2}||\mathbf{X}_k - \mathbf{H}\mathbf{S}_k\mathbf{V}^T||_F^2 + \\ & \frac{\lambda}{2}||\underline{\mathbf{Y}} - \mathbf{W}(\mathbf{C}\odot\mathbf{B})^T]||_F^2 + \sum_{k=1}^{K}(\frac{\mu_k}{2}||\mathbf{U}_k - \mathbf{Q}_k\mathbf{H}||_F^2) \\ & \textbf{s. t. } \mathcal{X}(:,:,k) = \mathbf{Q}_k^T\underline{\mathbf{X}}_k, \mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I}, \mathbf{U}_k \geq 0, \mathbf{S}_k \geq 0, \\ & \mathbf{W}(k,0) = diag(\mathbf{S}_k), \mathbf{B} \geq 0, \mathbf{C} \geq 0 \quad \forall k \in [1,K]\end{aligned} \tag{6.3}$$

6.2.2 Inference of Factors

We propose 3 types of algorithms to solve coupling namely CAPTION-ASET (unconstrained version), CAPTION-BPP (constrained), and CAPTION-ALS (constrained). First two methods are natural extension of [7] tensor coupling. Equation 6.3 is non-convex, our method utilizes instances of the non-negativity constrained least squares (NNLS) framework to divide problem into sub-problems. We use optimization method block principal pivoting for CAPTION-BPP [137], active set of Lawson and Hanson [136] for CAPTION-ASET and least square method for CAPTION-ALS to solve each sub-problem. Next, we summarize the solution for each latent factor.

Factor $\mathbf{Q}_k$ update

Consider first the update of factor $\mathbf{Q}_k$ obtained after fixing other factor matrices. For CAPTION-ASET and CAPTION-BPP, we update the $\mathbf{Q}_k$ by minimizing Equ. 6.1 using following method:

$$\underset{Q_k}{\arg\min}\ \frac{\mu_k}{2}||\mathbf{U}_k - \mathbf{Q}_k\mathbf{H}||_F^2 \quad s.t. \quad \mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I}$$

$$\arg\min_{Q_k} \mu_k(Tr(\mathbf{Q}_k\mathbf{H}\mathbf{H}^T\mathbf{Q}_k^T - 2\mathbf{Q}_k\mathbf{H}\mathbf{U}_k^T + \mathbf{U}_k\mathbf{U}_k^T)) = 0$$

Using $Tr(ABC) = Tr(CAB)$ property, we can re-write $Tr(\mathbf{Q}_k\mathbf{H}\mathbf{H}^T\mathbf{Q}_k^T) = Tr(\mathbf{Q}_k^T\mathbf{Q}_k\mathbf{H}\mathbf{H}^T)$.

As $\mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I}$, we can reformulate above equation w.r.t. $\mathbf{Q}_k$ as follows :

$$\arg\min_{Q_k} \mu_k\mathbf{Q}_k\mathbf{H}\mathbf{U}_k^T \quad s.t. \quad \mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I}$$

$$[\mathbf{Q}_k] = SVD[\mu_k\mathbf{H}\mathbf{U}_k^T] \quad s.t. \quad \mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I} \tag{6.4}$$

For CAPTION-ALS, this factor is computed using the simple SVD as:

$$[\mathbf{Q}_k] = SVD[\mathbf{H} \times diag(\mathbf{W}(k,:)) \times (\underline{\mathbf{X}}_k \times \mathbf{V})^T] \tag{6.5}$$

Note that each $\mathbf{Q}_k$ can contain negative values.

Factor H update

We update $\mathbf{H}$ by fixing $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{Q}_k$. We set derivative the loss $\mathcal{LS}$ w.r.t. $\mathbf{H}$ (Equ. 6.1, note that part 1 and part 2 are constant) to zero to find local minima as follows:

$$\frac{\delta\mathcal{LS}}{\delta\mathbf{H}} = \frac{\sum_{k=1}^{K}\frac{\mu_k}{2}Tr((\mathbf{U}_k - \mathbf{Q}_k\mathbf{H})(\mathbf{U}_k - \mathbf{Q}_k\mathbf{H})^T)}{\delta\mathbf{H}} = 0$$

$$\delta(\sum_{k=1}^{K}\mu_k Tr(\mathbf{Q}_k\mathbf{H}\mathbf{H}^T\mathbf{Q}_k^T - 2\mathbf{Q}_k\mathbf{H}\mathbf{U}_k^T + \mathbf{U}_k\mathbf{U}_k^T))/\delta\mathbf{H} = 0$$

$$\sum_{k=1}^{K}\mu_k\mathbf{Q}_k^T\mathbf{Q}_k\mathbf{H} - \sum_{k=1}^{K}\mu_k\mathbf{Q}_k^T\mathbf{U}_k = 0$$

as $\mathbf{Q}_k^T\mathbf{Q}_k = \mathbf{I}$, the update rule for latent factor $\mathbf{H}$ is given below:

$$\begin{aligned} &\textbf{For CAPTION-ASET:} \quad \mathbf{H} = \frac{\sum_{k=1}^{K}\mathbf{Q}_k^T\mathbf{U}_k}{\sum_{k=1}^{K}\mu_k} \\ &\textbf{For CAPTION-BPP:} \quad \mathbf{H} = \frac{\sum_{k=1}^{K}\mathbf{Q}_k^T\mathbf{U}_k}{\sum_{k=1}^{K}\mu_k} \quad \textbf{s. t. } \mathbf{H} \geq 0 \end{aligned} \tag{6.6}$$

For **CAPTION-ALS**, we set $\mu_k = 0$ and derive update rule from Equ. 6.3 as follows:

$$\mathbf{H} = \frac{(\mathcal{X}\mathbf{V}) * \mathbf{W}^T}{(\mathbf{V}^T\mathbf{V} * \mathbf{W}^T\mathbf{W})} \quad \textbf{s. t. } \mathbf{H} \geq 0 \tag{6.7}$$

Factor $\mathbf{S}_k$ or $\mathbf{W}$ update

This mode of the PARAFAC2 tensor is coupled with CP tensor. The objective function 6.1 with respect to $\mathbf{W}$ can be rewritten as:

$$\arg\min_{S_k} \frac{1}{2}||\underline{\mathbf{X}}_k - \mathbf{U}_k\mathbf{S}_k\mathbf{V}^T||_F^2 + \frac{\lambda}{2}||\underline{\mathbf{Y}} - \mathbf{W}(\mathbf{C} \odot \mathbf{B})^T]||_F^2 \tag{6.8}$$

For **CAPTION-ASET** Equation 6.8 can be rewritten as:

$$\arg\min_{\mathbf{S}_k} \frac{1}{2}||\begin{bmatrix} (\mathbf{V} \odot \mathbf{U}_k) \\ \sqrt{\lambda}(\mathbf{C} \odot \mathbf{B}) \end{bmatrix}\mathbf{W}(k,:)^T - \begin{bmatrix} \text{vec}(\underline{\mathbf{X}}_k) \\ \sqrt{\lambda}\text{vec}(\underline{\mathbf{Y}}(k,:,:)) \end{bmatrix}||_F^2 \tag{6.9}$$

For **CAPTION-BPP**, Equation 6.9 can be computed such that $\mathbf{W}(k,:) \geq 0$. The Khatri-rao product operation is expensive that can be replaced by element-wise (hadamard) product and matrix to tensor product can be replaced by slice wise dot product with factor matrices [159].

For **CAPTION-ALS**, we update $\mathbf{S}_k$ or $\mathbf{W}$ as:

$$\begin{aligned} \mathbf{S}_k &= \frac{(\mathbf{U}_k^T\mathbf{U}_k * \mathbf{V}^T\mathbf{V}) + (\sqrt{\lambda}(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})}{diag(\mathbf{U}_k^T\underline{\mathbf{X}}_k\mathbf{V}) + diag(\mathbf{B}^T\underline{\mathbf{Y}}(k,:,:)\mathbf{C})} \\ \mathbf{W}(k,:) &= diag(\mathbf{S}_k) \quad \textbf{s. t. } \mathbf{W}(k,:) \geq 0, \forall k \in [1,K] \end{aligned} \tag{6.10}$$

Factor $\mathbf{V}$ update

We solve Equation 6.1 with respect to $\mathbf{V}$ as given below:

$$\arg\min_{\mathbf{V}} \frac{1}{2}||\underline{\mathbf{X}}_k - \mathbf{U}_k\mathbf{S}_k\mathbf{V}^T||_F^2 \tag{6.11}$$

For **CAPTION-ASET**, Equ. (6.11) can be formulated as:

$$\mathbf{V}(:,k) = \frac{\mathbf{X}_k^T}{\mathbf{S}_k\mathbf{U}_k^T} \tag{6.12}$$

For **CAPTION-BPP**, Equ. (6.12) can be easily updated such that by $\mathbf{V} \geq 0$.

For **CAPTION-ALS**, we update $\mathbf{V}$ as given below:

$$\mathbf{V} = \frac{(\mathcal{X}^T\mathbf{H}) * \mathbf{W}^T}{(\mathbf{H}^T\mathbf{H} * \mathbf{W}^T\mathbf{W})} \quad \textbf{s. t.} \quad \mathbf{V} \geq 0 \tag{6.13}$$

Factor B or C update

Finally, factor matrices $\mathbf{B}$ and $\mathbf{C}$ represents the participation of CP tensor for user similarities. We solve Equation 6.1 w.r.t $\mathbf{B}$ as given below:

$$\arg\min_{\mathbf{B}} \frac{1}{2}||\underline{\mathbf{Y}} - \mathbf{B}(\mathbf{C} \odot \mathbf{W})^T||_F^2 \quad \text{s. t.} \quad \mathbf{B} \geq 0 \tag{6.14}$$

which can be easily updated via all-set method for CAPTION-ASET and via block principal pivoting for CAPTION $-$ BPP.

For CAPTION-ALS, we update $\mathbf{B}$ as given below:

$$\mathbf{B} = \frac{\text{MTTKRP}(\underline{\mathbf{Y}}, \mathbf{C}, \mathbf{W})}{(\mathbf{C}^T\mathbf{C} * \mathbf{W}^T\mathbf{W})} \quad \textbf{s. t.} \quad \mathbf{B} \geq 0 \tag{6.15}$$

Similarly, we update $\mathbf{C}$ as:

$$\mathbf{C} = \frac{\text{MTTKRP}(\underline{\mathbf{Y}}, \mathbf{B}, \mathbf{W})}{(\mathbf{B}^T\mathbf{B} * \mathbf{W}^T\mathbf{W})} \quad \textbf{s. t.} \quad \mathbf{C} \geq 0 \tag{6.16}$$

Factor $\mathbf{U}_k$ update

For CAPTION-ALS, this factor is computed using the simple multiplication $\mathbf{U}_k = \mathbf{Q}_k * \mathbf{H}$. For CAPTION-ASET and CAPTION-BPP (where $\mathbf{U}_k \geq 0$), the objective function with respect to $\mathbf{U}_k$ can be solved as:

$$\arg\min_{\mathbf{U}_k} \frac{1}{2}||\begin{bmatrix}(\mathbf{V}\mathbf{S}_k) \\ \sqrt{\mu_k}\mathbf{I}\end{bmatrix}\mathbf{U}_k^T - \begin{bmatrix}\underline{\mathbf{X}}_k^T \\ \sqrt{\mu_k}\mathbf{H}^T\mathbf{Q}_k^T\end{bmatrix}||_F^2 \tag{6.17}$$

6.3 Experiments

In this section we extensively evaluate the performance of CAPTION on multiple synthetic and real datasets, and compare its performance with state-of-the-art approaches. We focus on answering the following:

Q1. Does CAPTION preserve accuracy while being fast to compute and helps in Identifiability of latent factors?

Q2. How does CAPTION scale for increasing number of users (K)?

Q4. How can we use CAPTION for real-world utility?

6.3.1 Dataset

We provide the datasets used for evaluation in Table 6.1. Rank determination in the experiments is performed with the aid of the Core Consistency Diagnostic method [31, 193].

Synthetic Data: In order to fully explore the performance of CAPTION, in our experiments we generate synthetic tensor with varying density. Those tensors are created from a known set of randomly generated factors, so that we have full control over the ground truth of the full decomposition. The specifications of synthetic datasets are given in Table 6.1.

Real Data: In order to truly evaluate the effectiveness of CAPTION, we test its performance against four real datasets that have been used in the literature. Those datasets are summarized in Table 6.2 and details are below.

Dataset	Statistics (K: Thousands M: Millions)			
	$[I_{max}, J, K]$	[K, L, M]	R	$\#nnz$
SYN-I	$[500, 1K, 5K]$	$[5K, 5K, 500]$	40	$[0.5B, 1.5B]$
SYN-II	$[1K, 1K, 10K]$	$[10K, 10K, 1K]$	40	$[1.4B, 3.9B]$
SYN-III	$[1K, 5K, 50K]$	$[50K, 50K, 1K]$	10	$[6B, 9B]$
Collaboration	$[25, 10, 11K]$	$[11K, 11K, 5]$	$5-50$	$[1M, 1.2M]$
Movielens	$[121, 4K, 6K]$	$[6K, 6K, 5]$	$5-50$	$[1M, 4.5M]$
Adobe	$[1K, 1K, 31K]$	$[31K, 31K, 5]$	$5-50$	$[1.7M, 6.3]$
CMS	$[250, 1K, 98K]$	$[98K, 98K, 5]$	$5-50$	$[9.6M, 9.7M]$

Table 6.1: Details for the datasets.

Collaboration Data[191]: It is co-authorship network (where two authors are connected if they publish at least one paper together) of 11,176 authors over years 1990-2015 for International Conference on Data Mining (ICDM), International Conference on Machine Learning (ICML), Knowledge Discovery and Data Mining (KDD) conference.

Movielens[103]: MovieLens-1M 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: Adobe dataset is sequential data and it consists of tutorial sequence of anonymous 7 million users. We selected users (31K) who watched at least unique 15 tutorials. We created PARAFAC2 tensor as sequence-by-tutorial-by-user [max 1k $\times$ 1k $\times$ 31k] and CP tensor as user-by-user-by-similarity. We have semi synthetic ground truth values for this dataset and we assigned each user to class based on the type of tutorial watched.

Data	Metric	SCD	RCTF	TASTE	C-BPP	C-ASET	C-ALS
SYN-I	RMSE	0.43(0.055)	0.38(0.068)	0.21(0.041)	0.20(0.068)	0.26(0.032)	**0.18 (0.026)**
	NMI	0.45(0.010)	0.49(0.074)	0.78(0.021)	0.79(0.019)	0.65(0.012)	**0.92 (0.034)**
	Time	490.01(14.07)	548.32(16.36)	357.23(34.59)	336.43(11.59)	348.56(56.43)	**301.87 (34.43)**
SYN-II	RMSE	[[OoM]]	[[OoM]]	0.30(0.023)	0.29(0.013)	0.36(0.092)	**0.25 (0.065)**
	NMI			0.65(0.044)	0.68(0.023)	0.61(0.049)	**0.75 (0.063)**
	Time			2109.11(89.75)	2021.48(56.35)	2090.41(134.67)	**1689.68 (101.23)**
SYN-III	RMSE	[[OoM]]	[[OoM]]	0.35(0.022)	0.38(0.035)	0.43(0.056)	**0.32 (0.081)**
	NMI			0.72(0.069)	0.70(0.013)	0.65(0.086)	**0.76 (0.081)**
	Time			2387.56(72.85)	2304.68(89.63)	2360.41(112.34)	**1959.68 (91.23)**

Table 6.2: Performance of CAPTION in terms of RMSE, NMI and CPU Time (mins) for synthetic data. Numbers where our proposed method outperforms other baselines are bolded. For each dataset, we report the standard deviation between two parentheses along with average score. Remarkably, CAPTION-ALS better preserve accuracy which ultimately, improves task performance.

CMS [47]: This dataset is synthetically created by Centers for Medicare and Medicaid (CMS) by using 5% of real medicare data and includes 98K beneficiaries. We created PARAFAC2 tensor as visits-by-diagnosis-by-patient and CP tensor as user-by-user-by-similarity.

We create CP tensor using well known similarity methods such that cosine similarity, Jaccard similarty, LSH hasing [220], ABC hashing[220], K-mean and Edit distance.

6.3.2 Baselines

Here we briefly present the state-of-the-art baselines we used for comparison. Note that for each baseline we use the reported parameters that yielded the best performance in the respective publications.All comparisons were carried out over 10 iterations each, and

each number reported is an average with a standard deviation attached to it. We compared the following algorithms for coupling CP and PARAFAC2 tensors.

- **TASTE** [7]: This method based on block principle pivot for coupling between PARAFAC2 tensor and matrix. We run algorithm for all slices of CP tensor $\underline{\mathbf{Y}}$.
- **Soft Coupled Decomposition** [40]: SCD method is soft coupling of two PARAFAC2 data.
- **Robust Coupled Tensor Factorization** [76]: RCTF is robust coupling of two CP tensors via measuring distance between factor.
- Our proposed methods:
  - **CAPTION-ASET**: This is solving coupled tensor factorization via Active set methods with using the unconstrained least squares optimization.
  - **CAPTION-BPP**: This is solving coupled tensor factorization via block principal pivoting method using the non negativity-constrained least squares optimization.
  - **CAPTION-ALS**: This is solving coupled tensor factorization via alternating non-negative least squares optimization.

6.3.3 Evaluation Measures

We evaluate CAPTION and the baselines using three quantitative criteria: Root Mean Square Error and CPU-Time (in minutes). Briefly,

- **Root Mean Square Error**: Performance is evaluated as the Root Mean Square Error (RMSE) which is a well known evaluation measure used in coupled tensor fac-

torization literature. Mathematically,

$$RMSE = \sqrt{\frac{\sum_{k=1}^{K} \left(||\underline{\mathbf{X}}_k - \hat{\underline{\mathbf{X}}}_k||^2 \right) + ||\underline{\mathbf{Y}} - \hat{\underline{\mathbf{Y}}}||^2}{\sum_{k=1}^{K} (I_k \times J) + (K \times L \times M)}} \tag{6.18}$$

- **CPU time (sec)**: indicates how much faster does the decomposition runs as compared to baselines. The average running time is measured in seconds, and is used to validate the time efficiency of an algorithm.
- **Normalized Mutual Information**: Normalized Mutual Information (NMI) is a good measure for determining the quality of clustering. Mathematically,

$$NMI(Y, C) = \frac{2 * I(Y; C)}{[H(Y) + H(C)]} \tag{6.19}$$

where I(Y, C) is mutual information between cluster Y and C, H(Y) and H(C) are entropy of cluster and classes.

6.3.4 Experimental Result

Q1a. Effectiveness and Run Time

We evaluate performance of the algorithm for community detection where each node in a graph is assigned to a single label. In our study, we perform hard clustering over latent factor matrices.We run each method for 10 different random initialization and provide the average and standard deviation of RMSE and CPU Time (min) as shown in Table 6.2.

Synthetic Data: The baseline method SCD and RCTF unable to decompose SYN-II and SYN-III due to out of memory during intermediate computations. Our proposed methods CAPTION-BPP and CAPTION-ASET provide comparable accuracy and

Dataset	Metric	SCD	RCTF	TASTE	C-BPP	C-ASET	C-ALS
Collaboration	RMSE	0.39(0.075)	0.35(0.068)	0.17(0.041)	0.17(0.068)	0.23(0.032)	**0.14 (0.026)**
	Time	379.01(14.07)	427.88(16.36)	236.93(19.69)	210.43(11.59)	226.43(13.28)	**174.31 (11.41)**
Movielens	RMSE	0.24(0.021)	0.28(0.020)	0.19(0.082)	0.16(0.012)	0.21(0.093)	**0.14 (0.012)**
	Time	48.21(3.21)	45.20(2.34)	21.34(1.69)	20.89(4.83)	25.55(2.84)	**10.19 (1.36)**
Adobe	RMSE	0.29(0.033)	0.35(0.062)	0.28(0.015)	0.22(0.023)	0.26(0.042)	**0.20 (0.013)**
	NMI	0.42(0.05)	0.48(0.09)	0.53(0.02)	0.54(0.01)	0.49(0.06)	**0.58 (0.08)**
	Time	210.23(11.34)	198.20(16.96)	98.22(10.58)	93.23(23.52)	150.24(25.58)	**78.72 (21.74)**
CMS	RMSE	0.29(0.045)	0.34(0.098)	0.21(0.058)	**0.20 (0.052)**	0.24(0.021)	0.23(0.037)
	Time	466.23(13.44)	435.34(9.10)	150.24(10.92)	149.24(11.23)	202.24(19.13)	**112.33 (11.93)**

Table 6.3: Performance of CAPTION in terms of RMSE and CPU Time (mins) for real data decomposed. Numbers where our proposed method outperforms other baselines are bolded. For each dataset, we report the standard deviation between two parentheses along with average score. *Note: we have semi-synthetic labels for the Adobe dataset only.

runtime when compared to TASTE method. Overall, CAPTION-ALS achieves significant improvement on running time and average $3-8\%$ RMSE improvement. Therefore, our approach is the only one that achieves a fast and accurate solution.

For real dataset we do not have labels, so we provide only RMSE and CPU Time for these data as discussed below:

Collaboration Data: We observed that CAPTION-ALS provides the high-quality communities as shown in Table 6.3. We see similar behaviour with CAPTION-BPP also. We select top two communities based on size. Each community represents a group of scientists with the same research interests, such as Data Mining community (#3) and Information Retrieval and Web Mining community (#10) in Table 6.4. Researchers like "Jiawei Han" and "Philip S. Yu", have published a large number of papers in collaboration with people from various research communities. These authors considered as tightly related

to the same community in the network. We further analyze the outcome of baseline methods and observe that SCD and RCTF are not able to find few strongly connected communities and fails to merge the small groups even those share a strong connection. In terms of RMSE, CAPTION-ALS, outperformed the baselines as shown in Table 6.3

Community[#3]	**Community[#10]**
Jiawei Han	Rakesh Agrawal
Philip S. Yu	Ramakrishnan Srikant
Wei Fan	Panayiotis Tsaparas
Charu C. Aggarwal	H Lei Zhang
Jimeng Sun	Josh Attenberg
Jian Pei	Anitha Kannan
Bing Liu	Sreenivas Gollapudi
Bhavani M. Thuraisingham	Kamal Ali
Longbing Cao	Sunandan Chakraborty
Tanya Y. Berger Wolf	Rui Cai
Xindong Wu	Indu Pal Kaur

Table 6.4: Top two communities (based on size) discovered by CAPTION-ALS on *Collaboration* Dataset. Selected researchers are based on top 10 factor values of latent factor.

Movielens Data: We decompose movielens data using R=20 for all methods. Our proposed method CAPTION-ALS outperformed w.r.t RMSE and computation time. Here, we explore interesting observations. First, we observe that there is a rapid growth of sci-fi movies beginning of 1970, a few months after the first Moon landing. Secondly, we

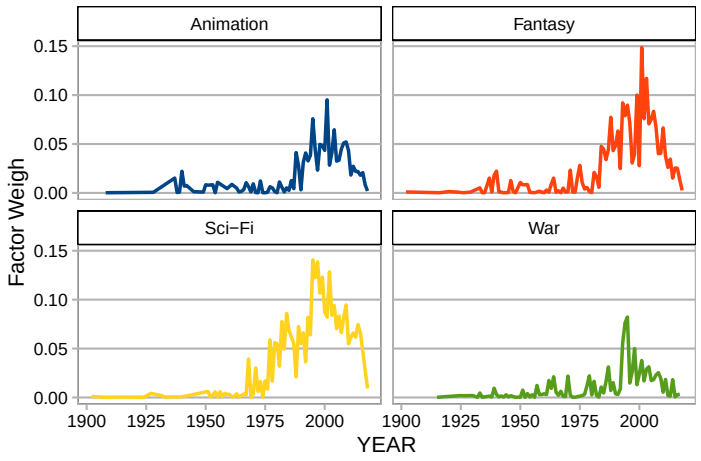


Figure 6.2: Movielens Data exploratory analysis for top movie genre.

observe that the rise of popularity of animation movies as shown in fig 6.2, the reason could be the advancement of the computer animation technology which made the development of such movies much easier. Next, the most of the war/action movies were popular around the time of World War II, Vietnam War and war in Afghanistan and Iraq. It's interesting to observe that how the cinematography world reflected the viewership of the real world. Another interesting observation is that most of the salesman and programmers mostly loved adventurer movies and lawyers liked most of drama and fantasy movies.

Adobe Data: CAPTION-ALS outperforms the baseline methods. Remarkably, it surpasses the baselines most when the data is sparse. In order to present the use of CAPTION towards community detection, we focus our analysis on a subset of tutorials watched by each community in this dataset. Figure 6.3 presents the top 5 (based on size) community's most frequent tutorial(s) sequence watched. Conceptually, those users share similar interest in terms of domain knowledge, learning or interests. We observe from the factors of the CP tensor that these communities are connected strongly within the

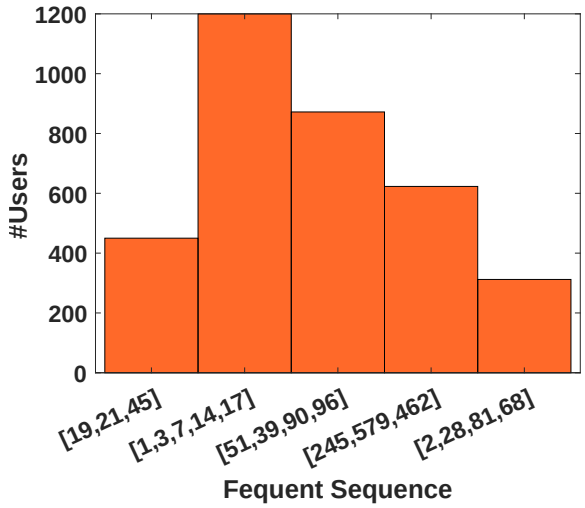


Figure 6.3: Frequent sequence of tutorials watched for top communities based on size.

group and have few connections outside the group. Nevertheless, CAPTION-ALS achieves significantly good performance in terms of $NMI \approx 0.58$ as shown in Table 6.3.

Q1b. Identifiability Analysis As we know that PARAFAC2 is known for the hardness in terms of optimization. In many instances, Alternate Least Square (ALS) based algorithms do not converge to good solutions, although the PARAFAC2 decomposition theoretically has identifiability. For identifiability analysis, PARAFAC2 tensor $\underline{\mathbf{X}} \in \mathbb{R}^{1000\times1000\times1000}$ and CP tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{1000\times1000\times500}$ is constructed with fixed target rank $R = 10$. Our aim is to recover latent factors as similar as possible to original latent factors. To simplify, we discuss identifiability of latent factor matrix $\mathbf{W}$ only. We compute the dot product for all permutations of columns between original latent factors (W_{org}) and latent factors (W_{pred}) obtained after decomposition. If the computed dot product is higher than the threshold value (80%), the two factors match, and we consider them as recovered factor. If the dot product between a column in W_{org} and with all the columns in W_{pred} has

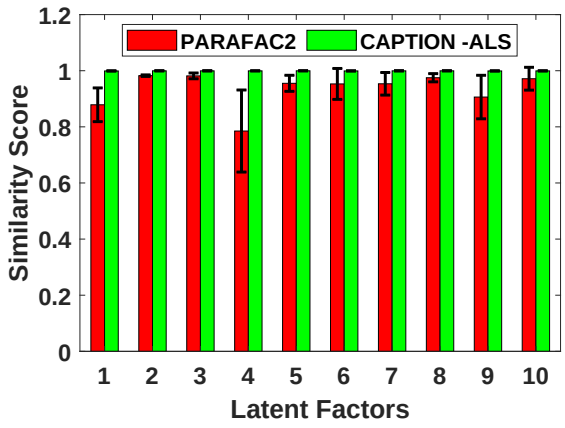


Figure 6.4: Identifiability analysis with and without coupling of PARAFAC2. Higher the value, better the identifiability.

a value less than the threshold, we consider it as an non-recovered factor. The Figure 6.4 shows that using coupled CP tensor data alongside PARAFAC2 data could help to alleviate the above discussed challenge and improve the identifiability of decomposition.

Q2. Scalability We also evaluate the scalability of our algorithm on synthetic dataset in terms of time needed for increasing load of input users (K). A PARAFAC2 tensors $\underline{\mathbf{X}} \in \mathbb{R}^{100\times100\times[1K-1M]}$ and CP tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{[1K-1M]\times[1K-1M]\times5}$ are decomposed with fixed target rank $R = 40$. The time needed by CAPTION increases very linearly with increase in non-zero elements. Our proposed method CAPTION-ALS, successfully decomposed the large coupled tensors in reasonable time as shown in Figure 6.5(a - b) and is up to 5× faster than baseline methods. Figure 6.5(c - h), we present the RMSE, and the computational time for the approaches under comparison for ML, Adobe and CMS data for increasing target rank from $5 - 50$. We remark that all methods achieve comparable RMSE values for three different data sets but proposed method CAPTION-ALS is up to average 2.5×

faster than baselines for all data sets. We remark the favorable scalability properties of CAPTION, rendering it practical to use for large tensors.

Q3. CAPTION at Work

Centers for Medicare and Medicaid (CMS) data files were created to allow researchers to gain familiarity using Medicare claims data while protecting beneficiary privacy. The CMS data contains multiple files per year. The file contains synthesized data taken from a 5% random sample of Medicare beneficiaries in 2008 and their claims from 2008 to 2010. We created CP tensor $\underline{\mathbf{Y}}$ from files that contain demographic characteristics (sex, race, state etc) of the beneficiary. The PARAFAC2 tensor $\underline{\mathbf{Y}}$ is created from files that has clinical variables such as chronic conditions. We decompose CP and PARAFAC2 tensor jointly with rank $R = 40$.

Model Interpretation: We propose the following model interpretation towards the target challenge:

- **Diagnosis feature factor:** Each column of factor matrix $\mathbf{V}$ represents a cluster and each row indicates a medical feature. Therefore an entry $\mathbf{V}(i, j)$ represents the membership of medical feature i to the j.
- **Irregular dimension factor (visits):** The r^{th} column of $\mathbf{U}_k$ presents the evolution of cluster r for all clinical visits for patient k.
- **Coupled factor (patients):** The coupled latent factor $\mathbf{W} = diag(\mathbf{S}_k)$ and CP latent factor $\mathbf{B}$, indicates the R communities of the patient.
- **Similarity factor:** The factor $\mathbf{C}$ indicates the importance of similarities membership which is responsible for creating clusters.

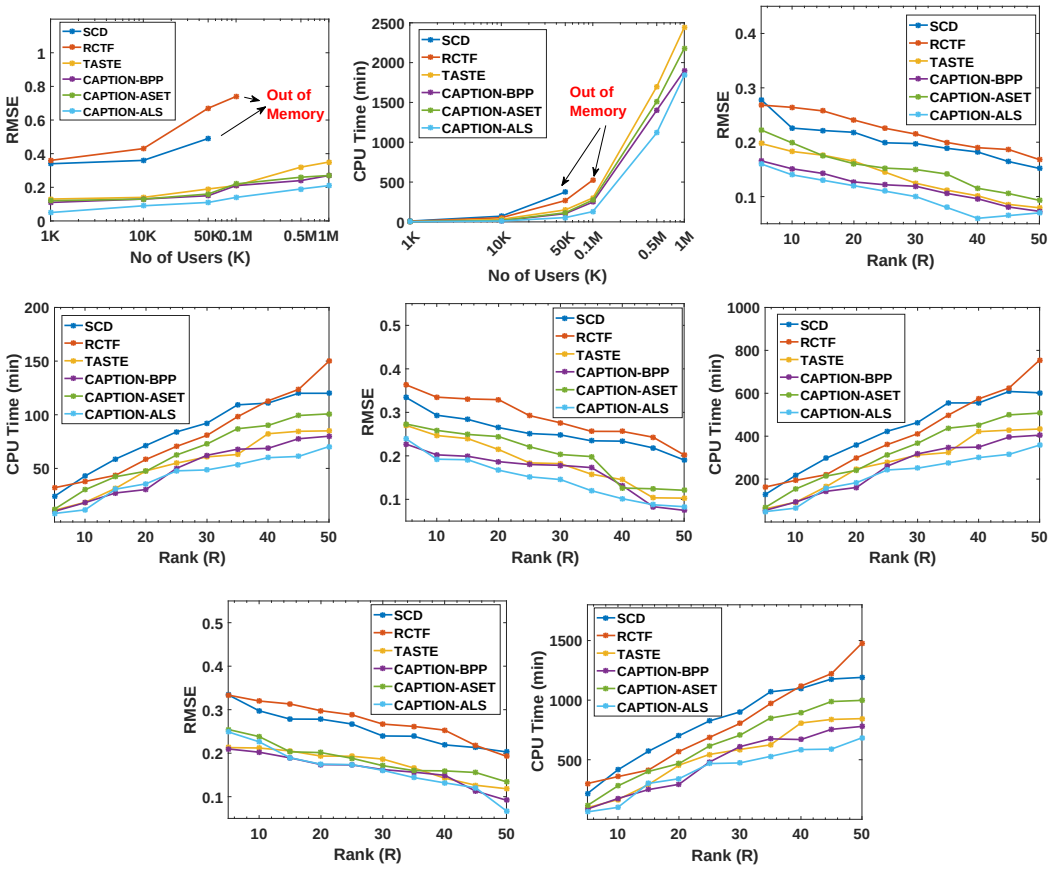


Figure 6.5: Scalability analysis of CAPTION method using synthetic and three real world datasets. (a-b): Scalability analysis of synthetic data with respect to varying number of users K, where K ranges $10^3 - 10^6$. Stable performance (RMSE) in the range $1K - 50K$, for most methods. Baseline method SCD and RCTF runs out of memory. (c-d): Scalability analysis with respect to Movielens dataset (c-d), Adobe dataset (e-f) and CMS heath record data (g-h). CAPTION-ALS significantly outperforms the other methods even when data is very sparse.

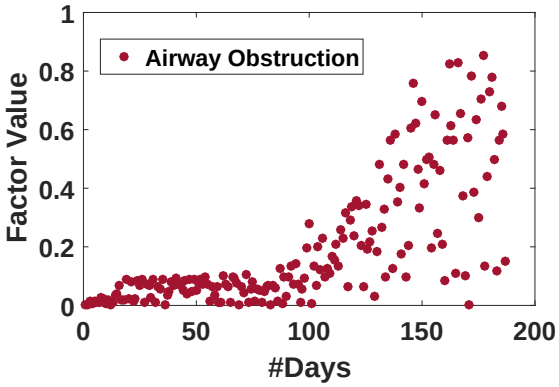


Figure 6.6: Visualization of time-frame captured of the patient no. 11426 created by CAPTION-ALS on CMS dataset.

Findings: In order to illustrate the use of CAPTION towards clustering, we focus our analysis on a subset of patients those are classified as medically complex. We observe that cluster number 32 has most patients with respiratory disease. These are the patients with high utilization ($>$ 50%), multiple clinical visits (avg 67) and high severity (death rate 8-10%). Most of the patients share ICD-9 code 492 (Emphysema), 496 (Chronic airway obstruction) and 511 (Pleurisy). These codes are characterized by obstruction of airflow that interferes with normal breathing. It is observed that these conditions frequently co-exists in real world and are hard to treat. In Figure 6.6, we provide time-frame captured by CAPTION-ALS for patient no. 11426 with chronic airway obstruction. In the patient's heath timeline, it shows that on the first few weeks visit, there is no sign of obstruction. The subsequent visits reflects a change in the patient's heath with a large number of diagnosis (day 84). Nevertheless, as shown in figure 6.5, CAPTION-ALS achieves significantly good performance in terms of RMSE (avg 60% better) and computation time (3$\times$ faster) by leveraging the coupling between CP and PARAFAC2 tensor data. .

6.4 Conclusion

This paper outlined our vision on exploring the coupling of CP and PARAFAC2 tensor decomposition using various optimization methods (BPP, ASET and ALS) to improve accuracy of factorization. We propose CAPTION, a framework that is able to offer interpretable results, and we provide a experimental analysis on synthetic as well as real world dataset. Extensive experiments with large synthetic dataset have demonstrated that the proposed method is capable of handling larger dataset for coupling for which most of the baselines are not able to performs due to lack of memory.

Furthermore, this paper outlines a set of interesting future research directions:

- How can we couple one of auxiliary tensor with irregular mode of PARAFAC2 tensor to obtain better approximation?
- What other constraints, other than non-negative, for the CAPTION are well suited for various application and have potential to offer more accurate results?
- How can we use coupling for incremental tensor data?

Chapter based on material published in ASONAM 2020 [88].

Chapter 7

Niche Detection in User Content Consumption Data

"How do we know what types of content a certain market likes and prioritizes?Given a particular type of content (e.g "food related" videos), which markets find it the most attractive?"

Explainable machine learning methods have attracted increased interest in recent years. In this work, we pose and study the *niche detection* problem, which imposes an explainable lens on the classical problem of co-clustering interactions across two modes. In the niche detection problem, our goal is to identify *niches*, or co-clusters with node-attribute oriented explanations. Niche detection is applicable to many social content consumption scenarios, where an end goal is to describe and distill high-level insights about user-content associations: not only that certain users like certain types of content, but rather the *types* of users and content, explained via node attributes. Some examples are an e-commerce

platform with who-buys-what interactions and user and product attributes, or a mobile call platform with who-calls-whom interactions and user attributes. Discovering and characterizing niches has powerful implications for user behavior understanding, as well as marketing and targeted content production. Unlike prior works, ours focuses on the intersection of explainable methods and co-clustering. First, we formalize the niche detection problem and discuss preliminaries. Next, we design an end-to-end framework, NED, which operates in two steps: *discovering* co-clusters of user behaviors based on interaction densities, and *explaining* them using attributes of involved nodes. Finally, we show experimental results on several public datasets, as well as a large-scale industrial dataset from Snapchat, demonstrating that NED improves in both co-clustering ($\approx 20\%$ accuracy) and explanation-related objectives ($\approx 12\%$ average precision) compared to state-of-the-art methods. The content of this chapter is adapted from the following paper:

Ekta Gujral, Neil Shaw, Leonardo Neves, and Evangelos E. Papalexakis, "NED: Niche Detection in User Content Consumption Data". The content of this chapter was under blind peer review at the time of thesis submission.

7.1 Introduction

Developing features that attract and appeal to customers is critical to companies, as it determines their success and revenue [86]. Recently, platforms that create, serve, and curate content such as Snapchat, Netflix, and Youtube use data-driven approaches to attract and maintain a large and diverse userbase. This approach has successfully promoted mass engagement and user traction on popular content via recommendation systems: the larger

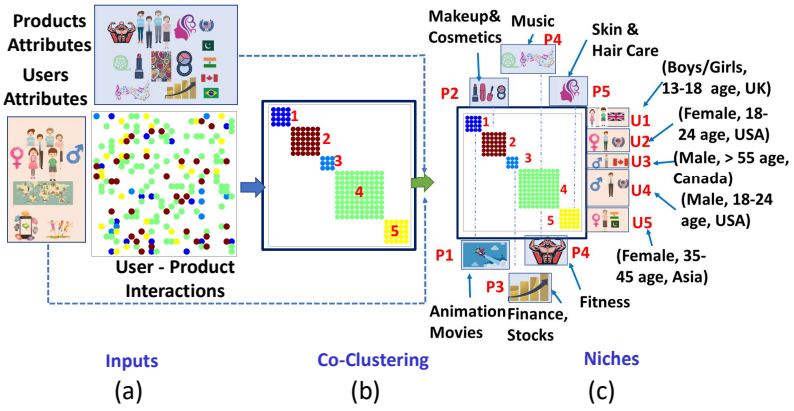


Figure 7.1: Our niche detection framework, NED: (a) takes as inputs user-content interactions, user attributes, and content attributes, (b) mines coherent co-clusters from interaction data, and (c) outputs niches, or co-clusters imbued with concise, user and content attribute-oriented explanations.

the number of users that interact with a piece of content, the more that piece of content tends to be promoted. While this model is successful, engineering the process of content creation to generate this level of appeal by automating the process of identifying market *niches*, is quite challenging.

Specifically, in creating content, it is beneficial to understand the attributes that make a particular type of content (e.g food-related videos) attractive to specific markets that might be underserved by the platform (e.g. 18-24 year old women in Australia). Thus, to improve targeting and to create and promote content that will likely better retain, engage, and satisfy target audiences, we propose a method for self-explaining *niche detection* in user content consumption data. Our work characterizes niches as outstanding co-clusters in user-content interaction graph data, imbued with user and content-oriented attributed co-cluster explanations which designate audience and content types. At its heart, our work addresses

the scenario: *Consider that we have a rich user-liked-content interaction graph, and nodal attributes describing the users (e.g. user profile, demographics information, device types) and content (e.g. creator profile, category), how can we explain and make sense of it?*

Our work lies at the intersection of co-clustering and explainable machine learning, though the problem we aim to solve is one that other works do not. Co-clustering is a method that aims to simultaneously cluster both users and content (interchangeably, products) simultaneously so that users and content can be organized into homogeneous blocks. This approach is helpful to partially answer our research question. In the literature, many co-clustering methods [109, 82, 23] have been proposed. For instance, [59, 23] take information theoretic approaches to derive co-clusters by maximizing preserved mutual information and entropy between users and items. More recently, [268, 269] utilize deep autoencoders to generate low-dimensional representations for users and products, and employ Gaussian Mixture Models to infer the co-clusters. Several works have also employed Non-Negative Matrix Tri-Factorization (NMTF) [85, 221, 185], turning the co-clustering problem into a matrix approximation problem [101]. Although these methods can be utilized to identify outstanding co-clusters, they are inherently limited in their capacity to *explain* the interactions. This is primarily because they cannot leverage user or item attributes to explain the discovered co-clusters, and thus have difficulty summarizing interactions between diverse user and item groups concisely. Explainability needs are paramount for the task we aim to solve in practice: ultimately, any discovered data-driven insights must be acted upon by creators to dictate a content creation strategy. Moreover, while explainability has become a focal point of recent works in machine learning [73, 133], there is no prior work on imbuing

co-clusters with explanations, which is an important facet of the niche detection problem we propose in this work.

Summarily, addressing the niche detection problem has clearly high value for content strategists and platforms, but it poses several notable challenges. Firstly, it requires powerful and performant, large-scale co-clustering, to discover coherent and outstanding statistical regularities in user-content interaction data. Secondly, it requires a component to leverage nodal attributes available on users and content to concisely explain the associations between users and content which a co-cluster is characterized by, in a fashion interpretable to humans who must act as decision-makers on the basis of the explanations. The output of such a method may deliver informative insights from crude user-content interaction data, such as "female users generally enjoy skin-care related content more than others" or "18-24 male users enjoy bodybuilding content more than others," which can be acted upon and guide creators and strategists. More importantly than discovering popular or globally intuitive patterns, the methodology of a successful niche detection framework offers the promise of enabling the discovery of smaller, underserved and non-apparent niches which can inform highly localized and tailored content solutions for a subgroup. In this work, we formally pose the niche detection problem, and propose a framework, NED, to tackle it by carefully designing the two components: Figure 7.1 illustrates the high-level process. We further conduct extensive experiments demonstrating NED's success in discovering coherent niches, and outperformance of alternative approaches.

In short, our contributions are:

Symbols	Definition
$\mathbf{X}$, $\mathbf{x}$, x	a matrix, column vector and scalar
$\mathbf{A}, \mathbf{B}, \mathbf{S}$	user cluster, product cluster and summary matrix
$\mathbf{U}_f, \mathbf{P}_f$	user attributes matrix, product attributes matrix
$\mathbf{U}_{p_f}, \mathbf{P}_{u_f}$	user to product attributes matrix, product to user attributes matrix
I, J, F_1, F_2	#users, #products, #user features, #product features,
M, N	No of user clusters, No of product clusters

Table 7.1: Table of symbols and their descriptions

- **Novel Problem Formulation**: We formally pose the niche detection problem for user behavior modeling. The problem is guided by two sub-problems: discovering co-clusters of user behaviors based on interaction densities, and explaining them using attributes of involved nodes via feature learning.
- **Novel Algorithm**: We propose an intuitive, simple, efficient, and effective method, NED, a niche detection framework, which proposes mutual information-inspired variant of NMTF for co-clustering, and a mutual information-inspired solution for explainable feature selection. To our knowledge, NED is the first approach at explainable co-clustering. Moreover, to evaluate niche quality, we propose a compression-based score to bridge the detected co-clusters and the feature explanations.
- **Extensive Experiments**: We evaluate NED on multiple synthetic and publicly available real-world datasets, as well as a private, large-scale sparse dataset with $500K$ users, 2500 products and $> 5M$ interactions from Snapchat, showing $\approx 14\%$ accu-

racy improvement against state-of-the-art co-cluster approaches and $\approx 20\%$ average precision improvement in explanation quality.

Our implementation and small synthetic data is publicly available[1].

7.2 Related work

Although several co-clustering and explainable machine learning methods have been previously developed, none of them produce co-clusters with node-attribute oriented explanations as ours does. We thus discuss tangent prior work in co-clustering and explainable machine learning.

7.2.1 Co-clustering

Most prior clustering literature has focused on one-sided clustering algorithms like k-means and its parameter-free variants [71, 147], spectral [285, 92], and probabilistic clustering [211, 281]. Our problem deals with simultaneous clustering of rows and columns, known as bi(dimensional)-, co-, or block clustering [45, 174] that can be categorized into following three main categories: information-theoretic, decomposition-based, and neural methods, which we discuss below.

[59] introduced a co-clustering that utilizes a lossy coding scheme to co-cluster a two dimensional joint probability distribution, and it requires the number of clusters as input. [74] proposes a parameter-free and a fast information-theoretic agglomerative co-clustering method. [266] proposed a co-clustering method that allow rows and column clusters to overlap with each other. [44] developed a hierarchical structure for rows and

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

columns with a minimum number of leaf clusters to realize co-clustering. Most recently, [214] proposed improvements to [59].

[85] proposed two regularization terms to use the geometric structure data of data graph and feature graph separately when using semi-NMTF decomposition. [48] proposed a fast approach to constrain factor matrices to be cluster indicator matrices. [226] proposed a co-clustering method from the perspective of dynamical synchronization. [185] proposed a method for learning a bipartite graph with explicit linked components by imposing a rank constraint on the Laplacian matrix. Most recently, [221] proposed a word co-occurrence NMTF method that leverages mutual information for co-clustering words and documents.

[268] used the deep auto-encoder to map data to a low-dimensional space and then minimize the KL difference between cluster assignments and an auxiliary distribution distribution. Most recently, [269] proposed a deep co-clustering model, DeepCC, which used a deep auto-encoder to generate low-dimensional representations for rows and columns and used GMM framework for cluster assignment prediction.

Unlike our work, none of these methods can explain associations in the produced co-clusters. Moreover, NED*'s co-clustering component outperforms the previously proposed works in NMI and accuracy on both simulated and real datasets.*

7.2.2 Explainable Machine Learning

In the last decade, explainable machine learning has gained considerable attention. Prior work shows that the ability of intelligent systems to justify their choices is important for their successful use; when users do not understand the decisions of an intelligent system, they become cynical and unwilling to use it, despite improved performance [84, 132]. Several

works aim to explain complex predictive models with simple rule-based explanations; rule-based explanations [169, 37, 64, 170] and deep learning based explanations [210] have been a popular approach to explain black-box models. However, these methods are often tailored to the specifics of the model which is being explained.

Recently, another line of work has focused on explaining predictions of complex models in terms of the importance of features in the classification. [240] proposed an ablation-style approach which removes all possible subsets of features and evaluates changes in predictions. However, such combinatorial approaches are computationally expensive. [165] improved the computations described in [240] and proposed efficient method to interpret model prediction using weights on features, representing their relative contribution on the prediction using Shapley values, and effectively applying to any downstream classification model. Tree-based ensemble methods [164] such as random forests [163] and gradient boosted trees [16, 133] achieve state-of-the-art performance in many domains. They have a successful history of use in machine learning, and new high-performance implementations are an active area of research [133, 215, 42]. Such models often outperform standard deep models [229] on datasets where features are individually meaningful and do not have strong temporal or spatial structures [42]. Most recently, [133] proposed LightGBM to enhance the performance of tree-based models.

Unlike our work, none of these methods aim to explain co-clusters, and rather focus on traditional supervised learning. Moreover, NED *'s explainability component outperforms feature explanations from the recent state-of-the-art LightGBM in terms of stability score, average precision, and compression ratio.*

7.3 Problem Formulation

Table 7.1 contains the symbols used throughout the chapter. Before we conceptualize the niche detection problem that our work tackles, we define certain terms necessary to set up the problem and formally define the problem statement.

7.3.1 Problem Context

We assume input data that consists of:

- A set of users and products (interchangeably, contents) represented by $\mathcal{U} = \{U_1, \ldots, U_I\}$ and $\mathcal{P} = \{P_1, \ldots, P_J\}$ respectively. The relationship between entities consists of: user-product interactions containing tuple of the form (U_i, P_j, x_{ij}), where $U_i \in \mathcal{U}$ and $P_i \in \mathcal{P}$ and $x_{ij} \in \{0, 1\}$, and arranged in the form of binary matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, and zero entries indicate absent interactions.
- A set of user features, arranged in a binary matrix $\mathbf{U}_f \in \mathbb{R}^{I \times F_1}$, where F_1 represents total number of user features, and zero entries indicate absent features.
- A set of product (or content) features stored in a binary matrix $\mathbf{P}_f \in \mathbb{R}^{J \times F_2}$, where F_2 represents total number of product features, and zero entries indicate absent features.

The eventual goal in solving the niche detection problem is the capacity to discover co-clusters with user and product attribute-oriented explanations. The problem can be decomposed into two subgoals: first, identifying quality co-clusters, and second, explaining the co-clusters via careful node feature selection.

7.3.2 Problem Statement

Next, we introduce a few basic definitions necessary to define our novel niche detection problem.

Definition 17 (Co-cluster) *Formally, given a $I \times J$ data matrix $\mathbf{X}$, a co-clustering can be defined by two maps ρ and γ, which groups users (or rows) and products (or columns) of $\mathbf{X}$ into M and N disjoint or hard clusters respectively. Specifically,*

$$\rho : \{U_0, U_1, ..., U_I\} \rightarrow \{\hat{U}_1, \hat{U}_2, ..., \hat{U}_M\} \tag{7.1}$$

$$\gamma : \{P_0, P_1, ..., P_J\} \rightarrow \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_N\} \tag{7.2}$$

where $\rho(U) = \hat{U}$ denotes that user U is in user cluster $\hat{U}$, and $\gamma(P) = \hat{P}$ denotes that product P is in product cluster $\hat{P}$. A co-cluster is an interaction block defined by $\{\hat{U}_m, \hat{P}_n\}$ for some $m < M, n < N$.

Definition 18 (Co-cluster Explanation) *A co-cluster explanation predicates the existence of binary feature matrices for users and products, $\mathbf{U}_f$, and $\mathbf{P}_f$ respectively. Let $sub(\mathbf{U}_f)$ and $sub(\mathbf{P}_f)$ respectively denote some subset of the F_1 user and F_2 product features, without loss of generality. An explanation for a co-cluster is indicated by a suitable pair $(sub(\mathbf{U}_f), sub(\mathbf{P}_f))$, which are associated with the co-cluster.*

Definition 19 (Niche) *A niche is the pairing of a co-cluster with a co-cluster explanation. Formally, a niche is indicated by*

$$\{\hat{U}_m, \hat{P}_n, sub(\mathbf{U}_f), sub(\mathbf{P}_f)\} \tag{7.3}$$

where $\hat{U}_m$ is m^{th} user cluster, $\hat{P}_n$ is the n^{th} product cluster, and $sub(\mathbf{U}_f)$ and $sub(\mathbf{P}_f)$ are subsets of user and product features.

We now have all the necessary definitions to formally define our problem. Hence, we pose the following:

Given (a) a set of users $\mathcal{U}$ and products $\mathcal{P}$ with user-product interactions $\mathbf{X}$, (b) user-feature relationships $\mathbf{U_f}$, and (c) product-feature relationships $\mathbf{P_f}$;

Design a framework to identify one or more coherent niches of the form in Equ. 7.3.

Note that our framework can apply to any rows/columns but we use users/products for ease of explanation.

7.4 Proposed Method: NED

Our proposed approach relies on two successive steps. First, the *co-clustering step* that co-clusters the user-product interaction data matrix and second, the *explaining step*, which focuses on learning the set of features that suitably characterize high-quality co-clusters discovered in the previous step. A two-step method is beneficial for two reasons: first, the *driving* features of the generative process may be missing in the observed data in lieu of strong correlates, in which case we may not want to try and infer a misleading process directly from these correlates; its sufficient in our case to try to identify strong correlative, instead of non-causal relationships. Second, with co-clustering independent of the features, we avoid *missed cluster* scenarios where a joint generative process may not identify a co-cluster simply because it is composed of a mix of features that the process is not sufficiently capable of capturing. By decoupling clustering and explanation, we prioritize recall on observed interaction clusters, acknowledging that some may be hard to adequately explain, but exist nonetheless. Although any co-clustering algorithm can be used in solving the

niche detection problem, we propose a variant of NMTF guided by mutual information as one suitable solution. In fact, through extensive experiments (see Section 7.5.4), we show that our proposed method is not only intuitive, but achieves state-of-the-art co-clustering performance compared to previously proposed methods in literature.

7.4.1 Step 1: Co-Clustering

We frame the co-clustering problem (Defn. 17) in an NMTF-inspired formulation: we seek a decomposition of the user-product matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ into three low dimensional non-negative [161, 221] latent factor matrices i.e. $\mathbf{A} \in \mathbb{R}_{+}^{I \times M}$ is user-clustering matrix, $\mathbf{B} \in \mathbb{R}_{+}^{J \times N}$ is product-clustering matrix and $\mathbf{S} \in \mathbb{R}_{+}^{M \times N}$ provides the summary of $\mathbf{X}$ due to co-clustering. The values M and N represent the number of user and product clusters, respectively ($M << \min(I, J)$ and $N << \min(I, J)$). The optimization problem can be represented as:

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \mathbf{A}, \mathbf{B}, \mathbf{S}) = \min_{\mathbf{A},\mathbf{B},\mathbf{S}} ||\mathbf{X} - \mathbf{A}\mathbf{S}\mathbf{B}^T||_2^F \\ s.t. \quad \mathbf{A} \geq 0, \quad \mathbf{B} \geq 0, \quad \mathbf{S} \geq 0 \end{aligned} \tag{7.4}$$

In this way, users and products are clustered simultaneously while satisfying constraints, keeping a good low-rank approximation.

Factor Inference

Here, we derive an alternating optimization algorithm that infer the latent factor matrices from the user-product interaction $\mathbf{X}$. The Equ. 7.4 can re-written as:

$$\begin{aligned} \mathcal{L} =& \frac{1}{2} Tr((\mathbf{X} - \mathbf{A}\mathbf{S}\mathbf{B}^T)(\mathbf{X} - \mathbf{A}\mathbf{S}\mathbf{B}^T)^T) \\ =& \frac{1}{2} = \frac{1}{2} Tr(\mathbf{X}\mathbf{X}^T - 2\mathbf{X}\mathbf{B}\mathbf{S}^T\mathbf{A}^T + \mathbf{A}\mathbf{S}\mathbf{B}^T\mathbf{B}\mathbf{S}^T\mathbf{A}^T) \end{aligned} \tag{7.5}$$

Next, as in the bilateral k-means algorithm [100], we derive iterative update rules for $\mathbf{A}$ and $\mathbf{B}$ under the non-negativity constraints. The update rule of $\mathbf{A}$ is given as:

$$\mathbf{A}_{im} = \begin{cases} 1 & m = \text{argmax}_i(\hat{\mathbf{A}}(i,:)), \quad \hat{\mathbf{A}}(i,:) = [\mathbf{X}\mathbf{B}\mathbf{S}^T](i,:) \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

There is only one element equal to 1 at m^{th} column and the rest are zeros in each i^{th} row of $\mathbf{A}$. Similarly, $\mathbf{B}$ can be updated as:

$$\mathbf{B}_{jn} = \begin{cases} 1 & n = \text{argmax}_j(\hat{\mathbf{B}}(j,:)), \quad \hat{\mathbf{B}}(j,:) = [\mathbf{X}^T\mathbf{A}\mathbf{S}](j,:) \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

As we impose constraints on user ($\mathbf{A}$) and product ($\mathbf{B}$) latent factors, the summary matrix $\mathbf{S}$ could be noisy since it is not optimized with any given criterion. It could represent an unclear structure (due to data noise, and high overlap among the categories represented by the clusters) where either there is no correlation between clusters, or every cluster is associated with other clusters. In order to mitigate the above issue, we compute the summary matrix $\mathbf{S}$ as positive point-wise mutual information (PPMI) matrix, which can extract clearer co-clusters and exploits its background knowledge for further convergence of the algorithm. The PMI is a theoretical measure of information widespread used to measure the association between pairs of results that arise from discrete random variables. In the literature [183], it is shown that this measure is highly co-related to conditional probability and resembles human judgment. Mathematically, the PMI between two random variable u and v is given by:

$$PMI(u,v) = \log\left(\frac{p(u,v)}{p(u)p(v)}\right) \tag{7.8}$$

Thus, we compute $\mathbf{S} \in \mathbb{R}_{+}^{M \times N}$ as follows:

$$\mathbf{S} = \log\Big(\frac{\mathbf{A}^T\mathbf{X}\mathbf{B}}{\sum_{j=1}^{J}\mathbf{A}^T\mathbf{X}\sum_{i=1}^{I}\mathbf{X}\mathbf{B}}\Big) \quad s.t. \quad \mathbf{S} \geq 0 \tag{7.9}$$

So, each element of $\mathbf{S}_{m,n}$ represents the PMI between a user cluster $\mathbf{A}_m$ and a product cluster $\mathbf{B}_n$. Each positive value of the matrix has surely to be considered in the identification of co-cluster.

Why PMI?: The intuition behind computing $\mathbf{S}$ using PMI comes from the observation that sometimes, in the co-clustering of user-product data, a user cluster is associated with another product cluster that does not exist. This leads to the idea of an update rule based on co-occurrence between users and products for all cluster pairs in the given data. PMI is an information-theoretic approach that measures how often two clusters $(\mathbf{A}_m, \mathbf{B}_n)$ occur as compared with what we expect if they were independent. The numerator of PMI informs us how often we observed the two clusters together in user-product context consumption. The denominator informs us how often we would anticipate both to co-cluster, assuming they are independent clusters. Thus, the ratio provides us an estimation of how much more the two clusters co-cluster than we anticipate by chance. Our choice for PMI is encouraged by [241]. Using PMI, we encourage users of similar interactions with products to have closer representation in latent space. For example, when representing co-clusters, we can easily think of positive relationship (e.g. "Female" and "Nurse") but find it much harder to relate negative ones ("Female" and "Carpenter"). Therefore, we focus on positive point-wise mutual information. Next, Equ. (7.6) and (7.7) can be re-written as:

$$\mathbf{A} = \mathbf{X}\mathbf{B}\log\Big(\frac{\mathbf{A}^T\mathbf{X}\mathbf{B}}{\sum_{i=1}^{I}\mathbf{A}^T\mathbf{X}\sum_{i=1}^{I}\mathbf{X}\mathbf{B}}\Big)^T \tag{7.10}$$

$$\mathbf{B} = \mathbf{X}^T\mathbf{A} \log\left(\frac{\mathbf{A}^T\mathbf{X}\mathbf{B}}{\sum_{j=1}^{J}\mathbf{A}^T\mathbf{X}\sum_{i=1}^{I}\mathbf{X}\mathbf{B}}\right) \tag{7.11}$$

Now, we assign each user/product to a single cluster by finding the cluster with maximum membership. This translates to finding the maximum column index for each row.

7.4.2 Step 2: Co-Cluster Explanation

After discovering outstanding co-clusters, we next aim to identify the user and product feature subsets that best explain the co-clusters. The process involves two steps: auxiliary feature matrix creation to infer the user preferences over product features to get insights about implicit user similarities, and feature selection using point wise mutual information to compute the association between features and user/product cluster centroids. Through experiments we show that the auxiliary feature matrix helps to improve the explainability of our proposed method (see Section 7.5.6). The quest for similarities plays an important role in co-cluster analysis [244]. In quadripartite graphs (see Figure 7.2), one can derive several semantics on similarity by considering different paths in a graph. Upon deriving these, we have all information to learn important co-cluster features: the final suitability score for each feature is captured as a linear combination of both steps. We briefly explain each step as follow:

Auxiliary Feature Matrix

We have the user and product feature vectors ($\mathbf{U}_f \in \mathbb{R}^{I\times F_1}$ and $\mathbf{P}_f \in \mathbb{R}^{J\times F_2}$) which are independent and do not infer any user preferences for the features that appear in products and vice-versa. This could lead to undermining the user/product similarities

for feature learning. To overcome this issue, we compute the user's proximity to a product feature through a meta-path as an indication of the user's possible "preference" towards the product feature and vice versa. The meta-path [244] is a powerful mechanism for a user to select an appropriate similarity semantics to learn from a set of examples of similar objects. Formally, a meta-path can be defined as:

Definition 20 (Meta-path) *A meta-path is a sequence of relations $\mathcal{R}$ between object types $\mathcal{O}$, which defines a new composite relation between its starting type and ending type. It is denoted in the form of $\mathcal{O}_1 \xrightarrow{\mathcal{R}_1} \mathcal{O}_2 \xrightarrow{\mathcal{R}_2} \dots \xrightarrow{\mathcal{R}_{m-1}} \mathcal{O}_m$, which defines a composite relation between $\mathcal{R}_1 \circ \mathcal{R}_2 \circ \cdots \circ \mathcal{R}_m$ between types $\mathcal{O}_1$ and $\mathcal{O}_m$, where $\circ$ denotes the composition operator on relation. For example, a meta-path (Figure 7.2, purple path) user (male) $\rightarrow$ product (movie) $\rightarrow$ feature (action) $\rightarrow$ product (movie) $\rightarrow$ feature (comedy) (denoted as UPFPF) indicates user preferences based on content similarities.*

The process of auxiliary feature matrix creation is adapted from [142, 244]. In [142], movie preferences are learned by leveraging user similarities defined through different types of meta paths or relations to successfully design a new movie. Here, the auxiliary feature matrix helps to leverage both explicit features and user/product similarities via a graph-theoretic approach. For user auxiliary feature matrix as shown in Figure 7.2, the red path $\mathbf{U}_{p_f}^{UPF}$ (i.e., starting from a user and ending on a product feature via a product) finds the preferences for the product features for each user based on its interactions. The blue path $\mathbf{U}_{p_f}^{UPUPF}$ finds user preferences for the product features based on user-user similarity, and the purple path $\mathbf{U}_{p_f}^{UPFPF}$ finds user preferences based on product-product (content) similarity.

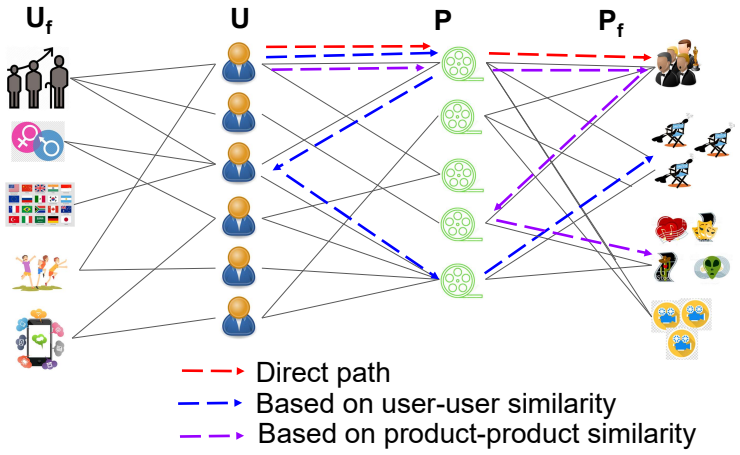


Figure 7.2: Quadripartite graph of the users $\mathbf{U} \in \mathbb{R}^I$, Product $\mathbf{P} \in \mathbb{R}^J$ and the user features $\mathbf{U}_f \in \mathbb{R}^{I \times F_1}$ and product features $\mathbf{P}_f \in \mathbb{R}^{J \times F_2}$. The resultant user auxiliary feature matrices is $\mathbf{U}_{p_f} \in \mathbb{R}^{I \times F_2}$.

For the final weighted matrix $\mathbf{U}_{p_f} \in \mathbb{R}^{I \times F_2}$, which represents user $\mathbf{U}$ preference over product features $\mathbf{P}_f$ is a linear combination of the weighted $\mathbf{U}_{p_f}$ over the three predefined meta-path types:

$$\mathbf{U}_{p_f} = \alpha \mathbf{U}_{p_f}^{UPF} + \beta \mathbf{U}_{p_f}^{UPUPF} + \gamma \mathbf{U}_{p_f}^{UPFPF} \tag{7.12}$$

This linear combination helps to smooth the information in case of sparse direct user-product preferences. Similarly, we compute weighted auxiliary matrix for product features as:

$$\mathbf{P}_{u_f} = \alpha \mathbf{P}_{u_f}^{PUF} + \beta \mathbf{P}_{u_f}^{PUPUF} + \gamma \mathbf{P}_{u_f}^{PUFUF} \tag{7.13}$$

where $\alpha, \beta, \gamma \in \mathbb{R}^+$ are combination parameters satisfying criterion $\alpha + \beta + \gamma = 1$. We set $\alpha = 0.5$, $\beta = 0.25$ and $\gamma = 0.25$ in both cases. We give higher importance to direct user/product preferences (α) and lower importance to 4-step path (β, γ) because it could

'obscure' the individual preferences by depending on 'similar' users/products. Section 7.5.7 shows sensitivity analysis for these parameters.

Feature Selection

For feature selection, we propose a PMI-based approach that leverages the association between user and product cluster centroids. We compute information about how often we observed the certain features for each user cluster using $\mathbf{A}$ and $\mathbf{U}_f$ as $\frac{(\mathbf{A}^T\mathbf{U}_f)_{ij}}{\sum_{i=1}^{I}\sum_{j=1}^{J}(\mathbf{A}^T\mathbf{U}_f)_{ij}} \in \mathbb{R}^{M\times F_1}$. Next, we compute information about their association while independently drawn as $p(\mathbf{U}_f) = \frac{\sum_{j=1}^{J} U_f}{\sum_{i=1}^{I}\sum_{j=1}^{J}(\mathbf{A}^T\mathbf{U}_f)_{ij}}$ and $p(\mathbf{A}) = \frac{\sum_{i=1}^{I} A}{\sum_{i=1}^{I}\sum_{j=1}^{J}(\mathbf{A}^T\mathbf{U}_f)_{ij}}$.

Finally, we compute PMI relation as:

$$\mathbf{U}^1_{PMI} = \log \frac{p(\mathbf{A}^T\mathbf{U}_f)}{p(\mathbf{U}_f)p(\mathbf{A})} = \frac{\mathbf{A}^T\mathbf{U}_f * \sum_{i=1}^{I}\sum_{j=1}^{J}(\mathbf{A}^T\mathbf{U}_f)_{ij}}{\sum_{j=1}^{J}\mathbf{U}_f^{\mathbf{T}} * \sum_{i=1}^{I}\mathbf{A}} \tag{7.14}$$

where $\mathbf{U}^1_{PMI} \in \mathbb{R}^{M\times F_1}$. Now, we compute PMI relation for user auxiliary features matrix $\mathbf{U}_{p_f}$ as:

$$\mathbf{U}^2_{PMI} = \log \frac{p(\mathbf{A}^T\mathbf{U}_{p_f})}{p(\mathbf{U}_{p_f})p(\mathbf{A})} \in \mathbb{R}^{M\times F_2} \tag{7.15}$$

Similarly, we compute both PMIs ($\mathbf{P}^1_{PMI} \in \mathbb{R}^{N\times F_2}$ and $\mathbf{P}^2_{PMI} \in \mathbb{R}^{N\times F_1}$) for product features also. Due to space limitations and symmetry, we do not include derivations for them.

To select the most relevant user and product features for the niche, we linearly combine the PMIs associated with co-clustering (i.e. via summary matrix $\mathbf{S}$), attribute matrices (via. Equ. 7.14), and auxiliary matrices (via. Equ. 7.15) as following:

$$\textbf{For users:} \quad \mathbf{e}^u = \mathbf{U}^1_{PMI} + \mathbf{S} * \mathbf{P}^2_{PMI} \in \mathbb{R}^{M\times F_1} \tag{7.16}$$

$$\textbf{For products:} \quad \mathbf{e}^p = \mathbf{P}^1_{PMI} + \mathbf{S}^T * \mathbf{U}^2_{PMI} \in \mathbb{R}^{N \times F_2} \tag{7.17}$$

Finally, we choose the top N highest values for each cluster (or each row) from $\mathbf{e}^u$ (Equ. 7.16) and $\mathbf{e}^p$ (Equ. 7.17) to explain the niche.

7.5 Experiments

In this section, we aim to answer the following research questions:

- **RQ1 Accuracy**: Can NED outperform state-of-the-art alternatives at effectively capturing co-clusters?
- **RQ2 Explainability**: Can NED help learn meaningful explanations of co-clusters, and thus better niches?
- **RQ3 Scalability** How efficient and scalable is NED with respect to the size of the input graphs?

We discuss these after detailing our experimental setup.

7.5.1 Datasets and Experiment Setup

The details of the synthetic and the real data used for experiments are given in Table 7.2.

Synthetic Data

In order to fully control and evaluate the niches in our experiments, we generate synthetic data i.e. user-product engagement graph $\mathbf{X} \in \mathbb{R}^{I \times J}$, user attributes $\mathbf{U}_f \in \mathbb{R}^{I \times F_1}$ (e.g. age, gender, country, app engagement etc.) and product attributes $\mathbf{P}_f \in \mathbb{R}^{J \times F_2}$ (e.g.

Dataset	#users	#products	#features (users, products)	#clusters (users, products)
Syn-I	$10K$	$1K$	$(22, 43)$	$(14, 20)$
Syn-II	$50K$	$5K$	$(22, 55)$	$(70, 35)$
Syn-III	$500K$	$10K$	$(22, 63)$	$(140, 50)$
Syn-IV	$1M$	$50K$	$(22, 86)$	$(280, 70)$
Cora	$3K$	$1.5K$	–	$(7, -)$
WebKB4	$4K$	$1K$	–	$(4, -)$
MovieLens	$6K$	$4K$	$(25, 23)$	$(-, 20)$
News20	$19K$	$61K$	–	$(20, -)$
Caltech	$2K$	300	–	$(3, -)$
Snapchat	$500K$	$2.5K$	$(22, 238)$	$(-, -)$

Table 7.2: Details for the datasets. "-" indicates unknown/unavailable public information. name, publisher name, country, category,subcategory etc.) as discussed in Supplementary Material.

Real Data

We employ several real-world public datasets from different domains: *Cora* is publications dataset, *WebKB4* [2] consists of classified web page information, *MovieLens* [3]has data has 6000 users and 4000 movie rating information, *News20* [4] is a collection of approximately 20,000 newsgroup documents, *Caltech* [146] is an image dataset. We also use private *Snapchat* dataset containing full-duration views between users and public feed

[2] http://membres-lig.imag.fr/grimal/data.html
[3] https://grouplens.org/datasets/movielens/
[4] http://qwone.com/~jason/20Newsgroups/

contents from the Snapchat platform; we assume full views to indicate persisted interest in the consumed content, as in [149, 130].

7.5.2 Baseline Methods

The two major components of NED are in (a) the discovering of coherent co-clusters, and (b) their concise explanation via nodal attributes. Thus, we conduct experiments with two categories of baseline to evaluate each contribution separately.

Co-clustering

We compare with recent, state-of-the-art co-clustering approaches:

- **NEO-CC** [266]: A non-exhaustive overlapping co-clustering method based on the minimum sum-squared residue objective.
- **DeepCC** [269]: A deep autoencoder based co-clustering method which employs a variant of Gaussian Mixture Model (GMM) to infer cluster assignments.
- **CoClusInfo** [214]: An information-theoretic approach which uses mutual information to define its objective function.
- **FNMTF** [48]: A fast, NMTF method based on projected gradients, coordinate descent, and alternating least squares optimization.
- **WC-NMTF** [221]: A word co-occurrence NMTF method that leverages mutual information for co-clustering the word and documents.
- **SNCC** [162]: a sparse neighbor constrained co-clustering with dual regularizers for learning category consistency.

Note that in [221], **WC-NMTF** method performed better than original NMF [270], orthogonal NMF (ONMF) [279], projective NMF (PNMF) [280], graph regularized NMF (GNMF) [32], NMTF [161], orthogonal NMTF (ONMTF) [60] and graph regularized NMTF (GNMTF) [225]. Similarly, in [162], paper evaluated performance of **SNCC** against K-means [158], NMF [270], SNMF [61], graph regularized NMF (GNMF) [32], dual regularization NMTF (DNMTF) [225], dual local learning co-clustering (DLLC) [262] and structured optimal bipartite graph (SOBG). Therefore, to avoid the repetitive comparison, we chose to compare our proposed method's performance with **SNCC** and **WC-NMTF**.

Explainability

Although we could not find any explicit co-clustering explainability baselines, we adapted the recently proposed **LightGBM** [133] for our explainability baseline. It is a boosting decision tree-based method that employs feature bundling to deal with a large number of features. To use it as our baseline, we fed co-clustering outcomes from the above-discussed method as labels for the user and product clusters along with user/product features data matrices to discern feature importance per co-cluster. We also compared our method with recently proposed method **BMGUFS** [22] which is rigorous approximation algorithms for block model guided unsupervised feature selection and helps in finding high-quality features for cluster explanation.

Dataset	Cluster	Metric	NEO-CC	DeepCC	CoClusInfo	FNMTF	WC-NMTF	SNCC	NED
I	User	NMI	0.672 ± 0.052	0.771 ± 0.031	0.879 ± 0.021	0.601 ± 0.023	0.824 ± 0.022	0.831 ± 0.016	$\mathbf{0.948 \pm 0.017}$
		Accuracy	0.532 ± 0.029	0.597 ± 0.034	0.781 ± 0.049	0.703 ± 0.021	0.695 ± 0.019	0.780 ± 0.021	$\mathbf{0.865 \pm 0.045}$
	Product	NMI	0.772 ± 0.075	0.736 ± 0.002	0.886 ± 0.001	0.851 ± 0.043	0.801 ± 0.032	0.854 ± 0.161	$\mathbf{0.949 \pm 0.096}$
		Accuracy	0.776 ± 0.105	0.651 ± 0.005	0.789 ± 0.057	0.719 ± 0.001	0.683 ± 0.045	0.741 ± 0.122	$\mathbf{0.865 \pm 0.032}$
II	User	NMI	0.793 ± 0.062	0.769 ± 0.047	0.901 ± 0.008	0.883 ± 0.041	0.746 ± 0.052	0.902 ± 0.021	$\mathbf{0.960 \pm 0.011}$
		Accuracy	0.682 ± 0.062	0.568 ± 0.075	0.794 ± 0.052	0.703 ± 0.028	0.683 ± 0.119	0.761 ± 0.061	$\mathbf{0.867 \pm 0.038}$
	Product	NMI	0.673 ± 0.024	0.743 ± 0.064	0.924 ± 0.064	0.904 ± 0.058	0.839 ± 0.072	0.911 ± 0.141	$\mathbf{0.947 \pm 0.014}$
		Accuracy	0.540 ± 0.086	0.694 ± 0.109	0.799 ± 0.034	0.834 ± 0.058	0.788 ± 0.001	0.801 ± 0.014	$\mathbf{0.853 \pm 0.042}$
III	User	NMI	[-]	[-]	0.720 ± 0.011	0.913 ± 0.046	0.763 ± 0.023	0.921 ± 0.046	$\mathbf{0.973 \pm 0.006}$
		Accuracy			0.694 ± 0.017	0.523 ± 0.034	0.492 ± 0.002	0.632 ± 0.021	$\mathbf{0.882 \pm 0.023}$
	Product	NMI			0.840 ± 0.001	0.842 ± 0.074	0.763 ± 0.028	0.856 ± 0.121	$\mathbf{0.933 \pm 0.019}$
		Accuracy			0.640 ± 0.068	0.653 ± 0.104	0.535 ± 0.112	0.716 ± 0.214	$\mathbf{0.799 \pm 0.023}$
IV	User	NMI	[-]	[-]	0.879 ± 0.021	0.879 ± 0.021	[-]	0.881 ± 0.012	$\mathbf{0.959 \pm 0.017}$
		Accuracy			0.534 ± 0.011	0.495 ± 0.133		0.563 ± 0.112	$\mathbf{0.587 \pm 0.072}$
	Product	NMI			0.947 ± 0.001	0.893 ± 0.129		0.891 ± 0.012	$\mathbf{0.971 \pm 0.007}$
		Accuracy			0.832 ± 0.064	0.793 ± 0.101		0.801 ± 0.102	$\mathbf{0.902 \pm 0.032}$

Table 7.3: Experimental results for NMI and Accuracy for synthetic data. Boldface indicates the best results.

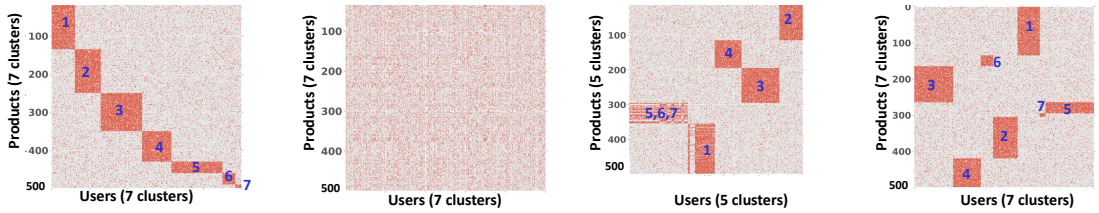


Figure 7.3: The co-clustering result of NED on synthetic data. Left to right: (a) original synthetic data with 7 users and 7 product clusters, (b) shuffled synthetic data, (c) the second-best performing baseline (CoClusInfo) result (91% accurate), and (d) NED's result (100% accurate). NED is successfully able to detect large as well as small sized co-clusters more accurately.

7.5.3 Evaluation Measures

Co-clustering

We evaluate NED and the baselines for co-clustering using three criteria: ***Normalized Mutual Information (NMI)***, ***Accuracy*** and ***CPU time (sec)***.

Explainability

We evaluate NED and the baselines for explainability using three criteria namely **Average Precision**, **Stability score** [188] and our proposed **Compression Score**. The MDL based compression score is given by:

$$\begin{aligned} Score = {} & \log^* r_m + \log^* c_n + \log^* uf_m + \log^* pf_n + E(B_{mn}) \\ & - r_m \log_2 \frac{r_m}{I} - c_n \log_2 \frac{c_n}{J} + \log_2(r_m c_n + 1) \\ & - uf_m \log_2 \frac{uf_m}{F_1} + \log_2(r_m uf_m + 1) + E(B_{mm}) \\ & - pf_n \log_2 \frac{pf_n}{F_2} + \log_2(c_n pf_n + 1) + E(B_{nn}) \end{aligned} \tag{7.18}$$

where $\log^*$ is the universal code length for integers [212], r_m is m^{th} the row cluster encoding bits, c_n is n^{th} the column cluster encoding bits, uf_m is the m^{th} feature cluster encoding bits corresponding to the m^{th} row cluster, pf_n is n^{th} feature cluster encoding bits corresponding to the n^{th} column cluster. $E(B_{mn})$, $E(B_{mm})$ and $E(B_{nn})$ are the number of bits required to encode block of user-product, user-feature and product-feature, respectively. The lower the value, the better is the compression.

7.5.4 Quantitative Analysis

In this section, we provide quantitative evaluation and analysis of our method for co-clustering and explainability on synthetic and real-world datasets. Recommendation is undoubtedly one key task for businesses. However, explainable and thematic engagement is useful for both individual content creators (influencers, etc.) and companies (Snapchat, Netflix, etc.) who aim to attract certain audiences with original content. The most concrete and evaluate-able components are discovering and describing the co-clusters and explanations found, upon which these creators can leverage findings. Thus, our quantitative evaluation is focused around these points, and we also show associated qualitative end-to-end findings on real data.

Co-clustering Performance

To answer the first experimental question, we report co-clustering performance of all methods in Table 7.3 and 7.4. We cap the allowed run-time of all methods to 24 hours, indicating unfinished/non-converged results as "-".

Synthetic Data: As we can see in Table 7.3, NED achieves the best performance for all synthetic datasets. As expected, NED significantly out-performs two related NMTF-based methods, SNCC, FNMTF and WC-NMTF, due to its use of mutual information via the summary matrix as described in Section 7.4. Moreover, NED outperforms CoClusInfo, NEO-CC and the neural method DeepCC, in both accuracy and NMI, surpassing them with substantial accuracy improvements (average $\approx 14\%$). We visualize the co-clustering ability of NED and second highest performing baseline (CoClusInfo) on an additional small synthetic data (SmallSYN). The SmallSYN data has 500 users, 500 features and 7 co-clusters, and is shown in Figure 7.3(a). The shuffled data fed into CoClusInfo and NED is shown in Figure 7.3(b). Subsequently, the users and products are rearranged to show discovered co-clusters as shown in Figure 7.3(d), according to which NED detects the co-clusters precisely (100% accuracy). CoClusInfo's results (achieving 91% accuracy) are shown in Figure 7.3(c). Note that the inferred co-cluster sequences in Figure 7.3(c-d) are not the same as in Figure 7.3(a) – this is because the cluster assignment for the same user/product clusters may be arbitrarily permuted during inference.

Real Data: Table 7.4 presents NED 's performance compared to state-of-the-art techniques on real datasets, using the NMI, accuracy and CPU Time (in seconds). For each dataset, we list the scores that correspond to a specific cluster type. For all real datasets except Movielens data, only user labels are available. For Movielens data, product labels are inferred from movie genre types. NED is the only method that shows consistently high performance on all different kinds of datasets, indicating its flexibility. Due to space

Dataset	Cluster	Metric	NEO-CC	DeepCC	CoClusInfo	FNMTF	WC-NMTF	SNCC	NED
Cora	User	NMI	0.034 ± 0.004	0.003 ± 0.001	0.152 ± 0.006	0.152 ± 0.011	0.173 ± 0.002	0.112 ± 0.001	**0.181 ± 0.023**
		Accuracy	0.206 ± 0.034	0.2943 ± 0.045	0.375 ± 0.032	0.373 ± 0.101	0.314 ± 0.133	0.213 ± 0.022	**0.399 ± 0.034**
		Time (sec)	1488.8 ± 126.6	1481.4 ± 256.3	6.6 ± 1.23	4.5 ± 1.2	121.3 ± 12.5	**4.1 ± 0.34**	7.9 ± 0.7
WebKB4	User	NMI	0.136 ± 0.001	0.379 ± 0.084	0.383 ± 0.095	0.255 ± 0.054	0.241 ± 0.058	0.149 ± 0.003	**0.489 ± 0.043**
		Accuracy	0.374 ± 0.075	0.568 ± 0.026	0.653 ± 0.082	0.549 ± 0.102	0.503 ± 0.121	0.461 ± 0.101	**0.752 ± 0.029**
		Time (sec)	4089.7 ± 118.8	1719.9 ± 493.5	4.7 ± 1.6	3.7 ± 0.63	286.3 ± 2.2	**3.2 ± 0.21**	3.8 ± 0.6
MovieLens	Product	NMI	0.356 ± 0.020	0.636 ± 0.102	0.742 ± 0.112	0.394 ± 0.021	0.689 ± 0.039	0.472 ± 0.102	**0.783 ± 0.124**
		Accuracy	0.413 ± 0.020	0.550 ± 0.192	0.649 ± 0.102	0.382 ± 0.048	0.592 ± 0.124	0.564 ± 0.116	**0.683 ± 0.017**
		Time (sec)	8402.3 ± 363.5	2067.78 ± 34.5	103.98 ± 3.6	34.78 ± 6.9	673.67 ± 5.7	**29.43 ± 6.4**	41.55 ± 3.6
News20	User	NMI	[-]	0.448 ± 0.018	0.499 ± 0.091	0.149 ± 0.031	0.392 ± 0.019	0.334 ± 0.016	**0.541 ± 0.019**
		Accuracy		0.452 ± 0.075	0.433 ± 0.012	0.108 ± 0.093	0.329 ± 0.121	0.316 ± 0.021	**0.477 ± 0.029**
		Time (sec)		4028.3 ± 42.6	114.4 ± 4.1	97.4 ± 12.3	1143.1 ± 41.6	**92.6 ± 10.4**	111.1 ± 16.3
Caltech	User	NMI	0.324 ± 0.086	0.636 ± 0.087	0.753 ± 0.059	0.211 ± 0.101	0.593 ± 0.109	0.462 ± 0.011	**0.756 ± 0.019**
		Accuracy	0.543 ± 0.001	0.778 ± 0.022	0.899 ± 0.012	0.531 ± 0.111	0.835 ± 0.291	0.791 ± 0.091	**0.911 ± 0.102**
		Time (sec)	2649.6 ± 132.4	281.4 ± 13.3	2.4 ± 0.72	3.5 ± 0.41	103.5 ± 17.4	**2.9 ± 0.21**	3.9 ± 0.24

Table 7.4: Experimental results for NMI, Accuracy and CPU Time in seconds for real data. The boldface means the best results.

limitations, here, we provide analysis of Movielens data below and analysis of other real datasets are provided in Supplementary Material.

Movielens Data: We observe that NED is able to detect 15 clusters for movies. This is reasonable outcome for this data, as there are overlapping movie categories; for example, "Toy Story (1995)" can be categorized in Animation as well as Comedy. FNMTF detected only $5-6$ clusters as shown in Figure 7.4. CoClusInfo, WC-NMTF, SNCC and DeepCC achieved better performance, but still lower than NED.

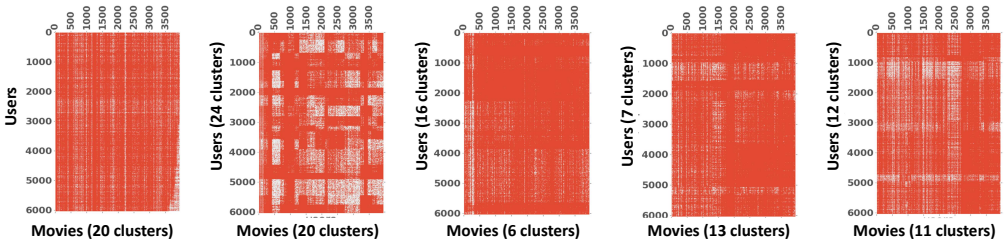


Figure 7.4: Visualized co-clustering result of NED on the Movielens data. From left to right: (a) Original data, (b) co-cluster detected by NED, (c) FNMTF, (d) WC-NMTF, and (e) DeepCC. Each method is run with $M = 50, N = 20$ (specifying a maximum of 50 user clusters and 20 product clusters). NED evidently produces the most coherent co-clustering structure.

We also visualize the co-clustering results on the Movielens dataset, to show a visual representation of improved co-clustering performance. We rearrange the original data matrix (Figure 7.4(a)) according to the user and product cluster assignments to show the co-clustering result. We observe that the co-clusters for NED (Figure 7.4(b)) are more salient compared to those for baselines FNMTF, WC-NMTF and DeepCC (Figures 7.4(c-

e)). Snapchat dataset is very interesting and detailed analysis is provided in Section 7.5.8.

Explainability Evaluation

To answer the second experimental question, we first analyze the explanations derived from NED, and compare the results with explanations from baseline methods. Table 7.5 presents NED's performance over LightGBM explanations on 2 synthetic and 2 real datasets, using the stability score, average precision, CPU time in sec. and our proposed compression score.

Setup: First, we obtain co-clusters using NED and other very closely related two matrix tri-factorization based methods, i.e FNMTF and WC-NMTF. Note that baselines CoclusInfo, DeepCC and NEO-CC provide only user and product cluster factor matrices but do not provide any summary matrix **S**, which is required for our computations (See Equ. 7.16 and 7.17). Next, co-clustering results are fed into explainability components for both LightGBM and NED.For NED, first, we create auxiliary matrices for both users and products (see Section 7.4.2), and then compute PMI using co-clustering outcomes (see Section 7.4.2). We then select the top-5 user and product features per co-cluster to explain it.

Results: We observe that NED drastically outperforms the baselines for all the datasets. For synthetic data SYN-I, each user cluster is created with combination of age and gender. Similarly, product clusters are focused on gender dominated viewership. For example, in our simulated data, women associate highly with "makeup & cosmetic, weight loss, and pop music" and men associate highly "card games, driving and racing games, and

body building." Our results suggest that LightGBM is not able to explain co-clusters well with the correct attributes. This is likely, partially because it trains a classifier using only co-cluster outcomes and does not leverage implicit similarities between users and products. Conversely, NED successfully captures these similarities in auxiliary matrices and is thus able to explain these relationships for each co-cluster. Similarly, NED is able to provide more complex explanations for SYN-II in which product clusters do not have any specific gender based dominance.

For Movielens, NED discovered an interesting co-cluster in which salesmen and programmers strongly associated with adventure movies, and in another co-cluster lawyers strongly associated with drama and fantasy movies. Interestingly, all co-clustering baselines with combination of LightGBM for explanation underperformed in our experiments, compared to the solution adopted by NED. NED consistently compressed the discovered co-clusters quite well, compared to LightGBM, suggesting that we can describe the co-clusters more concisely with better-quality attribute/feature explanations. Overall, NED outperforms state-of-the-art approaches by at least $\approx$ 15% stability and $\approx$ 20% average precision improvement. Also, NED achieves at least 5% compression bits and 20% runtime reduction (see Table 7.5), In the table 7.5) boldface means the best results. The 'a' represents Stability, 'b' represents Avg. Precision, 'c' represents CPU Time (sec) and 'd' represents Compression (Kb). Here 'U' = User, 'P' = Product and 'B' = Both..

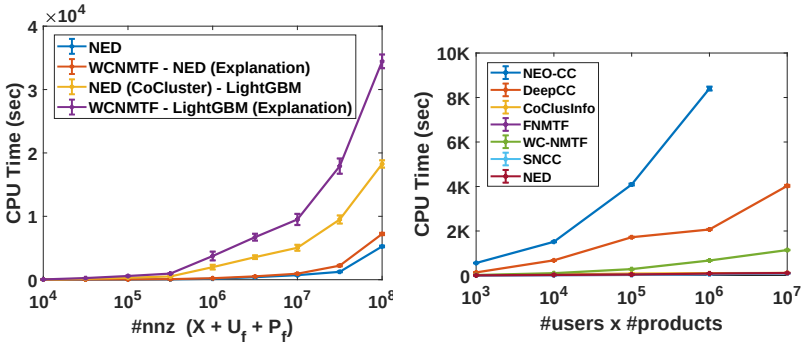


Figure 7.5: (a) Total running time (averaged over 10 runs) of NED versus the total number of non-zeros for Snapchat data (b) Co-clustering running time for synthetic data.

7.5.5 Scalability

Finally, to answer the third question, we experimentally study the runtime of NED with respect to the input size on real graph. For runtime measurements, we use a large private viewer-publisher Snapchat data with 5 million viewer and 7500 publisher. To generate real graphs of growing size, we increasingly sample the Snapchat user-product user engagement matrix rows and report NED runtime averaged over 10 runs in Figure 7.5. We can observe that the run-time of NED scales approximately linearly; notably, NED can handle interaction matrices with many millions of interactions in mere minutes. We show only linear scaling for Snapchat dataset because (a) it is the largest dataset, and (b) the scaling trends are consistent with other datasets.

7.5.6 Effectiveness of Auxiliary Feature Matrix

To show the effectiveness of using meta path based auxiliary feature matrices (in short AFM), we computed average precision scores with and without them in Equ. 7.16

and 7.17 for Movielens data. We fed NED with varied number of user cluster M (as this is unknown) but we fixed the number of movie clusters N to 20. Table 7.6 shows that NED performance improved by $\approx 7.5\%$ when using auxiliary feature matrices.

7.5.7 Parameter Sensitivity Analysis

We evaluate the sensitivity of the interpolation parameters α, β and γ in Equ. 7.12-7.13 which describe involvement of each meta-path matrix. We learn stability and average precision score for different combinations of $\{\alpha, \beta, \gamma\}$ on synthetic data SYN-I. Here, we kept $\beta = \gamma$ to give equal importance to 4-step meta path. Figure 7.6(left) shows that NED performs better when higher importance is given to direct paths i.e $\alpha \geq 0.5$. Beyond $\alpha = 0.5$, there is no significant change in performance. Moreover, Figure 7.6 (right) shows that average precision achieves optimal values around $\alpha = 0.5$, suggesting that incorporating indirect interactions via meta-paths beyond just direct paths does contribute to improved performance. Hence, we choose $\alpha = 0.5$,$\beta = \gamma = 0.25$ in our experiments.

7.5.8 NED at Work

We use NED on a private viewer-publisher interaction dataset from Snapchat, consisting of $500K$ viewers, 2500 publishers, and 5 million viewer-publisher interactions (here, viewers correspond to users, and publishers to products, for consistency with our prior discussion). The dataset is naturally unlabeled, making NMI and accuracy analysis infeasible; therefore, we resort to qualitative discussion. In this dataset, each viewer is described by 22 associated features (age, gender, country, etc.) and each publisher has 238

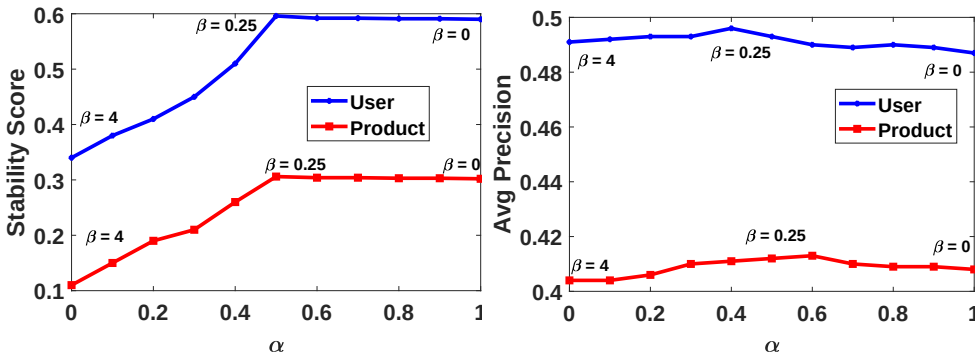


Figure 7.6: Sensitivity of interpolation parameters for co-cluster explanation; values of $\alpha = 0.5$ (interpolating direct path with indirect metapath matrices) produce best explanation results.

features (e.g. publisher demographics, publishing category etc.). We ran NED with the maximum number of user clusters $M = 100$ and product clusters $N = 25$. For explainability evaluation, we use top $N = 5$ highest feature values computed with Equ. 7.16 and 7.17. NED finds viewer-publisher co-clusters of various sizes in Snapchat data, as shown in Figure 7.7. Table 7.7 provide three major niches based on high PMI values from the summary matrix $\mathbf{S}$.

The viewers clusters in the green square labeled as '1' mostly belong to groups of young women (age between $13-20$) associated with publisher who publish content regarding 'Beauty & Lifestyles' category. We observe that viewers in this dense group have similar content consumption. The viewers cluster '2' contains viewers with connections across many publishers clusters like 'Home & Family' and 'Hobbies & Interests'. All of the viewers are between two age groups $25-34$ and > 35. This niche does not have any gender dominance in viewer cluster but we observe that most of the content creators or publishers are men.

Figure 7.7: NED on Snapchat data finds clusters of viewers/publishers with similar attributes coherence. The interaction matrix is carefully arranged by NED, revealing patterns: e.g., the young women heavily view content regarding 'Beauty' category published by women creators, and they focus on the same group of content.

Niche represented by '3' is the most prominent: this is a dense group of male viewers, $> 70\%$ of them are from North America and all of them are between age group $21 - 24$, highly associated with content related to 'World Music', published by group of male publishers between age group $25 - 34$. Most of the publishers are from Europe. Overall as shown in Table 7.5, NED outperforms state-of-the-art approaches by $\approx 6\%$ stability and $\approx 4\%$ average precision improvement. Also, NED achieves at least 10% compression bits and 25% run time reduction (See Table 7.5) for Snapchat dataset. All in all, by using NED, we are able to understand and explain the user content consumption graph in a completely unsupervised fashion.

7.6 Conclusions

In this work, we tackle the problem of discovering market niches for strategic content creation to satisfy diverse audience groups. We pose the niche detection problem as one which involves discovering coherent co-clusters in user-product (content) interaction graph data, as well as explaining the co-clusters using nodal attributes on user and product nodes. To our knowledge, ours is the first work which tackles such an explainable co-clustering problem. We proposed NED, the first niche detection framework for finding and explaining co-clusters in attributed interaction graphs. NED utilizes principles from mutual information to (a) propose and solve a non-negative matrix tri-factorization oriented objective to discover cohesive co-clusters, and (b) select important user and product features associated with these co-clusters using meta-path driven feature selection. In doing so, we find that NED successfully discovers niches in user-content consumption data, and demonstrates improvements over both state-of-the-art co-clustering approaches ($\approx 14\%$ accuracy) as well as candidate explanation approaches ($\approx 20\%$ average precision) on multiple simulated and real-world datasets. Finally, we show that NED provides an advantageous speed-quality tradeoff over alternative approaches, and scales effortlessly and discovers interesting insights on large-scale interaction data with over *60M* interactions, via a private dataset from Snapchat.

The content of this chapter was under blind peer review at the time of thesis submission.

Table 7.5: Experimental results for explainability evaluation.

Data	Score	FNMTF			WC-NMTF			NED		
		LightGBM	**BMGUFS**	**NED**	**LightGBM**	**BMGUFS**	**NED**	**LightGBM**	**BMGUFS**	**NED**
S-I	U-a	0.306 ± 0.04	0.294 ± 0.01	0.445 ± 0.06	0.381 ± 0.06	0.264 ± 0.02	0.447 ± 0.07	0.501 ± 0.06	0.416 ± 0.08	**0.596 ± 0.06**
	U-b	0.119 ± 0.08	0.115 ± 0.06	0.134 ± 0.01	0.112 ± 0.03	0.201 ± 0.09	0.289 ± 0.02	0.148 ± 0.01	0.296 ± 0.16	**0.496±0.07**
	P-a	0.288 ± 0.03	0.276 ± 0.04	0.301 ± 0.02	0.129 ± 0.04	0.131 ± 0.02	0.132 ± 0.01	0.116 ± 0.09	0.264 ± 0.12	**0.306 ± 0.04**
	P-b	0.218 ± 0.01	0.201 ± 0.06	0.311 ± 0.02	0.109 ± 0.06	0.146 ± 0.141	0.231 ± 0.03	0.274 ± 0.01	0.336 ± 0.17	**0.412 ± 0.02**
	B-c	43.34 ± 5.2	264.43 ± 21.64	8.45 ± 4.2	58.01 ± 4.3	124.6 ± 6.4	10.23 ± 3.1	28.26 ± 2.6	119.6 ± 10.4	**5.86 ± 1.9**
	B-d	518.55 ± 0.55	598.65 ± 0.84	499.69 ± 0.4	659.9 ± 0.16	652.6 ± 0.26	636.4 ± 0.07	340.26 ± 0.06	387.4 ± 0.04	**307.5± 0.05**
S-II	U-a	0.241 ± 0.05	0.224 ± 0.07	0.356 ± 0.1	0.202 ± 0.02	0.200 ± 0.06	0.331 ± 0.02	0.485 ± 0.07	0.483 ± 0.02	**0.8481± 0.02**
	U-b	0.105 ± 0.01	0.106 ± 0.03	0.321 ± 0.09	0.09 ± 0.01	0.121 ± 0.04	0.298 ± 0.04	0.185 ± 0.08	0.321 ± 0.02	**0.403 ± 0.02**
	P-a	0.637 ± 0.11	0.592 ± 0.09	0.812 ± 0.13	0.071 ± 0.01	0.062 ± 0.01	0.035 ± 0.01	0.585 ± 0.15	0.576 ± 0.14	**0.836 ± 0.06**
	P-b	0.121 ± 0.06	0.122 ± 0.08	0.224 ± 0.01	0.101 ± 0.03	0.102 ± 0.04	0.101 ± 0.02	0.100 ± 0.04	0.101 ± 0.03	**0.390± 0.05**
	B-c	1123.69 ± 12.7	1202.1.7 ± 0.04	102.454 ± 11.9	1229.82 ± 32.8	1364.7 ± 10.4	116.42 ± 18.4	1178.24 ± 8.9	962.4 ± 10.4	**64.1 ± 4.8**
	B-d	6294.58 ± 0.08	6746.24 ± 0.02	5928.72 ± 0.01	9761.52 ± 0.1	9842 ± 0.04	9285.6 ± 0.02	4590 ± 0.09	4656.8 ± 0.02	**3920.18 ± 0.07**
ML	U-a	0.197 ± 0.01	0.210 ± 0.01	0.126 ± 0.06	0.136 ± 0.04	0.141 ± 0.02	0.1679 ± 0.05	0.129 ± 0.08	0.196 ± 0.02	**0.215 ± 0.04**
	U-b	0.124 ± 0.02	0.132 ± 0.11	0.246 ± 0.05	0.136 ± 0.01	0.142 ± 0.06	0.389 ± 0.08	0.197 ± 0.01	0.271 ± 0.06	**0.466 ± 0.11**
	P-a	0.009 ± 0.07	0.101 ± 0.01	0.101 ± 0.02	0.129 ± 0.09	0.162 ± 0.03	0.196 ± 0.06	0.212 ± 0.02	0.108 ± 0.01	**0.228 v 0.06**
	P-b	0.246 ± 0.12	0.294 ± 0.02	0.301 ± 0.02	0.312 ± 0.01	0.264 ± 0.01	0.358 ± 0.03	0.326 ± 0.09	0.113 ± 0.05	**0.424 ± 0.03**
	B-c	38.6 ± 7.4	124.6 ± 2.5	30.5 ± 1.4	35.1 ± 2.9	112.2 ± 3.6	28.4 ± 6.4	21.24 ± 1.4	98.7 ± 3.3	**14.30 ± 2.9**
	B-d	749.29 ± 0.42	760.4 ± 0.01	730.95 ± 0.19	788.08 ± 0.12	796.6 ± 0.04	760.02 ± 0.18	696.17 ± 0.49	683.6 ± 0.01	**671.7 ± 0.32**
SC	U-a	0.289 ± 0.04	0.122 ± 0.04	0.305 ± 0.02	0.292 ± 0.03	0.161 ± 0.01	0.321 ± 0.04	0.329 ± 0.04	0.231 ± 0.02	**0.351± 0.08**
	U-b	0.374 ± 0.07	0.101 ± 0.04	0.518 ± 0.11	0.281 ± 0.03	0.009 ± 0.01	0.521 ± 0.04	0.277 ± 0.04	0.201 ± 0.01	**0.527 ± 0.05**
	P-a	0.298 ± 0.01	0.106 ± 0.07	0.344 ± 0.09	0.312 ± 0.01	0.112 ± 0.01	0.241 ± 0.09	0.311 ± 0.09	0.261 ± 0.09	**0.351 ± 0.01**
	P-b	0.178 ± 0.03	0.113 ± 0.02	0.236 ± 0.01	0.201 ± 0.02	0.121 ± 0.04	0.241 ± 0.10	0.200 ± 0.12	0.102 ± 0.01	**0.257 ± 0.03**
	B-c	1498.4 ± 12.5	2642.6 ± 23.7	149.5 ± 38.8	1298.5 ± 34.9	4364.36 ± 12.4	246 ± 13.9	1256.37 ± 56.2	3641.78 ± 34.7	**107.15 ±11.3**
	B-d	60457.47 ± 1.21	61362.2 ± 0.04	54538.27 ± 1.14	58694.03 ± 3.21	58961.7 ± 0.01	56022.87 ± 3.42	52889 ± 0.45	53316.8 ± 0.14	**47765.7 ± 0.34**

Given M	**Predicted M**	**Without AFM**	**With AFM**
15	10	0.105 ± 0.11	0.113 ± 0.14
30	24	0.198 ± 0.02	0.215 ± 0.04
50	27	0.201 ± 0.01	0.214 ± 0.06

Table 7.6: Experiment evaluation of Auxiliary Feature Matrix on Movielens dataset.

Beauty & Lifestyle	**Family & Hobbies**	**Music**
Cosmetics	Parenting & child care	Urban hip-hop
Women Fashion	Home Improvement	Electronic Dance Music
Women Lifestyle	Shopping	Pop Music
Spa beauty	Gardening	Concerts
Nail Art	Decor Design	Classical Opera

Table 7.7: Illustration of niches discovered by NED.

Part II

Mining Streaming Tensors

Chapter 8

Sampling-based Batch Incremental Tensor Decomposition

"How to summarize high-order data tensor? How to incrementally update those patterns over time?"

Tensor decompositions are invaluable tools in analyzing multimodal datasets. In many real-world scenarios, such datasets are far from being static, to the contrary they tend to grow over time. For instance, in an online social network setting, as we observe new interactions over time, our dataset gets updated in its "time" mode. How can we maintain a valid and accurate tensor decomposition of such a dynamically evolving multimodal dataset, without having to re-compute the entire decomposition after every single update? In this chapter we introduce SamBaTen, a *Sam*pling-based *Ba*tch Incremental *Ten*sor Decomposition algorithm, which incrementally maintains the decomposition given new updates to the tensor dataset. SamBaTen is able to scale to datasets that the state-of-the-art in

incremental tensor decomposition is unable to operate on, due to its ability to effectively summarize the existing tensor and the incoming updates, and perform all computations in the reduced summary space. We extensively evaluate SamBaTen using synthetic and real datasets. Indicatively, SamBaTen achieves comparable accuracy to state-of-the-art incremental and non-incremental techniques, while being up to *25-30 times faster*. Furthermore, SamBaTen scales to very large sparse and dense dynamically evolving tensors of dimensions up to $100K \times 100K \times 100K$ where state-of-the-art incremental approaches were not able to operate. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, Ravdeep Pasricha, and Evangelos E. Papalexakis. "Sambaten: Sampling-based batch incremental tensor decomposition." In Proceedings of the 2018 SIAM International Conference on Data Mining, pp. 387-395. Society for Industrial and Applied Mathematics, 2018.

8.1 Introduction

Tensor decomposition is a very powerful tool for many problems in data mining [139, 197]. The success of tensor decomposition lies in its capability of finding complex patterns in multi-way settings, by leveraging higher-order structure and correlations within the data. The dominant tensor decompositions are CP/PARAFAC (henceforth referred to as CP), which extracts interpretable latent factors from the data, and Tucker, which estimates the joint subspaces of the tensor. In this work we focus on the CP decomposition,

which has been shown to be extremely effective in exploratory data mining time and time again [197].

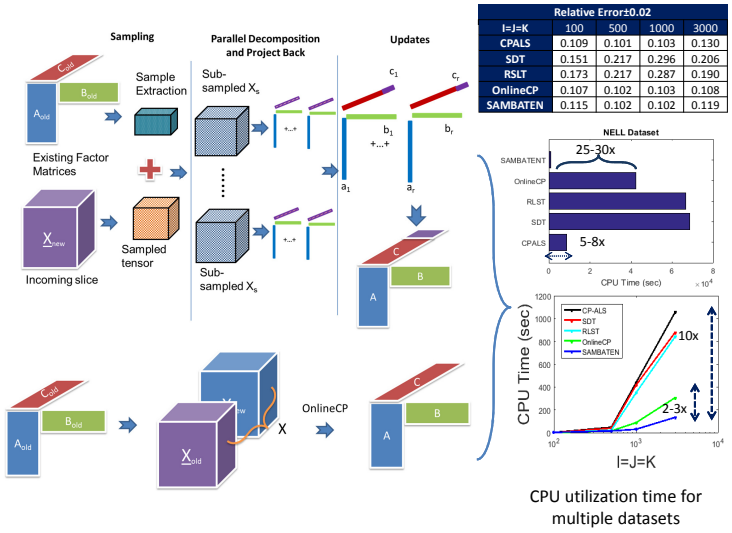


Figure 8.1: SamBaTen outperforms state-of-the-art baselines while maintaining competitive accuracy.

In a wide array of modern real-world applications, data are far from being static. To the contrary, data get updated dynamically. For instance, in an online social network, new interactions occur every second and new friendships are formed at a similar pace. In the tensor realm, we may view a large proportion of these dynamic updates as an introduction of new "slices" in the tensor: in the social network example, new interactions that happen as time evolves imply the introduction of new snapshots of the network, which grow the tensor in the "time" mode. A tensor decomposition in that tensor can discover *communities* and their evolution over time. How can we handle such updates in the data without having to re-compute the decomposition whenever an update arrives, but incrementally update the existing results given the new data? In the community detection example, how can we track

the evolution of the existing communities, and discover new ones, for the new updates that continuously arrive?

Computing the decomposition for a dynamically updated tensor is challenging, with the challenges lying, primarily, on two of the three V's in the traditional definition of Big Data: *Volume* and *Velocity*. As a tensor dataset is updated dynamically, its volume increases to the point that techniques which are not equipped to handle those updates *incrementally*, inevitably fail to execute due to the sheer size of the data. Furthermore, even though the applications that tensors have been successful so far do not require real-time execution per se, the decomposition algorithm must, nevertheless, be able to ingest the updates to the data at a rate that will not result in the computation being "drowned" by the incoming updates.

The majority of prior work has focused on the Tucker Decomposition of incrementing tensors [192, 243, 70], however very limited amount of work has been done on the CP. Nion and Sidiropoulos [186] proposed two methods namely Simultaneous Diagonalization Tracking (SDT) and Recursive Least Squares Tracking (RLST) and most recently, [284] introduced the OnlineCP decomposition for higher order online tensors. Even though prior work in incremental CP decomposition, by virtue of allowing for incremental updates to the already computed model, is able to deal with *Velocity*, when compared to the naive approach of re-executing the entire decomposition on the updated data, every time a new update arrives, it falls short when the *Volume* of the data grows.

We show a snapshot of our results in Figure 8.1: SamBaTen is faster than all state-of-the-art methods on data that the baselines were able to operate on. Furthermore,

SamBaTen was able to scale to, both dense and sparse, dynamically updated tensors, where none of the baselines was able to run. Finally, SamBaTen achieves comparable accuracy to existing incremental and non-incremental methods. Our contributions are summarized as follows:

- **Novel scalable algorithm**: The advantage of SamBaTen stems from the fact that it only operates on small summaries of the data at all times, thereby being able to maintain its efficiency regardless of the size of the full data. To the best of our knowledge, this is the first incremental tensor decomposition which effectively leverages sparsity in the data.
- **Extensive experimental evaluation**: Through experimental evaluation on six real-world datasets with sizes that range up to 70GB, and synthetic tensors that range up to $100K \times 100K \times 100K$, we show thatSamBaTen can incrementally maintain very accurate decompositions, faster and in a more efficient and scalable manner than state-of-the-art methods.

Reproducibility: We make our Matlab implementation publicly available at link [1]. Furthermore, all the datasets we use for evaluation are publicly available.

8.2 Related work

Incremental tensor methods in the literature can be categorized into three main categories: 1) Tucker decomposition, 2) CP decomposition, 3) Tensor completion

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

Tucker Decomposition: Online tensor decomposition was first proposed by Sun *el at.*[243] as ITA (Incremental Tensor Analysis), describing the three variants of Incremental Tensor Analysis. First, DTA i.e. Dynamic tensor analysis which is based on calculation of co-variance of matrices in traditional higher-order singular value decomposition in an incremental fashion. Second, with help of SPIRIT algorithm, they found approximation of DTA named as Stream Tensor Analysis (STA). Third, they proposed window-based tensor analysis (WTA). To improve the efficiency of DTA, it uses a sliding window strategy. Liu *el at.*[192] proposed an efficient method to diagonalize the core tensor to overcome this problem. Hadi *el at.* [70] proposed the multi-aspect-streaming tensor analysis (MASTA) method that and allows the tensor to simultaneously grow in all modes.

CP Decomposition: There is very limited study on online CP decomposition methods. Phan *el at.* [203] had developed a theoretic approach GridTF to large-scale tensors processing based on an extension to CP's fundamental mathematics theory. They used divide and concur technique to get sub-tensors and fuses the output of all factorization to achieve final factor matrices which is proved to be same as decomposing the whole tensor using CP decomposition. Its potential of concurrent computing methods to adapt the engineering applications remains unclear. Nion *el at.* [186], proposed two algorithms that focus on CP decomposition namely SDT (Simultaneous Diagonalization Tracking) that incrementally perform the SVD of the unfolded tensor; and RLST (Recursive Least Squares Tracking) , which recursively updates the decomposition factors by minimizing the mean squared error. The latest related work is OnlineCP, proposed by Zhou, *el at.* [284], is an online CP decomposition method, where the the latent factors are updated when there are new data.

Tensor Completion: The main difference between completion and decomposition techniques is that in completion "zero" values are considered "missing" and are not part of the model, and the goal is to impute those missing values accurately, rather than extracting latent factors from the observed data.

The earliest work on incremental tensor completion traces back to [168], and recently, Qingquan *el at.*[238], proposed streaming tensor completion based on block partitioning.

8.3 Problem Formulation

In many real-world applications, data grow dynamically. In a time-evolving social network, we observe user interactions every few seconds, which can be translated to new tensor slices, after fixing the temporal granularity (a problem which, on its own merit, is very hard to solve optimally, and we do not address in this chapter). This incremental property of data gives rise to the need for an on-the-fly update of the existing decomposition, which we name incremental tensor decomposition. Notice that the literature (and thereby this chapter) uses the terms "incremental", "dynamic", and "online" interchangeably. In such scenarios, data updates happen very fast which make traditional (non-incremental) methods to collapse because they need to recompute the decomposition for the entire dataset.

We focus on a 3-mode tensor one of whose dimensions are growing with time. However, the problem definition (and our proposed method) extends to any number of modes. Let us consider $\underline{\mathbf{X}}(t)=\mathbb{R}^{I\times J\times K_1(t)}$ at time t. The CP decomposition of $\underline{\mathbf{X}}(t)$ is given

as :

$$\underline{\mathbf{X}}^{(1)}(t) \approx (\mathbf{A}(t) \odot \mathbf{B}(t))\mathbf{C}^T(t) \approx \mathbf{L}(t)\mathbf{C}^T(t)$$

where $\mathbf{L}(t) = (\mathbf{A}(t) \odot \mathbf{B}(t))$ of dimension $IJ \times R$ and $\mathbf{C}^T(t)$ is of dimension $K_1 \times R$. When new incoming slice $\underline{\mathbf{X}}(t^{'})$=$\mathbb{R}^{I \times J \times K_2(t^{'})}$ is added in mode 3, required decomposition at time $t^{'}$ is :

$$\underline{\mathbf{X}}^{(1)}(t + t^{'}) \approx \mathbf{L}(t + t^{'})\mathbf{C}^T(t + t^{'})$$

where $\mathbf{L}(t+t^{'}) = (\mathbf{A}(t+t^{'}) \odot \mathbf{B}(t+t^{'}))$ of dimension $IJ \times R$ and $\mathbf{C}^T(t+t^{'})$ is of dimension $(K_1 + K_2) \times R$.

The problem that we solve is the following:

Given (a) an existing set of decomposition results $\mathbf{A}(t), \mathbf{B}(t)$ and $\mathbf{C}(t)$ of R components, which approximate tensor $\underline{\mathbf{X}}_{old}$ of size $I \times J \times K_1$ at time t , (b) new incoming batch of slices in form of tensor $\underline{\mathbf{X}}_{new}$ of size $I \times J \times K_2$ at any time $t^{'}$, find updates of $\mathbf{A}(t^{'}), \mathbf{B}(t^{'})$ and $\mathbf{C}(t^{'})$ **incrementally** to approximate tensor $\underline{\mathbf{X}}$ of dimension $I \times J \times K$, where K =$K_1 + K_2$ after appending new slice or tensor to 3^{rd} mode while maintaining a comparable accuracy with running the full CP decompositon on the entire updated tensor $\underline{\mathbf{X}}$.

To simplify notation, we will interchangeably refer to $\mathbf{A}(t)$ as $\mathbf{A}_{old}$ (when we need to refer to specific indices of that matrix), and similarly for $\mathbf{A}(t')$ we shall refer to it as $\mathbf{A}^{'}$.

8.4 Proposed Method: SamBaTen

As we mention in the introduction, there exists a body of work in the literature that is able to efficiently and incrementally update the CP decomposition in the presence of incoming tensor slices [186, 284]. However, those methods fall short when the size of of the dynamically growing tensor increases, and eventually are not able to scale to very large dynamic tensors. The reason why this happens is because these methods operate on the *full data*, and thus, even though they incrementally update the decomposition (avoiding to re-compute it from scratch), inevitably, as the size of the full data grows, it takes a toll on the run-time and scalability.

In this chapter we propose SamBaTen, which takes a different view of the solution, where instead of operating on the full data, it operates on a summary of the data. Suppose that the "complete" tensor (i.e., the one that we will eventually get when we finish receiving updates) is denoted by $\underline{\mathbf{X}}$. Any given incoming slice (or even a batch of slice updates) can be, thus, seen as a sample of that tensor, $\underline{\mathbf{X}}$ where the sampled indices in the third mode (which we assume is the one receiving the updates) are the indices of the incoming slice(s). Suppose, further, that given a set of sample tensors (which are drawn by randomly selecting indices from all the modes of the tensor) we can approximate the original tensor with high-accuracy (which, in fact, the literature has shown that it is possible [196, 67]). Therefore, when we receive a new batch of slices as an update, if we update those samples with the new indices, then we should be able to compute a decomposition very efficiently which incorporates the slice updates, and approximates the updated tensor well. A visual summary of SamBaTen is shown in Figure 8.2.

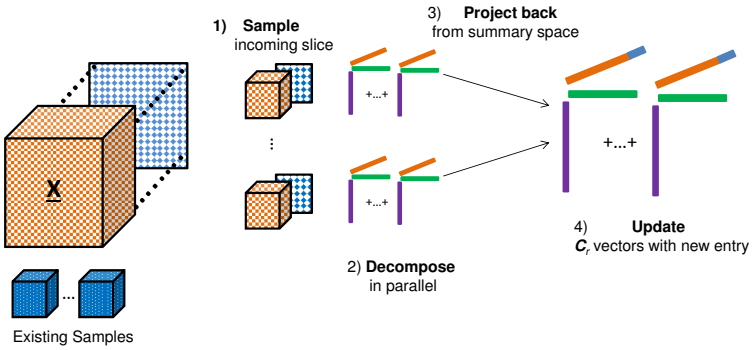


Figure 8.2: SAMBATEN: Sampling-based Batch Incremental Tensor Decomposition: 1) Sample incoming tensor into sub-tensors, 2) run parallel decompositions on the samples, 3) project back the results into the original space, and, finally, 4) update the incrementally growing factor matrix $\mathbf{C}$.

8.4.1 The heart of SamBaTen

The algorithmic framework we propose is shown in Figure 8.2 and is described below: We assume that we have an existing set of decomposition results, as well as a set of *summaries* of the tensor, before the update. Summaries are in the form of sampled sub-tensors, as described in the text below. For simplicity of description, we assume that we are receiving updated slices on the third mode, which in turn have to add new rows to the $\mathbf{C}$ matrix (that corresponds to the third mode). We, further, assume that the updates come in batches of new slices, which, in turn, ensures that we see a mature-enough update to the tensor, which contains useful structure. Trivially, however, SAMBATEN can operate on singleton batches. In the following lines, $\underline{\mathbf{X}}$ is the tensor prior to the update and $\underline{\mathbf{X}}_{new}$ is the batch of incoming slices. Given an update, SAMBATEN performs the following steps:

Sample: The rationale behind SamBaTen is that each batch $\underline{\mathbf{X}}_{new}$ can be seen as a sample of third-mode indices of (what is going to be) the full tensor. In this step, we are going to merge these incoming indices with an already existing set of sampled tensors. In order to obtain those pre-existing samples, we follow a similar approach to [196]. Namely, we sample indices from the tensor $\underline{\mathbf{X}}$ based on a measure of importance. To determine the importance for each mode m and then sample the indices using this measure as a sampling weight divided by its probability. An appropriate measure of importance (MoI) is the **sum-of-squares** of the tensor for each mode. For the first mode , MoI is defined as: $x_a(i) = \sum_{j=1}^{J} \sum_{k=1}^{K} \underline{\mathbf{X}}(i,j,k)^2$ for $i \in$ (1,I). Similarly, we can define the MoI for modes 2 and 3.

We sample each mode of $\underline{\mathbf{X}}$ without replacement, using the above MoI to bias the sampling probabilities. With s as sampling factor, i.e. if $\underline{\mathbf{X}}$ has size $I \times J \times K$, then $\underline{\mathbf{X}}_s$ will be of size $\frac{I}{s}, \frac{J}{s}, \frac{K}{s}$. Sampling rate for each mode is independent from each other, and in fact, different rates can be used for imbalanced modes. In the case of sparse tensors, the sample will focus on the dense regions of the tensor which contains most of the useful structure. In the case of dense tensors, the sample will give priority to the regions of the tensor with the highest variation.

After forming the sample summary $\underline{\mathbf{X}}_s$ for $\underline{\mathbf{X}}$, we merge it with the samples obtained from the intersection of the third-mode indices of $\underline{\mathbf{X}}_{new}$ and the already sampled indices in the remaining modes, so that the final sample is equal to $\underline{\mathbf{X}}_s$=$\underline{\mathbf{X}}(I_s, J_s, K_s \cup [K+1 \cdots K_{new}])$, where $K+1 \cdots K_{new}$ are the third-mode indices of $\underline{\mathbf{X}}_{new}$.

Due to the randomized nature of this summarization, we need to draw multiple samples, in order to obtain a reliable set of summaries. Each such independent sample is denoted as $\underline{\mathbf{X}}_s^{(r)}$. In the case of dense tensors, obtaining multiple, independent random samples helps summarize as much of the useful variation as possible. In fact, we will see in the experimental evaluation that increasing the number of samples, especially for dense tensors, improves accuracy.

In [196] the authors note that in order for their method to work, a set of anchor indices must be common between all samples, so that, later on, we can establish the correct correspondence of latent factors. However, in SamBaTen we do not have to actively fix a set of indices across sampling repetitions. When we sample I_s, J_s, K_s each time, those indices correspond to a portion of the decomposition that is already computed. Therefore, the entire set of indices I_s, J_s, K_s can serve as the set of anchors. This is a major advantage compared to [196], since SamBaTen 1) does not need to commit to a set of fixed indices for all samples a-priori, which, due to randomization may happen to represent a badly structured portion of the tensor, 2) does not need to be restricted in a "small" set of fixed common indices (which is required in [196] in order to ensure that sufficiently enough new indices are sampled across repetitions), but to the contrary, is able to use a larger number of anchor indices to establish correspondence more reliably, and 3) does not require any synchronization between different sampling repetitions, which results in higher parallelism potential.

Decompose: Having obtained $\underline{\mathbf{X}}_s$, from the previous step, SamBaTen decomposes the summary using any state-of-the-art algorithm, obtaining factor matrices $[\mathbf{A}_i', \mathbf{B}_i', \mathbf{C}_i']$.

For the purposes of this chapter, we use the Alternating Least Squares (ALS) algorithm for the CP decomposition, which is probably the most well studied algorithm for CP.

Algorithm 8: SamBaTen

Data: Tensor $\underline{\mathbf{X}}_{new}$ of size $I \times J \times K_{new}$, Factor matrices $\mathbf{A}_{old}, \mathbf{B}_{old}, \mathbf{C}_{old}$, sampling factor s and number of repetitions r.

Result: Factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, λ.

for $i = 1$ *to* r **do**

- Compute $\mathbf{x}_a, \mathbf{x}_b$ and $\mathbf{x}_c$.
- Sample a set of indices I_s, J_s, K_s from $\underline{\mathbf{X}}$ without replacement using $\mathbf{x}_a(i) / \sum_{i=1}^{I} x_a(i)$ as probability (accordingly for the rest).
- $\underline{\mathbf{X}}_s = \underline{\mathbf{X}}(I_s, J_s, K_s \cup [K+1 \cdots K_{new}])$
- $[\mathbf{A}_i^{'}, \mathbf{B}_i^{'}, \mathbf{C}_i^{'}] = \text{CP}(\underline{\mathbf{X}}_s, R)$.
- Normalize $\mathbf{A}_i^{'}, \mathbf{B}_i^{'}, \mathbf{C}_i^{'}$ (as in the text) and absorb scaling in $\boldsymbol{\lambda}$.
- Compute optimal matching between the columns of $\mathbf{A}_{old}, \mathbf{B}_{old}, \mathbf{C}_{old}$ and $\mathbf{A}_i^{'}, \mathbf{B}_i^{'}, \mathbf{C}_i^{'}$ as in the text
- Update only zero entries of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ that correspond to sampled entries of $\mathbf{A}_i^{'}, \mathbf{B}_i^{'}, \mathbf{C}_i^{'}$
- Obtain $\mathbf{C}_{new}$ of size $K_{new} \times R$ by taking last K_{new} rows of $\mathbf{C}^{'}$.
- Use a shared copy of $\mathbf{C}_{new}$ and average out its entries in a column-wise fashion across different sampling repetitions.

end

Update $\mathbf{C}$ of size $(K_{new} + K_{old}) \times R$ as : $C = \begin{bmatrix} \mathbf{C}_{old} \\ \mathbf{C}_{new} \end{bmatrix}$

Update scaling $\boldsymbol{\lambda}$ as the average of the previous and new value.

return $\mathbf{A}, \mathbf{B}, \mathbf{C}, \boldsymbol{\lambda}$

Project back: The CP decomposition is unique (under mild conditions) up to permutation and scaling of the components [197]. This means that, even though the existing decomposition $[\mathbf{A}_{old}, \mathbf{B}_{old}, \mathbf{C}_{old}]$ may have established an order of the components, the decomposition $[\mathbf{A}_i^{'}, \mathbf{B}_i^{'}, \mathbf{C}_i^{'}]$ we obtained in the previous step is very likely to introduce a different ordering and scaling, as a result of the aforementioned permutation and scaling ambiguity. Formally, the sampled portion of the existing factors and the currently computed factors are connected through the following relation Equ. 8.1:

$$[\mathbf{A}_{old}(I_s,:), \mathbf{B}_{old}(J_s,:), \mathbf{C}_{old}(K_s,:)] = [\mathbf{A}_i^{'}(I_s,:)\mathbf{\Lambda\Pi}, \mathbf{B}_i^{'}(J_s,:)\mathbf{\Pi}, \mathbf{C}(K_s,:)_i^{'}\mathbf{\Pi}] \tag{8.1}$$

where $\mathbf{\Lambda}$ is a diagonal scaling matrix (which without loss of generality we absorb on the first factor), and $\mathbf{\Pi}$ is a permutation matrix that permutes the order of the components (columns of the factors).

In order to tackle the scaling ambiguity, we need to normalize the results in a consistent manner. In particular, we normalize such that each column of the newly computed factors which spans the indices that are shared with $[\mathbf{A}_{old}, \mathbf{B}_{old}, \mathbf{C}_{old}]$ has unit norm: $\mathbf{A}_i^{'}(:,f) = \frac{\mathbf{A}_i^{'}}{||\mathbf{A}_i^{'}(I_s,f)||_2}$, and accordingly for the remaining factors. Note that for $\mathbf{A}_i^{'}$, trivially holds that $\mathbf{A}_i^{'}(I_s,f) = \mathbf{A}_i^{'}(:,f)$ and similarly for $\mathbf{B}_i^{'}$. After normalizing, the relation between the existing factors and the currently computed is $\mathbf{A}_{old}(I_s,:) = \mathbf{A}_i^{'}\mathbf{\Pi}$ (and similarly for the rest). Each iteration retains a copy of $[\mathbf{A}_{old}(I_s,:), \mathbf{B}_{old}(J_s,:), \mathbf{C}_{old}(K_s,:)]$ which will serve as the anchor for disambiguating the permutation of components. We normalize $[\mathbf{A}_{old}(I_s,:), \mathbf{B}_{old}(J_s,:), \mathbf{C}_{old}(K_s,:)]$ to unit norm as well, and the reason behind that lies in Lemma 21:

Lemma 21 *Consider* $\mathbf{a} = \mathbf{A}_i^{'}(:, f_1)$ *and* $\mathbf{b} = \mathbf{A}_{old}(:, f_2)$. *If* f_1 *and* f_2 *correspond to the same latent CP factor, in the noiseless case, then* $\mathbf{a}^T\mathbf{b} = 1$ *otherwise* $\mathbf{a}^T\mathbf{b} < 1$.

Proof. From Cauchy-Schwartz inequality $\mathbf{a}^T\mathbf{b} \leq \|\mathbf{a}\|_2\|\mathbf{b}\|_2$. The above inequality is maximized when $\mathbf{a} = \mathbf{b}$ and for unit norm $\mathbf{a}, \mathbf{b}$, $\mathbf{a}^T\mathbf{b} \leq 1$. Therefore, if $\mathbf{a} = \mathbf{b}$, which happens when f_1 and f_2 correspond to the same latent CP factor, $\mathbf{a}^T\mathbf{b} = 1$. ■

Lemma 21 is a guide for identifying the permutation matrix $\mathbf{\Pi}$: For every column of $\mathbf{A}_i^{'}$ we compute the inner product with every column of $\mathbf{A}_{old}(I_s, :)$ and compute a matching when the inner product is equal (or close) to 1. Given a large-enough number of rows for $\mathbf{A}_i^{'}$ (which is usually the case, since we require a large-enough sample of the tensor in order to augment it with the update and compute the factor updates accurately), this matching can be computed reliably even in noisy real-world data, as we show in the experimental evaluation.

Update results: After appropriately permuting the columns of $\mathbf{A}_i^{'}, \mathbf{B}_i^{'}, \mathbf{C}_i^{'}$, we have all the information needed to update our model. Returning to the problem definition of Section 8.3, $\mathbf{A}_i^{'}$ contains the updates to the rows within I_s for $\mathbf{A}(t)$ (and similarly for $\mathbf{B}$ and $\mathbf{C}$). Even though $\mathbf{A}, \mathbf{B}$ do not increase their number of rows over time, the incoming slices may contribute valuable new estimates to the already estimated factors. Thus, for the already existing portions of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ we only update the zero entries that fall within the range of I_s, J_s, and K_s respectively. Finally, $\mathbf{C}_i^{'}([K + 1 \cdots K_{new}], :)$ contains the factors for the newly arrived slices, which need to be merged to the already existing columns of $\mathbf{C}$. After properly permuting the columns of $\mathbf{C}_i^{'}$, we accumulate the lower portion of the

$\mathbf{C}_i^{'}$ (corresponding to the new slices) into $\mathbf{C}_{new}$ and we take the column-wise average of the rows to-be-appended to $\mathbf{C}$, across repetitions. Finally, we update $\mathbf{C}(t') = \begin{bmatrix} \mathbf{C}_{old} \\ \mathbf{C}_{new} \end{bmatrix}$.

Guarantees for Correctness Lemma 21 is essentially providing a guarantee for the correctness of SamBaTen. In particular, Lemma 21 ensures that, under mild conditions that make the CP decomposition unique and identifiable, SamBaTen will discover the correct latent factors, disambiguate the permutation and scaling ambiguity, and update the correct columns of the factor matrix that is to be augmented. As we will also empirically demonstrate in the experimental evaluation, indeed, SamBaTen is able to produce correct results that are on par with state-of-the-art methods.

8.4.2 Dealing with rank deficient updates

So far, the above algorithm description is based on the assumption that each one of the sampled tensors $\underline{\mathbf{X}}_s$ we obtain are full-rank, and the CP decomposition of that tensor is identifiable (assumption that is also central to previous works [196]). However, this assumption glosses over the fact that in reality, updates to our tensor may be rank-deficient. In other words, even though $\mathbf{A}(t), \mathbf{B}(t), \mathbf{C}(t)$ have R components, the update may contain R_{new} components, where $R_{new} < R$. If that happens, then the matching as described above is going to fail, and this inevitably leads to very low quality results. Here, we tackle this issue by adding an extra layer include graphics of quality control when we compute the CP decomposition, right before line 5 of Algorithm 8: we estimate the number of components R_{new} in $\underline{\mathbf{X}}_s$ and instead of the "universal" rank R, which may result in low quality factors, we use R_{new} and we accordingly match only those R_{new} to their most likely

matches within the existing R components. Estimating the number of components is a very hard problem, however, there exist efficient heuristics in the literature, such as the Core Consistency Diagnostic (CorConDia) [31] which gives a quality rating for a computed CP decomposition. By successively trying different candidate ranks for $\underline{\mathbf{X}}_s$, we estimate its actual rank, as shown in Algorithm 9 (GetRank), and use that instead. For efficiency we use a recent implementation of CorConDia that is especially tailored for exploiting sparsity [195]. In the experimental evaluation we demonstrate that using GetRank indeed results in higher-quality latent factors

Algorithm 9: GetRank method

Data: Tensor $\underline{\mathbf{X}}$, maximum rank R , maximum no. of iterations it.

Result: R_{new}

```
for 1 to R do
    for j = 1 to it do
        Run CP Decomposition on X with rank i and obtain f_i
        Run CorConDia (X_s, f_i) and obtain p(i, j)
    end
end
Sort p and get top 1 index idx_1.
return R_new = idx_1
```

8.5 Experiments

In this section we extensively evaluate the performance of SamBaTen on multiple synthetic and real datasets, and compare its performance with state-of-the-art approaches.

We experiment on the different parameters of SamBaTen and the baselines, and how that affects performance. We implemented SamBaTen in Matlab using the functionality of the Tensor Toolbox for Matlab [21] which supports very efficient computations for sparse tensors. Our implementation is available at link [2]. We used Intel(R) Xeon(R), CPU E5-2680 v3 @ 2.50GHz machine with 48 CPU cores and 378GB RAM.

8.5.1 Data-set description

Below, we describe the process for generating synthetic data and the real datasets we used.

Synthetic data generation

We generate random tensors of dimension $I = J = K$ with increasing I. Those tensors are created from a known set of randomly generated factors, so that we have full control over the ground truth of the full decomposition. We dynamically calculate the size of batch or slice for our all experiments to fit the data into memory. The machine used in this experiments has ≈380GB of memory. For example, in case of $I = J = K = 50000$, batch size is up to 5, the dense incoming slice requires memory space equivalent to ≈80 GB and the rest of memory space is used for the computations. The specifications of each synthetic dataset are given in Table 8.1.

[2] http://www.cs.ucr.edu/~egujr001/ucr/madlab/src/SAMBATEN.zip

Dimension ($I = J = K$)	Density-dense	Density-sparse	Batch size	Sampling factor
100	100%	65%	50	2
500	100%	65%	150	2
1000	100%	55%	150	2
3000	100%	55%	100	5
5000	100%	55%	100	5
10000	100%	55%	10	2
50000	100%	35%	5	2
100000	100%	35%	5	2

Table 8.1: Table of Datasets analyzed

Real Data Description

In order to truly evaluate the effectiveness of SamBaTen, we test its performance against six real datasets that have been used in the literature. Those datasets are summarized in Table 8.2 and are publicly available at [236] .

8.5.2 Evaluation Measures

We evaluate SamBaTen and the baselines using three criteria: Relative Error, Wall-Clock time and Fitness. These measures provide a quantitative way to compare the performance of our method. More Specifically, **Relative Error** is effectiveness measurement and defined as :

$$RelativeError = \frac{||\underline{\mathbf{X}}_{original} - \underline{\mathbf{X}}_{predicted}||}{||\underline{\mathbf{X}}_{original}||}$$

Name	Description	Dimensions	NNZ	Batch size	Sampling factor	Dataset File Size
NIPS [79]	(Paper,Author,Word)	2.4K x 2.8K x 14Km	3,101,609	500	10	57MB
NELL [35]	(Entity,Relation,Entity)	12K x 9K x 28K	76,879,419	500	10	1.4GB
Facebook-wall [259]	(Wall owner, Poster, day)	62K x 62K x 1,070	78,067,090	100	5	2.1GB
Facebook-links [259]	(User, Links, Day)	62K x 62K x 650	263,544,295	50	2	3.8GB
Patents[236]	(Term ,Term, Year)	239K x 239K x 46	3,596,640,708	10	2	73GB
Amazon[172]	(User, Product ,Word)	4.8M x 1.7M x 1.8M	1,741,809,018	50000	20	43GB

Table 8.2: Real datasets analyzed

where, the lower the value, the better.

CPU time (sec): The average running time denoted by T_{tot} for processing all slices for given tensor, measured in seconds, and is used to validate the time efficiency of an algorithm.

Relative Fitness: Relative Fitness is defined as:

$$RelativeFitness = \frac{||\underline{\mathbf{X}}_{original} - \underline{\mathbf{X}}_{\text{SamBaTen}}||}{||\underline{\mathbf{X}}_{original} - \underline{\mathbf{X}}_{BaseLine}||}$$

where, again, lower is better.

8.5.3 Baselines for Comparison

Here we briefly present the state-of-the-art baselines we used for comparison. Note that for each baseline we use the *reported parameters* that yielded the best performance in the respective publications. For fairness, we compare against the parameter configuration for SamBaTen that yielded the best performance in terms of low wall-clock timing, low relative error and fitness. Note that all comparisons were carried out over 10 iterations

each, and each number reported is an average with a standard deviation attached to it.

CP_ALS [21]: is considered the most standard and well optimized algorithm for CP. We use the implementation of the Tensor Toolbox for Matlab [21]. Here, we simply re-compute CP using CP_ALS after every update.

SDT [186]: Simultaneous Diagonalization Tracking (SDT) is based on incrementally tracking the Singular Value Decomposition (SVD) of the unfolded tensor $\underline{\mathbf{X}}_{(3)} = U \sum V^T$.

RLST [186]: Recursive Least Squares Tracking (RLST) is another online approach in which recursive updates are computed to minimize the Mean Squared Error (MSE) on incoming slice.

OnilneCP [284]: This is the most recent and state-of-the-art method in online computation of CP.

We conduct our experiments on multiple synthetic datasets and six real-world tensors datasets. We set the tolerance rate for convergence between consecutive iterations to 10^{-5} and the maximum number of iteration to 1000 for all the algorithms. The batch size and sampling factor is selected based on dimensions of first mode i.e. I, provided in Table 8.1 and 8.2 for synthetic and real dataset respectively. We use the publicly available implementations for the baselines, as provided by the authors. We only modified the interface of the baselines, so that it is consistent across all methods with respect to the way that they receive the incoming slices. No other functionality has been changed.

8.5.4 Experimental Results

The following major three aspects are analyzed.

Q1. Effectiveness and Accuracy: How effective is SamBaTen as compared to the

baselines on different synthetic and real world datasets?

Q2. Speed & Scalability: How fast is SamBaTen when compared to the state-of-the-art methods on very large sized datasets?

Q3. Parameter Sensitivity: What is the influence of sampling factor s and sampling repetitions r?

Baselines for Comparison

For all datasets we compute Relative Error,CPU time (sec) and Fitness. For SamBaTen, CP_ALS , OnlineCP, RSLT and SDT we use 10% of the data in each dataset as existing dataset. We experimented for both dense as well as sparse tensor to check the performance of our method. The results for the dense and sparse synthetic data are shown in Table 8.3 - 8.4. For each of datasets , the best result is shown in bold. OnlineCP, SDT and RLST address the issue very well. Compared with CP_ALS, SDT and RLST reduce the mean running time by up to 2x times and OnlineCP reduce mean time by up to 3× for small dataset (I up to 3000). Performance of RLST was better than SDT algorithm on 8 out of 8 third-order synthetic tensor datasets. In fact, the efficiency (in terms of CPU time (sec)) of SDT is quite close to RLST. However, the main issue of SDT and RLST is their estimation of relative error and fitness. For some datasets, such as $I = 100$ and $I = 3000$, they perform well, while for some others, they exhibit poor fitness and relative error, achieving only nearly half of the fitness of other methods. For *small size* datasets, OnlineCP's efficiency and accuracy is better than all methods. As the dimension grows, however, the performance of OnlineCP method reduces,and particularly

for datasets of dimension larger than $5000 \times 5000 \times 5000$. Same behavior is observed for sparse tensors. SamBaTen is comparable to baselines for small dataset and outperformed the baselines for large dataset. CP_ALS is the only baseline able to run on datasets up to size $3000 \times 3000 \times 3000$. These results answer **Q1** as the SamBaTen have comparable accuracy to other baseline methods.

I=J=K	CP_{ALS}	OnlineCP	SDT	RLST	SamBaTen
100	0.109 ± 0.01	**0.107± 0.02**	0.173± 0.02	0.151± 0.02	0.115± 0.02
500	**0.102 ± 0.09**	**0.102± 0.09**	0.217± 0.06	0.217± 0.06	**0.102± 0.09**
1000	0.103± 0.01	0.103± 0.01	0.287± 0.01	0.296± 0.01	**0.102± 0.01**
3000	0.119± 0.01	**0.108± 0.01**	0.189± 0.01	0.206± 0.01	0.109± 0.01
5000	N/A	0.122± 0.002	0.201± 0.002	0.196± 0.04	**0.115± 0.009**
10000	N/A	0.173± 0.04	0.225± 0.04	0.252± 0.06	**0.162± 0.01**
50000	N/A	0.215± 0.03	0.229± 0.03	0.26± 0.01	**0.169± 0.01**
100000	N/A	N/A	N/A	N/A	**0.275 ± 0.00**

Table 8.3: Experimental results for relative error for synthetic dense tensor. We see that SamBaTen gives comparable accuracy to baseline.

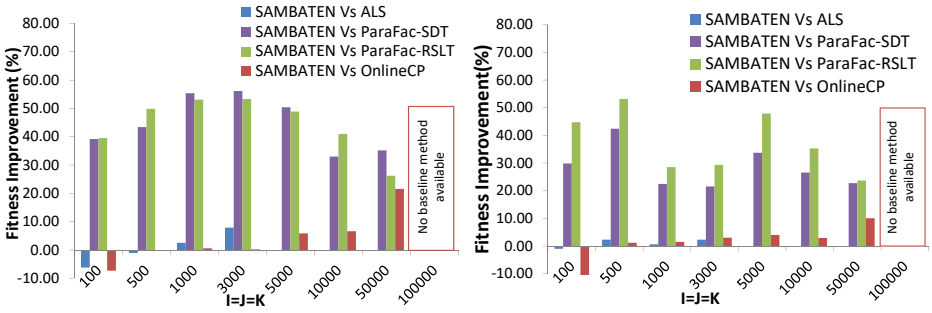


Figure 8.4: Experimental results for relative fitness improvement for (a) dense tensor (b) sparse tensor

I=J=K	CP_{ALS}	OnlineCP	SDT	RSLT	SamBaTen
100	0.169± 0.01	**0.154± 0.02**	0.306± 0.01	0.313± 0.01	0.178± 0.01
500	**0.175± 0.01**	0.188± 0.01	0.43± 0.01	0.421± 0.01	0.184± 0.01
1000	0.179± 0.01	0.185± 0.01	0.613± 0.01	0.813± 0.01	**0.178± 0.01**
3000	0.177± 0.02	**0.171± 0.04**	0.446± 0.03	0.513± 0.02	0.176± 0.03
5000	N/A	0.192± 0.02	0.494± 0.09	0.535± 0.13	**0.187± 0.04**
10000	N/A	**0.173± 0.01**	0.212± 0.01	0.224± 0.11	0.198± 0.12
50000	N/A	0.222± 0.00	0.262± 0.00	0.259± 0.00	**0.200± 0.00**
100000	N/A	N/A	N/A	N/A	**0.283± 0.00**

Table 8.4: Experimental results for relative error for synthetic sparse tensor. We see that SamBaTen works better in very large scale dataset such as 50000 × 50000 × 50000.

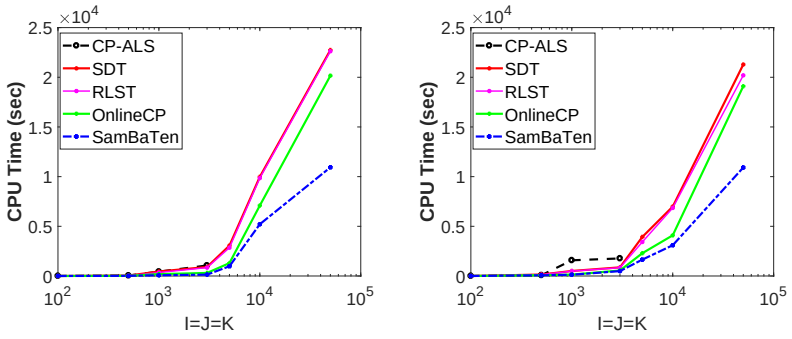


Figure 8.3: Experimental results for CPU time (sec) for (a) dense tensor (b) sparse tensor

SamBaTen is efficiently able to compute $100K \times 100K \times 100K$ sized tensor with batch size of 5 and sampling factor 2. It took **58095.72s** and **47232.2s** to compute online decomposition for dense and sparse tensor, respectively. On other hand, state-of-art methods are unable to handle such large scaled incoming data.

Table 8.5 shows the comparison between methods. SamBaTen outperforms other state-of-the-art approaches in most of the real dataset. In the case of NIPS datset, SamBaTen gives better results compared to the baselines, specifically in terms of CPU Time (*faster up to 20 times*) and Fitness (*better up to 15-20%*). SamBaTen outperforms for NELL, Facebook-Wall and Facebook-Links dataset in terms of efficiency comparable to CP_ALS. For the NIPS dataset, SamBaTen is *25-30 times faster* than OnlineCP method. Due to high dimensions of dataset, RSLT and SDT are unable to execute further. Note that all the real datasets we use are highly sparse, however, no baselines except CP_ALS actually take advantage of that sparsity, therefore, repeated CP_ALS tends to be faster because the baselines have to deal with dense computations which tend to be slower, when the data contain a lot of zeros. Most importantly, however, SamBaTen performed very well on Amazon and Patent datasets, arguably the hardest of the six real datasets we examined and have been analyzed in the literature, *where none of the baselines was able to run.* These result answer **Q1** and **Q2** and show that SamBaTen is able to handle large dimensions and sparsity.

Sensitivity of Sampling Factor s

The sampling factor plays an important role in SamBaTen. We performed experiments to evaluate the impact of changing sampling factor on SamBaTen. For these experiments , we fixed batch size to 50 for all datasets. We see in figure 8.5 that increasing sampling factor results in reduction of CPU time (as sparsity of sub sampled tensor increased) and it reduces the fitness of output up-to 2-3%. In sum, these observations demonstrate that: 1) a suitable sampling factor on sub-sampled tensor could improve the fitness and result in better tensor decomposition, and 2) the higher sampling factor is, the lower the CPU time. This result partially answers **Q3**.

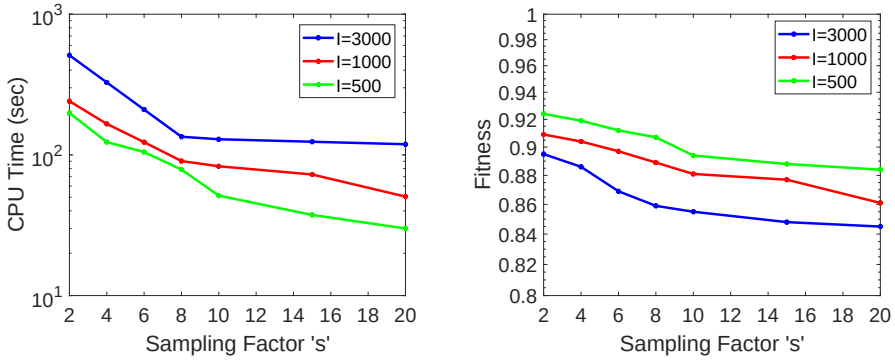


Figure 8.5: SamBaTen CPU Time (sec) and Relative Fitness vs. Sampling Factor s on different datasets (*lower is better*).

Sensitivity of Repetition Factor r

We evaluate the performance for parameter setting r i.e the number of parallel decompositions. For these experiments, we choose batch size and sampling rate for synthetic $500 \times 500 \times 500$ dataset and NIPS real world dataset as provided in table 8.1 and

8.2, respectively. We can see that with higher values of the repetition factor r, Relative Fitness (SamBaTen vs CP_ALS) is improved as shown in Figure 8.6(a). We experiment on varying repetition factor r with Sampling factor s on NIPS real world dataset to check the performance of our method as shown in Figure 8.6(b). The lower the relative fitness score, the better the decomposition. This result completes the answer to **Q3**.

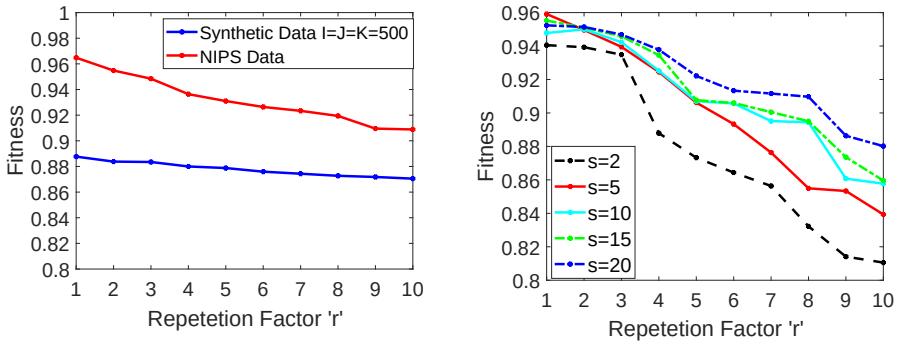


Figure 8.6: SamBaTen Relative Fitness vs. repetition factor r on synthetic and NIPS datasets (*lower is better*).

8.6 Conclusions

We introduce SamBaTen, a novel sample-based incremental CP tensor decomposition. We show its effectiveness with respect to approximation quality, with its performance being on par with state-of-the-art incremental and non-incremental algorithms, and we demonstrate its efficiency and scalability by outperforming state-of-the-art approaches (*25-30 times faster*) and being able to run very large incremental tensors where none of the baselines was able to produce results.

Chapter based on material published in SDM 2018 [94].

Dataset	CPU Time (sec)					Fitness SamBaTen w.r.t			
	CP_{ALS}	OnlineCP	SDT	RSLT	SamBaTen	CP_{ALS}	OnlineCP	SDT	RSLT
NIPS	177.46± 2.9	372.03± 8.9	1608.23± 37.5	1596.07± 15.2	**43.98± 0.6**	0.96± 0.01	0.98± 0.01	**0.78± 0.02**	0.82± 0.01
NELL	8783.27± 11.2	42325.22± 70.0	65325.22± 25.2	63485.98± 10.6	**983.83± 30.7**	0.95±0.02	0.81± 0.01	**0.76± 0.02**	0.81± 0.01
Facebook-wall	3041.98±3.8	N/A	N/A	N/A	**736.07±4.1**	0.97±0.01	N/A	N/A	N/A
Facebook-links	2689.69±7.9	N/A	N/A	N/A	**343.32±6.3**	0.96±0.06	N/A	N/A	N/A
Amazon	N/A	N/A	N/A	N/A	**4892.07±61.8**	N/A	N/A	N/A	N/A
Patent	N/A	N/A	N/A	N/A	**8068.27±55.4**	N/A	N/A	N/A	N/A

Table 8.5: SamBaTen performance for real datasets. SamBaTen outperforms the baselines for all the large tensors.

Chapter 9

Online Compression-based Tensor Decomposition

"How to leverage random compression for streaming high-order tensor decomposition? Can we guarantee the identifiability and speed-up with parallelism?"

Tensor decompositions are powerful tools for large data analytics as they jointly model multiple aspects of data into one framework and enable the discovery of the latent structure and higher-order correlations within the data. One of the most widely studied and used decompositions, especially in data mining and machine learning, is the Canonical Polyadic or CP decomposition. However, today's datasets are not static and these datasets often dynamically growing and changing with time. To operate on such large data, we present OCTen the first ever compression-based online parallel implementation for the CP decomposition.We conduct an extensive empirical analysis of the algorithms in terms of fitness, memory used and CPU time, and in order demonstrate the compression and

scalability of the method, we apply OCTen to big tensor data. Indicatively, OCTen performs on-par or better than to state-of-the-art online and offline methods in terms of decomposition accuracy and efficiency, while saving up to *40-250 %* memory space. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, Ravdeep Pasricha, Tianxiong Yang, and Evangelos E. Papalexakis. "Octen: Online compression-based tensor decomposition." In 2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pp. 455-459. IEEE, 2019.

9.1 Introduction

A Tensor is a multi-way array of elements that represents higher-order or multi-aspect data. In recent years, tensor decompositions have gained increasing popularity in big data analytics [197]. In higher-order structure, tensor decomposition are capable of finding complex patterns and higher-order correlations within the data. Much like matrix factorization tools like the Singular Value Decomposition, there exist generalizations for the tensor domain, with the most widely used being CANDECOMP/PARAFAC or CP [104] which extracts interpretable factors form tensor data, and Tucker decomposition [256], which is known for estimating the joint subspaces of tensor data. In this work we focus only on the CP decomposition, which is extremely effective in exploratory knowledge discovery on multi-aspect data.

In the era of information explosion, data are generated or modified at large volume. In such environments, data may be increased or decreased in any of its dimensions with high

velocity. When using tensors to represent this dynamically changing data, an instance of the problem is that of a "streaming", "incremental", or "online" tensors[1]. Considering an example of time evolving social network interactions, where a large number of users interact with each other every second (Facebook users update $\approx 684K$ piece of content and Twitter users send $\approx 100K$ tweets every single minute[2]); every such snapshot of interactions is a new incoming slice(s) to the tensor on its "time" or 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. In this chapter, our goal is, given an already computed CP decomposition, to *track* the CP decomposition of an online tensor, as it receives streaming updates, 1) *efficiently*, being much faster than re-computing the entire decomposition from scratch every update, and utilizing small amount of memory, and 2) *accurately*, incurring an approximation error that is as close as possible to the decomposition of the full tensor. For exposition purposes, we focus on the time-evolving/streaming scenario, where a three-mode tensor grows on the third ("time") mode, however, our work extends to cases where more than one modes is online.

As the volume and velocity of data grow, the need for time- and space-efficient online tensor decomposition is imperative. There already exists a modest amount of prior work in online tensor decomposition both for Tucker [18, 243] and CP [186, 284]. However, most of the existing online methods [18, 284, 186] , model the data in the full space, which can become very memory taxing as the size of the data grows. There exist memory efficient

[1]Notice that the literature (and thereby this chapter) uses the above terms as well as "dynamic" interchangeably.

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

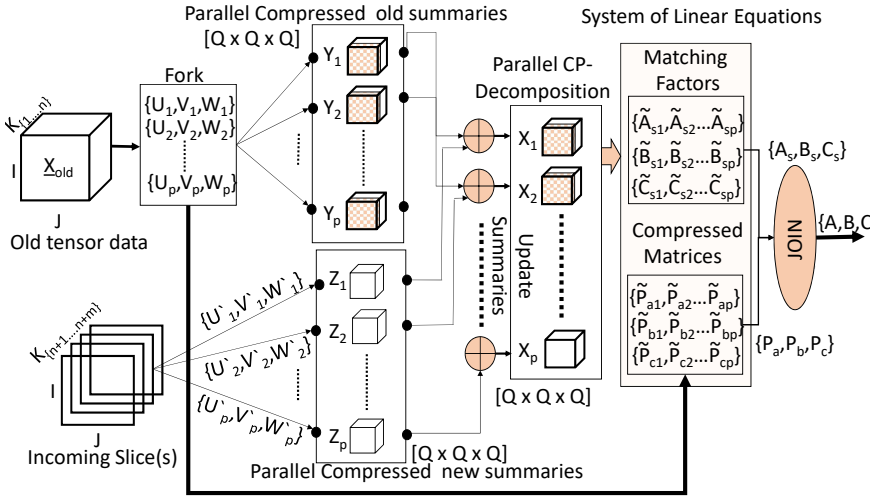


Figure 9.1: OCTEN framework. Compressed tensor summaries $\underline{\mathbf{Y_p}}$ and $\underline{\mathbf{Z_p}}$ is obtained by applying compression matrices $(\mathbf{U}_p, \mathbf{V}_p, \mathbf{W}_p)$ and $(\mathbf{U}_p^{'}, \mathbf{V}_p^{'}, \mathbf{W}_p^{'})$ to $\underline{\mathbf{X_{old}}}$ and $\underline{\mathbf{X_{new}}}$ or incoming slice(s) respectively. The updated summaries are computed by $\underline{\mathbf{X_p}} = \underline{\mathbf{Y_p}} + \underline{\mathbf{Z_p}}$. Each $\underline{\mathbf{X_p}}$ is independently decomposed in parallel. The update step anchors all factor matrices to a single reference, and solves a linear equation for the overall A, B, and C.

tensor decompositions, indicatively MET for Tucker [140] and PARACOMP [230] for CP, neither of which are able to handle online tensors. In this chapter, we fill exactly that gap.

Online tensor decomposition is a challenging task due to the following reasons. First, maintaining high-accuracy (competitive to decomposing the full tensor) using significantly fewer computations and memory than the full decomposition calls for innovative and, ideally, sub-linear approaches. Second, operating on the full ambient space of the data, as the tensor is being updated online, leads to super-linear increase in time and space complexity, rendering such approaches hard to scale, and calling for efficient methods that work on spaces that are significantly smaller than the original ambient data dimensions. Third, in many real settings, more than one modes of the tensor may receive streaming updates

at different points in time, and devising a flexible algorithm that can handle such updates seamlessly is a challenge. To handle the above challenges, in this chapter, we propose to explore how to decompose online or incremental tensors based on CP decomposition. We specifically study: (1) How to make parallel update method based on CP decomposition for online tensors? (2) How to identify latent component effectively and accurately after decomposition? Answering the above questions, we propose OCTen (Online Compression-based Tensor Decomposition) framework. Our contributions are summarized as follows:

- **Novel Parallel Algorithm** We introduce OCTen, a novel compression-based algorithm for online tensor decomposotion that admits an efficient parallel implementation. We do not limit to 3-mode tensors, our algorithm can easily handle higher-order tensor decompositions.
- **Correctness guarantees** By virtue of using random compression, OCTen can guarantee the *identifiability* of the underlying CP decomposition in the presence of streaming updates.
- **Extensive Evaluation** Through experimental evaluation on various datasets, we show that OCTen provides stable decompositions (with quality on par with state-of-the-art), while offering up to *40-250 %* memory space savings.

Reproducibility: We make our Matlab implementation publicly available at link [3]. Furthermore, all the small size datasets we use for evaluation are publicly available on same link.

[3] http://www.cs.ucr.edu/~egujr001/ucr/madlab/src/OCTen.zip

9.2 Related work

In this section, we provide review of the work related to our algorithm. At large, incremental tensor methods in the literature can be categorized into two main categories as described below:

Tensor Decomposition: Phan *el at.* [203] had purposed a theoretic method namely GridTF to large-scale tensors decomposition based on CP's basic mathematical theory to get sub-tensors and join the output of all decompositions to achieve final factor matrices. Sidiropoulos *el at.*[186], proposed algorithm that focus on CP decomposition namely RLST (Recursive Least Squares Tracking), which recursively updates the factors by minimizing the mean squared error. In 2014, Sidiropoulos *el at.* [230] , proposed a parallel algorithm for low-rank tensor decomposition that is suitable for large tensors. The Zhou, *el at.* [284] describes an online CP decomposition method, where the latent components are updated for incoming data. The most related work to ours was proposed by [94] which is sampling-based batch incremental tensor decomposition algorithm. These state of art techniques focus on only fast computation but not effective memory usage. Besides CP decomposition, tucker decomposition methods[243, 192] were also introduced. Some of these methods were not only able to handle data increasing in one-mode, but also have solution for multiple-mode updates using methods such as incremental SVD [70]. Latest line of work is introduced in [18] i.e TuckerMPI to find inherent low-dimensional multi-linear structure, achieving high compression ratios. Tucker is mostly focused on recovering subspaces of the tensor, rather than latent factors, whereas our focus is on the CP decomposition which is more suitable for exploratory analysis.

Tensor Completion: Another field of study is tensor completion, where real-world large-scale datasets are considered incomplete. In literature, wide range of methods have been proposed based on online tensor imputation[168] and tensor completion with auxiliary information[181, 4]. The most recent method in this line of work is by Qingquan *el at.*[238], who proposed a low-rank tensor completion with general multi-aspect streaming patterns, based on block partitioning of the tensor. However, these approaches can not directly applied when new batches of data arrived. This provides us a good starting point for further research.

9.3 Problem Formulation

In many real-world applications, data grow dynamically and may do so in many modes. For example, given a dynamic tensor in a movie-based recommendation system, organized as *users* $\times$ *movie* $\times$ *rating* $\times$ *hours*,the number of registered users, movies watched or rated, and hours may all increase over time. Another example is network monitoring sensor data where tons of information like source and target IP address, users ,ports etc., is collected every second. This nature of data gives rise to update existing decompositions on the fly or online and we call it incremental decomposition [4]. In such conditions, the update needs to process the new data very quickly, which makes non-incremental methods to fall short because they need to recompute the decomposition for the full dataset. The problem that we solve is the following:

[4]We uses the terms "incremental", "dynamic", and "online" interchangeably in chapter

Problem 22 Given *(a) an existing set of* summaries $\{\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2 \ldots \underline{\mathbf{Y}}_p\}$ *of* R *components, which approximate tensor* $\underline{\mathbf{X}}_{old}$ *of size* $\{ I^{(1)} \times I^{(2)} \times \ldots I^{(N-1)} \times t_{old}\}$ *at time* t *, (b) new incoming batch of slice(s) in form of tensor* $\underline{\mathbf{X}}_{new}$ *of size* $\{I^{(1)} \times I^{(2)} \times \ldots I^{(N-1)} \times t_{new}\}$*, find updates of* $(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)})$ **incrementally** *to approximate tensor* $\underline{\mathbf{X}}$ *of dimension* $\{I^{(1)} \times I^{(2)} \times \ldots I^{(N-1)} \times I^{(N)}\}$*, where* $I^{(N)} = (t_{old} + t_{new}) = I^{(N)}_{1\ldots n} + I^{(N)}_{(n+1)\ldots m}$ *after appending new slice or tensor to* N^{th} *mode.*

9.4 Proposed Method: OCTen

As we mention in the previous section, to the best of our knowledge, there is no algorithm in the literature that is able to efficiently compress and incrementally update the CP decomposition in the presence of incoming tensor slices. However, there exists a method for static data [230]. Since this method considers the tensor in its entirety, it cannot handle streaming data and as the data size grows its efficiency may degrade, since it handles the full data in one shot. In this section, we introduce OCTen, a new method for parallel incremental decomposition designed with two main goals in mind: **G1**: Compression, speed , simplicity, and parallelization; and **G2**: correctness in recovering compressed partial results for incoming data, under suitable conditions.

The algorithmic framework we propose is shown in Figure 9.1 and is described below:

We assume that we have a pre-existing set of *summaries* of the $\underline{\mathbf{X}}$ before the update. Summaries are in the form of compressed tensors of dimension $[Q \times Q \times Q]$. For simplicity of description, we assume that we are receiving updated slices on the third mode. We, further,

assume that the updates come in batches of new slices, which, in turn, ensures that we see a mature-enough update to the tensor, which contains useful structure. Trivially, however, OCTen can operate on singleton batches and for > 3 modes also.

In the following, $\underline{\mathbf{X}}_{old}$ is the tensor prior to the update and $\underline{\mathbf{X}}_{new}$ is the batch of incoming slice(s). Considering $S = \prod_{i=1}^{[N-1]} I^{(i)}$ and $T = \sum_{i=1}^{[N-1]} I^{(i)}$, we can write space and time complexity in terms of S and T. Given an incoming batch, OCTen performs the following steps:

Parallel Compression and Decomposition: When handling large tensors $\underline{\mathbf{X}}$ that are unable fit in main memory, we may compress the tensor $\underline{\mathbf{X}}$ to a smaller tensor that somehow apprehends most of the systematic variation in $\underline{\mathbf{X}}$. Keeping this in mind, for incoming slice(s) $\underline{\mathbf{X}}_{new}$, during the parallel compression step, we first need to create 'p' parallel triplets of random compression matrices $\{\mathbf{U_p}, \mathbf{V_p}, \mathbf{W_p}\}$ of $\underline{\mathbf{X}}$. Thus, each worker (i.e. Matlab parpool) is responsible for creating and storing the three compression matrices of dimension $\mathbb{R}^{I \times Q}$, $\mathbb{R}^{J \times Q}$ and $\mathbb{R}^{t_{new} \times Q}$. These matrices share at least 'shared' amount of columns among each other. Mathematically, we can describe it as follows:

$$\underline{\mathbf{X}} = \begin{bmatrix} \{\mathbf{U_1}, \mathbf{V_1}, \mathbf{W_1}\} \\ \{\mathbf{U_2}, \mathbf{V_2}, \mathbf{W_2}\} \\ \cdots \\ \{\mathbf{U_p}, \mathbf{V_p}, \mathbf{W_p}\} \end{bmatrix} = \begin{bmatrix} \{(\mathbf{u} \;\; \mathbf{U_{1'}}), (\mathbf{v} \;\; \mathbf{V_{1'}}), (\mathbf{w} \;\; \mathbf{W_{1'}})\} \\ \{(\mathbf{u} \;\; \mathbf{U_{2'}}), (\mathbf{v} \;\; \mathbf{V_{2'}}), (\mathbf{w} \;\; \mathbf{W_{2'}})\} \\ \cdots \\ \{(\mathbf{u} \;\; \mathbf{U_{p'}}), (\mathbf{v} \;\; \mathbf{V_{p'}}), (\mathbf{w} \;\; \mathbf{W_{p'}})\} \end{bmatrix} \tag{9.1}$$

where u, v and w are shared and have dimensions of $\mathbb{R}^{I \times Q_{shared}}$, $\mathbb{R}^{J \times Q_{shared}}$ and $\mathbb{R}^{t_{new} \times Q_{shared}}$.

For compression matrices, we choose to assign each worker to create a single row of each of the matrices to reduce the burden of creating an entire batch of $\{\mathbf{U_p'}, \mathbf{V_p'}, \mathbf{W_p'}\}$ of $\underline{\mathbf{X}}_{new}$. We see that for each worker is sufficient to hold $\{\mathbf{U_p}, \mathbf{V_p}, \mathbf{W_p}\}$ in memory. Now,

we created compressed tensors or replicas $\{\underline{\mathbf{Z_1}}, \underline{\mathbf{Z_2}} \dots \underline{\mathbf{Z_p}}\}$ from each triplets of 3-mode compression matrices generated from $\underline{\mathbf{X}}_{new}$;see Figure 9.1. $\underline{\mathbf{Z}}_p$ is 3-mode tensor of size $\mathbb{R}^{Q \times Q \times Q}$. Since Q is considerably smaller than [I ,J, K], we use $O(Q^3)$ of memory on each worker.

For $\underline{\mathbf{X}}_{old}$, we already have replicas $\{\underline{\mathbf{Y_1}}, \underline{\mathbf{Y_2}} \dots \underline{\mathbf{Y_p}}\}$ from each triplets of 3-mode compression matrices $\{\mathbf{U_p}, \mathbf{V_p}, \mathbf{W_p}\}$ generated from $\underline{\mathbf{X}}_{old}$;see Figure 9.1. In general, the compression comprises N-mode products, leads to complexity of $(Q_{(1)}St_{new} + Q_{(2)}St_{new} + Q_{(3)}St_{new} + \dots Q_{(N)}St_{new})$ overall for dense tensor $\underline{\mathbf{X}}$, if the first mode is compressed first, followed by the second, and then the third mode and so on. We choose to keep $Q_1, Q_2, Q_3 \dots Q_N$ of same order as well non-temporal dimensions are of same order in our algorithm, so time complexity of parallel compression step for N-mode data is $O(QSt_{new})$ for each worker. The *summaries* are always dense, because first mode product with tensor is dense, hence remaining mode products unable to exploit sparsity.

After appropriately computing *summaries* $\{\underline{\mathbf{Z_1}}, \underline{\mathbf{Z_2}} \dots \underline{\mathbf{Z_p}}\}$ for incoming slices, we need to update the old summaries $\{\underline{\mathbf{Y_1}}, \underline{\mathbf{Y_2}} \dots \underline{\mathbf{Y_p}}\}$. Each worker reads its segment and processing update parallel as given below.

$$\begin{bmatrix} \underline{\mathbf{X_1}} \\ \underline{\mathbf{X_2}} \\ \vdots \\ \underline{\mathbf{X_p}} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{Y_1}} \\ \underline{\mathbf{Y_2}} \\ \vdots \\ \underline{\mathbf{Y_p}} \end{bmatrix} \oplus \begin{bmatrix} \underline{\mathbf{Z_1}} \\ \underline{\mathbf{Z_2}} \\ \vdots \\ \underline{\mathbf{Z_p}} \end{bmatrix} \odot \begin{bmatrix} \mathbf{W_1'}(\mathbf{k}, \mathbf{q}) \\ \mathbf{W_2'}(\mathbf{k}, \mathbf{q}) \\ \vdots \\ \mathbf{W_p'}(\mathbf{k}, \mathbf{q}) \end{bmatrix} \implies \begin{bmatrix} (\mathbf{A_{s(1)}}, \mathbf{B_{s(1)}}, \mathbf{C_{s(1)}}) \\ (\mathbf{A_{s(2)}}, \mathbf{B_{s(2)}}, \mathbf{C_{s(2)}}) \\ \vdots \\ (\mathbf{A_{s(p)}}, \mathbf{B_{s(p)}}, \mathbf{C_{s(p)}}) \end{bmatrix} \tag{9.2}$$

Where k is the number of slices of the incoming tensor and q is the slice number for thecompressed tensor. Further, for the decomposition step, we processed 'p' *summaries* on

different workers, each one fitting the decomposition to the respective compressed tensor $\{\underline{\mathbf{X_1}}, \underline{\mathbf{X_2}} \dots \underline{\mathbf{X_p}}\}$ created by the compression step. We assume that the updated compressed tensor $\{\underline{\mathbf{X_1}}, \underline{\mathbf{X_2}} \dots \underline{\mathbf{X_p}}\}$ fits in the main memory, and performs in-memory computation. We denote p^{th} compressed tensor decompositions as $(\mathbf{A_{s(p)}}, \mathbf{B_{s(p)}}, \mathbf{C_{s(p)}})$ as discussed above. The data for each parallel worker $\underline{\mathbf{X_p}}$ can be uniquely decomposed, i.e. $(\mathbf{A_p}, \mathbf{B_p}, \mathbf{C_p})$ is unique up to scaling and column permutation.

Furthermore, parallel compression and decomposition is able to achieve Goal **G1**.

Algorithm 10: OCTEN for incremental tensor decomposition

Data: $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I \times J \times K_{(n+1)\dots m}}$,summary $\underline{\mathbf{Y}}_i \in \mathbb{R}^{Q \times Q \times Q}$, R,p,Q, shared S.

Result: Factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of size $(I \times R)$, $(J \times R)$ and $(K_{1\dots n,(n+1)\dots m} \times R)$.

while *new slice(s) coming* **do**

  $\underline{\mathbf{Z}}_i \leftarrow \{(\mathbf{U}_i^{'}, \mathbf{V}_i^{'}, \mathbf{W}_i^{'})\}$, $\underline{\mathbf{Z}}_i \in \mathbb{R}^{Q \times Q \times Q}$, $i \in (1, p)$

  $\underline{\mathbf{X}}_i \leftarrow \underline{\mathbf{Y}}_i \oplus \underline{\mathbf{Z}}_i$, $\underline{\mathbf{X}}_i \in \mathbb{R}^{Q \times Q \times Q}$, $i \in (1, p)$

  $(\widetilde{\mathbf{A}}_{s(i)}, \widetilde{\mathbf{B}}_{s(i)}, \widetilde{\mathbf{C}}_{s(i)}) \leftarrow CP(\underline{\mathbf{X}}_i, R)$, $i \in (1, p)$ $(\widetilde{\mathbf{P}}_{a(i)}, \widetilde{\mathbf{P}}_{b(i)}, \widetilde{\mathbf{P}}_{c(i)}) \leftarrow \{(\mathbf{U}^{'}(i, [S, :], :)^T, \mathbf{V}^{'}(i, [S, :], :)^T, \mathbf{W}^{'}(i, [S, :], :)^T)\}$, $i \in (1, p)$ **for** $i \leftarrow 1$ *to* $p - 1$ **do**

    $(\widetilde{\mathbf{A}}_s, \widetilde{\mathbf{B}}_s, \widetilde{\mathbf{C}}_s) \leftarrow \Pi\big[(\widetilde{\mathbf{A}}_{s(i)}, \widetilde{\mathbf{B}}_{s(i)}, \widetilde{\mathbf{C}}_{s(i)})$; $(\mathbf{A}_{s(i+1)}, \mathbf{B}_{s(i+1)}, \mathbf{C}_{s(i+1)})\big]$

    $(\widetilde{\mathbf{P}}_a, \widetilde{\mathbf{P}}_b, \widetilde{\mathbf{P}}_c) \leftarrow \big[(\widetilde{\mathbf{P}}_{a(i)}, \widetilde{\mathbf{P}}_{b(i)}, \widetilde{\mathbf{P}}_{c(i)})$; $(\widetilde{\mathbf{P}}_{a(i+1)}, \widetilde{\mathbf{P}}_{b(i+1)}, \widetilde{\mathbf{P}}_{c(i+1)})\big]$

  end

  $\mathbf{A} \leftarrow \widetilde{\mathbf{P}}_a^{-1} * \widetilde{\mathbf{A}}_s$; $\mathbf{B} \leftarrow \widetilde{\mathbf{P}}_b^{-1} * \widetilde{\mathbf{B}}_s$; $\mathbf{C} \leftarrow [\mathbf{C}_{old}; \widetilde{\mathbf{P}}_c^{-1} * \widetilde{\mathbf{C}}_s]$

end

return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$

Factor match for identifiability: According to Kruskal [105], the CP decomposition is unique (under mild conditions) up to permutation and scaling of the components i.e. $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ factor matrices. Consider an 3-mode tensor $\underline{\mathbf{X}}$ of dimension I, J and K of rank R. If rank

$$r_c = F \implies K \geq R \;\&\; I(I-1)(J-1) \geq 2R(R-1), \tag{9.3}$$

then rank 1 factors of tensor $\underline{\mathbf{X}}$ can be uniquely computable[105, 128]. With help of Kronecker product[27] property i.e. $(\mathbf{U}^T \otimes \mathbf{C}^T \otimes \mathbf{W}^T)(\mathbf{A} \odot \mathbf{B} \odot \mathbf{C}) = ((\mathbf{U}^T\mathbf{A}) \odot (\mathbf{V}^T\mathbf{B}) \odot (\mathbf{W}^T\mathbf{C})) \approx (\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$. Now Kruskal's combining uniqueness and Kronecker product property, we can obtain identifiable factors from *summaries* if

$$\min(Q, r_A) + \min(Q, r_B) + \min(Q, r_C) \geq 2R + 2 \tag{9.4}$$

where Kruskal-rank of A, denoted r_A, is the maximum r such that any r columns of $\mathbf{A}$ are linearly independent;see [230]. Hence, upon factorization of 3-mode $\underline{\mathbf{X_p}}$ into R rank-1 components, we obtain $\mathbf{A} = a_p^T A \Pi_p \lambda_p^{(1/N)}$ where a is shared among summaries decompositions , Π_p is a permutation matrix, and λ_p is a diagonal scaling matrix obtained from CP decomposition. To match factor matrices after decomposition step, we first normalize the shared columns of factor matrices $(\mathbf{A_{s(i)}}, \mathbf{B_{s(i)}}, \mathbf{C_{s(i)}})$ and $(\mathbf{A_{s(i+1)}}, \mathbf{B_{s(i+1)}}, \mathbf{C_{s(i+1)}})$ to unit norm $||.||_1$. Next, for each column of $(\mathbf{A_{s(i+1)}}, \mathbf{B_{s(i+1)}}, \mathbf{C_{s(i+1)}})$, we find the most similar column of $(\mathbf{A_{s(i)}}, \mathbf{B_{s(i)}}, \mathbf{C_{s(i)}})$, and store the correspondence. Finally, We can describe factor matrices as :

$$\widetilde{\mathbf{A}}_{\mathbf{s}} = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_p^T \end{bmatrix} * \mathbf{A}\Pi\lambda^{(1/N)}, \quad \widetilde{\mathbf{B}}_{\mathbf{s}} = \begin{bmatrix} b_1^T \\ b_2^T \\ \vdots \\ b_p^T \end{bmatrix} * \mathbf{B}\Pi\lambda^{(1/N)}, \quad \widetilde{\mathbf{C}}_{\mathbf{s}} = \begin{bmatrix} c_1^T \\ c_2^T \\ \vdots \\ c_p^T \end{bmatrix} * \mathbf{C}\Pi\lambda^{(1/N)} \tag{9.5}$$

where $\widetilde{\mathbf{A}}_s, \widetilde{\mathbf{B}}_s$, and $\widetilde{\mathbf{C}}_s$ are matrices of dimension $\widetilde{\mathbf{A}}_s \in \mathbb{R}^{pQ\times R}$, $\widetilde{\mathbf{B}}_s \in \mathbb{R}^{pQ\times R}$ and $\widetilde{\mathbf{C}}_s \in \mathbb{R}^{pQ\times R}$ respectively and N is number of dimensions of tensor. For 3-mode tensor , N is 3 and for 4-mode tensor, N is 4 and so on. Similarly, for 4-mode, we have $(\widetilde{\mathbf{A}}_{\mathbf{s}}, \widetilde{\mathbf{B}}_{\mathbf{s}}, \widetilde{\mathbf{C}}_{\mathbf{s}}, \widetilde{\mathbf{D}}_{\mathbf{s}})$ and so on. Even though for 3-mode tensor, $\mathbf{A}$ and $\mathbf{B}$ do not increase their number of rows over time, the incoming slices may contribute valuable new estimates to the already estimated factors.Thus, we update the all factor matrices in same way. This is able to partially achieve Goal **G2**.

Update results: Final step is to remove the all singleton dimensions from the sets compression matrices $\{\mathbf{U_p}, \mathbf{V_p}, \mathbf{W_p}\}$ and stack them together. A singleton dimension of tensor or matrix is any dimension for which size of matrix or tensor with given dimension becomes one. Consider the 5-by-1-by-5 array A. After removing its singleton dimension, the array A become 5-by-5. Now, we can write compression matrices as:

$$\widetilde{\mathbf{P}}_a = \begin{bmatrix} \mathbf{U}(:,:,\mathbf{1})^\mathbf{T} \\ \mathbf{U}(:,:,\mathbf{2})^\mathbf{T} \\ \vdots \\ \mathbf{U}(:,:,\mathbf{p})^\mathbf{T} \end{bmatrix}, \quad \widetilde{\mathbf{P}}_b = \begin{bmatrix} \mathbf{V}(:,:,\mathbf{1})^\mathbf{T} \\ \mathbf{V}(:,:,\mathbf{2})^\mathbf{T} \\ \vdots \\ \mathbf{V}(:,:,\mathbf{p})^\mathbf{T} \end{bmatrix}, \quad \widetilde{\mathbf{P}}_c = \begin{bmatrix} \mathbf{W}(:,:,\mathbf{1})^\mathbf{T} \\ \mathbf{W}(:,:,\mathbf{2})^\mathbf{T} \\ \vdots \\ \mathbf{W}(:,:,\mathbf{p})^\mathbf{T} \end{bmatrix} \tag{9.6}$$

where $\widetilde{\mathbf{P}}_a, \widetilde{\mathbf{P}}_b$, and $\widetilde{\mathbf{P}}_c$ are matrices of dimension $\widetilde{\mathbf{P}}_a = \mathbb{R}^{pQ \times I}$, $\widetilde{\mathbf{P}}_b = \mathbb{R}^{pQ \times J}$ and $\widetilde{\mathbf{P}}_c = \mathbb{R}^{pQ \times K}$ respectively. The updated factor matrices (A, B, and C) for 3-mode tensor $\underline{\mathbf{X}}$ (i.e. $\underline{\mathbf{X}}_{old} + \underline{\mathbf{X}}_{new}$) can be obtained by :

$$\mathbf{A} = \widetilde{\mathbf{P}}_a^{-1} * \widetilde{\mathbf{A}}_s, \quad \mathbf{B} = \widetilde{\mathbf{P}}_b^{-1} * \widetilde{\mathbf{B}}_s, \quad \mathbf{C} = \left[\mathbf{C}_{old}; \widetilde{\mathbf{P}}_c^{-1} * \widetilde{\mathbf{C}}_s\right] \tag{9.7}$$

where $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are matrices of dimension $\mathbb{R}^{I \times R}$, $\mathbb{R}^{J \times R}$ and $\mathbb{R}^{K_{1\ldots n,(n+1)\ldots m} \times R}$ respectively. Hence, we achieve Goal **G2**.

Complexity Analysis: As discussed previously, compression step's time and space complexity is $O(QSt_{new})$ and $O(Q^3)$ respectively. Identifiability and update can be calculated in $O(pQI + pQR)$. Hence, time complexity is considered as $O(p^2QI + p^2QIR + QSt_{new})$. Overall, as S is larger than any other factors, the time complexity of OCTen can be written as $O(QSt_{new})$. In terms of space consumption, OCTen is quite efficient since only the compressed matrices, previous factor matrices and summaries need to be stored. Hence, the total cost of space is $pQ(pT + t_{new} + R) + (T + t_{old})R + Q^3$.

Dataset	Time Complexity	Space Complexity	Reference
OCTen	$O(QSt_{new})$	$pQ(pT + t_{new} + R) + (T + t_{old})R + Q^3$	
OnlineCP	$O(NSt_{new}R)$	$St_{new} + (2T + t_{old})R + (N-1)R^2$	[284]
SambaTen	$O(nnz(X)R(N-1)$	$nnz(X) + (T + t_{old} + 2S)R + 2R^2$	[94]
RLST	$O(R^2S)$	$St_{new} + (T + t_{old} + 2S)R + 2R^2$	[186]
ParaComp	$O(QSt_{new}R)$	$St_{new} + (T + t_{old})R + pQ^3$	[230]

Table 9.1: Complexity comparison between OCTen and state-of-art methods.

9.4.1 Extending to Higher-Order Tensors

We now show how our approach is extended to higher-order cases. Consider N-mode tensor $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times T_1}$. The factor matrices are $(\mathbf{A}_{old}^{(1)}, \mathbf{A}_{old}^{(2)}, \ldots, \mathbf{A}_{old}^{(N-1)}, \mathbf{A}_{old}^{(T_1)})$ for CP decomposition with N^{th} mode as new incoming data. A new tensor $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times T_2}$ is added to $\underline{\mathbf{X}}_{old}$ to form new tensor of $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times T}$ where $T = T_1 + T_2$. In addition, sets of *compression matrices* for old data are $\{\mathbf{U}_p^{(1)}, \mathbf{U}_p^{(2)}, \ldots, \mathbf{U}_p^{(N-1)}, \mathbf{U}_p^{(T)}\}$ and for new data it is $\{\mathbf{U}_p^{'(1)}, \mathbf{U}_p^{'(2)}, \ldots, \mathbf{U}_p^{'(N-1)}, U_p^{'(T)}\}$ for p number of summaries.

Each compression matrices are converted into compressed cubes i.e. for $\underline{\mathbf{X}}_{old}$ compressed cube is of dimension $\underline{\mathbf{Y_p}} \in \mathbb{R}^{Q^{(1)} \times Q^{(2)} \cdots \times Q^{(N-1)} \times Q^{(N)}}$ and same follows for $\underline{\mathbf{X}}_{new}$. The updated summaries are computed using $\underline{\mathbf{X_p}} = \underline{\mathbf{Y_p}} + \underline{\mathbf{Z_p}}$ s.t. $\underline{\mathbf{X_p}} \in \mathbb{R}^{Q^{(1)} \times Q^{(2)} \cdots \times Q^{(N-1)} \times Q^{(N)}}$. After CP decomposition of $\underline{\mathbf{X_p}}$, factor matrices and random compressed matrices are stacked as :

$$
\begin{aligned}
(\widetilde{\mathbf{A}}_s^{(1)}, \widetilde{\mathbf{A}}_s^{(2)}, \ldots, \widetilde{\mathbf{A}}_s^{(N-1)}, \widetilde{\mathbf{A}}_s^{(N)}) \leftarrow \Pi\big[(\widetilde{A}_{s(i)}^{(1)}, \widetilde{\mathbf{A}}_{s(i)}^{(2)}, \ldots, \widetilde{\mathbf{A}}_{s(i)}^{(N-1)}, \widetilde{\mathbf{A}}_{s(i)}^{(N)}) \;\; ; \\
(\widetilde{\mathbf{A}}_{s(i+1)}^{(1)}, \widetilde{\mathbf{A}}_{s(i+1)}^{(2)}, \ldots, \widetilde{\mathbf{A}}_{s(i+1)}^{(N-1)}, \widetilde{\mathbf{A}}_{s(i+1)}^{(N)})\big], i \in (1, p-1)
\end{aligned}
$$

$$
\& \quad (\widetilde{\mathbf{P}}^{(1)}, \widetilde{\mathbf{P}}^{(2)}, \ldots, \widetilde{\mathbf{P}}^{(N-1)}, \widetilde{\mathbf{P}}^{(N)}) = \begin{bmatrix} \widetilde{\mathbf{P}}_{(1)}^{(1)}, \widetilde{\mathbf{P}}_{(1)}^{(2)}, \ldots, \widetilde{\mathbf{P}}_{(1)}^{(N-1)}, \widetilde{\mathbf{P}}_{(1)}^{(N)} \\ \widetilde{\mathbf{P}}_{(2)}^{(1)}, \widetilde{\mathbf{P}}_{(2)}^{(2)}, \ldots, \widetilde{\mathbf{P}}_{(2)}^{(N-1)}, \widetilde{\mathbf{P}}_{(2)}^{(N)} \\ \vdots \\ \widetilde{\mathbf{P}}_{(p)}^{(1)}, \widetilde{\mathbf{P}}_{(p)}^{(2)}, \ldots, \widetilde{\mathbf{P}}_{(p)}^{(N-1)}, \widetilde{\mathbf{P}}_{(p)}^{(N)} \end{bmatrix} \tag{9.8}
$$

Finally, the update rule of each non-temporal mode $\in (1, N-1)$ and temporal mode $\in (N)$ is :

$$
(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{N}) \quad \leftarrow \quad (\widetilde{\mathbf{P}}^{(1)^{-1}} \quad * \quad \widetilde{\mathbf{A}}_s^{(1)}, \widetilde{\mathbf{P}}^{(2)^{-1}} \quad * \quad \widetilde{\mathbf{A}}_s^{(2)}, \ldots,
$$

$$\widetilde{\mathbf{P}}^{(N-1)^{-1}} * \widetilde{\mathbf{A}}_s^{(N-1)}, \left[\mathbf{A}_{old}^{(N)}; \widetilde{\mathbf{P}}^{(N)^{-1}} * \widetilde{\mathbf{A}}_s^{(N)})\right] \tag{9.9}$$

Finally, by putting everything together, we obtain the general version of our OCTen for 3-mode tensor, as presented in Algorithm 9.4.The matlab implementation for higher modes is also available at $link$[1].

9.4.2 Necessary characteristics for uniqueness

As we mention above, OCTen is able to identify the solution of the online CP decomposition, as long as the parallel CP decompositions on the compressed tensors are also identifiable. Empirically, we observe that if the decomposition was given data that have exact or near-trilinear structure (or multilinear in the general case), i.e., obeying the low-rank CP model with some additive noise, OCTen is able to successfully, accurately, and using much fewer resources than state-of-the-art, track the online decomposition. On the other hand, when given data that do not have a low trilinear rank, the results were of lower quality. This observation is of potential interest in exploratory analysis, where we do not know 1) the (low) rank of the data, and 2) whether the data have low rank to begin with (we note that this is an extremely hard problem, out of the scope of this chapter, but we refer the interested reader to previous studies [263, 193] for an overview). If OCTen provides a good approximation, this indirectly signifies that the data have appropriate trilinear structure, thus CP is a fitting tool for analysis. If, however, the results are poor, this may indicate that we need to reconsider the particular rank used, or even analyzing the data using CP in the first place. We reserve further investigations of what this observation implies for future work.

9.5 Experiments

We design experiments to answer the following questions:

Q1. Memory Efficient: How much memory OCTen required for updating incoming data?

Q2. Fast and Accurate: How fast and accurately are updates in OCTen compared to incremental algorithms?

Q3. Scalability: How does the running time of OCTen increase as tensor data grow (in time mode)?

Q4. Sensitivity: What is the influence of parameters on OCTen?

Q5. Effectiveness: How OCTen used in real-world scenarios?

For our all experiments, we used Intel(R) Xeon(R), CPU E5-2680 v3 @ 2.50GHz machine with 48 CPU cores and 378GB RAM.

9.5.1 Evaluation Measures

We evaluate OCTen and the baselines using three criteria: fitness, processor memory used, and wall-clock time. These measures provide a quantitative way to compare the performance of our method. More specifically, **Fitness** measures the effectiveness of approximated tensor and defined as :

$$Fitness(\%) = 100 * \Big(1 - \frac{||\underline{\mathbf{X}} - \underline{\mathbf{X}}_{approx}||_F^2}{||\underline{\mathbf{X}}||_F^2}\Big) \tag{9.10}$$

higher the value, the better approximation.

CPU time (sec): indicates the average running time for processing for processing all slices for given tensor, measured in seconds, is used to validate the time efficiency of an algorithm.

Process Memory used(MB): indicates the average memory required to process each slices for given tensor, is used to validate the space efficiency of an algorithm.

9.5.2 Baselines

In this experiment, four baselines have been selected as the competitors to evaluate the performance.

OnlineCP[284]: it is online CP decomposition method, where the latent factors are updated when there are new data.

SambaTen[94]: Sampling-based Batch Incremental Tensor Decomposition algorithm is the most recent and state-of-the-art method in online computation of canonical parafac and perform all computations in the reduced summary space.

RLST[186]: Recursive Least Squares Tracking (RLST) is another online approach in which recursive updates are computed to minimize the Mean Squared Error (MSE) on incoming slice.

ParaComp[230]: an implementation of non-incremental parallel compression based tensor decomposition method. The model is based on parallel processing of randomly com-

pressed and reduced size replicas of the data. Here, we simply re-compute decomposition after every update.

9.5.3 Experimental Setup

The specifications of each synthetic dataset are given in Table 9.2. We generate random tensors of dimension I = J = K with increasing I and other modes. Those tensors are created from a known set of randomly generated factors with known rank R, so that we have full control over the ground truth of the full decomposition. We dynamically calculate the size of incoming batch or incoming slice(s) for our all experiments to fit the data into 10% of memory of machine. Left over memory of machine is used for computations for algorithm. We use 20 parallel workers for every experiment.

Dimension	#number of non-zeros	Batch size	p	Q	shared
50 x 50 x 50	$125K$	5	20	30	5
100 x 100 x 100	$1M$	10	30	35	10
500 x 500 x 500	$125M$	50	40	30	6
1000 x 1000 x 1000	$1B$	20	50	40	10
5000 x 5000 x 5000	$7B$	10	90	70	25
10000 x 10000 x 10000	$1T$	10	110	100	20
50000 x 50000 x 50000	$6.25T$	4	140	150	30

Table 9.2: Table of Datasets analyzed

Note that all comparisons were carried out over 10 iterations each, and each number reported is an average with a standard deviation attached to it. Here, we only care about the relative comparison among baseline algorithms and it is not mandatory to have the best rank decomposition for every dataset. Although, there are few methods [176, 193] available in literature to find rank of tensor. But for simplicity, we choose the rank R to 5 for all datasets. In case of method-specific parameters, for the ParaComp algorithm, the settings are choose to give best performance. For OnlineCP, we choose the batchsize which gave best performance in terms of approximation fitness. For fairness, we also compare against the parameter configuration for OCTen that yielded the best performance for baselines. Also, during processing , for all methods we removed the unwanted variable from baselines to fairly compare with our methods.

9.5.4 Results

Memory efficient, Fast and Accurate

For all datasets we compute Fitness(%),CPU time (seconds) and Memory(MB) space required. For OCTen, OnlineCP, ParaComp,Sambaten and RLST we use 10% of the time-stamp data in each dataset as existing old tensor data. The results for qualitative measure for data is shown in Table 9.3-9.4 . For each of tensor data ,the best performance is shown in bold. All state-of-art methods address the issue very well. Compared with OnlineCP, ParaComp,Sambaten and RLST, OCTen give comparable fitness and reduce the mean CPU running time by up to 2x times for big tensor data. For all datasets, PARACOMP's accuracy (fitness) is better than all methods. But it is able to handle upto

$\mathbb{R}^{10^4 \times 10^4 \times 250}$ size tensor only. For small size datasets, OnlineCP's efficiency is better than all methods. For large size dataset, OCTen outperforms the baseline methods w.r.t fitness, average running time (improved 2x-4x) and memory required to process the updates. It significantly saved 200-250% of memory as compared to Online CP, Sambaten and RLST as shown in Table 9.4. It saved 40-80% memory space compared to Paracomp. Hence, OCTen is comparable to state-of-art methods for small dataset and outperformed them for large dataset. These results answer Q1 & Q2 as the OCTen have comparable qualitative measures to other methods.

Scalability Evaluation

To evaluate the scalability of our method, firstly, a tensor $\underline{\mathbf{X}}$ of small slice size ($I \in [20, 50, 100]$) but longer time dimension ($K \in [10^2 - 10^6]$) is created. Its first $\leq$10% timestamps of data is used for $\underline{\mathbf{X}}_{old}$ and each method's running time for processing batch of $\leq$10 data slices at each time stamp measured.

As can be seen from Figure 9.2, increasing length of temporal mode increases time consumption quasi-linearly. However the slope is different for various non-temporal mode data sizes. In terms of memory consumption, OCTen also behaves linearly. This is expected behaviour because with increase in temporal mode, the parameters i.e. p and Q also grows. Once again, our proposed method illustrates that it can efficiently process large size temporal data. This answers our Q3.

Size	**Fitness(%)**				
I=J=K	OCTEN	OnlineCP	SambaTen	ParaComp	RLST
50	97.1±4.9	86.9±7.3	76.9±16.1	**100.0±0**	95.0±0.1
100	95.8±2.2	93.4±6.9	72.1±3.7	**100.0±0**	97.7±0.1
500	96.9±1.3	90.9±8.7	84.1±0.6	**100.0±0**	94.6±0.1
1000	96.1±0.1	71.6±10.7	85.6±0.1	**100.0±0**	98.3±0.1
5000	90.8±14.7	90.6 ±42.7	80.5±15.5	**100.0±0**	98.8±0.1
10000	**98.13±2.5**	54.34±1.4	56.12±1.5	NA	97.3±0.2
50000	**68.13±4.2**	56.89±3.7	53.95±12.7	NA	NA

Dimension	**CPU Time(sec)**				
I=J=K	OCTEN	OnlineCP	SambaTen	ParaComp	RLST
50	1.32±0.01	**0.95±0.1**	2.53±0.1	1.09±0.1	1.35±0.1
100	1.92±0.3	**1.35±0.3**	5.19±0.3	1.91±0.3	4.97±0.3
500	**19.7±0.1**	20.73±4.4	99.21±2.9	26.3±0.1	37.20±0.2
1000	**290.9±6.7**	455.7±24.5	606.3±1.7	511.37±1.6	401.08±1.8
5000	**4398.8±45.5**	5835.5±0.2K	6779.7±0.3K	5157.98±3.9	6464.52±0.3K
10000	**15406.5±156.9**	21287.3±85.6	20589.81±69.4	NA	22974.76±0.1K
50000	**78892.8±98.2**	80159.6±23.5	79922.80±47.3	NA	NA

Table 9.3: Experimental results for speed and accuracy of approximation of incoming slices. We see that OCTEN gives comparable accuracy and speed to baseline.This answers our Q2.

Parameter Sensitivity

We extensively evaluate sensitivity of number of compressed cubes required 'p' , size of compressed cubes and number of shared columns required for OCTEN. we fixed

Size	Process Memory (MB)				
I=J=K	OCTEN	OnlineCP	SambaTen	ParaComp	RLST
50	**1.4±0.001**	1.6±0.002	8.8±0.001	5.5±0.003	6.271±0.001
100	**13.2±0.04**	16.8±0.03	30.1 ±0.01	14.5±0.02	19.255 ±0.02
500	**25.7±0.09**	2018.3±0.91	2959.4±0.15	26.3±0.01	1070.138 ±0.01
1000	**85.1±4.1**	16037.4±56.5	13830.8±21.7	110.9±5.1	7789.841 ±23.2
5000	**3138.1±10.1**	19457.7±25.7	19409.4±56.8	4707.1±4.1	11295.270 ±12.9
10000	**21179.5±56.9**	67145.7 ±0.0	61935.7±0.0	NA	32459.1 ±456.1
50000	**59613.5±30.0**	89657.8 ±0.0	78387.2±0.0	NA	NA

Table 9.4: Experimental results for memory required to process of incoming slices. We see that OCTEN remarkably save the memory as compared to state-of-art techniques.This answers our Q1.

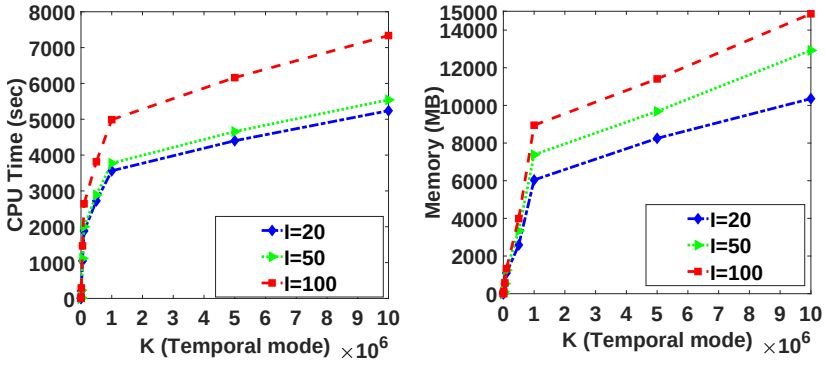


Figure 9.2: CPU time (in seconds) and Memory (MB) used for processing slices to tensor $\underline{\mathbf{X}}$ incrementing in its time mode. The time and space consumption increases quasi-linearly. The mean fitness is ≥90% for all experiments

batch size to $\approx 0.1 * K$ for all datasets ,where K is time dimension of tensor . As discus in section 9.4, it is possible to identify unique decomposition . In addition, if we have

$$p \geq \max([(I - shared)/(Q - shared) \quad J/Q \quad K/Q]) \tag{9.11}$$

for parallel workers, decomposition is almost definitely identifiable with column permutation and scaling. We keep this in mind and evaluate the OCTEN as follow.

Sensitivity of *p* :The number of compressed cubes plays an important role in OCTEN. We performed experiments to evaluate the impact of changing the number of cubes i.e. p with fixed values of other parameters for different size of tensors. We see in figure 9.3 that increasing number of cubes results in increase of Fitness of approximated tensor and CPU Time and Memory (MB) is super-linearly increased. Consider the case of $I = J = K = 1000$, from above condition, we need $P \geq \max\left(\left[\frac{1000-10}{50-10} \quad \frac{1000}{50} \quad \frac{100}{50}\right]\right) \approx 25$. We can see from figure 9.3, the condition holds true.

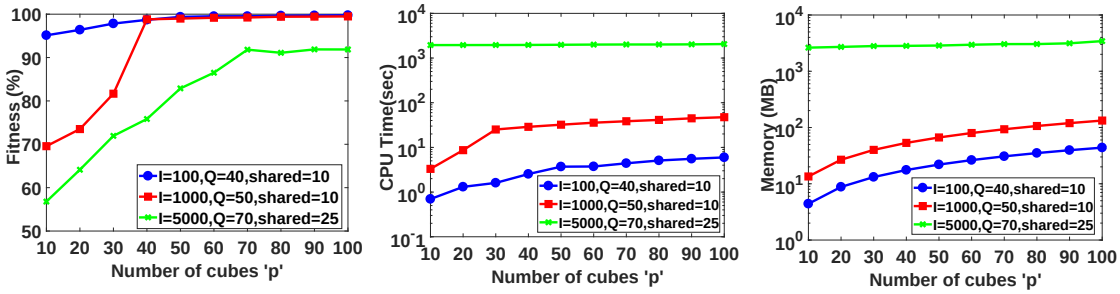


Figure 9.3: OCTEN Fitness, CPU Time (sec) and memory used vs. Number of compressed tensors 'p'on different datasets. With large 'p', high fitness is achieved.

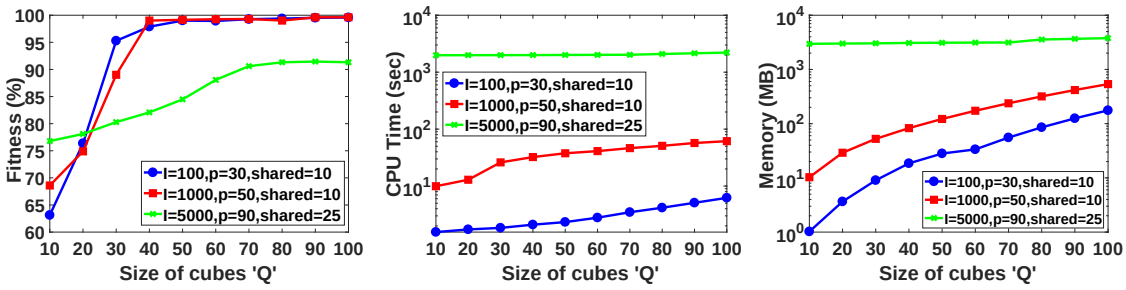


Figure 9.4: OCTEN Fitness, CPU Time (sec) and memory used vs. size of compressed tensors 'Q' on different datasets.

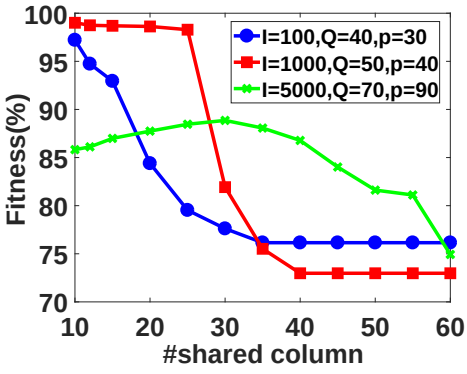


Figure 9.5: OCTEN fitness vs. shared columns of compressed tensors 'shared' on different datasets. It is observed that parameter 'shared' has negligible effect on CPU time (sec) and memory used(MB).

Sensitivity of *Q* : To evaluate the impact of Q , we fixed other parameters i.e. 'p' and 'shared'. We can see that with higher values of the 'Q', Fitness is improved as shown in Figure 9.4. Also It is observed that when equation 9.11 satisfy, fitness become saturated. Higher the size of compressed cubes, more memory is required to store them.

Sensitivity of *shared* : To evaluate the impact of 'shared' , we fixed other parameters i.e. 'p' and 'Q'.We observed that this parameter does not have impact on CPU Time (sec) and Memory space(MB). The best fitness is found when $shared \leq \frac{Q}{2}$ as shown in figure 9.5. Fitness decreases when $shared \geq \frac{Q}{2}$ because the new compressed cubes

completely losses its own structure when joined to old compressed cubes. To retain both old and new structure we choose to keep parameter $shared \leq \frac{Q}{2}$ for all experiments.

In sum, these observations demonstrate that: 1) a suitable number of cubes and its size i.e. p, Q on compressed tensor could improve the fitness and result in better tensor decomposition, and 2) For identifiability 'p' must satisfy the condition, $p \geq \max([(I - shared)/(Q - shared) \quad J/Q \quad K/Q])$, to achieve better fitness,lower CPU Time (seconds) and low memory space(MB). This result answers Q4.

Effectiveness on real world dataset

In order to truly evaluate the effectiveness of OCTen, we test its performance against five sparse (density $\approx 10^{-5}$) real datasets present in Table 9.5. American College Football Network (ACFN) [115 x 115 x 10k] dataset includes interaction between players of Division IA colleges games during Fall 2000; Foursquare-NYC [1k x 40k x 310] dataset contains 10 months check-in data in New York city collected from Foursquare; Facebook Links [63k x 63k x 650] dataset contains a list of all of the user-to-user links from the Facebook New Orleans sub-network; NELL [12k x 9k x 28k] dataset is an entity-relation-entity tuple snapshot of the Never Ending Language Learner knowledge base and NIPS Publications [2.5k x 2.8k x 14k] dataset consists of papers published from 1987 to 2003 in NIPS. The OCTen's performance in terms of timing and memory utilization on datasets are summarized in Table 9.5 and 9.6.

OCTen outperforms other baseline methods in most of the real dataset. In the case of Foursquare-NYC, Facebook-links, NELL and NIPS dataset, OCTen gives better

Dataset	OnlineCP	SambaTen	RLST	ParComp	OCTen
ACFN [227]	**12.91**	16.45	43.52	20.23	14.48
Foursquare-NYC [272]	712.21	746.20	761.42	1.2k	**642.37**
Facebook Links [259]	n/a	4.7k	n/a	n/a	**3.9k**
NELL [35]	42k	37k	n/a	n/a	**35k**
NIPS Publications [79]	372.03	343.98	1.6k	448.63	**315.47**

Table 9.5: Average CPU Time (sec) over all the batches. The lower the better. The best performance is shown in bold.

results compared to the baselines, specifically in terms of memory used (*better up to average 50-70 times*) and CPU time (*better up to average 5-8%*). OCTen outperforms for ACFN dataset in terms of memory usage and CPU time is comparable to other methods. Due to high dimensions of dataset, RSLT and PARACOMP are unable to execute within 12 hours for Facebook-links and NELL datasets. OCTen took advantage of parallel compression and decomposition of incoming slices and save huge amount of memory requirements along with giving comparable run time.

9.5.5 OCTen at work

Beyond our memory and CPU time analysis of the real world datasets in Tbl. 9.6 and 9.5, we consider American College Football Network (ACFN) [227] and Foursquare-NYC dataset for further analysis.

Case study 1: We construct the ACFN tensor data with the player-player interaction to a 115 x 115 grid, and considering the time as the third dimension of the tensor.

Dataset	OnlineCP	SambaTen	RLST	ParComp	OCTen
ACFN	1.47	1.21	1.11	2.06	**0.02**
Foursquare-NYC	2.36	2.18	2.13	4.34	**0.34**
Facebook Links	n/a	45.49	n/a	n/a	**12.78**
NELL	17.11	16.61	n/a	n/a	**9.43**
NIPS Publications	3.18	2.08	3.86	6.7	**0.45**

Table 9.6: Average Memory (GB) usage over all the batches. The lower the better. The best saving performance is shown in bold.

Therefore, each element in the tensor is an integer value that represents the number of interactions between players at a specific moment of time. Our aim is to find the players communities (ground truth communities = 12) changed over time in football dataset. In order to evaluate the effectiveness of our method on football dataset, we compare the ground-truth communities against the communities found by the our method. Figure 9.6 shows a visualization of the football network over time, with nodes colored according to the observed communities. American football leagues are tightly-knit communities because of very limited matches played across communities. Since these communities generally do not overlap, we perform hard clustering. We find that communities are volatile and players belongs to community #12 (from subfigure (a)) are highly dynamic in forming groups. We observe that OCTen is able to find relevant communities and also shows the ability to capture the changes in forming those communities in temporal networks.

Case study 2: Foursquare-NYC dataset includes long-term ($\approx$ 10 months) check-in data in New York city collected from Foursquare from 12 April 2012 to 16 February 2013.

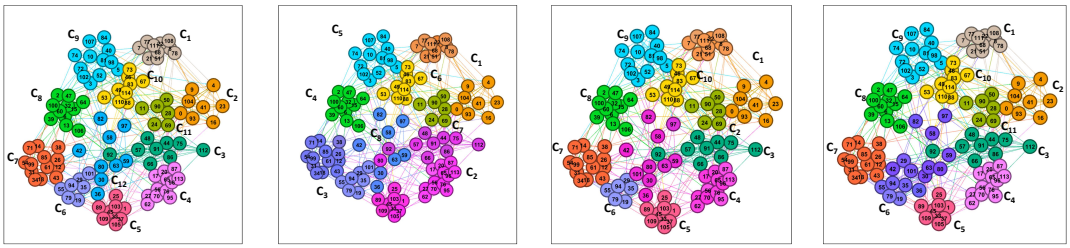


Figure 9.6: Visualization of the ground truth communities vs. the identified communities using OCTen on ACFN dataset, which has 12 observed players communities i.e. $C \in \{C_1, C_2 \dots C_{12}\}$. **(a):** Represents the visualization of the network colored with ground truth communities. **(b):** Shows the visualization of the network colored with predicted communities at time $\frac{1}{3}T$, where T is total time stamps. **(c):** Shows the visualization of the network colored with predicted communities at time $\frac{2}{3}T$. **(d):** Shows the visualization of the network colored with predicted communities at time T. We see that reconstructed views using OCTen helps to identify the communities changing over time.

The tensor data is structured as [user (1k), Point of Interest (40k), time (310 days)] and each element in the tensor represents the total time spent by user for that visit. Our aim is to find next top@5 places to visit in NYC per user. We decompose the tensor data into batches of 7 days and using rank = 15 estimated by AutoTen [193]. For evaluation, we reconstruct the complete tensor from factor matrices and mask the known POIs in the tensor and then rank the locations for each user. Since we do not have human supplied relevance rankings for this dataset, we choose to visualize the most significant factor values (locations) using maps provided by Google. If the top ranked places are with-in 5 miles radius of user's previous places visited, then we consider the decomposition is effective.

Figure 9.7: OCTEN's five highest values of the factor are represented as red markers.

In the Figure 9.7, the five red markers corresponds to the five highest values of the factor. These locations correspond to well-known area in NYC : Brooklyn Bridge , Square garden , Theater District and Museum of Art. The high density of activities (green points) verifies their popularity.

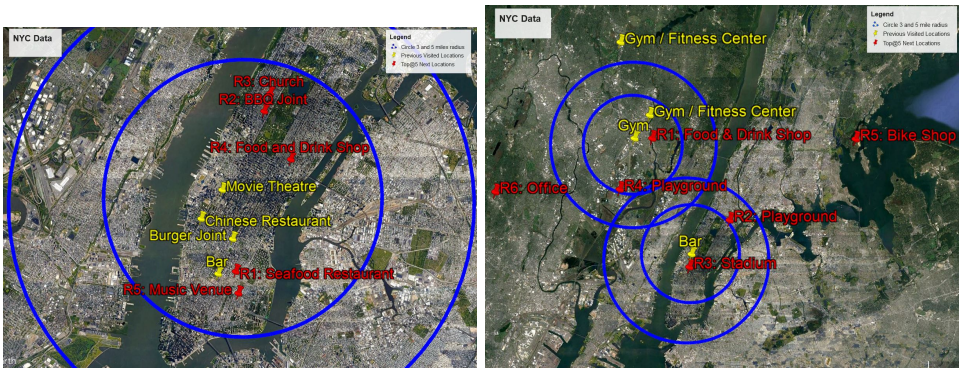


Figure 9.8: Visualization of the top@5 POIs of the **user#192 and user#902** obtained from reconstructed tensor using factor matrices. The yellow markers are user's previous visited POIs and red markers are recommended POIs.

Figure 9.8 shows the top@5 results for users #192 and user #902, the red marker shows the next locations to visit and yellow marker shows the previous visited locations. More interestingly, we can see that user #192 visited coffee shops and restaurants most of the time, top@5 ranked locations are also either restaurants or food & drink shops. Similarity, user #902, most visited places are Fitness center, top@5 ranked locations are park, playground and Stadium.

This shows the effectiveness of the decomposition and confirms that the OCTen can be used for various types of data analysis and this answers **Q5**.

9.6 Conclusion

In this work, we focus on online tensor decomposition problem and proposed a novel compression based OCTen framework. The proposed framework effectively identify the low rank latent factors of compressed replicas of incoming slice(s) to achieve online tensor decompositions. To further enhance the capability, we also tailor our general framework towards higher-order online tensors. Through experiments, we empirically validate its effectiveness and accuracy and we demonstrate its memory efficiency and scalability by outperforming state-of-the-art approaches (40-200 % better). Regardless, future work will focus on investigating different tensor decomposition methods and incorporating various tensor mining methods into our framework.

Chapter based on material published in CAMSAP 2019 [96].

Chapter 10

Streaming Algorithms to Track the Block Term Decomposition of Large Tensors

"How to incrementally update the low or multi-linear rank data effectively?"

In data mining, block term tensor decomposition (BTD) is a relatively under-explored but very powerful multi-layer factor analysis method that is ideally suited for modeling for batch processing of data which is either low or multi-linear rank, e.g., EEG/ECG signals, that extract "rich" structures ($> rank - 1$) from tensor data while still maintaining a lot of the desirable properties of popular tensor decompositions methods such as the interpretability, uniqueness, and etc. These days data, however, is constantly changing which hinders its use for large data. The tracking of the BTD decomposition for the dynamic

tensors is a very pivotal and challenging task due to the variability of incoming data and lack of efficient online algorithms in terms of accuracy, time and space.

In this paper, we fill this gap by proposing an efficient method OnlineBTD to compute the BTD decomposition of streaming tensor datasets containing millions of entries. In terms of effectiveness, our proposed method shows comparable results with the prior work, BTD, while being computationally much more efficient. We evaluate OnlineBTD on six synthetic and three diverse real datasets, indicatively, our proposed method shows $10-60\%$ speedup and saves $40-70\%$ memory usage over the traditional baseline methods and is capable of handling larger tensor streams for which the classic BTD fails to run. To the best of our knowledge, OnlineBTD is the first approach to track streaming block term decomposition while not only being able to provide stable decompositions but also provides better performance in terms of efficiency and scalability. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, and Evangelos E. Papalexakis. "OnlineBTD: Streaming Algorithms to Track the Block Term Decomposition of Large Tensors." In 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), pp. 168-177. IEEE, 2020.

10.1 Introduction

Tensor decomposition methods are very vital tool in various applications like biomedical imaging [222], social networks [276], and recommender systems [150] to solve various challenging problems. Tensors are higher-order matrix generalization that can reveal more details compared to matrix data, while maintaining most of the computational

efficiencies. Each such order of the data is an impression of the same underlying phenomenon e.g, the formation of friendship in social networks or the evolution of communities over time. A main task of the tensor analysis is to decompose the multi-modal data into its latent factors, which is widely known as CANDECOMP/PARAFAC (CP) [36, 104] and Tucker Decomposition [256] in the literature. The CP decomposition has found many applications in machine learning [193], statistical learning [41] and computational neuroscience [209] to understand brain generated signals.

Motivating example: Given the importance of tensor analysis for large-scale data science applications, there has been a growing interest in scaling up these methods to handle large real-world data [284, 94, 237, 95]. However, the CP and Tucker decomposition make a strong assumption on the factors, namely that they are rank-1 (see def. 4). In various application domains, like biomedical images [180], text data [93], it is often controversial whether this assumption is satisfied for all the modes of the problem. The low or multi-linear rank may be a better approximation of the real-world applications [93, 257]. For example, fetal electrocardiogram (fECG) tracking is extremely important for evaluating fetal health and analyzing fetal heart conditions during pregnancy. The electrical activity of the fetal heart is recorded during labor between 38 and 41 weeks of gestation by electrodes (non-invasive) placed on mother's abdomen or an electrode attached to the fetal scalp (invasive, not routinely used) while the cervix is dilated (i.e. during delivery). This signal is produced from a small heart, therefore the amplitude of the signal is low and quite similar to the adult ECG, with a higher heart rate. The recorded signal also has interference from muscle noise, motion artifacts and etc. The fetal and maternal ECG have temporal and spectral

overlap. The separation of an accurate fetal electrocardiogram signal from the abdominal mixed signals is complicated but very crucial for many reasons like early detecting fetal distress to avoid painful emergency caesarean delivery ($> 9.6\%$) [99] and reduce brain damage or cerebral palsy in new-born. How to identify and separate fetal ECG over time which can signify a potential issues from noise? How to monitor the dynamic fetal signal behavior? In [9], the Tucker decomposition is used for fECG extraction. Because of the fetal low-amplitude signal compared to the mother's heart signal, the drawback of this method lies in the use of only the fetal periodicity constraints to get rank-1 components. Such limitations can be resolved by Block Term Decomposition (BTD) [53]. BTD helps to tensorize (low multi-linear blocks) abdominal mixed signals into separate subspaces of the mother, fetus and noise signals. The block term decomposition writes a given tensor as a sum of terms with a low multi-linear rank, without rank-1 being required.

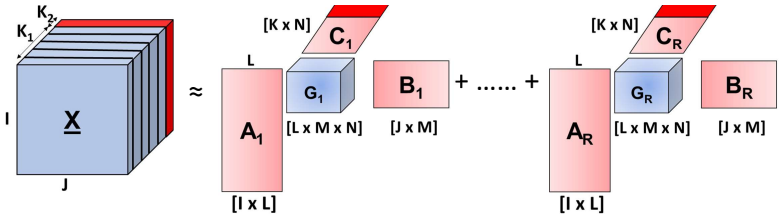


Figure 10.1: OnlineBTD procedure. Left: the original tensor is extended with an extra slice (red) in the third mode. Right: the BTD of the tensor is updated by adding a new vector (red) to the factor matrix in the third mode and modifying the existing factor matrices (pink).

Previous Works: The Block Term Decomposition (BTD) unifies the CP and Tucker Decomposition. The BTD framework offers a coherent viewpoint on how to gen-

eralize the basic concept of rank from matrices to tensors. The author [122] presented an application of BTD where epileptic seizures pattern was non-stationary, such a trilinear signal model is insufficient. The epilepsy patients suffer from recurring unprovoked seizures, which is a cause and a symptom of abrupt upsurges. They showed the robustness of BTD against these sudden upsurges with various model parameter settings. The author [41] used a higher-order BTD for the first time in fMRI analysis. Through extensive simulation, they demonstrated its effectiveness in handling strong instances of noise. A deterministic block term tensor decomposition (BTD) - based Blind Source Separation [54, 209] method was proposed and offered promising results in analyzing the atrial activity (AA) in short fixed segments of an AF ECG signals. The paper [58] extends the work [54] for better temporal stability. The paper [180] proposed doubly constrained block-term tensor decomposition to extract fetal signals from maternal abdominal signals. Recently, the paper [93] proposed ADMM based constrained BTD method to find structures of communities within social network data. However, these classic methods and applications of BTD are limited to small-sized static dense data ($1K \times 1K \times 100$) and require a large amount of resources (time and space) to process big data. In this era, data is growing very fast and a recipe for handling the limitations is to adapt existing approaches using online techniques. For example, health monitoring data like fECG represents a large number of vibration responses measured over time by many sensors attached at different parts of abdomen. In such applications, a naive approach would be to recompute the decomposition from scratch for each new incoming data. Therefore, this would become impractical and computationally expensive.

Challenges. For many different applications like sensor network monitoring or evolving social network, the data stream is an important model. Streaming decomposition is a challenging task due to the following reasons. First, **accuracy**: high-accuracy (competitive to decomposing the full tensor) using significantly fewer computations than the full decomposition calls for innovation. Second, **speed**: the velocity of incoming data into the system is very high and require real-time execution. Third, **space**: operating on the full ambient space of data, as the tensor is being updated online, leads to increase in space complexity, rendering offline approaches hard to scale, and calling for efficient methods that work on memory spaces which are significantly smaller than the original ambient data dimensions. Lastly, **beyond rank-1 data** [93]: there are certain instances wherein rank-1 decomposition (CP or Tucker) can not be useful, for example, EEG signals [122] needed to be modeled as a sum of exponentially damped sinusoids and allow the retrieval of poles by singular value decomposition. The rank-one terms can only model components of data that are proportional along columns and rows, and it may not be realistic to assume this. Alternatively, it can be handled with blocks of decomposition.

Mitigating Challenges: Motivated by the above challenges, the objective of our work is to develop an algorithm for large multi-aspect or multi-way data analysis that is scalable and amenable to incremental computation for continuously incoming data. In this paper, we propose a method to decompose online or streaming tensors based on BTD decomposition. Our goal is, given an already computed BTD decomposition, to *track* the BTD decomposition of an online tensor, as it receives streaming updates, a) *efficiently*, being much quicker than recomputing the entire decomposition from scratch after each update,

and using less memory, and b) *accurately*, obtaining an approximation error which is as similar to the complete tensor decomposition as possible. Answering the above questions, we propose OnlineBTD framework (Figure 10.1). Our OnlineBTD achieves the best of both worlds in terms of accuracy, speed and memory efficiency: 1) in all cases considered for real and synthetic data (Table 10.4) it is faster than a highly optimized baseline method, achieving up to $10-60\%$ performance improvement; 2) simultaneously, the proposed method is more robust and scalable, since it can execute large problem instances in a reasonable time whereas the baseline fails due to excessive memory consumption (Figure 10.3).

To our best knowledge, there is no work in the literature that deals with streaming or online Block Term Decomposition. To fill the gap, we propose a scalable and efficient method to find the BTD decomposition for streaming large-scale high-order multi-aspect or temporal data and maintain comparable accuracy. Our contributions can be summarized as:

- **Novel and Efficient Algorithm**: We introduce OnlineBTD, a novel and efficient algorithm for tracking the BTD decompositions of streaming tensors that admits an accelerated implementation described in Section 7.4. We do not limit to three-mode tensors, our algorithm can easily handle higher-order tensor decompositions.
- **Stable Decomposition**: Based on the empirical analysis (Section 6.3) on synthetic datasets, our algorithm produces similar or more stable decompositions than to existing offline approaches, as well as a better scalability.

- **Real-World Utility**: We performed a case study of applying OnlineBTD on the dataset by ANT-Social Network which consists of $>$ 1 million interactions over 41 days and EEG signal data to understand the human body movement.

Reproducibility: To promote reproducibility, we make our MATLAB implementation publicly available at Link[1].

10.2 Proposed Method: OnlineBTD

In this section, we present our proposed method to track the BTD decomposition of streaming tensor data in an incremental setting. Initially a case of the third-order will be discussed for simplicity of presentation. Further, we expand further to more general conditions, where our proposed algorithm can handle tensors with a higher modes. Formally, the problem that we solve is the following:

Given (a) an existing set of BTD decomposition i.e. $\mathbf{A}_{old}, \mathbf{B}_{old}$ and $\mathbf{C}_{old}$ factor matrices, having (L_r, M_r, N_r) latent components, that approximate tensor $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I \times J \times K_1}$ with Rank R at time t , (b) new incoming slice (s) in form of tensor $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I \times J \times K_2}$ at any time Δt,

Find updates of $\mathbf{A}_{new}, \mathbf{B}_{new}$ and $\mathbf{C}_{new}$ **incrementally** to approximate BTD tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times (K_1 + K_2)}$ after appending new slice(s) at $t = t_1 + \Delta t$ in last mode while maintaining a comparable accuracy with running the full BTD decomposition on the entire updated tensor $\underline{\mathbf{X}}$.

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

10.2.1 The Principle of OnlineBTD

To address the online BTD problem, our proposed method follows the same alternating update schema as ALS, such that only one factor matrix is updated at one time by fixing all others. Our proposed method is the first work to do an online BTD algorithm, so we need to make some assumptions that will ground the problem. Practically, all other online decomposition (CP/Tucker) based works [284, 95, 186, 94] presume the same thing, either implicitly or explicitly.

Assumptions

- We assume the last mode of a tensor growing, while the size of the other modes remain unchanged with time.
- The factor matrices $\mathbf{A}_{old}, \mathbf{B}_{old}$ and $\mathbf{C}_{old}$ and core tensor $\mathcal{G}_{old}$ for old data ($\underline{\mathbf{X}}_{old}$) at time stamp t_1 is available.
- The tensor $\underline{\mathbf{X}}$ rank R and Block rank L_r, M_r, N_r are available where $r \in [1, R]$.

Update Temporal Mode

Consider first the update of factor $\mathbf{C}_{new}$ obtained after fixing $\mathbf{A}_{old}$, $\mathbf{B}_{old}$ and $\mathcal{G}_{old}$, and solving the corresponding minimization in Equ 10.1.

$$\mathbf{C}_{new} \leftarrow \arg\min_{\mathbf{C}} ||\mathbf{X}^{(3)} - \mathbf{C}.bd(\mathcal{G}).(\mathbf{B} \odot \mathbf{A})^T||_2^F \tag{10.1}$$

The above Equ. (10.1) resembles similar to solving problem for CP [141, 284] but it is different and more challenging problem because offline/online CP has rank-1 latent factors (single block) for decomposition without any core tensor. In BTD, we deals with beyond rank-1

and decomposition consists of R number of blocks. Each block is solved with partition-wise Kronecker product (See def. (13)) instead of column-wise Khatri-Rao product (See def. (12)). Further Equ. (10.1) can be written as:

$$\begin{aligned}\mathbf{C}_{new} &= \arg\min_{\mathbf{C}} || \begin{bmatrix} \mathbf{X}^{(3)}_{old} \\ \mathbf{X}^{(3)}_{new} \end{bmatrix} - \begin{bmatrix} \mathbf{C}_{r_{old}} \\ \widetilde{\mathbf{C}_r} \end{bmatrix} .[\mathcal{G}^{(3)}_r (\mathbf{B}_r \otimes \mathbf{A}_r)^T]||^F_2 \\ &= \arg\min_{\mathbf{C}} || \begin{bmatrix} \mathbf{X}^{(3)}_{old} - \mathbf{C}_{r_{old}} .[\mathcal{G}^{(3)}_r (\mathbf{B}_r \otimes \mathbf{A}_r)^T] \\ \mathbf{X}^{(3)}_{new} - \widetilde{\mathbf{C}_r} .[\mathcal{G}^{(3)}_r (\mathbf{B}_r \otimes \mathbf{A}_r)^T] \end{bmatrix} ||^F_2 \end{aligned} \tag{10.2}$$

where $r \in [1, R]$ and the above equation presents that the first part is minimized with respect to $\mathbf{C}_{r_{old}}$, since $\mathbf{A}_r$, $\mathbf{B}_r$ and $\mathcal{G}^{(3)}_r$ are fixed as $\mathbf{A}_{r_{old}}$, $\mathbf{B}_{r_{old}}$ and $\mathcal{G}^{(3)}_{r_{old}}$ from the last time stamp. The $\widetilde{\mathbf{C}_r}$ can be obtained after minimizing above equation as :

$$\begin{aligned}\widetilde{\mathbf{C}} &= \mathbf{X}^{(3)}_{new} * [\mathcal{G}_{r_{old}} (\mathbf{B}_{r_{old}} \otimes \mathbf{A}_{r_{old}})^T]^\dagger \quad \forall r \in [1, R] \\ &= \mathbf{X}^{(3)}_{new} * (\mathcal{G}_{1_{old}} .(\mathbf{B}_{1_{old}} \otimes \mathbf{A}_{1_{old}})^T \dots \mathcal{G}_{R_{old}} .(\mathbf{B}_{R_{old}} \otimes \mathbf{A}_{R_{old}})^T)^\dagger \end{aligned} \tag{10.3}$$

Observation 1: The classic Matricized Tensor Kronecker product is expensive process because of high computations during Kronecker product, henceforth referred as *classic MTTKRONP*. Thus the *accelerated MTTKRONP* is required. Consider matrix $\mathbf{A} \in \mathbb{R}^{I \times L}$ and $\mathbf{B} \in \mathbb{R}^{J \times M}$ and its Kronecker product $\mathbf{AB} \in \mathbb{R}^{IJ \times LM}$. To speed up the process, we avoid use of def (11) and also avoid explicit allocation of memory by following steps:

- Reshaping matrix into 4-D array : $\mathbf{A} \in \mathbb{R}^{1 \times I \times 1 \times L}$; $\mathbf{B} \in \mathbb{R}^{J \times 1 \times M \times 1}$;
- Multiplies 4-D array $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$ by multiplying corresponding elements; $\mathbf{Kr} = \underline{\mathbf{A}} . * \underline{\mathbf{B}}$
- Reshape $\mathbf{Kr}$ as $\in \mathbb{R}^{IJ \times LM}$ and multiple its transpose with matrix form of given core tensor.

I=J,Rank	**Classic** (sec)	**Accelerated** (sec)	**Improvement**
$1000, 5$	0.16	0.09	43%
$1500, 15$	0.73	0.54	26%
$5000, 25$	37.55	28.91	23%

Table 10.1: Computational gain of accelerated vs classic MTTKRONP.

The above method saves $\approx$ 30% (average) of computational time as provided in Table 10.1, when compared to classic (product of tensor and output of kron) method available in MATLAB [171].

The factor matrix $\mathbf{C}_{new} \in \mathbb{R}^{(K_1+K_2)\times N}$ is updated by appending the projection $\mathbf{C}_{old} \in \mathbb{R}^{K_1\times N}$ of previous time stamp, to $\widetilde{\mathbf{C}} \in \mathbb{R}^{K_2\times N}$ of new time stamp, i.e.,

$$\mathbf{C}_{new} = \begin{bmatrix} \mathbf{C}_{old} \\ \widetilde{\mathbf{C}} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{old} \\ \mathbf{X}^{(3)}_{new} * [\mathcal{G}_{r_{old}} * \mathbf{Kr}_r^T]^\dagger \end{bmatrix} \tag{10.4}$$

where $r \in [1, R]$ and the accelerated MTTKRONP is efficiently calculated in linear complexity to the number of non-zeros.

Update Non-Temporal Mode

We update $\mathbf{A}_{new}$ by fixing $\mathbf{B}_{old}$, $\mathcal{G}_{old}$ and $\mathbf{C}_{new}$. We set derivative of the loss $\mathcal{LS}$ w.r.t. $\mathbf{A}$ to zero to find local minima as :

$$\frac{\delta([\mathbf{X}^{(1)}_{new} - \mathbf{A}_{r_{new}}.[bd(\mathcal{G}_{r_{old}}).(\mathbf{C}_{r_{new}} \otimes \mathbf{B}_{r_{old}})]^T]}{\delta A_{r_{new}}} = 0 \tag{10.5}$$

By solving above equation, we obtain:

$$\begin{aligned}\mathbf{A}_{new} &= (\begin{bmatrix}\mathbf{X}^{(1)}_{old}\\ \mathbf{X}^{(1)}_{new}\end{bmatrix} * [\mathcal{G}_{r_{old}}.(\begin{bmatrix}\mathbf{C}_{r_{old}}\\ \widetilde{\mathbf{C}_r}\end{bmatrix} \otimes \mathbf{B}_{r_{old}})^T]^\dagger \\ &= \mathbf{X}^{(1)}_{new} * [\mathcal{G}_{r_{old}}.(\widetilde{\mathbf{C}_r} \otimes \mathbf{B}_{r_{old}})^T]^\dagger + \mathbf{X}^{(1)}_{old} * [\mathcal{G}_{r_{old}}.(\mathbf{C}_{r_{old}} \otimes \mathbf{B}_{r_{old}})^T]^\dagger \\ &= \mathbf{X}^{(1)}_{new} * \mathbf{G}^\dagger + \mathbf{A}_{old}, \quad \mathbf{G} = [\mathcal{G}_{r_{old}}(\widetilde{\mathbf{C}_r} \otimes \mathbf{B}_{r_{old}})^T]\end{aligned} \tag{10.6}$$

In this way, the factor update equation consists of two parts: the historical part; and the new data part that makes computation fast using *accelerated MTTKRONP*. The $\mathbf{A}_{new} \in \mathbb{R}^{I \times LR}$ is then partitioned into block matrices using corresponding rank (i.e. L) per block.

Similarly, $\mathbf{B}_{new}$ can be updated for mode-2 as :

$$\mathbf{B}_{new} = \mathbf{X}^{(2)}_{new} * \mathbf{G}^\dagger + \mathbf{B}_{old}, \quad \mathbf{G} = [\mathcal{G}_{r_{old}}(\widetilde{\mathbf{C}_r} \otimes \mathbf{A}_{r_{new}})^T] \tag{10.7}$$

The $\mathbf{B}_{new} \in \mathbb{R}^{J \times MR}$ is then partitioned into block matrices using corresponding rank (i.e. M) per block.

Update core tensor

The updated core tensor is obtained from updated factors $\mathbf{A}_{new}$, $\mathbf{B}_{new}$ and $\widetilde{\mathbf{C}}$ using following equation:

$$\begin{aligned}\begin{bmatrix}(\mathcal{G}_1)_{LMN}\\ \dots \\ (\mathcal{G}_R)_{LMN}\end{bmatrix} &= \begin{bmatrix}(\widetilde{\mathbf{C}}_{1_{new}} \otimes \mathbf{B}_{1_{new}} \otimes \mathbf{A}_{1_{new}})\\ \dots \\ (\widetilde{\mathbf{C}}_{R_{new}} \otimes \mathbf{B}_{R_{new}} \otimes \mathbf{A}_{R_{new}})\end{bmatrix}^\dagger .\underline{\mathbf{X}}_{new}(:) \\ &= \mathbf{H}^\dagger.\underline{\mathbf{X}}_{new}(:)\end{aligned} \tag{10.8}$$

Observation 2: The above pseudo-inverse ($\mathbf{H}^{\dagger}$) or generalized inverse is very expensive in terms of time and space. This can be accelerated using reverse order law [205, 49] and modified LU Factorization (provided in Algorithm 11) and equation can be re-written as:

$$\begin{aligned} \mathbf{L} &= LU_{modified}(\mathbf{H}) \\ \mathcal{G}_{new} &= (\mathbf{L}(\mathbf{L}^T\mathbf{L})^{-1}(\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T\mathbf{H}^T)\underline{\mathbf{X}}_{new}(:) \end{aligned} \tag{10.9}$$

The main reason to use LU factorization over traditional pseudo-inverse is because a back tracing error is lower[114] as:

$$E_{forward} \leq cond(\mathbf{H}) \times E_{backward} \tag{10.10}$$

where $E_{forward}$ is forward error, $E_{backward}$ backward error and *cond* represents condition number of matrix. Since condition number does not depend on an algorithm used to solve given problem, so choosing LU algorithm gives smaller backward error and it will lead to lower forward error. Also, in this we are dealing with triangular matrices (L and U), which can be solved directly by forward and backward substitution without using the Gaussian elimination (Gauss - Jordan) process[115] used in pseudo-inverse. The $\mathcal{G}_{new} \in \mathbb{R}^{RLMN \times 1}$ is then partitioned into R blocks and reshaped using corresponding rank (i.e. $[L, M, N]$) per

block.

Algorithm 11: Modified LU Factorization for OnlineBTD

Data: $\mathbf{A} \in \mathbb{R}^{n \times n}$

Result: Lower matrix $\mathbf{L}$, Upper matrix $\mathbf{U}$, Permutation matrix $\mathbf{P}$

$\mathbf{L} = eye(n)$; $\mathbf{P} = \mathbf{L}$; $\mathbf{U} = \mathbf{A}$;

for $k \leftarrow 1$ *to* n **do**

  $[val \quad m] = max(abs(\mathbf{U}(k:n,k)));; m = m + k - 1$

  if $m \neq k$ **then**

    Interchange rows m and k in $\mathbf{U}$ and $\mathbf{P}$

    if $k \geq 2$ **then**

      Interchange rows m and k in $\mathbf{L}$ for $(k-1)$ columns

    end

  end

  for $j \leftarrow (k+1)$ *to* n **do**

    $\mathbf{L}(j,k) = \mathbf{U}(j,k)/\mathbf{U}(k,k)$; $\mathbf{U}(j,:) = \mathbf{U}(j,:) - \mathbf{L}(j,k) * \mathbf{U}(k,:)$;

  end

  $\mathbf{L}(:,k) = \mathbf{L}(:,k) * \sqrt{\mathbf{A}(k,k) - \mathbf{L}(k,1:k-1) * \mathbf{L}(k,1:k-1)^T}$;

end

return $\mathbf{L}$, $\mathbf{U}$, $\mathbf{P}$

Summary: For a 3-mode tensor that grows with time or at its 3^{rd} mode, we propose an efficient algorithm for tracking its BTD decomposition on the fly. We name this algorithm as OnlineBTD, comprising the following two stages:

- Initialization stage (Algorithm (1) in supplementary section (A.3)): in case factors from old tensor $\underline{\mathbf{X}}_{old}$ are not available, then we obtain its BTD decomposition as ($\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathcal{G}$)
- Update stage (Algorithm 10.2): for each new incoming data $\underline{\mathbf{X}}_{new}$, it is processed as:
  - For the temporal mode 3, $\mathbf{C}$ is updated using Equ. (10.4)
  - For non-temporal modes 1 and 2, $\mathbf{A}$ and $\mathbf{B}$ is updated using Equ. (10.6) and Equ. (10.7), respectively.
  - For core-tensor, $\mathcal{G}$ is updated using Equ. (10.8) and accelerated using Equ. (10.9).

10.2.2 Extending to Higher order tensors

We now show how our approach is extended to higher-order cases. Consider N-mode tensor $\underline{\mathbf{X}}_{old} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times K_1}$. The factor matrices are $(\mathbf{A}^{(1)}_{old}, \mathbf{A}^{(2)}_{old}, \ldots, \mathbf{A}^{(N-1)}_{old}, \mathbf{A}^{(N)}_{old})$ for BTD decomposition with N^{th} mode as new incoming data. A new tensor $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times K_2}$ is added to $\underline{\mathbf{X}}_{old}$ to form new tensor of $\mathbb{R}^{I_1 \times I_2 \times \cdots \times K}$ where $K = K_1 + K_2$. The subscript $i \neq n$ indicated the n^{th} matrix is not included in the operation.

The Temporal mode can be updated as :

$$\mathbf{A}^{(N)} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{A}^{(N)}_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{X}^{(N)}_{new} * [\mathcal{G}_{r_{old}} . (\otimes_{i=1}^{N-1} \mathbf{A}^{(i)})^T]^\dagger \end{bmatrix} \tag{10.11}$$

The Non-Temporal modes can be updated as:

$$\begin{aligned} \mathbf{A}^{(i)} &= \mathbf{X}^{(i)}_{new} * [\mathcal{G}_{r_{old}} (\otimes_{i \neq n}^{N} \mathbf{A}^{(i)}_r)^T]^\dagger + \mathbf{A}^{(i)}_{old} \\ &= \mathbf{X}^{(i)}_{new} * [\mathcal{G}_{r_{old}} (Kr^{(n)}_r)^T]^\dagger + \mathbf{A}^{(i)}_{old} \end{aligned} \tag{10.12}$$

Algorithm 12: OnlineBTD Update Framework

Data: $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N-1} \times K_2}$, old data factors

$$(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)}), \underline{\mathbf{D}} \text{ , Rank } R \text{ and } L.$$

Result: Updated factor matrices $(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)}, \underline{\mathbf{D}})$,

Update temporal modes of tensor $\underline{\mathbf{X}}$ as:

$$\mathbf{A}^{(N)} \leftarrow \begin{bmatrix} \mathbf{A}^{(N)}_{old} \\ \mathbf{X}^{(N)}_{new} * [\mathcal{G}_{r_{old}} \cdot (\otimes_{i=1}^{N-1} \mathbf{A}^{(i)})^T]^\dagger \end{bmatrix} \quad \forall r \in [1, R]$$

for $n \leftarrow 1$ *to* $N-1$ **do**

  Update other modes of tensor $\underline{\mathbf{X}}$ as:

  $$\mathbf{A}^{(i)} \leftarrow \mathbf{X}^{(i)}_{new} * [\mathcal{G}_{r_{old}} (\otimes_{i \neq n}^{N} \mathbf{A}^{(i)}_r)^T]^\dagger + \mathbf{A}^{(i)}_{old} \quad \forall i \in [1, N] \quad \forall r \in [1, R]$$

end

Update core tensor using $\underline{\mathbf{X}}_{new}$

$$\underline{\mathbf{D}} \leftarrow \begin{bmatrix} \otimes_{i=1}^{N} \mathbf{A}^{(i)}_1 \\ \ldots \\ \otimes_{i=1}^{N} \mathbf{A}^{(i)}_R \end{bmatrix}^\dagger * \underline{\mathbf{X}}_{new}(:) \quad \forall i \in [1, N] \quad \forall r \in [1, R]$$

return *Updated* $(\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}, \mathbf{A}^{(N)}, \underline{\mathbf{D}})$

where $i \in [1, N-1]$ and we denote the Kronecker(Kr) product of the first (N- 1) but the n^{th} loading matrices, $(\otimes_{i \neq n}^{N} \mathbf{A}_r^{(i)})$ as $Kr_r^{(n)}$.

The core tensor of BTD decomposition can be updated as:

$$\underline{\mathbf{D}} = \begin{bmatrix} \otimes_{i=1}^{N} \mathbf{A}_1^{(i)} \\ \dots \\ \otimes_{i=1}^{N} \mathbf{A}_R^{(i)} \end{bmatrix}^{\dagger} * \underline{\mathbf{X}}_{new}(:) \tag{10.13}$$

Avoid duplicate Kronecker product: To avoid duplicate Kronecker product in above calculations, we use a dynamic programming method adapted from [284] to compute all the Kronecker products in one run as shown in Figure (10.2). The process uses intermediate results to avoid duplicate Kronecker product. The method goes through the factor matrix from both sides, until it reaches the results of first and last Kronecker product in a block and repeats the process for all the blocks.

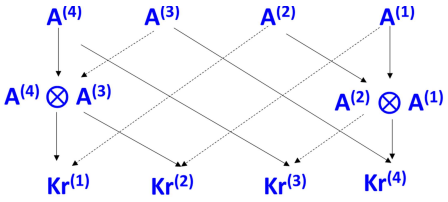


Figure 10.2: Kronecker products for the 5^{th}-order.

Accelerated computation summary: we accelerate computation in OnlineBTD via a) using *accelerated MTTKRONP* instead of *classic MTTKRONP*, b) using modified-LU factorization instead of classic pesudo-inverse, and c) by avoiding duplicate Kronecker product for higher order tensors. Finally, by putting everything together, we obtain the general version of our OnlineBTD algorithm, as presented in Algorithm 12.

Dataset	Statistics (K: Thousands M: Millions)			
	$I = J$	**K**	**[L,M,N]**	***Batch***
I	100	500	$[5, 6, 7]$	50
II	250	$1K$	$[5, 6, 7]$	50
III	$1K$	$1K$	$[5, 6, 7]$	10
IV	$1K$	$10K$	$[5, 6, 7]$	10
V	$1K$	$100K$	$[3, 4, 5]$	10
VI	$1K$	$1M$	$[3, 4, 5]$	10

Table 10.2: Details for the synthetic datasets.

10.3 Experiments

We design experiments to answer the following questions: **(Q1)** How fast, accurate and memory efficient are updates in OnlineBTD compared to classic BTD algorithm? **(Q2)** How does the running time of OnlineBTD increase as tensor data grow (in time mode)? **(Q3)** What is the influence of parameters on OnlineBTD? **(Q4)** How OnlineBTD used in real-world scenarios?

10.3.1 Experimental Setup

Synthetic Data

We provide the datasets used for evaluation in Table 10.2. For all synthetic data we use rank $R = 3$. In the entries of factor matrix $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ Gaussian noise is added such that the signal-to-noise ratio is $10dB$.

Dataset	I	J	K	[L,M,N]	*Batch*
ANT-Network [216]	822	822	41	$[3, 3, 3]$	10
EU Core [276]	1004	1004	526	$[8, 8, 8]$	10
EEG Signal [222]	109	896	$20K$	$[5, 5, 5]$	20

Table 10.3: Details for the real datasets.

Real Data

We provide the datasets used for evaluation in Table 10.3. Rank determination in the experiments is performed with the aid of the Core Consistency Diagnostic (CorConDia) method [31, 193] and the triangle method ,implemented by Tensorlab 3.0[258].

- **EU-Core**[276]: consists of e-mail data from a large European research institution over 526 days.
- **Ant Social Network**[216]: This dataset consists of information of all social interaction among ants, their behavior and spatial movement from all ants in six colonies over 41 days.
- **EEG Signal**[222]: This dataset includes 109 subjects who performed 14 experimental different motor/imagery tasks while 64-channel EEG were recorded using the BCI2000 system.

aseline method

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

- **BTD-ALS** : an implementation of standard ALS fitting algorithm BTD. Implementation of BTD-ALS is not provided in Tensorlab [258]. Hence, we provide an efficient implementation of BTD-ALS to promote reproducibility.
- **BTD-NLS** [258] : an implementation of standard NLS fitting algorithm BTD with noisy initialization.

Note that there is no method in literature for online or incremental BTD tensors. Hence, we compare our proposed method against algorithms that decompose full tensor. Also, extending BTD-NLS to online settings is challenging and requires further research both to find the best fit and to interpret the role of the independent variables used in various inherit methods. It faces difficulties in fitting due to the narrow boundaries on the model and less flexibility.

SYN	Approximation Loss			CPU Time (sec)			Memory Usage (MBytes)		
	BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD
I	0.12 ± 0.01	**0.04 ± 0.03**	0.07 ± 0.02	79.62 ± 4.5	**12.7 ± 3.1**	13.36 ± 6.31	7.725 ± 0.01	14.7 ± 0.02	**4.23 ± 0.01**
II	0.16 ± 0.04	**0.09 ± 0.03**	**0.09 ± 0.01**	466.91 ± 23.6	325.3 ± 34.9	**106.99 ± 12.61**	157.2 ± 0.01	219.4 ± 4.5	**32.70 ± 0.05**
III	[OoM]	[OoM]	**0.11 ± 0.01**	[OoM]	[OoM]	**831.52 ± 43.82**	[OoM]	[OoM]	**315.11 ± 0.01**
IV	[OoM]	[OoM]	**0.13 ± 0.06**	[OoM]	[OoM]	**2858.45 ± 59.45**	[OoM]	[OoM]	**314.21 ± 0.01**
V	[OoM]	[OoM]	**0.16 ± 0.03**	[OoM]	[OoM]	**7665.23 ± 89.81**	[OoM]	[OoM]	**316.45 ± 0.01**
VI	[OoM]	[OoM]	**0.39 ± 0.11**	[OoM]	[OoM]	**89349.62 ± 253.06**	[OoM]	[OoM]	**312.21 ± 0.01**

Table 10.4: Experimental results for approximation error, CPU Time in seconds and Memory Used in MB for synthetic tensor. We see that OnlineBTD gives stable decomposition in reasonable time and space as compared to classic BTD method. The boldface means the best results.

Evaluation Metrics

We evaluate OnlineBTD and the baselines using three criteria: 1) **approximation loss**, 2) **CPU time** in second, and 3) **memory usage** in Megabytes. These measures provide a quantitative way to compare the performance of our method. For all criterion, lower is better.

10.3.2 Experimental Results

Accurate, Fast and Memory Efficient

First, as shown in Table 10.4, we remark that OnlineBTD is both more memory-efficient and faster than the baseline methods and at the same time to a large extent comparable in terms of accuracy. In particular, the baseline methods fail to execute in the large problems i.e. SYN-III to SYN-VI for given target rank due to out of memory problems during the formation of core tensor $\underline{\mathcal{G}}$. This improvement stems from the fact that baseline methods attempt to decompose in full tensor, whereas OnlineBTD can do the same in streaming mode or on the fly, thus having higher immunity to large data volumes in short time intervals and small memory space with comparable accuracy.

For OnlineBTD, we use $1 - 10\%$ of the time-stamp data in each dataset as existing old tensor data. The results for qualitative measure for data are provided in Table 10.4. For each of the tensor data, the best performance is shown in bold. All state-of-art methods address the issue very well for small datasets. Compared with BTD-ALS and BTD-NLS, OnlineBTD gives lower or similar approximation loss and reduce the mean CPU running time by up to avg. 45% times for big tensor data. For all datasets, BTD-

NLS's loss is lower than all methods. But it is able to handle up to $1K \times 1K \times 100$ size only. Most importantly, however, OnlineBTD performed very well on SYN-V and SYN-VI datasets, arguably the hardest of the six synthetic datasets we examined *where none of the baselines was able to run efficiently (under 48-72 hours)*. It significantly saved $10-60\%$ of computation time and saved $40-80\%$ memory space compared to baseline methods as shown in Table 10.4. Hence, OnlineBTD is comparable to state-of-art methods for the small datasets and outperformed them for the large datasets. These results answer Q1 as the OnlineBTD has better qualitative measures to other methods.

Scalability Evaluation

To evaluate the scalability of our method, firstly, a dense tensor $\underline{\mathbf{X}}$ of small slice size $I = J = 100$ but longer 3^{rd} dimension ($K \in [10^2 - 10^8]$) is created. Its first $1-10\%$ timestamps of data is used for $\underline{\mathbf{X}}_{old}$ and each method's running time for processing batch of 100 data slices at each timestamp is measured. We decomposed it using fixed target rank $R = 3$ and fixed block rank $L = M = N = 5$. The baseline approach consumes more time as we increase the K. The baseline method runs up to 10^9 non-zero elements or 10^5 slices and runs out of memory for further data. However, our proposed method, successfully decomposed the tensor in reasonable time as shown in Figure 10.3. Overall, our method achieves up to 27% (average) speed-up regarding the time required and average 76% gain over memory saving. This answers our Q2. 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

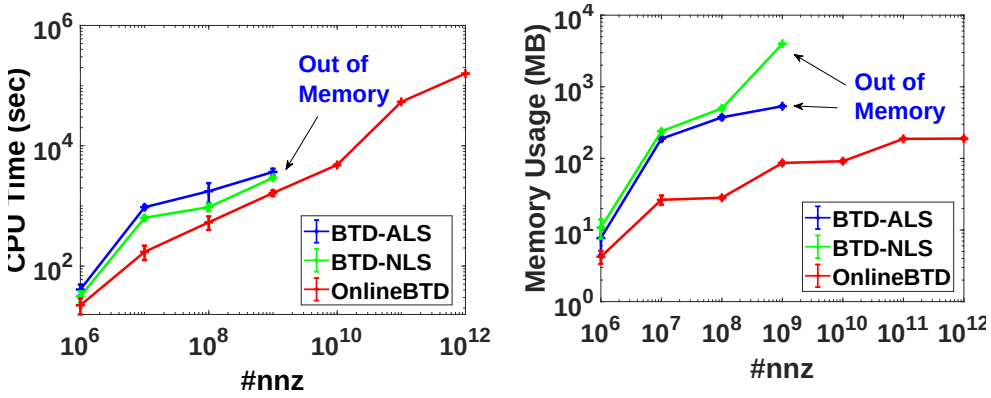


Figure 10.3: CPU time (sec) and Memory (MB) used for processing slices to tensor $\underline{\mathbf{X}}$ incrementing in its time mode. The time and space consumption increases linearly. The mean approximation error is $\leq$10% for all experiments. $\#nnz$: Number of non-zero elements.

slopes vary with different rank used. The analysis is included in the supplementary section (B) due to the limitation of space here.

Sensitivity of OnlineBTD

We extensively evaluate sensitivity of OnlineBTD w.r.t. target rank R, block rank (L, M, N) and noise added during initialization process. For all experiments, we use tensor $\underline{\mathbf{X}} \in \mathbb{R}^{250\times250\times10^5}$ and batch size of 10 slices at a time.

Sensitivity w.r.t tensor rank-R: We fixed the block rank ($L = 5, M = 6, N = 7$) with initialization factor noise is fixed at 10dB.The number of blocks play an important role in OnlineBTD. We see in Figure 10.4 (a) that increasing number of blocks result in decrease of approximation error of reconstructed tensor. The CPU Time and Memory (MB) is linearly (slope 1.03) increased as shown in figure 10.4 (b) and 10.4 (c).

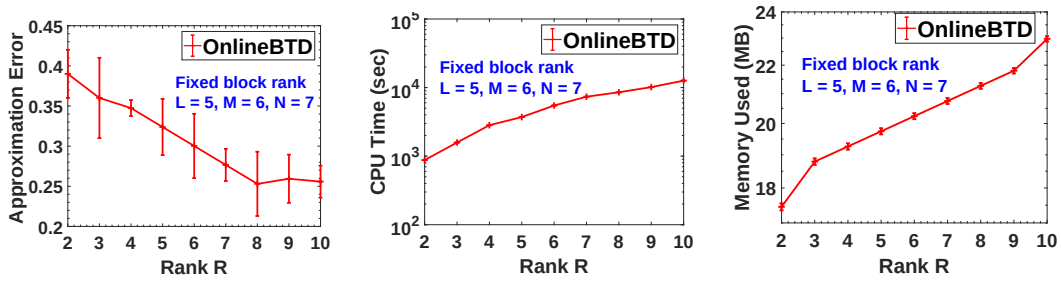


Figure 10.4: The average approximation error, time and memory usage for varying target rank 'R' on different datasets.

Sensitivity w.r.t block rank (L, M, N): To evaluate the impact of block rank - (L, M, N), we fixed tensor rank $R = 5$ and noise added to $10dB$. We can see that with higher values of the (L, M, N), approximation error is improved as shown in Figure 10.5 (a) and become saturated when original block rank is achieved. Higher the block size, more memory and time is required to compute them as shown in figure 10.5.

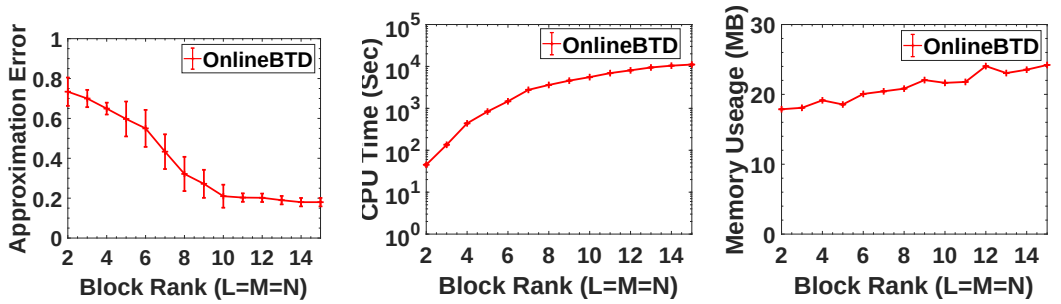


Figure 10.5: The average approximation error, CPU time in seconds and memory usage in MBytes for varying block rank - (L,M,N) of $\underline{\mathbf{X}}$ with original $L = M = N = 10$. As the rank increases, lower approximation error is achieved. Increase in time and memory consumption is expected behaviour.

R	REAL	Approximation Loss			CPU Time (sec)			Memory Usage (MBytes)		
		BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD	BTD-ALS	BTD-NLS	OnlineBTD
5	A	0.36 ± 0.13	**0.34 ± 0.14**	0.37 ± 0.13	385.51 ± 67.52	432.47 ± 46.89	**127.32 ± 13.31**	353.35 ± 0.01	211.34 ± 0.02	**45.37 ± 0.01**
	B	[OoM]	[OoM]	**0.31 ± 0.09**	[OoM]	[OoM]	**742.08 ± 65.37**	[OoM]	[OoM]	**312.34 ± 0.01**
	C	[OoM]	[OoM]	**0.34 ± 0.03**	[OoM]	[OoM]	**4834.3 ± 169.42**	[OoM]	[OoM]	**572.7 ± 0.01**
10	A	0.22 ± 0.11	**0.21 ± 0.09**	0.23 ± 0.02	593.1 ± 53.5	602.4 ± 77.34	**283.45 ± 43.47**	490.3 ± 0.03	381.59 ± 0.01	**90.43 ± 0.01**
	B	[OoM]	[OoM]	**0.23 ± 0.14**	[OoM]	[OoM]	**1267.4 ± 121.3**	[OoM]	[OoM]	**583.4 ± 0.01**
	C	[OoM]	[OoM]	**0.27 ± 0.05**	[OoM]	[OoM]	**8639.7 ± 392.36**	[OoM]	[OoM]	**845.5 ± 0.01**

Table 10.5: A ← ANT-Network; B ← EU-Core; C ← EEG Signals dataset. The average and standard deviation of memory usage and time metric comparison on real world dataset using two different target for five random initialization. For EU-Core and EEG signal datasets, baselines are unable to create intermediate core tensor. The baseline method has less approximation loss as compared to our proposed method. However, the time and memory saving ($> 50\%$) with OnlineBTD is significant.

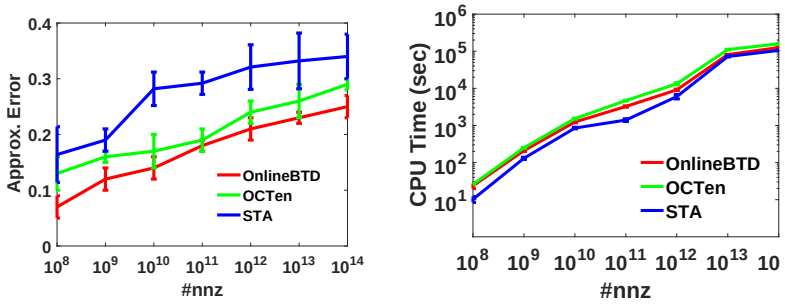


Figure 10.6: The CPU time in seconds and approximation loss for CP/Tucker/BTD online tensor decomposition.

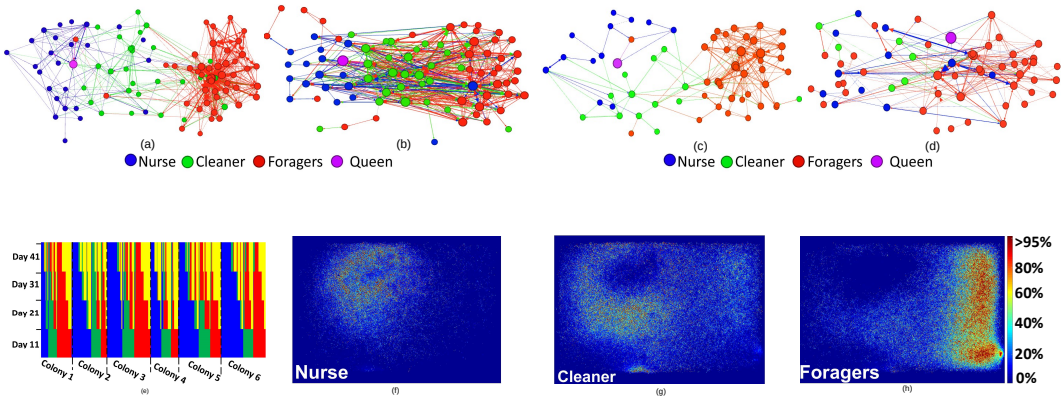


Figure 10.7: (a)-(d) Community detection results of colony 1 on days 11, 21, 31 and 41 of the experiment; (e) The community profiling of the each ant for every 10-day period for all colonies. Blue: nurse community; green: cleaning community; red: foraging community. Ants that disappeared because they are lost or dead are indicated in yellow; (f)-(h) Spatial distribution of nurses, cleaners, and foragers.

Comparison to Online Tucker and Online CP

We also evaluate OnlineBTD performance in terms of computation time with other CP and Tucker online methods as given below:

- **Online Tucker Decomposition** [242]: *STA* is a streaming tensor analysis method, which provides a fast, streaming approximation method that continuously track the changes of projection matrices using the online PCA technique.
- **Online CP Decomposition** [96]: *OCTen* is a compression-based online parallel implementation for the CP decomposition.

Here, we use dense tensor $\underline{\mathbf{X}}$ of slice size $I = J = 1000$ but longer 3^{rd} dimension ($K \in [10^2 - 10^8]$) for evaluation. For CP and Tucker decomposition, we use rank $R = 10$ and for OnlineBTD we use $L = 10$ and $R = 1$. We see in Figure 10.6, OnlineBTD performance is better than OCTen, however, STA outperforms the all the methods. In terms of Fitness, OnlineBTD is better ($> 35\%$) than all the baseline methods.

10.3.3 Effectiveness on Real-world data

We evaluate the performance of OnlineBTD approach for the real datasets as well. The empirical results are given in Table (10.5).

Ant Social Network

Here, we discuss the usefulness of OnlineBTD towards extracting meaningful communities or clusters of ants from Ant Social Network[216] data. It is well known fact that ants live in highly organized societies with a specific division of labor among workers, such as cleaning the nest, caring for the eggs, or foraging, but little is known or researched

about how these division of labor are generated. This network consists of more than 1 million interactions within 41 days between 822 ants. The challenge in community detection of such large data is to capture the functional behavioral based on spatial and temporal distribution of interactions. Therefore, there is a serious need to learn more useful representation.

Qualitative Analysis: For this case study, we decompose tensor in batch of 10 days to extract communities. We observed that the ant organization in this dataset is type Pleometrosis where multiple egg laying queen create there own colonies within organization. There are three communities [173] namely nurse (N), cleaner (C) and forager (F) present in each colony. We compute *F1-score* to evaluate the communities quality.

We focus our analysis on communities within each colony of ants in this dataset. The word "ant colony" refers to groups of workers, fertile individuals, and brood that non-aggressively stay together and work jointly. Our proposed method helped us to track temporal changes among the groups by performing community detection analyses on the batches of 10-day periods. Figure 10.7(e) shows behavioral trajectories of three communities of ants over the 41 days. We observed that ants exhibit preferred behavioral trajectory i.e. move from nursing (located near the queen) to cleaning (move throughout the colony) to foraging (moving in and out of the colony) as they age. The most common transition among them was from cleaner to forager. Conceptually, those ants share similar behaviour in terms of movement and work load. As a result, it becomes a very important challenge to accurately cluster them. Nevertheless, OnlineBTD tracked the behaviour changes very well and achieves significantly good performance in terms of $F1 - score$ i.e ≈ 0.79 as compared to baseline (max F1-Score ≈ 0.63). The communities of colony 1 for day 11, 21, 31 and 41 is

shown in Figure 10.7(a)-(d). The heatmap (Figure 10.7(f)-(h)) provides spatial distribution of community interactions. As any of the baseline does not have capability to capture this temporal behaviour efficiently, our proposed method gives advantage to decompose the data in streaming fashion in reasonable time.

EU-Core [276]

This is a temporal network dataset of communication via. emails between students, professor and staff members of the research institution during October 2003 to May 2005 (18 months) of 42 departments. EU-Core consists of multiple strongly connected communities corresponding to many instances of communication within department. However, we also find numerous re-occurred groups which indicate communication across departments. Interestingly, we note that between Oct,2004 -Jan,2005 one community of researchers established a continuous communication consisting of $80-85$ researchers who interacted each month from the same department. Suddenly, some members disappeared during Dec'04 and again in Jan'05 communication resumed back normally as shown in Figure 10.8. We believe that this may reflect the days of the week of Christmas and New year holidays.

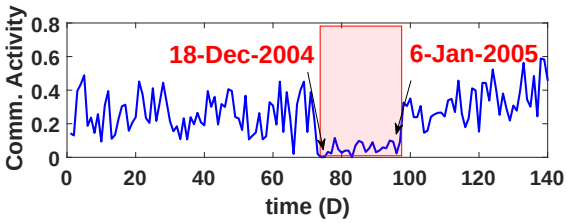


Figure 10.8: A community activity profile of EU-Core.

EEG Signals [222]

The challenge in movement detection of EEG signal data is to capture behaviour regarding the spatio-temporal representations of raw streams. EEG signals usually consist of various noises e.g. cardiac signal. Apart from the systems noises, such as power line interference etc. EEG signals consist from some unavoidable noises like eye blinks, heart beat and muscle activity, all harm to collecting high signal-to-noise ratio EEG signals. It is difficult to make sure that the subjects concentrate on the performing tasks during the whole experiment period. The offline tensor decomposition model assumes that the subject maintains the same spectral structure and topography within the observed window. However, these are typically characterised by evolving repetitive sharp waves. Our proposed

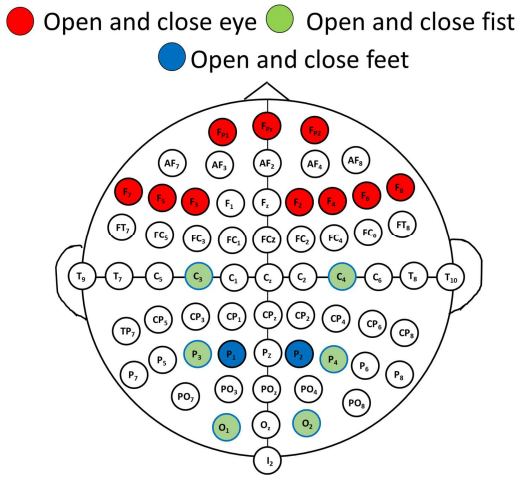


Figure 10.9: EEG Electrode map.

method allows more variability and more interaction between the factors in order to capture such non-stationarities. We find that frontal electrode lobes i.e F_2 through F_8 and front polar electrode lobes F_{P1} and F_{P2} gives the better separations results to differentiate the

motor movements tasks between eye open and eye closed. The parietal (P_3, P_4), central (C_3, C_4) and occipital (O_1, O_2) electrode lobes as shown in Figure 10.9 gives results at temporal scales for open and close left or right fist. This movement capturing over temporal mode is beneficial to users with severe disabilities.

The results in Table 10.5 and above qualitative analysis shows the effectiveness of the decomposition and confirms that the OnlineBTD can be used for various types of data analysis and this answers **Q4**.

10.4 Conclusion

We proposed OnlineBTD, a novel online method to learn Block Term Decomposition (BTD). The performance of the proposed method is assessed via experiments on six synthetic as well as three real-world networks. We summarize our contribution as:

- The proposed framework effectively identify the beyond rank-1 latent factors of incoming slice(s) to achieve online block term tensor decompositions. To further enhance the capability, we also tailor our general framework towards higher-order online tensors.
- Through experimental evaluation on multiple datasets, we show that OnlineBTD provides stable decompositions and have significant improvement in terms of run time and memory usage.
- Utility: we provide a clean and effective implementation of BTD-ALS and all accelerated supporting implementations along with OnlineBTD source code.

There is still room for improving our method. One direction is to explore online BTD for NLS (Non-Linear Square). Another direction is to further extend it for more general

dynamic tensors that may be changed on any modes so that our method can be more suitable for applications such as computer vision.

Chapter based on material published in DSAA 2019 [91].

Chapter 11

Streaming PARAFAC2 Decomposition for Sparse Datasets

"How to incrementally update the decomposition of irregular tensors?"

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 chapter, 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 ($\approx$ 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. The content of this chapter is adapted from the following published paper:

Gujral, Ekta, Georgios Theocharous, and Evangelos E. Papalexakis. "SPADE: S treaming PA RAFAC2 DE composition for Large Datasets." In Proceedings of the 2020 SIAM International Conference on Data Mining, pp. 577-585. Society for Industrial and Applied Mathematics, 2020.

11.1 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 [141], Bro [28], and Papalexakis et al. [197]. Regardless of recent development on temporal data through classic tensor decomposition approaches [18, 94, 186, 243, 284], there are certain instances [117, 118] 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 11.1. The PARAFAC2 decomposition, proposed by Harshman [106], 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 11.1) that is particularly suitable for chromatographic data [12, 233] and electronic health records[202]. d) Similar to the PARAFAC1/CP decomposition method, the PARAFAC2 method provides unique solutions under certain mild assumptions [247, 239], but this model is more loosely constrained and hence, provides more relevant details than PARAFAC1/CP[135].

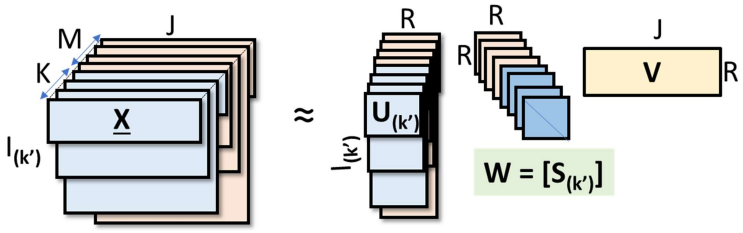


Figure 11.1: An illustration of the tensor decomposition on streaming PARAFAC2 data.

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 [202] data as shown in Figure 11.1, 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 $\mathbf{X}_k$ 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 11.1, PARAFAC2 approximates entire data (old + new) as: $\mathbf{X}_k \approx \mathbf{U}_{k'}\mathbf{S}_{k'}\mathbf{V}^T$, 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 11.3) and synthetic (Table 11.2, 11.3, 11.4) 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 11.2). 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].

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

- **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.

11.2 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:

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 $\mathbf{X}_{old} \in \mathbb{R}^{I_k \times J}$ at time t for $k \in [1, \ldots, K]$, (b) new incoming slice (s) in form of tensor $\mathbf{X}_{new} \in \mathbb{R}^{I_n \times J}$ at any time Δt for $n \in [1, \ldots, N]$,

Find updates of $\mathbf{U}_{new}, \mathbf{V}_{new}$ and $\mathbf{W}_{new}$ **incrementally** to approximate PARAFAC2 tensor $\mathbf{X} \in \mathbb{R}^{I_{(kn)} \times J}$ for $kn \in [1, \ldots, K+N]$ 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 $\underline{\mathbf{X}}$.

11.2.1 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[201] in the data.

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:

$$\begin{aligned}[\mathbf{P}_n, \Sigma_n, \mathbf{Z}_n] = & SVD[\mathbf{H}_{old} \times diag(\mathbf{W}_{rand}(n,:)) \\ & \times (\mathbf{X}_{new_n} \times \mathbf{V}_{old})^T]\end{aligned} \tag{11.1}$$

Given the SVD of above equation, the minimum of Eq. (2.25) 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. (2.26) is achieved by executing the decomposition on a tensor $\underline{\mathbf{Y}} = \mathbf{Q}_n^T \mathbf{X}_{new}(n) \in \mathbb{R}^{R \times J \times N}$ with

frontal slices as :

$$\mathcal{LS} = \text{argmin}\, \frac{1}{2}||\underline{\mathbf{Y}} - [\![\mathbf{H}_{new}; \mathbf{V}_{new}; \mathbf{W}_{new}]\!]||_F^2 \tag{11.2}$$

Note that our loading or factor matrices for CP tensor decomposition in Eq. 11.2 are: $\mathbf{H}_{new} \in \mathbb{R}^{R \times R}$, $\mathbf{V}_{new} \in \mathbb{R}^{J \times R}$ and $\mathbf{W}_{new} \in \mathbb{R}^{N \times R}$.

Initialization of Supporting Matrices

$\mathbf{M}$ can be initialized as MTTKRP [141, 202] 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.

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 11.3.

$$\underset{\mathbf{W}}{\text{argmin}}\, \frac{1}{2}||\underline{\mathbf{Y}}^{(3)}_{new} - \mathbf{W}_{rand}(\mathbf{V}_{old} \odot \mathbf{H}_{old})^T)||_2^F \tag{11.3}$$

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 :

$$\begin{aligned} \widetilde{\mathbf{W}} &= \underline{\mathbf{Y}}^{(3)}_{new} * ((\mathbf{V}_{old} \odot \mathbf{H}_{old})^T)^\dagger \\ &= \underline{\mathbf{Y}}^{(3)}_{new} * (\mathbf{V}^T_{old}\mathbf{V}_{old} * \mathbf{H}^T_{old}\mathbf{H}_{old})^\dagger * (\mathbf{V}_{old} \odot \mathbf{H}_{old}) \end{aligned} \tag{11.4}$$

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 (11.4) can be written as:

$$\widetilde{\mathbf{W}} = \frac{\underline{\mathbf{Y}}_{new}^{(3)} * (\mathbf{V}_{old} \odot \mathbf{H}_{old})}{(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old})} \tag{11.5}$$

The existing MTTKRP is expensive process [202, 283]. Thus the *accelerated MTTKRP* [202] regarding the mode-3 could be written as the inner product between the corresponding r^{th} columns of $\mathbf{H}_{old}$ and $[\underline{\mathbf{Y}}_{new}^{(3)}\ \mathbf{V}_{old}]$, respectively. Thus, in order to retrieve a row $\mathbf{M}^{(3)}(n,:)$, we can simply operate as:

$$\mathbf{M}^{(3)}(n,:) = dot(\mathbf{H}_{old}, \underline{\mathbf{Y}}_{new}^{(3)}\mathbf{V}_{old}) \tag{11.6}$$

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

$$\widetilde{\mathbf{W}} = \frac{\mathbf{M}_{new}^{(3)}}{(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old}))} \tag{11.7}$$

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

$$\mathbf{W}_{new} = \begin{bmatrix} \mathbf{W}_{old} \\ \frac{\mathbf{M}_{new}^{(3)}}{(\mathbf{V}_{old}^T\mathbf{V}_{old} * \mathbf{H}_{old}^T\mathbf{H}_{old}))} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{old} \\ \widetilde{\mathbf{W}} \end{bmatrix} \tag{11.8}$$

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

Algorithm 13: SPADE Update Framework

Data: $\underline{\mathbf{X}}_{new} \in \mathbb{R}^{I_k \times J_2 \times \cdots \times J_{N-1}} \quad \forall k = [1, K]$, 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.

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

Initialize $\mathbf{A}_{rand}^{(N)} \in \mathbb{R}^{K \times R}$ from $\mathbf{A}_{old}^{(N)}$

for $k \leftarrow 1$ *to* K **do**

$[\mathbf{P}_k, \Sigma_k, \mathbf{Z}_k] \leftarrow \text{SVD}(\mathbf{A}^{(1)} \mathbf{A}_{rand}^{(N)}(k) \underline{\mathbf{X}}_{new_k}^T \odot_{i=2}^{N-1} \mathbf{A}^{(i)})$ with Rank R.

$\mathbf{Q}_k = \mathbf{Z}_k \mathbf{P}_k^T$

$\underline{\mathbf{Y}}_k = \mathbf{Q}_k^T \underline{\mathbf{X}}_{new_k}$

end

Update temporal modes of CP tensor $\underline{\mathbf{Y}}$

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

for $n \leftarrow 1$ *to* $N-1$ **do**

Update other modes of CP tensor $\underline{\mathbf{Y}}$ $\mathbf{A}^{(i)} = \frac{\mathbf{M}_{old}^{(i)} + \odot_{i \neq n}^{N} \mathbf{A}^{(i)}}{\mathbf{L}_{old}^{(i)} + \circledast_{i \neq n}^{N} \mathbf{A}^{(i)}} \quad \forall i \in [1, N]$

end

Update first mode of PARAFAC2 tensor $\underline{\mathbf{X}}_{new}$ $\mathbf{U} = \begin{bmatrix} \mathbf{U}_{old} \\ \mathbf{Q}_n * \mathbf{A}^{(1)} \end{bmatrix}$

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

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 :

$$\frac{\delta([\underline{\mathbf{Y}}^{(1)}_{new} - \mathbf{H}_{new}(\mathbf{W}_{new} \odot \mathbf{V}_{old})^T]}{\delta H_{new}} = 0 \tag{11.9}$$

By solving above equation, we obtain:

$$\begin{aligned}\mathbf{H}_{new} =& \frac{\underline{\mathbf{Y}}^{(1)} * (\mathbf{W}_{new} \odot \mathbf{V}_{old})^T}{(\mathbf{W}_{new} \odot \mathbf{V}_{old})^T(\mathbf{W}_{new} \odot \mathbf{V}_{old})} \\ =& \frac{\underline{\mathbf{Y}}^{(1)}_{old} * (\mathbf{V}_{old} \odot \mathbf{W}_{old})^T + \underline{\mathbf{Y}}^{(1)}_{new} * (\widetilde{\mathbf{W}} \odot \mathbf{V}_{old})^T}{(\mathbf{W}^T_{old}\mathbf{W}_{old} * \mathbf{V}^T_{old}\mathbf{V}_{old}) + (\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}} * \mathbf{V}^T_{old}\mathbf{V}_{old})} \\ =& \frac{\mathbf{M}^{(1)}_{old} + \underline{\mathbf{Y}}^{(1)}_{new} * (\widetilde{\mathbf{W}} \odot \mathbf{V}_{old})^T}{\mathbf{L}^{(1)}_{old} + (\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}} * \mathbf{V}^T_{old}\mathbf{V}_{old})}\end{aligned} \tag{11.10}$$

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:

$$\mathbf{M}^{(1)}_{new} = \sum_{n=0}^{N} \underline{\mathbf{Y}}_{new}\mathbf{T}_n = (\underline{\mathbf{Y}}_{new}\mathbf{V}_{old}) * \mathbf{W}^{'} \tag{11.11}$$

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 [202]. Hence $\mathbf{H}_{new}$ can be updated as :

$$\mathbf{H}_{new} = \frac{\mathbf{M}^{(1)}_{new}}{\mathbf{L}^{(1)}_{new}} = \frac{\mathbf{M}^{(1)}_{old} + (\underline{\mathbf{Y}}_{new}\mathbf{V}_{old}) * \mathbf{W}^{'}}{\mathbf{L}^{(1)}_{old} + (\mathbf{W}^{'T}\mathbf{W}^{'} * \mathbf{V}^T_{old}\mathbf{V}_{old})} \tag{11.12}$$

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 $\mathbf{M}^{(2)} = \sum_{n=0}^{N} \underline{\mathbf{Y}}_{new}^{T} \mathbf{T}_n$ as :

$$\mathbf{V}_{new} = \frac{\mathbf{M}_{new}^{(2)}}{\mathbf{L}_{new}^{(2)}} = \frac{\mathbf{M}_{old}^{(2)} + (\underline{\mathbf{Y}}_{new}^{T} \mathbf{H}_{new}) * \mathbf{W}'}{\mathbf{L}_{old}^{(2)} + (\mathbf{H}_{new}\mathbf{H}_{new} * \mathbf{W}'^{T}\mathbf{W}')} \tag{11.13}$$

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:

$$\mathbf{U}_{new} = \begin{bmatrix} \mathbf{U}_{old} \\ \\ \mathbf{Q}_n * \mathbf{H}_{new} \end{bmatrix} \tag{11.14}$$

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*[202]. 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 13.

11.2.2 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 $(\mathbf{U}_{old}, \mathbf{A}_{old}^{(1)}, \mathbf{A}_{old}^{(2)}, \ldots, \mathbf{A}_{old}^{(N-1)}, \mathbf{A}_{old}^{(T_1)})$ 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 :

$$\mathbf{A}^{(N)} = \begin{bmatrix} \mathbf{A}_{old}^{(N)} \\ \mathbf{A}_{new}^{(N)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{old}^{(N)} \\ \frac{\mathbf{M}^{(N)}}{\circledast_{i=1}^{N-1} \mathbf{A}^{(i)}} \end{bmatrix} \tag{11.15}$$

The Non-Temporal modes can be updated as:

$$\mathbf{A}^{(i)} = \frac{\mathbf{M}_{old}^{(n)} + \odot_{i \neq n}^{N} \mathbf{A}^{(i)}}{\mathbf{L}_{old}^{(n)} + \circledast_{i \neq n}^{N} \mathbf{A}^{(i)}} \tag{11.16}$$

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

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

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_{old} \\ \mathbf{Q}_n * \mathbf{A}^{(1)} \end{bmatrix} \tag{11.17}$$

We obtain the general version of update rule of our SPADE for N-mode tensor, as presented in Algorithm 13. **NOTE**: Line 3-5 can be executed in parallel.

11.3 Experiments

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.

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 11.1: 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**

11.3.1 Experimental Setup

Synthetic Data Generation

The specifications of each synthetic dataset are given in Table 11.1. For all synthetic data we use rank $R = 15$. The entries of loading matrix $\mathbf{V}$ and $\mathbf{W}$ are Gaussian with unit variance, and orthogonality is imposed on factors $\mathbf{H}$ and $\mathbf{Q}$, then a few entries are clipped to zero randomly to create a sparse PARAFAC2 tensor. The MATLAB script is provided on *link*[1].

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

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

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**[103] 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

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

as sequence-by-tutorial-by-user. We have semi synthetic ground truth values for each user based on tutorial watched.

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.

Baselines

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

- **PARAFAC2** [135] : an implementation of standard fitting algorithm PARAFAC2 with random initialization.

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

- **SPARTan** [202] : a scalable PARAFAC2 fitting algorithm was proposed for large and sparse data.

Note: Our proposed method is natural extension of scalable PARAFAC2[202].

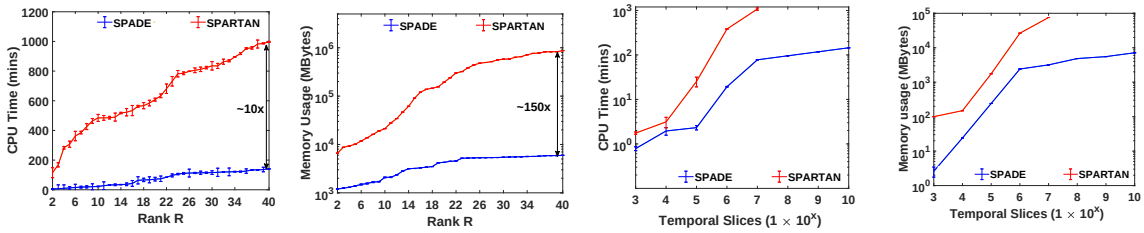


Figure 11.2: The average time in minutes and memory usage in MBytes for varying target rank (1^{st},2^{nd}) of synthetic data of size ($1000 \times 1000 \times 10^5$) and varying number of subjects(K) (3^{rd}, 4^{th}) for synthetic data of size ($100 \times 100 \times [10^3, 10^{10}]$).

11.3.2 SPADE is fast and memory-efficient

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 (11.2), it can be concluded that, the SPADE best performance is on par to the state-of-the-art (i.e. classic PARAFAC2 as 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 (11.3, 11.4). 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.

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 (11.5). 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.

Scalability Evaluation

We also valuate the scalability of our algorithm on synthetic and real dataset. Firstly, a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{1000\times1000\times10^5}$ is decomposed with increasing target rank.The baseline approach consumes more time as we increase the target rank as shown in Figure 11.2 (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 $\underline{\mathbf{X}} \in \mathbb{R}^{100\times100\times10^{10}}$ 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 11.2 (3^{rd}, 4^{th}).

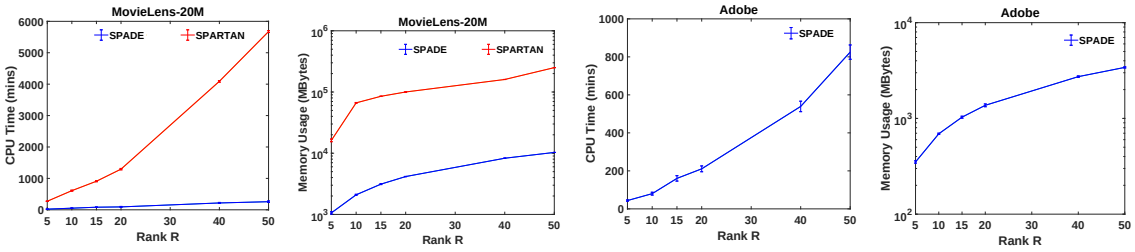


Figure 11.3: The average time in minutes and memory usage in MBytes for varying target rank for both the real datasets. For Adobe dataset baseline method runs out of memory.

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 11.3. 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 11.3) 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 included due to limitation of space here.

11.4 Community discovery on Adobe data

11.4.1 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.

11.4.2 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.

11.4.3 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 [80] 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 11.4(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 Figure11.4(b), we show top 5 communities of the dataset as clustered by SPADE with $R = 24$. Qualitatively, we see that the method's 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.

11.5 Conclusions

We propose SPADE, a novel online PARAFAC2 decomposition method. We demonstrate its efficiency and scalability over synthetic and real datasets. The SPADE

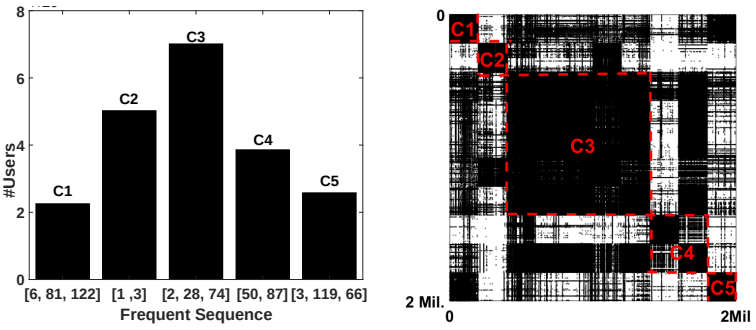


Figure 11.4: For top 5 communities (a) frequent sequence of tutorials watched (b) spy-plot of user-user view.

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.

Chapter based on material published in SDM 2020 [98].

Chapter 12

Automatic PARAFAC2 Tensor Analysis

"How to automatic mine data using PARAFAC2 in a data driven and unsupervised way.?"

In data mining, PARAFAC2 is a powerful and multi-layer tensor decomposition method that is ideally suited for unsupervised modeling of "irregular" tensor data, e.g., patient's diagnostic profiles, where each patient's recovery timeline does not necessarily align with other patients. In real-world applications, where no ground truth is available for this data, how can we automatically choose how many components to analyze? Although extremely trivial, finding the number of components is very hard. So far, under traditional settings, to determine a reasonable number of components, when using PARAFAC2 data, is to compute decomposition with a different number of components and then analyze the outcome manually. This is an inefficient and time-consuming path, first, due to large data

volume and second, the human evaluation makes the selection biased. In this chapter, we introduce APTERA, a novel automatic PARAFAC2 tensor mining that is based on locating the L-curve corner. The automation of the PARAFAC2 model quality assessment helps both novice and qualified researchers to conduct detailed and advanced analysis. We extensively evaluate APTERA 's performance on synthetic data, outperforming existing state-of-the-art methods on this very hard problem. Finally, we apply APTERA to a variety of real-world datasets and demonstrate its robustness, scalability, and estimation reliability. The content of this chapter is adapted from the following paper:

Guiral, Ekta, and Evangelos E. Papalexakis. "Aptera: Automatic PARAFAC2 Tensor Analysis." The content of this chapter was under blind peer review at the time of thesis submission.

12.1 Introduction

Tensors are the generalization of vectors and matrices. They are ubiquitous (e.g. images, videos, and social networks) and ever-increasing in popularity. With the opportunity to handle large volumes and velocity of data due to recent technical developments, such as mobile connectivity [189], digital tools [166], biomedical technology [24], and modern medical testing techniques [47], we face multi-source and multi-view [93] datasets. Suppose, for example, we are given health care record data, such as Centers for Medicare and Medicaid (CMS) [47], and we have information about patients who visited the hospital, or who got what kind of diagnosis in which visit, and when. Time modeling is difficult for the regular tensor factorization methods (e.g. CP [36] and Tucker [256]), due to either data

irregularity or time-shifted latent factor appearance of such data. Hence, data is formulated as a 3-mode PARAFAC2 tensor [106] as shown in Figure 12.1.

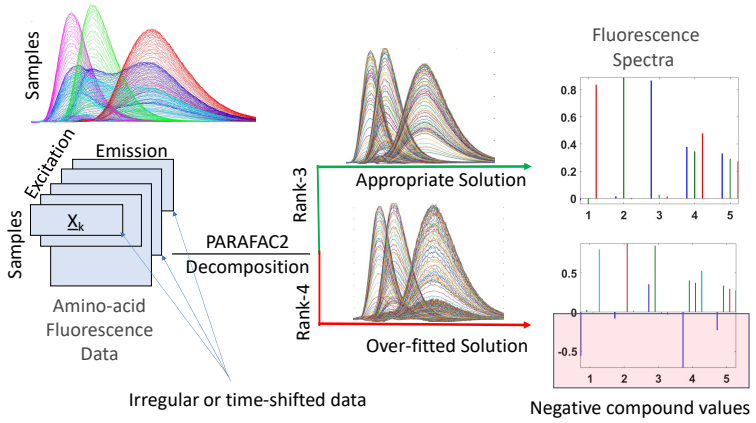


Figure 12.1: Amino acid data PARAFAC2 decomposition. The correct model has three components namely tryptophan, tyrosine and phenylalanin (which are chemically verified [134]), so a rank-4 solution would overfit by introducing misleading components and further introducing negative values in order to allow for cancellation of artifacts introduced.

With a rich variety of applications, PARAFAC2 decomposition [106] is a very effective analytical method and if performed correctly, it can reveal underlying structures of data. However, there are research issues that need to be tackled in the field of data mining in order for PARAFAC2 decompositions to assert their role as an effective tool for practitioners. One challenge, which has received considerable attention, is finding the correct number of components aka rank of PARAFAC2 decomposition. The tensor rank calculation has been proven to be NP-Hard [116] and in various cases NP-Complete [110]. Additionally, irregularity of data makes it a more complex and trivial task. This is the reason that various tensor mining papers [6, 135], understandably, set the number of components manually.

PARAFAC2 decomposition is able to handle various chromatographic data and choosing the correct number of components allows it to separate each variability source by using spectral information. Consider amino acid data [134] where three compounds tyrosine, tryptophan and phenylalanine dissolved in phosphate-buffered water. In Figure (12.1), PARAFAC2 decomposition with rank-3 resembles the pure spectra of tryptophan, tyrosine and phenylalanin. When PARAFAC2 decomposition with rank-4 is applied to this data, the fourth component does not resemble any of the compounds and in fact, it does not seem to reflect any chemical information. Therefore, it becomes very important to select the correct number of components to solve real-world problems.

In literature, one popular approach to find the rank of CP tensor is core consistency diagnostic (CORCONDIA) [31]. The CORCONDIA essentially assesses significant deviations from a super-diagonal core tensor. This would suggest that the CP decomposition is not optimal either because the selected rank is not correct, or the CP model cannot describe the data well enough. This approach is widely studied and explored among the tensor mining community. AutoTen [193] is a powerful method that uses CORCONDIA as a building block to provide unsupervised detection of multi-linear low-rank structure in tensors. Over the last few years, there has been various methods [228, 252, 282] proposed to find the number of components of fixed dimension tensor data. However, only one method namely Autochrome [129] estimates rank for irregular data. Unfortunately, this method uses various computation diagnostics that require the conversion of irregular data to regular data. This is expensive in terms of memory utilization.

To fill the gap, we propose a effective and efficient method APTERA to estimate the rank of irregular 'PARAFAC2' data that discover the number of components (interchangeably rank) through higher-order singular values. We observe that the nature of higher-order singular values is like 'L' shape (See Figure (12.2)) i.e. singular values start with high values and slowly dies to the end of computations. Also, we observed that the correct number of components falls exactly on the maximum curvature of the L-curve. Hence, we exploit the well known numerical method approach called L-curve corner detection [38] in our work to estimate the number of components. Our contributions are summarized as follows:

- **An efficient and simple rank estimation method**: We introduce APTERA, a scalable and effective algorithm for estimating the number of components of PARAFAC2 decompositions of irregular tensors that admits an efficient parallel implementation. We make our Matlab implementation publicly available on the link[1].
- **Extensive Evaluation**: We evaluate APTERA on synthetic and multiple real-world data, in order to study the behavior of our method in comparison to other baselines.
- **Technology Transfer:** This work provides an efficient way to apply the numeric method idea of L-curve corner to find the rank of PARAFAC2 tensor data, aiming to explore its capabilities and promote it within the data mining community.

12.2 Related work

As outlined in the introduction, rank detection and low-rank structure discovery are very hard problems, and there are currently no general-purpose methods that can achieve

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

these tasks efficiently. In the tensor mining literature, there exist most effective and efficient methods by the name of Core Consistency Diagnostic or CORCONDIA [29, 31], AutoTen [193] and NSVD [252], that can serve as a guide to judging how well a tensor is modeled by a given PARAFAC/CP decomposition. Most recently, a Bayesian robust tensor factorization (BRTF) [282] employs a fully Bayesian generative model for automatic CP-rank estimation. However, this method often under/over-estimates the true rank of tensors and has a high computational cost. To automatically estimate the Tucker-rank, an automatic relevance determination (ARD) algorithm is applied for sparse Tucker decomposition [176]. ARD is a hierarchical Bayesian approach widely used in many methods [204], but its efficiency is quite low. However, these methods are not directly applicable to the irregular tensor data.

There is very limited work done for PARAFAC2 data rank estimation. There exists a method named Autochrome [129] which uses PARAFAC2 decomposition for estimating the rank of tensor data. The method is based on a number of model diagnostics (quality criteria) collected from models with different numbers of factors. They combining these diagnostics to assess what are the appropriate number of components of data. However. this method is limited to gas chromatography–mass spectrometry data and also various diagnostics computations require regular CP/PARAFAC1 tensor as input instead of the irregular (PARAFAC2) tensor.

To our best knowledge, there is no work in the literature that deals with the reveling a number of components of PARAFAC2 decomposition without using expensive computations of Core Consistency Diagnostics and not limited to a specific type of data.

To fill the gap, we propose a scalable and efficient method that reveals the number of components of the PARAFAC2 model.

12.3 Proposed Method: Aptera

In data mining applications (e.g. chromatography, health care), we are given a very large irregular multi-layer data which is required to analyze by domain researchers, and we are asked to identify various useful patterns that could potentially help to grow the business or provide valuable insights about data. Most of the time, this analysis is done unsupervised as collecting ground truth is extremely expensive and requires human intervention. Unfortunately, it is not straightforward to determine the proper number of components for PARAFAC2 tensors. Since CORCONDIA based methods have instabilities in the quality estimations[252] and, therefore, we propose a new method for finding the structure in PARAFAC2 tensor data using the L-corner approach that reduces the human intervention and trial-and-error fine-tuning. Our proposed method consists of three steps as described below.

Informal Problem: **Given** a 'irregular' tensor without labelled data and maximum possible rank R_{max}, how can we analyze it using the PARAFAC2 decomposition so that we can also

- Determine automatically a good number of components for the decomposition.
- Minimize human involvement and trial-and-error fine-tuning.

12.3.1 PARAFAC2 decomposition

Here, we solve R_{max}-component PARAFAC2 decompositions as given in Equ. (12.1) by using random initialization. For each decomposition, we keep same initial parameters i.e. number of maximum iterations, tolerance for convergence etc.

$$\begin{aligned} \mathcal{L} = \sum_{k=1}^{K} \underset{\mathbf{Q}_k}{\arg\min} \frac{1}{2} ||\mathbf{X}_k - \mathbf{Q}_k \mathbf{H} \mathbf{W} \mathbf{V}^T||_F^2 \quad \forall k \in [1, K] \\ \text{subject to} \quad \mathbf{Q}_k \mathbf{Q}_k^T = \mathbf{I}_{R_{max}} \end{aligned} \tag{12.1}$$

Due to the irregular nature of the first mode of PARAFAC2 data, we use its resultant latent factors to create CP tensors using the Khatri-Rao product on factors $\underline{\mathbf{Y}} = (\mathbf{H} \odot \mathbf{V} \odot \mathbf{W}) \in \mathbb{R}^{R_{max} \times J \times K}$. This gives us a flexibility to use any existing method to discover the rank of the reconstructed tensor. Unfortunately, CORCONDIA based methods like AutoTen [193], Autochrome [129] get confused because the input, i.e the CP tensor, is created using outcome of PARAFAC2 decomposition instead of actual data which could have a different number of components. For example, consider the PARAFAC2 data has total of 10 components and we factorize this data with $R_{max} = 20$. When we provide the CP tensor with $R_{max} = 20$ to CORCONDIA based methods, it is highly likely possible that Core Consistency diagnostic metric is close to 100% at $R_{max} = 20$, because it can trivially produce "super-diagonal" core. To overcome such instabilities, we use multi-linear orthogonal projections via Higher Order Singular Value Decomposition (HOSVD) for discovering the number of component.

12.3.2 Formation of L-curve using Pareto Optimal Truncation

The Singular Value Decomposition (SVD) gives the best low-rank approximation of a matrix. In the sense of multi-linear rank, a generalization of the SVD is the higher-order SVD (HOSVD). Nowadays, it is better known with the effort of de Lathauwer et al. [55], who analyzed the structure of core tensor and proposed to use multi-linearity to discover the rank of the tensor. Motivated by this, we compute HOSVD of $\underline{\mathbf{Y}}$ as given in Equ. (12.2).

$$[\underline{\mathbf{G}}, \mathbf{A}, \sigma] = HOSVD(\underline{\mathbf{Y}}) \tag{12.2}$$

where $\underline{\mathbf{G}}$ is decomposed core tensor, $\mathbf{A}$ is set of matrices for each dimension and σ is set of n-mode multi-linear (interchangeably higher-order) non-negative singular values which appear in decreasing order. We can reconstruct 3-mode CP tensor using $\underline{\mathbf{G}}$ and $\mathbf{A}$ as given below Equ. (12.3).

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times \mathbf{A}^1 \times \mathbf{A}^2 \times \mathbf{A}^3 \tag{12.3}$$

Selecting the appropriate degree of compression is equivalent to estimating the rank of the tensor. Though the best rank approximation is NP-hard [116], a satisfying result can always be estimated by choosing a proper degree of truncation. Here, we use Pareto optimal truncation [11, 112] based on the upper bound on the singular values. For any possible 3-mode tensor dimensions, the corresponding relative error E can be defined as

$$vec(E_{rjk}) = \sum_{r=1}^{R_{max}} \sigma\{1\}(r) + \sum_{j=1}^{J} \sigma\{2\}(j) + \sum_{k=1}^{K} \sigma\{3\}(k) \tag{12.4}$$

$$E(n) = E_{rjk} = \frac{\sqrt{E_{rjk}}}{||\sigma\{1\}||} \tag{12.5}$$

where $n \in \{1, 2, 3, \ldots, R_{max}JK\}$ is linearized index pairs of R_{max}, J and K e.g. $(n = 1) \leftarrow [r = 1, j = 1, k = 1]$. Next, we define the points on the 2D plane with possible tensor dimension $\mathbf{d}$ as:

$$P(n) = \sqrt{(x(n))^2 + (y(n))^2}, \quad x(n) = ||d(n)||; \quad y(n) = E(n) \tag{12.6}$$

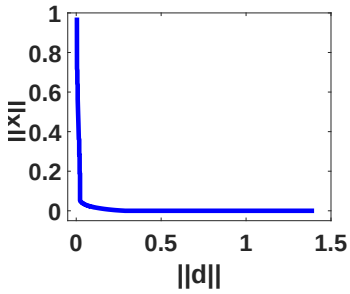


Figure 12.2: The L-curve formed after Pareto optimal truncation on multi-linear singular values of synthetic tensor.

where $\mathbf{d}$ is a vector of multi-indices and represents as $d(1) \leftarrow [r = 1, j = 1, k = 1]$, $d(2) \leftarrow [r = 2, j = 1, k = 1]$, and so on. The $||d(n)||$ is computed as $\frac{(r*R_{max})+(j*J)+(k*K)+(rjk)}{R_{max}JK}$. Now, we sort the points P and update residual norm (x) and solution norm (y) accordingly. By eliminating the P values that do not satisfy the monotonic condition, we can get a Pareto front end [102]. Having realized the important roles played by the norms of the solution y and norms of the residual x, it is quite natural to plot these two quantities versus each other, i.e., a trade-off curve as shown in Figure (12.2). This is precisely the L-curve that can be utilized for estimation of the rank of the tensor. Algorithm 14 provides the pseudocode of computing Pareto front end.

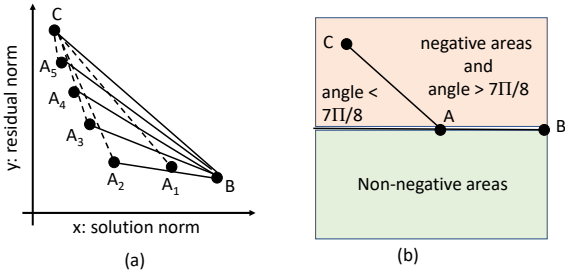


Figure 12.3: (a) Pictorial view to find the corner point of an L-curve. Fixing point B and C and forming triangles with A_1, A_2, A_3, A_4, and A_5. (b) Scenario that shows a case when point A could be a corner of the L-curve. The corner point will be the point A with the smallest angle which is also less than $7\pi/8$ and with the corresponding triangle ABC having negative area.

12.3.3 Rank Estimation with L-curve Corner

In this step, we use the L-curve corner method [50] to estimate the number of components of a tensor. To improve the efficiency of the method, we can adapt a triangle method [38] that uses geometric properties like the angle and direction of the triangle to estimate the L-curve curvature as shown in Figure (12.3).

Although, above process gives estimated rank for each dimension, but note that PARAFAC2 requires only a single rank value. Therefore, we report minimum rank predicted across tensor regular modes. Putting everything together, we end up with Algorithm (10) which is an efficient solution to the minimization problem of Equ. (12.1) and automatically

determine the good number of components for the decomposition.

Algorithm 14: Pareto Optimal Truncation

Data: E, tensor dimensions (R, J, K).

Result: Estimated rank R_{out}.

$\mathbf{d} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ \vdots & \vdots & \vdots \\ R & J & K \end{bmatrix}$

$x \leftarrow \frac{R*\mathbf{d}(n,1)+J*\mathbf{d}(n,2)+K*\mathbf{d}(n,3)+product(\mathbf{d}(n,:))}{(RJK)}$

$y \leftarrow E$; $P = \sqrt{x^2 + y^2}$; $p = 1$

//Pareto Optimal Truncation

Rearranging x and y based on P.

while $p \leq length(x)$ **do**

  $idx \leftarrow (x >= x(p) \& y > y(p)) | (x > x(p) \& y >= y(p))$

  if $\forall idx$ **then**

    $d = d(\neg idx, :); x = x(\neg idx); y = y(\neg idx)$

  end

  $p = p + 1 - \sum idx(1 : p)$

end

//L-curve corner detection

$\underline{\mathbf{AB}} \leftarrow [x - x'; y - y']$; $\underline{\mathbf{AC}} \leftarrow [x(n) - x'; y(n) - y']$

return $\underline{\mathbf{AB}}, \underline{\mathbf{AC}}$

Summary: First, we compute PARAFAC2 decompositions with either R_{max} or $\max(J, K)$ (see Section 12.3.1). Then, we create the CP tensor $\underline{\mathbf{Y}}$ using outer product of factor matrices (See Section 12.3.1). Next, we compute the HOSVD on $\underline{\mathbf{Y}}$ and perform

Pareto optimal truncation (see Section 12.3.2). Finally, find L-curve corner (final "best rank") using triangle method as discussed in Section 12.3.3.

Algorithm 15: APTERA: Automatic PARAFAC2 Tensor Mining

Data: T

ensor $\underline{\mathbf{X}}$ and maximum budget for component search R_{max} (Optional), No of Experiments N. **Result:** P

ARAFAC2 decomposition $\mathbf{U}, \mathbf{V}, \mathbf{W}$ of $\underline{\mathbf{X}}$, R_{est} **for** $n = 1 \dots N$ **do**

  Run PARAFAC2 decomposition for R_{max} (if given) or for $\max(J, K)$ components.

  Create CP Tensor $\underline{\mathbf{Y}}$ using above components as described in the text.

  Compute HOSVD; $[\sigma] = HOSVD(\underline{\mathbf{Y}})$ to obtain multi-linear singular values.

  Perform truncation on singular values (refer Section 12.3.2 and Algorithm (1a)).

  Compute L-corner using Section12.3.3 as $R_{out}(n)$.

end

Choose most repeating $R_{out}(n)$ to select R_{est}.

Output the chosen R_{est} and the corresponding decomposition.

12.4 Experiments

We design experiments to answer the following questions: **(Q1)** How accurately APTERA detect rank as compared to baseline algorithm? **(Q2)** How APTERA used in real-

Dataset	Dimension	Components
Syn-I	$200 \times 500 \times 1000$	5 (Synthetic)
Amino Acid	$5 \times 201 \times 61$	3 (See[134])
Wine-GCMS	$2700 \times 200 \times 44$	4 (See [233])
EU-Core	$986 \times 986 \times 827$	28 (See [276])
CMS	$250 \times 1000 \times 98000$	NA

Table 12.1: Details for the datasets.

world scenarios? **(Q3)** How does the running time of APTERA increase as tensor data grow (in 3rd mode)?

12.4.1 Synthetic Data Description

A first step in evaluating our method is to check its performance on simulated data whose rank and factors can be pre-defined. We create synthetic tensors by generating two-factor matrices with R columns each, where their elements are drawn as Gaussian with unit variance. Then, factors are column-wise normalized. The set of factor matrices for irregular mode is created in such a way that it retains the property of orthogonality. By considering these matrices as the PARAFAC2 factor matrices, therefore, the rank of the PARAFAC2 tensor will be exactly R. We considered a setup with 1000 subjects, 500 feature variables, and a maximum of 200 observations for each subject with rank-5. Also, we deformed the generated tensor data by an additive noise tensor that has the rank higher than 5 but has norm 2× less than actual synthetic data.

12.4.2 Real Data Description

We evaluate the performance of the proposed method APTERA for the real datasets to assess the practicality in real-world scenarios. For this reason, in our experiments we includes real data sets as shown in Table (12.1).

- **Chemical Data**: We used two chemical data i.e. Wine-GCMS (Gas Chromatography Mass Spectroscopy) [233] and Amino acids fluorescence data [134]. The wine data consists 44 samples of red wines, 44 samples, produced from the same grape (Cabernet Sauvignon), harvested in different geographical areas. For each sample a mass spectrum scan is measured at 2700 elution time-points was obtained providing a data for 44 samples have size 2700×200. The Amino acid data set consists of 5 simple laboratory-made samples. Each sample has size 333. The dimensions are Each sample contains different amounts of tyrosine, tryptophan and phenylalanine dissolved in phosphate buffered water. Through core consistency experiments, it is found that amino acid data has rank-3 [2].
- **Social Network Data**: We also analyze EU-Core [276] data and it consists of emails between members of the research institution during October 2003 to May 2005 (18 months) and messages can be sent to multiple recipients of 42 departments. For this dataset, we created tensor as days-by-researcher-by-researcher. For this data previous research has identified 28 components [276] so we assume that the true rank is equal or around that number.

[2] http://www.models.life.ku.dk/Amino_Acid_fluo

- **Healthcare Data**: This dataset is synthetically created by Centers for Medicare and Medicaid (CMS) [47] by using 5% of real medicare data and includes $98K$ subjects. We created PARAFAC2 tensor as visits - by - diagnosis - by - patient.

Baselines

In this experiment, three baselines have been used as to evaluate the performance.

- **Autochrome**[129]: an implementation of detecting components for PARAFAC2 model which uses Core Consistency Diagnostic on CP tensor $\underline{\mathbf{Y}}$.
- **Iteration based** [119, 129]: We postulate that this could be a decent baseline because if appropriate number of components are selected, the PARAFAC2 decomposition converges fast within few iteration. Therefore, we consider it as one of our baseline.
- **NSVD based** [252]: A Normalized Singular Value Deviation (NSVD) method used to find the rank via CP tensor $\underline{\mathbf{Y}}$.
- **Tucker ARD based** [176]: it is an automatic relevance determination algorithm for Tucker decomposition using the gradient based sparse coding algorithm. We use this method to find the rank via CP tensor $\underline{\mathbf{Y}}$.
- **BRTF Based** [282]: a Bayesian robust tensor factorization which employs a fully Bayesian generative model for automatic CP-rank estimation

All methods except Autochrome require a maximum bound R_{max} on the rank; for fairness, we set $R_{max} = 2R_{original}$ for all methods. Note that all comparisons were carried out over 10 iterations (or experiments) for all methods and most repeated rank is reported here.

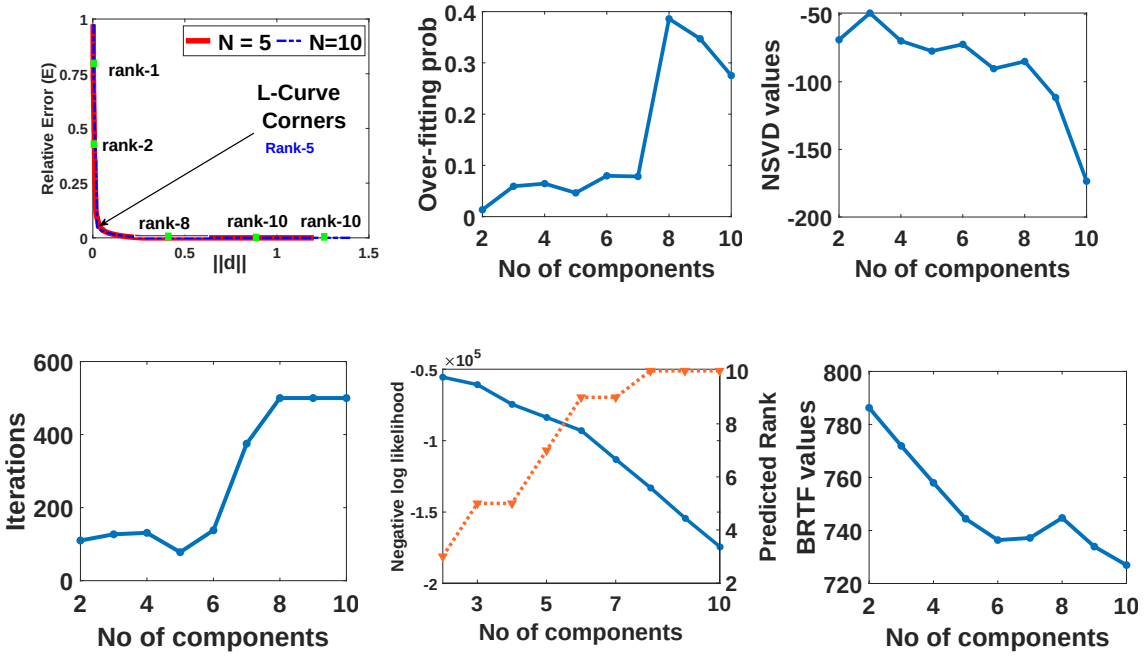


Figure 12.4: Baselines comparison on the synthetic dataset. From left to right represents, (a) our proposed method APTERA for 5^{th} and 10^{th} experiment run, (b) Autochrome, (c) NSVD based, (d) Iteration based, (e) Tucker ARD based, and (f) BRTF based method.

12.4.3 Q1: Rank Structure of Synthetic Dataset

For our synthetic dataset, we observe in Figure (12.4) that APTERA presents a quite distinct L-curve corner at rank-5 for all given random initilizations which is the correct answer. On the other hand, even though Autochrome and NSVD methods seem to approximate a region around 7 components (tested for $2-10$ rank), it struggles to give a definitive answer and leaves open the possibility of up to 8 or more components. Both, Tucker ARD and BRTF based methods not able to provide certain solution for synthetic data. Interestingly, even iteration based baseline seems to be working better than Autochrome and NSVD, showing a subtle indication at 5 components.

12.4.4 Q2: Rank Structure of Real Datasets

While our purposed method APTERA performs reasonable on synthetic tensor data, indicating a L-curve corner exactly where the predefined number of component is, but in order to evaluate its practicality in real-world scenarios, it is also important to research its performance and behavior on real-world data. For this reason, we analyze a range of real data sets as shown in Table (12.1).

Chemical Data

In Table (12.2), we can see that APTERA estimates the 4 components of wine-GC data, as opposed to iteration based and BRTF based baseline methods which suggests only 3 components. NSVD and Tucker ARD based baselines also fails to identify the correct answer by vaguely indicating only 6 or 7 components, and on other hand, Autochrome also suggests correct 4 components. It is interesting to observe that for amino acid data, our proposed method APTERA, Iteration based and Tucker-ARD methods are able to estimate the correct rank r=3 where all other baselines fail to discover 3 components in the data, as shown in Table (12.2). Figure (12.5), shows the importance of estimating correct rank. Correct rank (Figure 12.5(b)) can extract all relevant patterns without overlapping them (i.e. under-fitted model) in case low rank estimated than actual rank (Figure 12.5(a)). Also, more negative values indicated over-fitted model (Figure 12.5(c)) for such datasets. Our proposed method clearly outperforms most of the baselines.

Methods	Wine	Amino	EUCore	CMS	Wine	Amino	EUCore	CMS
$R_o \rightarrow$	4	3	28	–	Percent Deviation (%)			
Autochrome	4	2	26	*OoM*	**0.00**	−33.33	−7.15	–
NSVD	7	6	13 and 35	50	75.00	100.00	−53.57	–
Iterations	3	**3**	25	11	−25.00	**0.00**	−10.71	–
Tucker ARD	6	**3**	33	2	50.00	**0.00**	17.78	–
BRTF	3	2	49	*OoM*	−25.00	−33.33	75.00	–
Aptera	**4**	**3**	29	9	**0.00**	**0.00**	**3.57**	–

Table 12.2: Performance of APTERA for rank estimation. Numbers where our proposed method outperforms other baselines are bolded. The negative sign indicates solution is under-fitted and positive values (> 0 for deviation) indicates over-fitted solution.

Social Network Data

In Table (12.2), we see that the behavior of NSVD on the EUcore dataset is more complicated, providing multiple estimation at 13 and 35 components. On other hand, Tucker ARD and BRTF provides 33 and 49 components, respectively and clearly over-fitting the data, while Iteration based and Autochrome detect 25 and 26 components. Note that even though it is hard to obtain ground truth for this type of data, APTERA is able to at least suggest potential structure at all the points where other baselines do as well. It is also interesting to observe that the 28 components or clusters that the authors identify in [276] make sense considering that not only APTERA, Iteration based and Autochrome also provide an indication near this number of components.

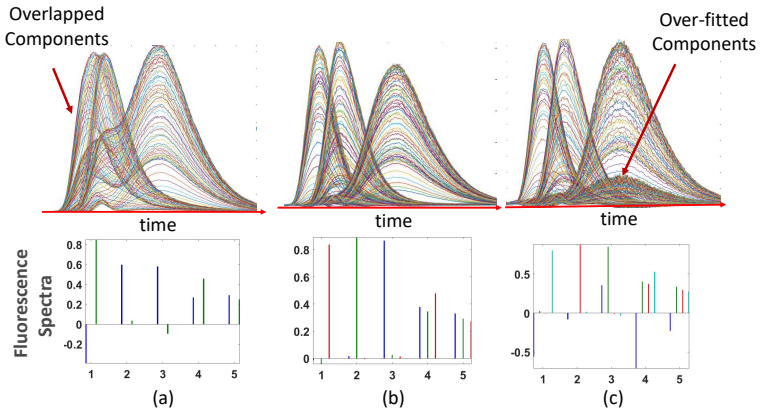


Figure 12.5: Weighted elution profiles for amino acid rank estimation. From left to right represents, (a) under-fitted model ($R = 2$), (b) correct estimation i.e. $R = 3$, (c) over-fitted model with $R = 4$.

Healthcare Data

Centers for Medicare and Medicaid (CMS) data files were created to allow researchers to gain familiarity using Medicare claims data while protecting beneficiary privacy. The CMS data contains multiple files per year. The file contains synthesized data taken from a 5% random sample of Medicare beneficiaries in 2008 and their claims from 2008 to 2010. We decompose PARAFAC2 tensor with rank between $R = 2$ to $R = 50$. Our aim is to estimate appropriate rank to find clinically-meaningful groups of features. For this data, BRTF based and Autochrome baselines unable to proceed due to out of memory after computing PARAFAC2 decompositions. Iteration based baseline and our method APTERA, estimated 11 and 9 components, respectively. NSVD does not provide any estimation of rank for this data. We observe one of the the component (or cluster) in which most of the patients has respiratory disease. These are the patients with high utilization ($> 50\%$), multiple clinical visits (avg 67) and high severity (death rate 8-10%). Most of the patients share

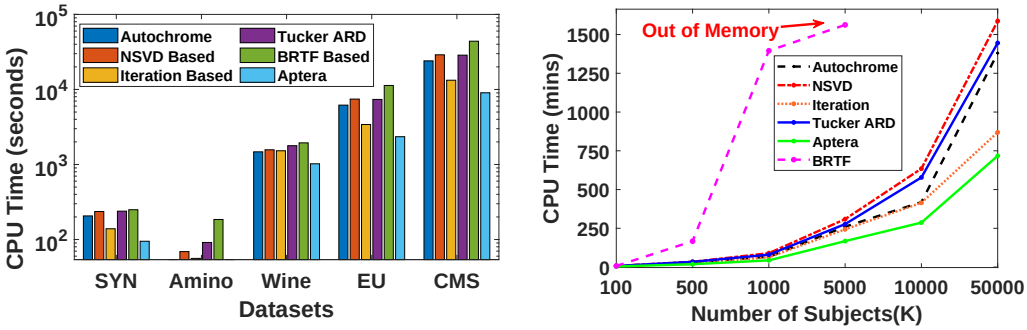


Figure 12.6: (a) Computation time of rank estimation for synthetic and real data. (b) Scalability Analysis on synthetic data.

ICD-9 code 492 (Emphysema), 496 (Chronic airway obstruction) and 511 (Pleurisy). These codes are characterized by obstruction of airflow that interferes with normal breathing.

Running Time Analysis

Figure (12.6a), shows the time taken by each method for synthetic and real dataset. We remark that our proposed method is faster that all baselines, even when compared to iteration based method where only PARAFAC2 decomposition is considered and no further computations are considered to find rank. This is because APTERA only requires to compute decomposition for R_{max} and other calculations once for each experiment.

12.4.5 Q3: Scalability Analysis

We also evaluate the scalability of our algorithm on synthetic dataset in terms of time needed for increasing number of subjects (K). A PARAFAC2 tensors $\underline{\mathbf{X}} \in \mathbb{R}^{100\times100\times[10050000]}$ are decomposed with fixed target rank $R = 10$. The time needed by APTERA increases very linearly with increase in nonzero elements. Our proposed method APTERA, successfully estimate the rank of the large PARAFAC2 tensors in reasonable time as shown in Figure

(12.6b) and is up to average 820% faster than baseline methods. We remark the favorable scalability properties of APTERA, rendering it is practical to use for large tensors.

12.5 Conclusions

In this paper, we work towards an automatic, PARAFAC2 tensor mining algorithm that minimizes human intervention. We encourage reproducibility by making our code publicly available. Our main contributions are:

- **Algorithm:** We proposed a new scalable method called APTERA for discovering low-rank structure in irregular data, which is based on the finding lcurve corner of higher order singular values.
- **Technology Transfer:** This work provides an efficient way to use Lcurve corner for rank detection in tensor mining, aiming to explore its capabilities and promote it within the community.
- **Evaluation:** We evaluate our method on synthetic data, showing their robustness compared to the baselines, as well as a wide variety of real datasets.

The content of this chapter was under blind peer review at the time of thesis submission.

Part III

Concluding Remarks

Chapter 13

Conclusions

In this thesis, we address the problem of mining large multi-aspect and dynamic graphs. Specifically, we develop models and algorithms for graph mining, and online tensor decomposition.

We develop a semi-supervised clustering algorithm SMACD that simultaneously consider multi-view clustering and semi-supervised learning to improve the clustering accuracy by utilizing partially available labels of nodes. Our algorithms exploit non-negativity and sparsity constraints to find patterns in multi-aspect graphs. We discovers meaningful rich structure of communities in multi-aspect graphs. Specifically, our algorithm RichCom exploits the Block Term Decomposition to extract higher than rank-1 but still interpretable structure from a multi-aspect dataset and uses AO-ADMM to speed up decomposition. Another line of work on community detection focuses on PARAFAC2 tensor mining where we proposed a novel method CAPTION uses the coupling between CP and PARAFAC2 tensor and POPLAR, where we introduce the concept of Laplacian regularization on the

PARAFAC2 decomposition, which improves community detection in time-evolving social networks, by leveraging graph-based auxiliary information. We also study the problem of niche detection and proposed method NED which discover co-clusters of user behaviors based on interaction densities, and explaining them using attributes of involved nodes.

We expand our scope to incremental multi-aspect data, in which we leverage network information over time. We develop fast, scalable and efficient methods i.e. SamBaTen, OCTen to tackle streaming CP decomposition, SPADE to tackle irregular tensors (PARAFAC2) and OnlineBTD to handle beyond rank-1 incremental tensor decomposition. For above methods we assumed that rank of the tensor is known to us. But in real world, this is far from truth. So to fill the gap, we study the automatic mining of PARAFAC2 decomposition and proposed efficient and scalable method namely Aptera.

Regardless of the diverse topics, this thesis has consistently used a combination of science and engineering grounded on theoretical foundations. Intending to impact the practice of computer science, this thesis requires design, experimentation, quantitative and qualitative evaluation, and analytic modeling to answer questions about community detection, incremental tensor analysis, deep learning, and methodologies related to their application. Although there is lot of work to be done, this thesis provides methods, tools and insights that can facilitate follow up research.

13.1 Future Directions

Tensor Mining: Hinton et. al [218] proposed a novel neural network architecture called Capsule Network (CapsNet). It outperforms state-of-the-art Convolutional Neural

Networks (CNN) on simple challenges like MNIST data. In general, capsules are a vector, and their elements consist of the size and direction of the object and its likelihood. Even though the focus of the thesis is on multi-aspect graphs, the methods developed herein can generalize to a broad spectrum of real-world applications within and beyond graph mining, where the key requirement is that multi-aspect contextual information is present, and we can model it as a tensor. Thus, future research goal is to gain insights on tensor compression for Capsule Networks, a relatively new deep learning paradigm, by examining its behavior in different practical settings.

Explainable AI: In the near term, we plan to explore the temporal evolution of niches in a number of ways, first, we like to understand when certain *niches* are *"born"*, whether they have certain periodic or other temporal patterns, and when do those *niches "retire"*?. Second, correlate those niches with external factors or events, which can be a second layer of explanation, in addition to the explanation provided by the attributes currently (For example, the birth of a certain Niche temporally correlates with a certain celebrity talking about "ABC" on the platform).

Another interesting topic here is to explore the "locale variation of niches": Are there certain niches that are particular to a location? Are there any attribute that could help with an explanation? Which niches persist across locations? And for those that do, is there any temporal variation in their birth and propagation (e.g., location 1 is inspired by location 2 and adopts a niche with some time delay). If niche is timely detected, we can expand its scope to different markets. For example, on the Cheerio Challenge, parents must balance Cheerios on their sleeping children's heads, it was started by "Life of Dad media"

company in Los Angles and went viral in USA. Later on, after few months, it went viral in India and took on a new dimension because of Father's Day.

AI for Healthcare: Artificial Intelligence (AI) is bringing a revolution to the healthcare industry. With the increasingly important role of AI in healthcare, there are growing worries over the lack of transparency and explainability in addition to potential bias encountered by model predictions. The medical diagnosis model is responsible for human life and we need to be confident enough to treat a patient as instructed by a black-box model. If the model cannot explain itself in the healthcare domain, then there is a huge risk of making a wrong decision may override its advantages of accuracy, speed, and decision-making efficacy. This would, in turn, severely limit its scope and utility. This is where Explainable Artificial Intelligence (XAI) comes into the picture. XAI increases the trust placed in an AI system by medical practitioners as well as AI researchers, and thus, eventually, leads to increasingly widespread deployment of AI in healthcare. This is perfectly aligned with the work on Niche Detection, where we try to find explainable co-clusters of viewers and publishers. Our work on PARAFAC2 tensor mining mostly focuses on health care data like CMS (Centers for Medicare and Medicaid Services). In this work, we would like to expand the research dimension via developing models/tools based on Graph Neural Networks for healthcare applications.

Another research area that we would like to explore is "Deep Learning for Networks" specifically for capsule networks. The broad goal of this research will be to address the challenge: to develop analysis techniques that enable us to understand when and why Capsule network methods will be successful and to design effective methods with explicit

performance guarantees. It is a fact that training deep learning models requires a computationally expensive empirical process. This slow process raises resource costs, wastes energy, and impedes user productivity. Under the same hood, work will focus on how to reduce resource costs and energy needs for training capsule networks on large data, and in turn, help democratize Capsule networks use to more application domains.

For reproducibility and the benefit of the community, we make most of the algorithms used throughout this thesis available at link[1].

[1] https://sites.google.com/view/gujralekta/research-work/software

Bibliography

[1] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23(13):i10–i18, 2007. 26

[2] Evrim Acar, Rasmus Bro, and Age K Smilde. Data fusion in metabolomics using coupled matrix and tensor factorizations. *Proceedings of the IEEE*, 103(9):1602–1620, 2015. 125

[3] Evrim Acar, Seyit A Camtepe, Mukkai S Krishnamoorthy, and Bülent Yener. Modeling and multiway analysis of chatroom tensors. In *International Conference on Intelligence and Security Informatics*, pages 256–268. Springer, 2005. 26

[4] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011. 48, 56, 62, 124, 125, 220

[5] Evrim Acar, Evangelos E Papalexakis, Morten A Rasmussen, Anders J Lawaetz, Mathias Nilsson, and Rasmus Bro. Structure-revealing data fusion. *BMC bioinformatics*, 15(1):239, 2014. 51, 124, 125

[6] Ardavan Afshar, Ioakeim Perros, Evangelos E Papalexakis, Elizabeth Searles, Joyce Ho, and Jimeng Sun. Copa: Constrained parafac2 for sparse & large datasets. *arXiv:1803.04572*, 2018. 2, 29, 31, 41, 83, 86, 111, 117, 303

[7] Ardavan Afshar, Ioakeim Perros, Haesun Park, Christopher deFilippi, Xiaowei Yan, Walter Stewart, Joyce Ho, and Jimeng Sun. Taste: temporal and static tensor factorization for phenotyping electronic health records. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, pages 193–203, 2020. 124, 125, 128, 135

[8] Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. A study of distinctiveness in web results of two search engines. In *Proceedings of the 24th International Conference on World Wide Web*, pages 267–273, 2015. 27

[9] Hassan Akbari, Mohammad B Shamsollahi, and Ronald Phlypo. Fetal ecg extraction using πtucker decomposition. In *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 174–178. IEEE, 2015. 248

[10] Saba A Al-Sayouri, Ekta Gujral, Danai Koutra, Evangelos E Papalexakis, and Sarah S Lam. t-pine: Tensor-based predictable and interpretable node embeddings. *arXiv preprint arXiv:1805.01889*, 2018. 4, 116

[11] Ayesha Haider Ali and Mohsin Nazir. Finding a pareto optimal solution for a multi-objective problem of managing radio resources: A qos aware algorithm. *Wireless Personal Communications*, 107(4):1661–1685, 2019. 309

[12] José Manuel Amigo, Marta J Popielarz, Raquel M Callejón, Maria L Morales, Ana M Troncoso, Mikael A Petersen, and Torben B Toldam-Andersen. Comprehensive analysis of chromatographic data by using parafac2 and principal components analysis. *Journal of Chromatography a*, 1217:4422–4429, 2010. 110, 280

[13] Charlotte Møller Andersen and Rasmus Bro. Practical aspects of parafac modeling of fluorescence excitation-emission data. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(4):200–215, 2003. 26

[14] Carl J Appellof and Ernest R Davidson. Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53(13):2053–2056, 1981. 26

[15] Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E Papalexakis, and Danai Koutra. Com2: fast automatic discovery of temporal ('comet') communities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014. 27

[16] Maria Athanasiou, Konstantina Sfrintzeri, Konstantia Zarkogianni, Anastasia C Thanopoulou, and Konstantina S Nikita. An explainable xgboost-based approach towards assessing the risk of cardiovascular disease in patients with type 2 diabetes mellitus. *arXiv preprint arXiv:2009.06629*, 2020. 154

[17] Dillen Augustijn, Alina Kulakova, Sujata Mahapatra, Pernille Harris, and Åsmund Rinnan. Isothermal chemical denaturation: Data analysis, error detection, and correction by parafac2. *Analytical chemistry*, 92(10):6958–6967, 2020. 31

[18] Woody Austin, Grey Ballard, and Tamara G Kolda. Parallel tensor compression for large-scale scientific data. In *PDPS, 2016 IEEE Int.*, pages 912–922. IEEE, 2016. 41, 216, 219, 279

[19] Brett W Bader, Michael W Berry, and Murray Browne. Discussion tracking in enron email using parafac. In *Survey of Text Mining II*, pages 147–163. Springer, 2008. 26

[20] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, 2015. 64

[21] Brett W Bader, Tamara G Kolda, et al. Matlab tensor toolbox version 2.6, available online, february 2015, 2015. 21, 27, 79, 94, 202, 205

[22] Zilong Bai, Hoa Nguyen, and Ian Davidson. Block model guided unsupervised feature selection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1201–1211, 2020. 168

[23] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8(Aug):1919–1986, 2007. 149

[24] Riccardo Bellazzi, Marianna Diomidous, Indra Neil Sarkar, Katsuhiko Takabayashi, Andreas Ziegler, and Alexa T McCray. Data analysis and data mining: current issues in biomedical informatics. *Methods of information in medicine*, 50(6):536, 2011. 123, 302

[25] Michele Berlingerio, Michele Coscia, and Fosca Giannotti. Finding redundant and complementary communities in multidimensional networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2181–2184. ACM, 2011. 49, 78, 82

[26] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, pages 1–122, 2011. 86

[27] John Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, 25(9):772–781, 1978. 225

[28] R. Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997. 110, 279

[29] Rasmus Bro. Multi-way analysis in the food industry-models, algorithms, and applications. In *MRI, EPG and EMA," Proc ICSLP 2000*. Citeseer, 1998. 306

[30] Rasmus Bro, Claus A Andersson, and Henk AL Kiers. Parafac2—part ii. modeling chromatographic data with retention time shifts. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 13(3-4):295–309, 1999. 31

[31] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of chemometrics*, 17(5):274–286, 2003. 2, 132, 201, 263, 304, 306

[32] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010. 168

[33] Bokai Cao, Lifang He, Xiangnan Kong, S Yu Philip, Zhifeng Hao, and Ann B Ragin. Tensor-based multi-view feature selection with applications to brain diseases. In *Data Mining (ICDM), 2014 IEEE Int. Conf. on*, pages 40–49. IEEE, 2014. 84

[34] Bokai Cao, Chun-Ta Lu, Xiaokai Wei, S Yu Philip, and Alex D Leow. Semi-supervised tensor factorization for brain network analysis. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer, 2016. 51, 52, 53

[35] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010. 204, 240

[36] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970. 21, 26, 27, 79, 247, 302

[37] Richard G Carter and John Levine. An investigation into tournament poker strategy using evolutionary algorithms. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 117–124. IEEE, 2007. 154

[38] J Longina Castellanos, Susana Gómez, and Valia Guerra. The triangle method for finding the corner of the l-curve. *Applied Numerical Mathematics*, 43(4):359–373, 2002. 305, 311

[39] Raymond B Cattell. "parallel proportional profiles" and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283, 1944. 31

[40] Christos Chatzichristos, Mike Davies, Javier Escudero, Eleftherios Kofidis, and Sergios Theodoridis. Fusion of eeg and fmri via soft coupled tensor decompositions. In *2018 26th EUSIPCO*, pages 56–60. IEEE, 2018. 124, 125, 135

[41] Christos Chatzichristos, Eleftherios Kofidis, Yiannis Kopsinis, Manuel Morante Moreno, and Sergios Theodoridis. Higher-order block term decomposition for spatially folded fmri data. In *Int. Conf. on Latent Variable Analysis and Signal Separation*. Springer, 2017. 37, 42, 83, 247, 249

[42] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. 154

[43] Wei Cheng, Xiang Zhang, Zhishan Guo, Yubao Wu, Patrick F Sullivan, and Wei Wang. Flexible and robust co-regularized multi-domain graph clustering. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 320–328. ACM, 2013. 47

[44] Wei Cheng, Xiang Zhang, Feng Pan, and Wei Wang. Hicc: an entropy splitting-based framework for hierarchical co-clustering. *Knowledge and Information Systems*, 46(2):343–367, 2016. 152

[45] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8-2000, pages 93–103, 2000. 152

[46] P.A. Chew, B.W. Bader, T.G. Kolda, and A. Abdelali. Cross-language information retrieval using parafac2. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 143–152. ACM, 2007. 110

[47] CMS. Data. https://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs, 2008. [Online]. 123, 134, 302, 316

[48] Andrej Čopar, Blaž Zupan, and Marinka Zitnik. Fast optimization of non-negative matrix tri-factorization. *PloS one*, 14(6):e0217994, 2019. 153, 167

[49] Pierre Courrieu. Fast computation of moore-penrose inverse matrices. *arXiv preprint arXiv:0804.4809*, 2008. 257

[50] Alessandro Cultrera and Luca Callegaro. A simple algorithm to find the l-curve corner in the regularization of inverse problems. *arXiv preprint arXiv:1608.04571*, 2016. 311

[51] Ian Davidson, Sean Gilpin, Owen Carmichael, and Peter Walker. Network discovery via constrained tensor analysis of fmri data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 194–202, 2013. 26

[52] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms–part i: Lemmas for... In *SIAM J. Matrix Anal. Appl.* Citeseer, 2008. 32, 33, 35, 80, 83

[53] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, pages 1033–1066, 2008. 32, 33, 35, 80, 83, 248

[54] Lieven De Lathauwer. Blind separation of exponential polynomials and the decomposition of a tensor in rank-(l_r,l_r,1) terms. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1451–1474, 2011. 37, 42, 249

[55] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000. 309

[56] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms-part iii: Alternating least squares algorithms. *SIAM journal on Matrix Analysis and Applications*, 30(3):1067–1083, 2008. 20, 21, 32

[57] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms—part iii: Alternating least squares algorithms. *SIAM journal on Matrix Analysis and Applications*, pages 1067–1083, 2008. 35, 36, 80, 83

[58] Pedro Marinho Ramos de Oliveira and Vicente Zarzoso. Block term decomposition of ecg recordings for atrial fibrillation analysis: Temporal and inter-patient variability. *Journal of Communication and Information Systems*, 34(1):111–119, 2019. 37, 42, 249

[59] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98, 2003. 149, 152, 153

[60] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135, 2006. 168

[61] Chris HQ Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2008. 168

[62] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering with multi-layer graphs: A spectral perspective. *IEEE Transactions on Signal Processing*, 60(11):5820–5831, 2012. 65

[63] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering on multi-layer graphs via subspace analysis on grassmann manifolds. *IEEE Transactions on signal processing*, 62(4):905–918, 2014. 47

[64] David R Duling. Use machine learning to discover your rules. *Paper SAS579-2017*, 2017. 154

[65] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):1–27, 2011. 27

[66] Georgios Theocharous ekta Gujral and Evangelos E. Papalexakis. Constrainted coupled cp and parafac2 tensor decomposition. In *Signals, Systems and Computers (ASILOMAR), 2020 Conference Record of the Forty Fifth Asilomar Conference on*. IEEE, 2020. 125

[67] Dóra Erdos and Pauli Miettinen. Walk'n'merge: A scalable algorithm for boolean tensor factorization. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1037–1042. IEEE, 2013. 193

[68] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. Zoobp: belief propagation for heterogeneous networks. *Proceedings of the VLDB Endowment*, 10(5), 2017. 51, 68

[69] Evangelos E. Papalexakis, Nicholas D Sidiropoulos, and Rasmus Bro. From k-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. *IEEE Transactions on Signal Processing*, 2013. 51

[70] Hadi Fanaee-T and João Gama. Multi-aspect-streaming tensor analysis. *Knowledge-Based Systems*, 89:332–345, 2015. 40, 188, 190, 219

[71] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters*, 138:185–192, 2020. 152

[72] Santo Fortunato. Community detection in graphs. *Physics reports*, 2010. 78

[73] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 149

[74] Tiantian Gao and Leman Akoglu. Fast information-theoretic agglomerative co-clustering. In *Australasian Database Conference*, pages 147–159. Springer, 2014. 152

[75] Wolfgang Gatterbauer, Stephan Günnemann, Danai Koutra, and Christos Faloutsos. Linearized and single-pass belief propagation. *Proceedings of the VLDB Endowment*, 8(5):581–592, 2015. 51

[76] Matthieu Genicot, P-A Absil, Renaud Lambiotte, and Saber Sami. Coupled tensor decomposition: a step towards robust components. In *2016 24th EUSIPCO*, pages 1308–1312. IEEE, 2016. 124, 125, 135

[77] Sean Gilpin, Chia-Tung Kuo, Tina Eliassi-Rad, and Ian Davidson. Some advances in role discovery in graphs. *arXiv preprint arXiv:1609.02646*, 2016. 51

[78] Vladimir Gligorijević, Yannis Panagakis, and Stefanos Zafeiriou. Fusion and community detection in multi-layer graphs. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1327–1332. IEEE, 2016. 50, 68, 74

[79] A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research*, 8:2265–2295, 2007. 204, 240

[80] Alexander Gorovits, Ekta Gujral, Evangelos E Papalexakis, and Petko Bogdanov. Larc: Learning activity-regularized overlapping communities across time. In *Proceedings of the 24th ACM SIGKDD Int. Conf. on KDD*, pages 1465–1474. ACM, 2018. 4, 78, 79, 83, 299

[81] Alexander Gorovits, Lin Zhang, Ekta Gujral, Evangelos E Papalexakis, and Petko Bogdanov. Mining bursty group behavior from online interactions. In *TBD*. ACM, 2021. 4

[82] Gérard Govaert and Mohamed Nadif. *Co-clustering: models, algorithms and applications*. John Wiley & Sons, 2013. 149

[83] Derek Greene and Pádraig Cunningham. Producing a unified graph representation from multiple social network views. In *Proceedings of the 5th annual ACM web science conference*, pages 118–121. ACM, 2013. 65

[84] Shirley Gregor and Izak Benbasat. Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS quarterly*, pages 497–530, 1999. 153

[85] Quanquan Gu and Jie Zhou. Co-clustering on manifolds. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 359–368, 2009. 149, 153

[86] PMARCA GUIDE. The only thing that matters. https://pmarchive.com/guide_to_startups_part4.html, 2007. [Online]. 147

[87] Ekta Guiral, Georgios Theocharous, and Evangelos E Papalexakis. C 3 aption: Constrainted coupled cp and parafac2 tensor decomposition. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 401–408. IEEE, 2020. 6

[88] Ekta Guiral, Georgios Theocharous, and Evangelos E Papalexakis. C 3 aption: Constrainted coupled cp and parafac2 tensor decomposition. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 401–408. IEEE, 2020. 145

[89] Ekta Gujral and Evangelos E Papalexakis. Smacd: Semi-supervised multi-aspect community detection. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 702–710. SIAM, 2018. 3, 6, 75, 78, 79, 82, 84, 123

[90] Ekta Gujral and Evangelos E Papalexakis. Smacd: Semi-supervised multi-aspect community detection. *WSDM HetroNam 2018*, 2018. 3, 6, 75

[91] Ekta Gujral and Evangelos E Papalexakis. Onlinebtd: Streaming algorithms to track the block term decomposition of large tensors. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 168–177. IEEE, 2020. 4, 6, 277

[92] Ekta Gujral, Evangelos E Papalexakis, Georgios Theocharous, and Anup Rao. Hacd: Hierarchical agglomerative community detection in social networks. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019. 4, 152

[93] Ekta Gujral, Ravdeep Pasricha, and Evangelos Papalexakis. Beyond rank-1: Discovering rich community structure in multi-aspect graphs. In *Proceedings of The Web Conference 2020*, pages 452–462, 2020. 3, 6, 10, 42, 108, 123, 247, 249, 250, 302

[94] Ekta Gujral, Ravdeep Pasricha, and Evangelos E Papalexakis. Sambaten: Sampling-based batch incremental tensor decomposition. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 387–395. SIAM, 2018. 4, 6, 212, 219, 227, 231, 247, 253, 279

[95] Ekta Gujral, Ravdeep Pasricha, Tianxiong Yang, and Evangelos E Papalexakis. Octen: Online compression-based tensor decomposition. *arXiv preprint arXiv:1807.01350*, 2018. 4, 6, 247, 253

[96] Ekta Gujral, Ravdeep Pasricha, Tianxiong Yang, and Evangelos E Papalexakis. Octen: Online compression-based tensor decomposition. In *2019 IEEE 8th International Workshop on CAMSAP*. IEEE, 2019. 244, 272

[97] Ekta Gujral, Georgios Theocharous, and Evangelos E Papalexakis. Poplar: Parafac2 decomposition using auxiliary information. In *2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 1–5. IEEE, 2020. 4, 6, 121

[98] Ekta Gujral, Georgios Theocharous, and Evangelos E Papalexakis. Spade: S treaming pa rafac2 de composition for large datasets. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 577–585. SIAM, 2020. 4, 6, 300

[99] Maimoona Hafeez, Asthma Yasin, Nazia Badar, Muhammad Irfan Pasha, Nishat Akram, and Bushra Gulzar. Prevalence and indications of caesarean section in a teaching hospital. *JIMSA*, 27(1):15–6, 2014. 248

[100] Junwei Han, Kun Song, Feiping Nie, and Xuelong Li. Bilateral k-means algorithm for fast co-clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31-1, 2017. 159

[101] Junwei Han, Kai Xiong, and Feiping Nie. Orthogonal and nonnegative graph reconstruction for large scale clustering. In *IJCAI*, pages 1809–1815, 2017. 149

[102] PC Hansen. The l-curve and its use in the numerical treatment of inverse problems. computational inverse problems in electrocardiology. *Advances in Computational Bioengineering*, 5:119, 2001. 310

[103] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems*, 5:19, 2016. 133, 292

[104] R.A. Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. *University of California at Los Angeles*, 1970. 21, 27, 79, 215, 247

[105] Richard A Harshman. Determination and proof of minimum uniqueness conditions for parafac1. *UCLA Working Papers in phonetics*, 22(111-117):3, 1972. 225

[106] Richard A Harshman. Parafac2: Mathematical and technical notes. *UCLA working papers in phonetics*, 22(3044):122215, 1972. 12, 27, 41, 110, 279, 303

[107] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. *Analytical Chemistry*, 1970. 26

[108] Richard A Harshman and Margaret E Lundy. Uniqueness proof for a family of models sharing features of tucker's three-mode factor analysis and parafac/candecomp. *Psychometrika*, 61(1):133–154, 1996. 28, 29

[109] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972. 149

[110] Johan Håstad. Tensor rank is np-complete. In *International Colloquium on Automata, Languages, and Programming*, pages 451–460. Springer, 1989. 2, 303

[111] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990. 97

[112] Lei He, Haiyan Wang, and Muhang Zhang. Identification of underwater propeller noise by low-rank approximation of cyclic spectrum. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–6. IEEE, 2018. 309

[113] Nathaniel E Helwig. Estimating latent trends in multivariate longitudinal data via parafac2 with functional and structural constraints. *Biometrical Journal*, 59(4):783–803, 2017. 110

[114] Nicholas J Higham. *Accuracy and stability of numerical algorithms*, volume 80. Siam, 2002. 257

[115] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987. 257

[116] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013. 2, 303, 309

[117] Joyce C Ho, Joydeep Ghosh, Steve R Steinhubl, Walter F Stewart, Joshua C Denny, Bradley A Malin, and Jimeng Sun. Limestone: High-throughput candidate phenotype generation via tensor factorization. *Journal of biomedical informatics*, 52:199–211, 2014. 279

[118] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 115–124, 2014. 27, 110, 279

[119] Jamin C Hoggard and Robert E Synovec. Parallel factor analysis (parafac) of target analytes in gc× gc- tofms data: automated selection of a model with an appropriate number of factors. *Analytical chemistry*, 79, 2007. 316

[120] Kejun Huang, Nicholas D Sidiropoulos, and Athanasios P Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, pages 5052–5065, 2016. 83, 86, 88, 104

[121] Kejun Huang, Nikos D Sidiropoulos, and A Swamiy. Nmf revisited: New uniqueness results and algorithms. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 4524–4528. IEEE, 2013. 55

[122] Borbála Hunyadi, Daan Camps, Laurent Sorber, Wim Van Paesschen, Maarten De Vos, Sabine Van Huffel, and Lieven De Lathauwer. Block term decomposition for modelling epileptic seizures. *EURASIP Journal on Advances in Signal Processing*, 2014(1):1–19, 2014. 10, 36, 42, 249, 250

[123] MatLab Inc. Matlab https://www.mathworks.com. Version MATLAB R2016b. 89

[124] ByungSoo Jeon, Inah Jeon, Lee Sael, and U Kang. Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 811–822. IEEE, 2016. 125

[125] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 570–586. Springer, 2010. 51

[126] Guanbo Jia, Zixing Cai, Mirco Musolesi, Yong Wang, Dan A Tennant, Ralf JM Weber, John K Heath, and Shan He. Community detection in social and biological networks using differential evolution. In *Learning and Intelligent Optimization*. Springer, 2012. 78

[127] He Jiang, Yangqiu Song, Chenguang Wang, Ming Zhang, and Yizhou Sun. Semi-supervised learning over heterogeneous information networks by ensemble of meta-graph guided random walks. In *IJCAI*, pages 1944–1950, 2017. 50

[128] Tao Jiang and Nikos D Sidiropoulos. Kruskal's permutation lemma and the identification of candecomp/parafac and bilinear models with constant modulus constraints. *IEEE Transactions on Signal Processing*, 52(9):2625–2636, 2004. 225

[129] Lea G Johnsen, José Manuel Amigo, Thomas Skov, and Rasmus Bro. Automated resolution of overlapping peaks in chromatographic data. *J. Chemometrics*, 28(2):71–82, 2014. 2, 304, 306, 308, 316

[130] Parisa Kaghazgaran, Maarten Bos, Leonardo Neves, and Neil Shah. Social factors in closed-network content consumption. In *CIKM*, 2020. 167

[131] Masayuki Karasuyama and Hiroshi Mamitsuka. Multiple graph label propagation by sparse integration. *IEEE transactions on neural networks and learning systems*, 24(12):1999–2012, 2013. 48, 68

[132] Ujwal Kayande, Arnaud De Bruyn, Gary L Lilien, Arvind Rangaswamy, and Gerrit H Van Bruggen. How incorporating feedback mechanisms in a dss affects dss evaluations. *Information Systems Research*, 20(4):527–546, 2009. 153

[133] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*, pages 3146–3154, 2017. 149, 154, 168

[134] Henk AL Kiers. A three–step algorithm for candecomp/parafac analysis of large data sets with multicollinearity. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 12(3):155–171, 1998. xxi, 303, 304, 314, 315

[135] Henk AL Kiers, Jos MF Ten Berge, and Rasmus Bro. Parafac2—part i. a direct fitting algorithm for the parafac2 model. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 13(3-4):275–294, 1999. 2, 117, 280, 293, 303

[136] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications*, 30(2):713–730, 2008. 128

[137] Jingu Kim and Haesun Park. Fast nonnegative tensor factorization with an active-set-like method. In *High-Performance Scientific Computing*, pages 311–326. Springer, 2012. 128

[138] Jungeun Kim and Jae-Gil Lee. Community detection in multi-layer graphs: A survey. *ACM SIGMOD Record*, pages 37–48, 2015. 79, 95, 102, 104

[139] Tamara G Kolda, Brett W Bader, and Joseph P Kenny. Higher-order web link analysis using multilinear algebra. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005. 186

[140] Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 363–372. IEEE, 2008. 217

[141] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3), 2009. 15, 16, 21, 29, 33, 51, 56, 74, 78, 110, 253, 279, 285

[142] Danai Koutra, Abhilash Dighe, Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis, Christos Faloutsos, and Jean Bolot. Pnp: Fast path ensemble method for movie design. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1527–1536, 2017. 162

[143] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8(3):183–202, 2015. 79, 83, 96, 98, 101

[144] Danai Koutra, Tai-You Ke, U Kang, Duen Horng Polo Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 245–260. Springer, 2011. 48, 51, 68

[145] Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977. 25

[146] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In *NeurIPS*, pages 1413–1421, 2011. 166

[147] Rashmi Kumari, MK Singh, R Jha, NK Singh, et al. Anomaly detection in network traffic using k-mean clustering. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 387–393. IEEE, 2016. 152

[148] Sangeetha Kutty, Lin Chen, and Richi Nayak. A people-to-people recommendation system using tensor space models. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 187–192, 2012. 27

[149] Hemank Lamba and Neil Shah. Modeling dwell time engagement on visual multimedia. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1104–1113, 2019. 167

[150] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007. 78, 246

[151] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650. ACM, 2010. 95, 100

[152] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. 78

[153] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, pages 29–123, 2009. 78

[154] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010. 46, 78

[155] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012. xvii, 103

[156] Joseph Levin. Three-mode factor analysis. *Psychological Bulletin*, 64(6):442, 1965. 37

[157] Na Li. *Variants of ALS on Tensor Decompositions and Applications*. PhD thesis, Clarkson University, 2013. 20, 21

[158] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003. 168

[159] Shuangzhe Liu and Gõtz Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *International Journal of Information and Systems Sciences*, 4(1):160–177, 2008. 130

[160] Xiangqian Liu and Nikos D Sidiropoulos. Cramér-rao lower bounds for low-rank decomposition of multidimensional arrays. *IEEE Transactions on Signal Processing*, 49(9):2074–2086, 2001. 26

[161] Bo Long, Zhongfei Zhang, and Philip S Yu. Co-clustering by block value decomposition. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 635–640, 2005. 158, 168

[162] Zhoumin Lu, Genggeng Liu, and Shiping Wang. Sparse neighbor constrained co-clustering via category consistency learning. *Knowledge-Based Systems*, 201:105987, 2020. 167, 168

[163] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. Explainable ai for trees: From local explanations to global understanding. *arXiv preprint arXiv:1905.04610*, 2019. 154

[164] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):2522–5839, 2020. 154

[165] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017. 154

[166] Anant Madabhushi and George Lee. Image analysis and machine learning in digital pathology: Challenges and opportunities, 2016. 123, 302

[167] Hing-Hao Mao, Chung-Jung Wu, Evangelos E Papalexakis, Christos Faloutsos, Kuo-Chen Lee, and Tien-Cheu Kao. Malspot: Multi 2 malicious network behavior patterns analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1–14. Springer, 2014. 26

[168] Georgios B Mardani, Gonzalo. Subspace learning and imputation for streaming big data matrices and tensors. *IEEE Signal Processing*, 2015. 41, 191, 220

[169] David Martens, Bart Baesens, Tony Van Gestel, and Jan Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European journal of operational research*, 183(3):1466–1476, 2007. 154

[170] David Martens and Foster Provost. Explaining data-driven document classifications. *Mis Quarterly*, 38(1):73–100, 2014. 154

[171] MATLAB. Kron product. https://www.mathworks.com/help/matlab/ref/kron.html?s_tid=mwa_osa_a. 255

[172] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013. 204

[173] Danielle P Mersch, Alessandro Crespi, and Laurent Keller. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science*, 340(6136):1090–1093, 2013. 273

[174] Boris Mirkin. *Mathematical classification and clustering*, volume 11. Springer Science & Business Media, 2013. 152

[175] J Mocks. Topographic components model for event-related potentials and some biophysical considerations. *IEEE transactions on biomedical engineering*, 35(6):482–484, 1988. 26

[176] Morten Mørup and Lars Kai Hansen. Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363, 2009. 97, 233, 306, 316

[177] Morten Mørup, Lars Kai Hansen, Sidse Marie Arnfred, Lek-Heng Lim, and Kristoffer Hougaard Madsen. Shift-invariant multilinear decomposition of neuroimaging data. *NeuroImage*, 42(4):1439–1450, 2008. 26

[178] Morten Mørup, Lars Kai Hansen, Christoph S Herrmann, Josef Parnas, and Sidse M Arnfred. Parallel factor analysis as an exploratory tool for wavelet transformed event-related eeg. *NeuroImage*, 29(3):938–947, 2006. 26

[179] Morten Mørup, Lars Kai Hansen, and Kristoffer Hougaard Madsen. Modeling latency and shape changes in trial based neuroimaging data. In *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 439–443. IEEE, 2011. 26

[180] Iman Mousavian, Mohammad Bagher Shamsollahi, and Emad Fatemizadeh. Noninvasive fetal ecg extraction using doubly constrained block-term decomposition. *Mathematical biosciences and engineering: MBE*, 17(1):144–159, 2019. 37, 42, 247, 249

[181] Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. Tensor factorization using auxiliary information. In *European Conf. on MLKDD*. Springer, 2011. 220

[182] Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery*, 25(2):298–324, 2012. 111, 113

[183] David Newman, Sarvnaz Karimi, and Lawrence Cavedon. External evaluation of topic models. In *in Australasian Doc. Comp. Symp., 2009*. Citeseer, 2009. 159

[184] Feiping Nie, Jing Li, Xuelong Li, et al. Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification. In *IJCAI*, pages 1881–1887, 2016. 48, 51, 68

[185] Feiping Nie, Xiaoqian Wang, Cheng Deng, and Heng Huang. Learning a structured optimal bipartite graph for co-clustering. In *NeurIPS*, pages 4129–4138, 2017. 149, 153

[186] D. Nion and N.D. Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *Signal Processing, IEEE Transactions on*, 57(6):2299–2310, 2009. 40, 188, 190, 193, 205, 216, 219, 227, 231, 253, 279

[187] Dimitri Nion, Kleanthis N Mokios, Nicholas D Sidiropoulos, and Alexandros Potamianos. Batch and adaptive parafac-based blind separation of convolutive speech mixtures. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1193–1207, 2009. 27

[188] Sarah Nogueira and Gavin Brown. Measuring the stability of feature selection. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 442–457. Springer, 2016. 170

[189] Olivera Novović, Sanja Brdar, and Vladimir Crnojević. Evolving connectivity graphs in mobile phone data. In *NetMob, The main conference on the scientific analysis of mobile phone datasets*, pages 73–75, 2017. 123, 302

[190] University of UMICH. http://topology.eecs.umich.edu/data.html. 95, 100

[191] omputer science bibliography. Kdd data. 133

[192] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st international conference on Very large data bases*, pages 697–708. VLDB Endowment, 2005. 40, 188, 190, 219

[193] Evangelos E Papalexakis. Automatic unsupervised tensor mining with quality assessment. In *Proceedings of the 2016 SIAM SDM*, pages 711–719. SIAM, 2016. 2, 97, 132, 229, 233, 242, 247, 263, 304, 306, 308

[194] Evangelos E Papalexakis, Leman Akoglu, and Dino Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *Proceedings of the 16th International Conference on Information Fusion*, pages 899–905. IEEE, 2013. 26, 47, 50, 53, 64, 68, 78, 79, 82, 84

[195] Evangelos E Papalexakis and Christos Faloutsos. Fast efficient and scalable core consistency diagnostic for the parafac decomposition for big sparse tensors. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5441–5445. IEEE, 2015. 201

[196] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012. 27, 193, 195, 196, 200

[197] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):16, 2016. 15, 18, 19, 20, 21, 51, 55, 110, 186, 187, 198, 215, 279

[198] Evangelos E Papalexakis, Tom M Mitchell, Nicholas D Sidiropoulos, Christos Faloutsos, Partha Pratim Talukdar, and Brian Murphy. Scoup-smt: Scalable coupled sparse matrix-tensor factorization. *arXiv preprint arXiv:1302.7043*, 2013. 125

[199] Papalexakis, Evangelos E. Automatic unsupervised tensor mining with quality assessment. In *SIAM SDM*, 2016. 52, 62

[200] SungJin Park, Suan Lee, and Jinho Kim. Estimating revenues of seoul commercial alley services using tensor decomposition & generating recommendation system. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 287–294. IEEE, 2020. 27

[201] Ravdeep Pasricha, Ekta Gujral, and Evangelos 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. 284

[202] Ioakeim Perros, Evangelos E Papalexakis, Fei Wang, Richard Vuduc, Elizabeth Searles, Michael Thompson, and Jimeng Sun. Spartan: Scalable parafac2 for large & sparse data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 375–384, 2017. 2, 29, 31, 41, 110, 280, 285, 286, 288, 289, 294

[203] Anh Huy Phan and Andrzej Cichocki. Parafac algorithms for large-scale problems. *Neurocomputing*, 74(11):1970–1984, 2011. 39, 190, 219

[204] Yuan Qi, Thomas P Minka, Rosalind W Picard, and Zoubin Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings of the twenty-first international conference on Machine learning*, page 85, 2004. 306

[205] Medhat A Rakha. On the moore–penrose generalized inverse matrix. *Applied Mathematics and Computation*, 158(1):185–200, 2004. 257

[206] Shebuti Rayana and Leman Akoglu. Less is more: Building selective anomaly ensembles. *ACM Transactions on Knowledge Discovery from Data*, 2016. 96, 102

[207] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013. 88, 104

[208] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010. 27

[209] Lucas N Ribeiro, Antonio R Hidalgo-Muñoz, Gérard Favier, João Cesar M Mota, André LF De Almeida, and Vicente Zarzoso. A tensor decomposition approach to noninvasive atrial activity extraction in atrial fibrillation ecg. In *2015 23rd EUSIPCO*, pages 2576–2580. IEEE, 2015. 37, 42, 247, 249

[210] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 154

[211] Tommaso Rigon, Amy H Herring, and David B Dunson. A generalized bayes framework for probabilistic clustering. *arXiv preprint arXiv:2006.05451*, 2020. 152

[212] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, pages 416–431, 1983. 89, 171

[213] Georgios Rizos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Multilabel user classification using the community structure of online networks. *PloS one*, 12(3):e0173347, 2017. 66

[214] François Role, Stanislas Morbieu, and Mohamed Nadif. Coclust: a python package for co-clustering. *Journal of Statistical Software*, 2018. 153, 167

[215] Steven J Rosansky, Donald R Hoover, Lisa King, and James Gibson. The association of blood pressure levels and change in renal function in hypertensive and nonhypertensive subjects. *Archives of internal medicine*, 150(10):2073–2076, 1990. 154

[216] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. 263, 272

[217] Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. Discovering dynamic communities in interaction networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 678–693. Springer, 2014. xvii, 92, 95, 99, 104

[218] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017. 325

[219] Tara Safavi, Chandra Sripada, and Danai Koutra. Scalable hashing-based network discovery. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 405–414. IEEE, 2017. 116

[220] Tara Safavi, Chandra Sripada, and Danai Koutra. Fast network discovery on sequence data via time-aware hashing. *Knowledge and Information Systems*, 61(2):987–1017, 2019. 134

[221] Aghiles Salah, Melissa Ailem, and Mohamed Nadif. Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 149, 153, 158, 167, 168

[222] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004. 246, 263, 275

[223] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008. 65

[224] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD Int. Conf. on KDD*, pages 1055–1064. ACM, 2015. 79, 83, 96, 98, 101, 102

[225] Fanhua Shang, LC Jiao, and Fei Wang. Graph dual regularization non-negative matrix factorization for co-clustering. *Pattern Recognition*, 45(6):2237–2250, 2012. 168

[226] Junming Shao, Chongming Gao, Wei Zeng, Jingkuan Song, and Qinli Yang. Synchronization-inspired co-clustering and its application to gene expression data. In *2017 IEEE Int. Conf. on Data Mining (ICDM)*, pages 1075–1080. IEEE, 2017. 153

[227] Fatemeh Sheikholeslami, Brian Baingana, Georgios B. Giannakis, and Nikolaos D. Sidiropoulos. Egonet tensor decomposition for community identification. In *GlobalSIP*, 2016. 50, 240

[228] Qiquan Shi, Haiping Lu, and Yiu-ming Cheung. Tensor rank estimation and completion via cp-based nuclear norm. In *Proc. of ACM on Conf. on IKM*, pages 949–958, 2017. 2, 304

[229] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017. 154

[230] N Sidiropoulos, Evangelos E. Papalexakis, and C Faloutsos. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *IEEE Signal Processing Magazine*, 2014. 40, 217, 219, 221, 225, 227, 231

[231] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017. 27

[232] Nicholas D Sidiropoulos, Georgios B Giannakis, and Rasmus Bro. Blind parafac receivers for ds-cdma systems. *IEEE Transactions on Signal Processing*, 48(3):810–823, 2000. 27

[233] Thomas Skov, Davide Ballabio, and Rasmus Bro. Multiblock variance partitioning: A new approach for comparing variation in multiple data blocks. *Analytica chimica acta*, 615:18–29, 2008. 110, 280, 314, 315

[234] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005. 26

[235] Shaden Smith, Alec Beri, and George Karypis. Constrained tensor factorization with accelerated ao-admm. In *2017 46th International Conference on Parallel Processing (ICPP)*, pages 111–120. IEEE, 2017. 83, 86, 87, 104

[236] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. FROSTT: The formidable repository of open sparse tensors and tools, 2017. 203, 204

[237] Shaden Smith, Kejun Huang, Nicholas D Sidiropoulos, and George Karypis. Streaming tensor factorization for infinite data sources. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 81–89. SIAM, 2018. 247

[238] Qingquan Song, Hancheng Ge Xiao Huang, James Caverlee, and Xia Hu. Multi-aspect streaming tensor completion. In *KDD*. ACM, 2017. 41, 191, 220

[239] Alwin Stegeman and Tam TT Lam. Multi-set factor analysis by means of p arafac2. *British Journal of Mathematical and Statistical Psychology*, 69(1):1–19, 2016. 29, 280

[240] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010. 154

[241] Qi Su, Kun Xiang, Houfeng Wang, Bin Sun, and Shiwen Yu. Using pointwise mutual information to identify implicit features in customer reviews. In *International Conference on Computer Processing of Oriental Languages*, pages 22–30. Springer, 2006. 160

[242] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383. ACM, 2006. 272

[243] Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Trans. Knowl. Discov. Data*, 2(3):11:1–11:37, October 2008. 40, 188, 190, 216, 219, 279

[244] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, 2011. 161, 162

[245] Wei Tang, Zhengdong Lu, and Inderjit S Dhillon. Clustering with multiple graphs. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 1016–1021. IEEE, 2009. 47

[246] Chayant Tantipathananandh and Tanya Y Berger-Wolf. Finding communities in dynamic social networks. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1236–1241. IEEE, 2011. 50

[247] Jos MF ten Berge and Henk AL Kiers. Some uniqueness results for parafac2. *Psychometrika*, 61(1):123–132, 1996. 28, 29, 280

[248] Jos MF ten Berge and Nikolaos D Sidiropoulos. On uniqueness in candecomp/parafac. *Psychometrika*, 67(3):399–409, 2002. 26

[249] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. 57

[250] Hannu Toivonen, Fang Zhou, Aleksi Hartikainen, and Atte Hinkka. Compression of weighted graphs. In *Proceedings of the 17th ACM SIGKDD Int. Conf. on KDD*, pages 965–973. ACM, 2011. 83

[251] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*, pages 613–622. IEEE, 2006. 51

[252] Yorgos Tsitsikas and Evangelos E Papalexakis. Nsvd: N ormalized s ingular v alue d eviation reveals number of latent factors in tensor decomposition. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 667–675. SIAM, 2020. 2, 304, 306, 307, 316

[253] Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15:122–137, 1963. 37

[254] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. 37

[255] Ledyard R Tucker et al. The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology*, 110119, 1964. 37

[256] L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. 37, 79, 215, 247, 302

[257] M. Alex O. Vasilescu and Eric Kim. Compositional hierarchical tensor factorization: Representing hierarchical intrinsic and extrinsic causal factors. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19): Tensor Methods for Emerging Data Science Challenges*, Aug. 2019. 247

[258] N Vervliet, O Debals, L Sorber, M Van Barel, and L De Lathauwer. Tensorlab 3.0, available online, mar. 2016. *URL: http://www. tensorlab. net*, 2016. 33, 94, 263, 264

[259] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 37–42. ACM, 2009. 204, 240

[260] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 78

[261] Mengting Wan, Yunbo Ouyang, Lance Kaplan, and Jiawei Han. Graph regularized meta-path based transductive regression in heterogeneous information network. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 918–926. SIAM, 2015. 50

[262] Shiping Wang and Wenzhong Guo. Robust co-clustering via dual local learning and high-order matrix factorization. *Knowledge-Based Systems*, 138:176–187, 2017. 168

[263] Xiaofei Wang and Carmeliza Navasca. Low-rank approximation of tensors via sparse optimization. *Numerical Linear Algebra with Applications*, 25(2), 2018. 229

[264] Xiaogang Wang, Xiaoxu Ma, and W Eric L Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *IEEE Transactions on pattern analysis and machine intelligence*, pages 539–555, 2009. 78

[265] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel Kho, You Chen, Bradley A Malin, and Jimeng Sun. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1265–1274. ACM, 2015. 51

[266] Joyce Jiyoung Whang and Inderjit S. Dhillon. Non-exhaustive, overlapping co-clustering. In *Proceedings of the 26th ACM Conference on Information and Knowledge Management (CIKM)*, pages 2367–2370, 2017. 152, 167

[267] Barry M Wise, Neal B Gallagher, and Elaine B Martin. Application of parafac2 to fault detection and diagnosis in semiconductor etch. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 15(4):285–298, 2001. 31

[268] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016. 149, 153

[269] Dongkuan Xu, Wei Cheng, Bo Zong, Jingchao Ni, Dongjin Song, Wenchao Yu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. Deep co-clustering. In *Proceedings of the 2019 SIAM SDM*, pages 414–422. SIAM, 2019. 149, 153, 167

[270] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, 2003. 168

[271] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *ICDM 2003. Proceedings. 2002 IEEE Int. Conf. on*, pages 721–724. IEEE, 2002. 83

[272] Dingqi Yang, Daqing Zhang, Vincent. W. Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015. 240

[273] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *Data Mining (ICDM), 2013 IEEE 13th international conference on*, pages 1151–1156. IEEE, 2013. 62

[274] Kai Yang, Xiang Li, Haifeng Liu, Jing Mei, Guotong Xie, Junfeng Zhao, Bing Xie, and Fei Wang. Tagited: Predictive task guided tensor decomposition for representation learning from electronic health records. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31-1, 2017. 27

[275] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005. 48, 50

[276] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD Int. Conf. on KDD*, pages 555–564. ACM, 2017. 95, 98, 104, 246, 263, 274, 314, 315, 319

[277] Kejing Yin, William K Cheung, Benjamin CM Fung, and Jonathan Poon. Tedpar: Temporally dependent parafac2 factorization for phenotype-based disease progression modeling. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 594–602. SIAM, 2021. 31

[278] Kejing Yin, William K Cheung, Yang Liu, Benjamin CM Fung, and Jonathan Poon. Joint learning of phenotypes and diagnosis-medication correspondence via hidden interaction tensor factorization. In *IJCAI*, pages 3627–3633, 2018. 27

[279] Jiho Yoo and Seungjin Choi. Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on stiefel manifolds. *Information processing & management*, 46(5):559–570, 2010. 168

[280] Zhijian Yuan, Zhirong Yang, and Erkki Oja. Projective nonnegative matrix factorization: Sparseness, orthogonality, and clustering. *Neural Process. Lett*, pages 11–13, 2009. 168

[281] Peng Zhang and Kun She. A novel hierarchical clustering approach based on universal gravitation. *Mathematical Problems in Engineering*, 2020, 2020. 152

[282] Qibin Zhao, Guoxu Zhou, Liqing Zhang, Andrzej Cichocki, and Shun-Ichi Amari. Bayesian robust tensor factorization for incomplete multiway data. *IEEE transactions on NN and learning systems*, pages 736–748, 2015. 2, 304, 306, 316

[283] Shuo Zhou, Sarah Erfani, and James Bailey. Online cp decomposition for sparse tensors. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1458–1463. IEEE, 2018. 286

[284] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating online cp decompositions for higher order tensors. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1375–1384, 2016. 40, 188, 190, 193, 205, 216, 219, 227, 231, 247, 253, 261, 279

[285] Xiaofeng Zhu, Yonghua Zhu, and Wei Zheng. Spectral rotation for deep one-step clustering. *Pattern Recognition*, 105:107175, 2020. 152

[286] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. *University of Wisconsin-Madison*, 2005. 51