MACHINE LEARNING APPROACHES TO REVEAL DISCRETE SIGNALS IN GENE EXPRESSION

by

Changlin Wan

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Chi Zhang, Co-Chair

School of Medicine

Indiana University

Dr. Mireille Boutin, Co-Chair

Electrical and Computer Engineering

Purdue University

Dr. Edward J. Delp

Electrical and Computer Engineering

Purdue University

Dr. Zina Ben-Miled

Electrical and Computer Engineering

Indiana University Purdue University Indianapolis

Approved by:

Dr. Michael A. Capano

ACKNOWLEDGMENTS

I would start by thanking my advisor Prof. Chi Zhang for his care and support along the way. I still remember the first time that I met Prof. Zhang and his wife Prof. Sha Cao. We were at the same table of ICIBM2016 dinner reception. That is also the very first time I had stirred lobster, BTW it was so good. Knowing I was seeking graduate study, Prof. Zhang not only showed his intention to recruit me to his own lab, but he also introduced me to nearly all his professor friends at that reception for opportunities in case he is not able to offer. I was very moved at that moment. More importantly, I know Prof. Zhang is a kind, candid, supportive and selfless person, who is exactly the type of person you need as an advisor. And he has been the same kind of person throughout my Ph.D training. I can not express how grateful I am for his mentorship. He introduced to me the mathematical perspective to every biological problems. He taught me how to properly model and generate hypothesis for a specific task. He also trained me how to present and develop a scientific story. He works efficiently and tirelessly. He always joked about he is a senior post-doc. You can really feel and be inspired by his passion while working with him. Till now, I still feel tremendously lucky that I met him at that dinner reception and had this fruitful five years under his wing.

I would also like to give a shout out to Prof. Cao. Throughout the years, I have developed a hobby. That I knew my manuscript is ready once she gave me the green light. She gave me a lot guidance on statistical learning. But more importantly, her positivity and passion boost my confidence to deal with any scientific problems. She and Prof. Zhang is a perfect team together. Noted, this team not only thrives on scientific discovery. I am very happy that their family is about to double the size. Their son is almost three years old, and they are expecting a baby girl this May.

I would like to thank the rest of member in the lab for their support. Especially, Wennan. We are in the same Ph.D program. He is the big brother to me. I could not imagine how difficult this Ph.D training could be without his support.

I would not ask for a better Ph.D committee. Besides, Prof. Zhang, I would like to thank Prof. Mireille Boutin, for her care, guidance and chairing my Ph.D committee. And to Prof. Edward Delp and Prof. Zina Ben-Miled for their insightful feedbacks and suggestions.

I would also like to thank all my colleges at Amazon Product Graph, Memorial Sloan Kettering Cancer Center (MSKCC) and Genentech. Especially, Dr. Tong Zhao, my mentor at Amazon, Prof. Christina Leslie, my host at MSKCC, and Dr.Aviv Regev and Dr. Kai Liu, my managers at Genentech for their guidance and support.

Furthermore, I would like to thank Sherrie Tucker, Prof. Brian King at IUPUI ECE program and Elisheba Van Winkle, Matt Golden at Purdue West Lafayette ECE program. They are always kind to me and they are the first person I went to if I met administrative issues.

Lastly, I would like to thank my family. Without their unconditional love and support, I could never go this far and pursue my dreams.

PREFACE

This journey started at one afternoon in the first year of my high school, that I happened to be in the seminar, where the teacher illustrate how HIV virus escapes our immune surveillance by integrating its genetic information with our genome. It was like a thunder strike. I was amazed by how living beings adopted surprising strategies for life and make the planet earth a complected but wonderful ecological system.

After high school, I entered University of Science and Technology of China (USTC) to pursue my bachelor degree in Biology. As the name revealed, USTC is very focus on the scientific training. In the first two years, all the undergraduate students are required to take classes from every department. Thus, even I am majored in Biology, I was required to study programming language, quantum physics, organic chemistry, calculus, etc, which build the foundation for me to conduct interdisciplinary researches. Moreover, USTC encourages and provides external research internships. At my senior year, I was interned at Rice University, where I was trying to conduct high throughput screens to study the genetic interactions in C.elegans. This internship serves as a wake up call that make me realize biological research is not only about conducting good experiments but also how to embrace and interpret the big data generated from experiments.

Purdue Electrical and Computer Engineering Ph.D program is the exact training I need to formulate my modeling skills to understand, interpret and generate knowledge from biological data. This thesis serves as a report of my five year training and endeavour on utilizing current resources and technologies to reveal the discrete signals in gene expression. Primarily, I focus on the genome arrangement and identify gene expression state from single cell RNA sequencing data. In each chapter, I mathematically formulate one specific problem in biology and propose our novel solutions. Noted, our proposed methods can be applied to problem in other field with similar formulations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Gene expression is an intricate process that determines different cell types and functions in metazoans, where most of its regulation is communicated through discrete signals, like whether the DNA helix is open, whether an enzyme binds with its target, etc. Understanding the regulation signals of the selective expression process is essential to the full comprehension of biological mechanism and complicated biological systems. In this research, we seek to reveal the discrete signals in gene expression by utilizing novel machine learning approaches. Specifically, we focus on two types of data chromatin conformation capture (3C) and single cell RNA sequencing (scRNA-seq). To identify potential regulators, we utilize a new hypergraph neural network to predict genome interactions, where we find the gene co-regulation may result from the shared enhancer element. To reveal the discrete expression state from scRNA-seq data, we propose a novel model called LTMG that considered the biological noise and showed better goodness of fitting compared with existing models. Next, we applied Boolean matrix factorization to find the co-regulation modules from the identified expression states, where we revealed the general property in cancer cells across different patients. Lastly, to find more reliable modules, we analyze the bias in the data and proposed BIND, the first algorithm to quantify the column- and row-wise bias in binary matrix.

We will first introduce the background of the thesis in the first chapter. In the second chapter, we will discuss how we formulate the genome interaction prediction task as hyperedge prediction problem and proposed a theoretically driven neural network HIGNN which achieved 30% performance increase comparing with other methods. Next, we thought to identify the discrete gene expression states. Specifically, in the third chapter, we proposed a left truncated mixture Gaussian model that retrieve the state information from single cell RNA sequencing data. In the fourth and fifth chapter, we introduce fast and efficient Boolean matrix/tensor factorization method to identify functional patterns from the expression states. In the sixth and seventh chapter, we further discussed the bias issue in binary data and proposed the first bias aware Boolean matrix factorization method that mitigate the impact from row- and column-wise bias in a binary matrix.

1. INTRODUCTION

The live beings of metazoans (multicellular animals), are started from a single zygote. Under the same genome, this zygote reproduces and differentiates into multitude of cell types that forms individual systems with distinct functionalities. For example, in our human genome, there are over 26000 genes [1]. Selective expression of specific genes established approximately 37.2 trillions cells with different functionalities that shape our blood system, cardiovascular system, digestive system, etc [2].

As eukaryotic genomes are finely packaged in the nucleus of cells, gene expression is an intricate process [3]–[5] (Figure 1.1). To start the expression of a gene, RNA polymerase recognize its promoter region and open up the DNA duplex structure for mRNA synthesis [5]. The newly generated mRNA undergoes prepossessing and is transported to cytoplasm [6], where it binds with ribosome and translate into protein [4]. These processes are majorly regulated by discrete signals. For example, on genome level, histone modification and DNA methylation impact the DNA accessiblity to RNA polymerases [7]–[9]. On transcription level, transcription factors (TF) can bind near promoter or enhancer that guides RNA polymeraze to the promoter of target genes [5], [10], [11]. And on protein level, the binding between eukaryotic initiation factors and mRNA determines its translation into protein [12].

In this thesis, we mainly focus on the gene expression up to its transcription, where we regard the generation of mRNA representing the formation of different cell types [5], [13]. Specifically, we focus on the arrangement of the genome and analysis of mRNA, where

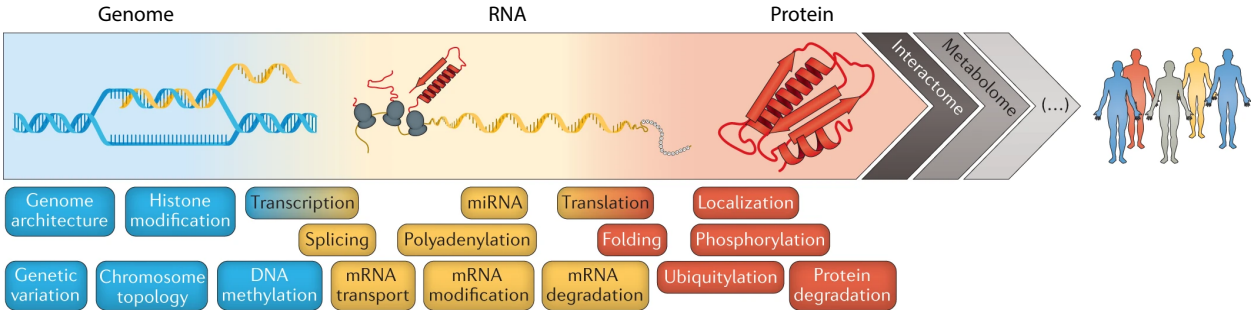

Figure 1.1. Overview of gene expression, adopted from [3]

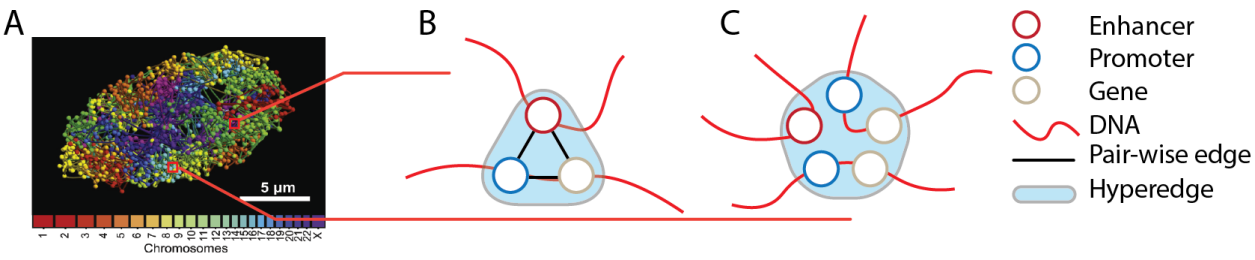

Figure 1.2. Hyperedge reflects higher order genome interactions, A is adopted from [20]

we apply different machine learning approaches to reveal the discrete signals of the genes expression.

1.1 Problem formulation and available resources

Biological research has embraced the fast development of the genetic sequencing technologies, where it provides unprecedented resolution on different levels of gene expression [14]. In this research, we are particular focusing on two techniques, chromosome conformation capture (3C) and single cell RNA sequencing (scRNA-seq) [15], [16].

1.1.1 Modeling chromosome 3D arrangement as hypergraph

How the genome organize in three dimensional space (3D) is the first step in regulating gene expression, where the arrangement of different genetic elements in the nucleus impacts the accessibility of target genes [7], [8] (Figure 1.2A). Chromosome conformation capture (3C) is the technique to characterize such information [17]. Specifically, in this research, we focus on the ligation-free 3C techniques like SPRITE and GAM [18], [19]. Conveniently, we could model the genome organization as hypergraph, where each node is the genetic elements (enhancer, promoter, gene, etc), and each hyperedge is a cluster of the genetic elements that are close in 3D (Figure 1.2B,C) [18]. Noted, such closeness is determined by experimental measurement. For example in SPRITE, after DNA cross-linking and digestion, the cluster is the remaining DNA elements that are still connected together (for detail refer to [18]).

The limited experimental resolution of 3C methods restricted the fully elucidation of DNA connections in the nucleus. To fill the missing pieces, here we formulated this problem

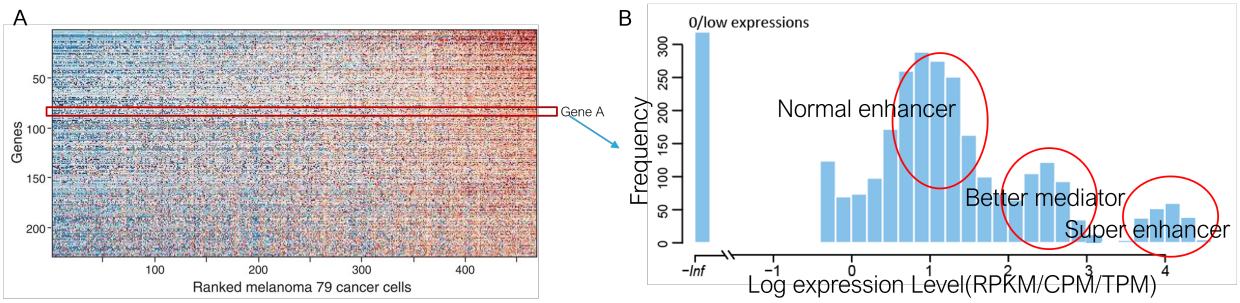

Figure 1.3. Single cell RNA sequencing and gene expression, A is adopted from [21]

as hyperedge link prediction tasks. Given the captured DNA interaction (hypregeraph), we want to predict the existence of potential hyperedges that reveals different cooperation of genetic elements in 3D. In chapter 2, we will discuss in detail about our hypergraph neural network for hyperedge prediction. The novelty of our work is that we focus on the ambiguities issue of graph neural networks for link prediction task in the case of hypergraph, namely node-level ambiguity and edge-level ambiguity. As a result, our proposed neural network model achieved 30% performance increase on the F1 score for hyperedge prediction. Our top prediction also potentially revealed the co-expression of two genes is related to a shared enhancer in 3D.

1.1.2 Gene expression state from single cell RNA sequencing

Single cell RNA sequencing is to detect RNA molecules of different genes in every individual cell [16], [22]. In general, it could be considered as non-negative matrix (Figure 1.3A), where every row represent one of the 26000 genes and every column is one of the individual cells in the biological sample. Conventional approaches aim to reveal the property of different cell types, which treat each column of the matrix as the vector representation of the cell. Then it utilize dimension reduction and clustering methods to find the different cell cluster that showed phenotipic variance [21], [23], [24]. In this research, rather than the individual cell, we focus on the expression of genes across different cells. Here we regard gene expression is multimodal across different cell types, where captured RNA expression is the manifesto of the expression states resulted from varied upstream regulations in different cell

types. For example, whether they connected to different enhancer elements, or they have better mediator that boosted the expression of genes in the cells (Figure 1.3B).

In this thesis, we focus on two problems in terms of gene expression state. 1. How to accurately infert gene expression state? 2. How to utilize the gene expression state to reveal the cell type property? Capturing the gene expression in the single cell is a difficult task with a strong background noise. How to deal with the noise is the core issue to reveal different expression state. In chapter 3, we propose a left truncated mixture Gaussian distribution (LTMG) for this task. LTMG treated the undetectable expression as left censored data and identify different expression state as mixture Gaussian peaks. For the second task, we utilize Boolean matrix factorization (BMF) to find the different cell type behavior over the identified gene expression state information. As BMF is an NP-hard problem [25], in chapter 3 and 4, we propose our fast and efficient Boolean matrix factorization method MEBF and Boolean tensor factorization method GETF that could deal with the large size of gene expression data. Moreover, we also discuss the bias issue in binary data in chapter 6 and proposed another BMF method BABF that could deal with bias and conduct Boolean factorization simultaneously in chaper 7.

1.2 Contribution of this thesis

- We seeks to reveal the discrete signals in regulating gene expression in the complicated biological systems.
- We focus on two data resources 3C and scRNA-seq, and formulated the problem as hyperedge prediction, statistical modeling and matrix factorization.
- We propose a novel hypergraph neural network that deals ambiguities issue in the hyperedge prediction tasks, which achieved 30% performance increase compared with existing model.
- We propose a sophisticated designed model, LTMG to accurately capture gene expression state while dealing with noise.

- We propose first of its kind fast and efficient Boolean matrix and tensor factorization methods, MEBF and GETF to reveal the data patterns in binary data.
- We also discussed bias issue in binary data and proposed the first bias-aware BMF approache, BABF.

1.3 Publications results from this work

1. **Wan, Changlin**, Muhan Zhang, Wei Hao, Sha Cao, Pan Li, and Chi Zhang. "Principled hyperedge prediction with structural spectral features and neural networks." arXiv preprint arXiv:2106.04292 (2021).
2. **Wan, Changlin**, Wennan Chang, Yu Zhang, Fenil Shah, Xiaoyu Lu, Yong Zang, Anru Zhang et al."LTMG: a novel statistical modeling of transcriptional expression states in single-cell RNA-Seq data." Nucleic acids research 47, no. 18 (2019): e111-e111.
3. **Wan, Changlin**, Wennan Chang, Tong Zhao, Mengya Li, Sha Cao, and Chi Zhang."Fast and efficient boolean matrix factorization by geometric segmentation." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 6086-6093. 2020.
4. **Wan, Changlin**, Wennan Chang, Tong Zhao, Sha Cao, and Chi Zhang. "Geometric All-Way Boolean Tensor Decomposition." Advances in Neural Information Processing Systems 33 (2020): 2848-2857.
5. **Wan, Changlin**, Dongya Jia, Yue Zhao, Wennan Chang, Sha Cao, Xiao Wang, and Chi Zhang. "A data denoising approach to optimize functional clustering of single cell RNA-sequencing data." In 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 217-222. IEEE, 2020.
6. **Wan, Changlin**, Wennan Chang, Tong Zhao, Sha Cao, and Chi Zhang. "Denoising Individual Bias for Fairer Binary Submatrix Detection." In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2245-2248. 2020.

7. **Wan, Changlin**, Pengtao Dang, Tong Zhao, Yong Zang, Sha Cao, and Chi Zhang. "Bias aware probabilistic Boolean Matrix Factorization." Under Review.

8. Fang, Yuanzhang†, Lifei Wang†, **Changlin Wan**†, Yifan Sun, Kevin Van der Jeught, Zhuolong Zhou, Tianhan Dong et al. "MAL2 drives immune evasion in breast cancer by suppressing tumor antigen presentation." The Journal of clinical investigation 131, no. 1 (2021).

9. Zhang, Yu†, **Changlin Wan**†, Pengcheng Wang, Wennan Chang, Yan Huo, Jian Chen, Qin Ma, Sha Cao, and Chi Zhang. "M3S: a comprehensive model selection for multi-modal single-cell RNA sequencing data." BMC bioinformatics 20, no. 24 (2019): 1-5.

† Co-first author.

2. HYPERGRAPH NEURAL NETWORK FOR HYPEREDGE PREDICTION

Graph has been broadly utilized to study relational data in various domains such as e-commence, drug design, social network analysis, and recommendation system [26]–[30]. While many methods have been devoted to represent pair-wise relations in ordinary *graphs*, it has been recognized that many real world relationships are characterized with more than two participating partners and thus not bilateral [31]–[35]. Taking the genetic interaction as an example (Figure 2.1A, 2.1B). An accurate characterization of gene expression involves the joint interaction among gene, promoter and enhancer, and capturing only their pair-wise interaction (enhancer-promoter, promoter-gene or gene-enhancer) will not fully recapitulate the gene regulatory relationship (Figure 2.1B) [5], [36], [37]. The same issue also exists in the analysis of multi-component drug design (Figure 2.1C), multi-ingredient recipes (Figure 2.1D), where multilateral relationships are not compatible with ordinary graph edges [38]–[40]. To overcome such conceptual limitations, *hypergraph* has been developed to model the higher-order interaction data [41]. In a hypergraph, any higher-order connection is represented by a hyperedge that could join an any number of entities (blue shadow Figure 2.1B, 2.1C, 2.1D). Hence, predicting the higher-order relation is transformed into a hyperedge prediction problem in a hypergraph.

Earlier works on hyperedge prediction rely on structural heuristics of hypergraph clique expansion, such as common neighbor, geometric mean, Adar index [29], as well as random

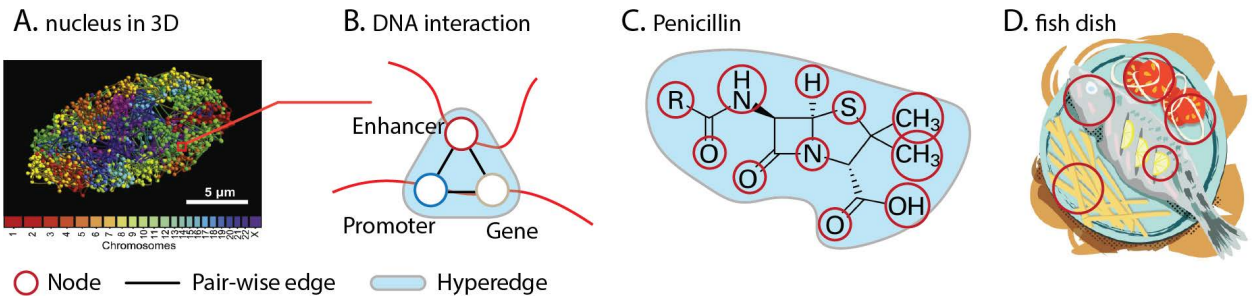

Figure 2.1. Hyperedge reflects higher-order interaction in many real world data. A adopted from [20]

walk related methods [42]. Yoon et al showed that combining clique expansion and higher order information of hypergraph enhances the prediction accuracy [43]. However, these methods are often restricted to predicting specific k-node hyperedges with mixed generalization power. Zhang et al leveraged matrix factorization to predict hyperedges from a given pool [40]. On the other hand, GNN has been introduced as a powerful method for hyperedge prediction, and showed improved performance [32], [34], [35], [44], [45]. Essentially, most existing methods can be considered as integrating information of individual nodes for hyperedge representations.

However, these methods could suffer from severe **ambiguity** issues. For instance, two hypergraphs can have nodes with identical pair-wise connections but different hyperedges (such as $\{v_1, v_2\}$ in Figure 2.2A and $\{v_1, v_2\}$ in Figure 2.2B). Methods based on pair-wise node relationship (clique expansion) like common neighbor, geometric mean will fail to tell such differences. Another example is to consider two different hyperedges whose connected nodes themselves are highly similar ($\{v_1, v_2, v_3\}$ and $\{v_2, v_3, v_6\}$ in Figure 2.2D). Methods like GNNs that rely on aggregating node embeddings as hyperedge representations will wrongly predict these two hyperedges to be the same (Figure 2.2D).

In this chapter, we formalize the above ambiguity issues into two classes, namely **node-level ambiguity** and **edge-level ambiguity**, which cause the major challenges of the link-prediction problem in *hypergraphs* compared to ordinary *graphs*. Such size flexibility enables different arrangement of hyperedges. As a result, hypergraph can no longer be bijectively mapped with pair-wise *graph* (node-level ambiguity). Also unlike two-node edge in *graph*, whose characteristics could largely be explained by aggregating the node information. Whereas in the case of hypergaph, the aggregation scheme could not represent the hyperedge in full as it omits the information of the node dependency within the hyperedge (edge-level ambiguity).

To address these issues for a better representation/prediction of hyperedges, we propose **HIGNN** which utilizes a bipartite GNN and hyperedge-specific node structural features to avoid such information loss. Compared with most recent models, HIGNN achieved a large margin of performance increase on the hyperedge prediction task. We also applied HIGNN to higher order genome interaction data, where HIGNN showed consistent stability across

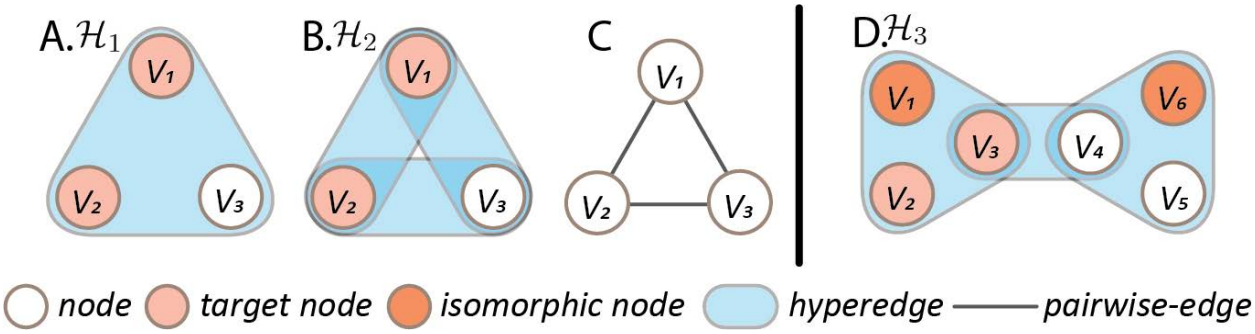

Figure 2.2. Ambiguity of different hypergraphs and isomorphic nodes.

different chromosomes. Moreover, HIGNN gives plausible DNA interaction prediction as the top predicted result is validated by existing literature [46].

We summarize our contributions as:

1. We mathematically describe the two challenges in hyperedge representation learning as node- and edge-level ambiguities, which prevent simple models from learning good hyperedge representations

2. We introduce a general framework HIGNN to tackle the two ambiguities, i.e., by using bipartite graph neural network to handle node-level ambiguity and hyepredge-specific node structure features to handle edge-level ambiguity.

3. Experiments show consistent performance improvement compared with recent state-of-the-art models for hyperedge prediction.

2.1 The two ambiguities in hyperedge representation

We first introduce the notations used in this work. Denote a set as an uppercase character (e.g. X), elements in a set as lowercase characters (x), a vector as a bold lower case character ($\boldsymbol{x}$), and a matrix as a bold uppercase character ($\boldsymbol{X}$), respectively. The dimension and indices of entries of a matrix are represented by its upper-script (e.g. $\boldsymbol{X}^{n\times m}$) and lower-script (e.g. i-th row: $\boldsymbol{X}_{\mathrm{i:}}$, j-th column: $\boldsymbol{X}_{\mathrm{:j}}$, and the entry of the i-th row and j-th column: $\boldsymbol{X}_{\mathrm{ij}}$), respectively. Let $\mathcal{H} = (V, E)$ represent a hypergraph, where V is the vertex set $V = \{v_1, .., v_n\}$ and E is the edge set $E = \{S_1, ..., S_m\}$, $E \subseteq P(V)$, and $P(V)$ represents the

powerset of V. The cardinality of a hyperedge S is defined by the number of nodes in S, which is denoted by $\|S\|$. The incidence matrix of a hypergraph is defined as $\boldsymbol{H} \in \{0,1\}^{\|V\|\times\|E\|}$, in which $\boldsymbol{H}_{\mathrm{ij}} = 1$ indicates $v_{\mathrm{i}} \in \mathrm{e_j}$, and otherwise $\boldsymbol{H}_{\mathrm{ij}} = 0$. Denote a hyperedge representation learning function as $f : S, \boldsymbol{H} \to \mathbb{R}^k$, $S \subset V$. The hyperedge prediction problem can be generally formulated as training f as well as a prediction function p that takes $f(S, \boldsymbol{H})$ as input, by which $p(f(S, \boldsymbol{H}))$ predicts if S forms a hyperedge. Collectively, we denote the overall prediction function $p(f(S, \boldsymbol{H}))$ as $p \circ f$. When $\boldsymbol{H}$ is clear from the context, we sometimes also write $f(S, \boldsymbol{H})$ as $f(S)$.

In this work, we only consider undirected and non-attributed hypergraph, so that the representation learning function f only captures the topological characteristics of the hypergraph. Nevertheless, the method described in this study can be easily extended to the representation learning of hypergraphs with directions and node-/edge-attributes.

Definition 2.1.1. ***Hypergraph isomorphism.*** *Two hypergraphs $\mathcal{H} = (V, E)$ and $\mathcal{H}' = (V', E')$ are isomorphic if $\exists$ a bijective mapping $\boldsymbol{\pi} : V \to V', s.t.\ \boldsymbol{\pi}(V) = V'$ and $\boldsymbol{\pi}(E) = \{\boldsymbol{\pi}(S) \| S \subset E\} = E'$, where $\boldsymbol{\pi}(S) = \{\boldsymbol{\pi}(v) \| v \in S\}$. Such a bijective mapping is called an isomorphic mapping. Specifically, a permutation operation $\boldsymbol{\pi} : V \to V$ is isomorphic if $\boldsymbol{\pi}(E) = E$ (automorphism). As isomorphic permutations are exchangeable, we can define the set of all isomorphic permutations of (V, E) as Π_I. For any node v, its isomorphic node set is defined by $I(v) = \{v' \| \exists\ \boldsymbol{\pi} \in \Pi_I\ s.t.\ \boldsymbol{\pi}(v) = v'\}$ (orbit), and isomorphic edge set of any edge S is defined by $\Pi_I(S) = \{S' \| \exists \boldsymbol{\pi} \in \Pi_I\ s.t.\ \boldsymbol{\pi}(S') = S\}$ (edge orbit). It is noteworthy that Π_I generates a segmentation of $P(V)$, denoted as $\Pi_I(\mathcal{H})$, which can be represented as $\Pi_I(\mathcal{H}) = \{\Pi_I(S_{(\mathrm{i})}) \| S_{(\mathrm{i})} \in P(V); \cup \Pi_I(S_{(\mathrm{i})}) = P(V); \Pi_I(S_{(\mathrm{i})}) \cap \Pi_I(S_{(\mathrm{j})}) = \emptyset, \forall \mathrm{i}, \mathrm{j}\}$.*

Definition 2.1.2. ***Isomorphic invariance.*** *A hyperedge representation learning function f is called isomorphic-invariant if for $\forall S \subset V$ and $\forall \boldsymbol{\pi} \in \Pi_I$, $f(S, \boldsymbol{H}) = f(\boldsymbol{\pi}(S), \boldsymbol{\pi}(\boldsymbol{H}))$.*

Intuitively, a good hyperedge learning function f should be isomorphic invariant, as it ensures the generalization of f on isomorphic hyperedges, i.e., hyperedge with identical topological structure will get same representation. The isomorphic invariant property of f is a necessary but insufficient condition of a valid hyperedge predictor ($\Pi_I(S)$). To measure

the efficacy of f, we introduce F-invariance ($\Pi_f(S)$) as follows. Noted, isomorphic invariant f is a sufficient but unnecessary condition of the isomorphic invariant property of $p \circ f$.

Definition 2.1.3. ***F-invariance.*** *Two hyperedges S and S' are F-invariant w.r.t a isomorphic invariant hyperedge representation function f, if $\exists \boldsymbol{\pi} : V \to V'$ s.t. $f(S, \boldsymbol{H}) = f(\boldsymbol{\pi}(S'), \boldsymbol{\pi}(\boldsymbol{H'}))$. Here, we define the F-invariant edge set of $S \in V$ as $\Pi_f(S) = \{S' \| \exists \boldsymbol{\pi} \in \Pi_n$ s.t. $f(S) = f(\boldsymbol{\pi}(S'))\}$ (edge orbit w.r.t f).*

Similarly to isomorphic permutations, Π_f also generates a segmentation of $P(V)$, which is defined by $\Pi_f(\mathcal{H}) = \{\Pi_f(S_{(\mathrm{i})}) \| S_{(\mathrm{i})} \in P(V); \cup \Pi_f(S_{(\mathrm{i})}) = P(V); \Pi_f(S_{(\mathrm{i})}) \cap \Pi_f(S_{(\mathrm{j})}) = \emptyset, \forall \mathrm{i}, \mathrm{j}\}$.

Lemma 2.1.1. *Any isomorphic invariant function $f(S, \boldsymbol{H})$ is permutation invariant, and $\forall S \subset V$, $\Pi_I(S) \subseteq \Pi_f(S)$.*

Proof. If a hypergraph does not have any non-trivial isomorphic permutation, $\forall S \subset V, \Pi_I(S)$ only has one element, $\Pi_I(S) \subseteq \Pi_f(S)$. For a hypergraph having at least one non-trivial isomorphic permutation, and an isomorphic invariant function f, $f(S, \boldsymbol{H}) = f(\boldsymbol{\pi}(S'), \boldsymbol{\pi}(H)) = f(S', \boldsymbol{H}), \forall S' \in \Pi_I(S)$, i.e., S and all of its isomorphic hyperedge $S' \in \Pi_I(S)$ share the same output of f. Hence f is permutation invariant and $\Pi_I(S) \subseteq \Pi_f(S)$. □

Lemma 2.1.2. *For a hyperedge $S \subset V$, if $\exists$ a permutation $\boldsymbol{\pi}$ s.t.$f(S, \boldsymbol{H}) = f(\boldsymbol{\pi}(S), \boldsymbol{\pi}(\boldsymbol{H}))$ and f is isomorphic invariant, then $\boldsymbol{\pi}(S) \in \Pi_f(S)$.*

Proof. Considering the bijective mapping $\boldsymbol{\pi}_0 : \boldsymbol{\pi}_0(S) = \boldsymbol{\pi}(S), \boldsymbol{\pi}_0(\boldsymbol{\pi}(S)) = \boldsymbol{\pi}^{-1}(\boldsymbol{\pi}(S)) = S$, and $\boldsymbol{\pi}_0(v) = v, v \in V \setminus (S \cup \boldsymbol{\pi}(S))$, $\boldsymbol{\pi}_0$ is a permutation. Since $f(S, \boldsymbol{H}) = f(\boldsymbol{\pi}(S), \boldsymbol{\pi}(\boldsymbol{H}))$ and f is permutation invariant, $f(A) = f(\boldsymbol{\pi}_0(A))$, $\forall A \subset (S \cup \boldsymbol{\pi}(S))$. And $\boldsymbol{\pi}_0$ is an identical mapping for $v \in V \setminus (S \cup \boldsymbol{\pi}(S))$. Hence, $\boldsymbol{\pi}_0$ is an F-invariant permutation w.r.t. f, i.e., $\boldsymbol{\pi}(S) \in \Pi_f(S)$. □

Lemma 2.1.1 suggests that the segmentation $\Pi_I(\mathcal{H})$ is always finer than $\Pi_f(\mathcal{H})$. Because $p(f(S, \boldsymbol{H}))$ has the same output for $S \subset \Pi_f(S)$. Specifically, $\Pi_f(S) \setminus \Pi_I(S)$ could reflect the ratio of false-discoveries in distinguishing $\Pi_I(S)$. Lemma 2.1.2 characterizes a general condition of the hyperedges in a same F-invariant edge set. For two permutation invariant functions f_1 and f_2, if $\forall S \subset V$, $\Pi_{f_1}(\mathcal{S}) \subseteq \Pi_{f_2}(\mathcal{S})$, we call f_1 is more informative than f_2 in

representing $\mathcal{H}$. Thus, the **objective** of hyperedge prediction task, or in general hyperedge representation task, is to find the permutation invariant representation function f, whose F-invariant edge set $\Pi_f(S)$ is expected to be close to the isomorphic set of hyperedges, i.e., $\Pi_f(S) \approx \Pi_I(S)$. In the following, we discuss current heuristic and GNN based methods for hyperedge prediction and illustrate edge- and node-level ambiguity that lead to $\Pi_I(S) \subset \Pi_f(S)$, i.e., $\exists S_1, S_2$ s.t. $S_2 \in \Pi_f(S_1), S_2 \notin \Pi_I(S_1)$.

2.1.1 Heuristics and GNNs for hyperedge prediction

A classic way to represent hyperedge is to utilize structural heuristics [29]. Take common neighbor (CN) as an example. Let $N(v)$ denote the neighbor set of node v in hypergraph $\mathcal{H}$. The hyperedge S could be represented as

$$f_{CN}(S, \boldsymbol{H}) = \cap_{v \in S} N(v).$$

Logistic regression p on the output of f is then used to predict the existence of hyperedge.

Recent development of GNN on representation learning tasks also achieved unprecedented performance [47]–[49]. The representation learning of GNN takes a general form as

$$\boldsymbol{X}^{l+1} = \sigma(\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\boldsymbol{X}^{l}\boldsymbol{W}^{l}),$$

where $\boldsymbol{X} \in \mathbb{R}^{\|V\| \times k}$ represents the learned node embedding, σ is a non-linear activation function, $\boldsymbol{A} \in \{0,1\}^{\|V\| \times \|V\|}$ is the adjacency matrix of the input graph. In hypergraph embedding, the adjacency matrix is defined by a clique expansion of the incidence matrix, $\boldsymbol{A} = sign(\boldsymbol{H}\boldsymbol{H}^T) \in \{0,1\}^{\|V\| \times \|V\|}$, in which $\boldsymbol{A}_{\mathrm{ij}} = 1$ if node v_{i} and v_{j} belong to at least one hyperedge, and otherwise, $\boldsymbol{A}_{\mathrm{ij}} = 0$ [50]. $\boldsymbol{D}_{\mathrm{ii}} = \sum_{\mathrm{j}} \boldsymbol{A}_{\mathrm{ij}}$ is the degree matrix and $\boldsymbol{W}^l \in \mathbb{R}^{k \times k}$ is the layer-specific weight matrix for the lth layer. A hyperedge S is then represented by aggregating information from the learned node embedding $\boldsymbol{X}$ in a permutation invariant fashion (e.g. sum-pooling, mean-pooling, max-pooling et al), i.e.,

$$f_{\mathrm{GNN}}(S, \boldsymbol{A}) = AGG(\boldsymbol{X}_{v:} \| v \in S).$$

We abuse the notation of $\boldsymbol{X}_{v:}$ to represent the node embedding of v. Since $\boldsymbol{X}_{v:}$ could be considered as the output of $f_{\text{GNN}}(v, \boldsymbol{A})$, the function f could also be written as

$$f_{\text{GNN}}(S, \boldsymbol{A}) = AGG(f_{\text{GNN}}(v, \boldsymbol{A}) \| v \in S).$$

Followed by a fully connected neural network p as prediction function, GNN-based methods $(p \circ f)$ could be trained end to end, compared with heuristic approaches.

2.1.2 Node-level ambiguity

Node-level ambiguity is defined as a false assignment of identical node embeddings to non-isomorphic nodes. Adjacency matrix over-simplifies the topological characteristics of a hypergraph, which can cause a node-level ambiguity as showcased in Figure 2.2A, 2.2B, 2.2C. Clearly, any two nodes from two hypergraphs $\mathcal{H}_1 = \{V_1, E_1\}$ and $\mathcal{H}_2 = \{V_2, E_2\}$ are not isomorphic. However, due to $\mathcal{H}_1$ and $\mathcal{H}_2$ having the same clique expansion $\boldsymbol{A}_{\mathcal{H}_1} = \boldsymbol{A}_{\mathcal{H}_2}$, f_{CN} and f_{GNN} assign the same common neighbor or node embedding to any nodes from them. Hence, for $\forall S_1 \subset V_1,\ S_2 \subset V_2$, where S_1 and S_2 have the same cardinality, CN and GNN have $p(f(S_1)) = p(f(S_2))$, i.e. $S_2 \in \Pi_f(S_1)$ and $S_2 \notin \Pi_I(S_1)$.

2.1.3 Edge-level ambiguity

Edge-level ambiguity is defined by a false assignment of identical embeddings to non-isomorphic edges.

Lemma 2.1.3. *For any isomorphic invariant edge representation learning function follows* $f(S, \boldsymbol{H}) = AGG(f(v_1, \boldsymbol{H}), f(v_2, \boldsymbol{H}), ..., f(v_m, \boldsymbol{H})), v_1, ..., v_m \in S$, *and* $\forall S' = \{v'_1 \in \Pi_f(v_1), v'_2 \in \Pi_f(v_2), ..., v'_m \in \Pi_f(v_m)\}$, *then* $S' \in \Pi_f(S)$.

Proof. As f is isomorphic invariant, $f(S, \boldsymbol{H}) = f(S', \boldsymbol{H})$, and by Lemma 2.1.2, $S' \in \Pi_f(S)$. □

A simple aggregation of node embedding ensures a high computational feasibility and an easy handling of the edges of different cardinality. However, Lemma 2.1.3 suggests that

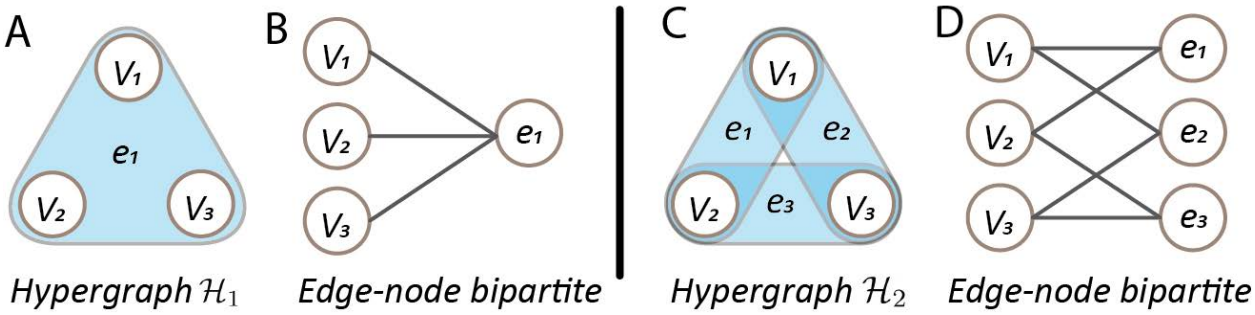

Figure 2.3. Node-level ambiguity is caused by different hyperedges with same clique expansion.

isomorphic invariant f ignores the topological dependency of nodes with S when it adopts the aggregation based formulation, i.e. $f(S, \boldsymbol{H}) \perp \boldsymbol{H} \| \{f(v_1, \boldsymbol{H}), f(v_2, \boldsymbol{H}), ..., f(v_m, \boldsymbol{H})\}$. Hence, all aforementioned methods suffer an over-simplified edge embedding. Figure 2.2D illustrates one example of edge-level ambiguity caused by such over-simplification. In the hypergraph, $\{v_1, v_2, v_5, v_6\}$ are isomorphic and $\{v_3, v_4\}$ are isomorphic. If f satisfies Lemma 2.1.3, $f(v_1) = f(v_6)$, then $p(f(v_1, v_2, v_3)) = p(AGG(f(v_1), f(v_2), f(v_3))) = p(f(v_2, v_3, v_6))$. However, the node sets $S_1 = \{v_1, v_2, v_3\}$ and $S_2 = \{v_2, v_3, v_6\}$ clearly have different topological structures, i.e. $S_2 \in \Pi_f(S_1)$ and $S_2 \notin \Pi_I(S_1)$.

2.2 Methodology

In this section, we discuss techniques to solve the above two ambiguities. Specifically, (1) to address the node-level ambiguity, we adopt a bipartite message passing neural network; and (2) to address the edge-level ambiguity, we propose Hyperedge-Specific Node Structural Features which encode each node's structural relationship w.r.t. the target hyperedge to predict. We show that by using these two techniques, the two ambiguities can be effectively alleviated.

2.2.1 Bipartite message passing neural network.

Considering each hyperedge as an individual object, the hypergraph $\mathcal{H} = (V, E)$ could be manifested as a bipartite graph, where one partite represents nodes V and the other

represents the hyperedges E (Figure 2.3). The edge-node bipartite graph is equivalent to the incidence matrix $\boldsymbol{H}$, which conceive more information than the clique expansion $\boldsymbol{A} = sign(\boldsymbol{H}\boldsymbol{H}^T)$. Bipartite message passing neural networks have been utilized in previous studies [51], [52], that takes the general form as

$$\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{W}_E) \quad \boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{W}_V)$$

The nonlinear activation in updating $\boldsymbol{X}_E$ and $\boldsymbol{X}_V$ enables a flexible and optimized information retrieval from $\boldsymbol{H}$, which is more informative than a clique expansion based representation ($\boldsymbol{A}$), i.e.

$$\boldsymbol{H}\sigma(\boldsymbol{H}^T\boldsymbol{X}_V\boldsymbol{W}_E)\boldsymbol{W}_V \neq \boldsymbol{W}'\boldsymbol{A}\boldsymbol{X}_V\boldsymbol{W}_V \qquad (\star)$$

The awareness of $\boldsymbol{H}$ distinguishes edges of different cardinality ($\star$ left hand side). Thus, it **avoids the node-level ambiguity** introduced by clique expansion in general graph neural network models ($\star$ right hand side).

In HIGNN, we apply this framework with a slight modification by introducing a one-side normalization term $\boldsymbol{D}_E^{-1}$ when updating $\boldsymbol{X}_E$,

$$\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T\boldsymbol{X}_V^l\boldsymbol{D}_E^{-1}\boldsymbol{W}_E) \quad \boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H}\boldsymbol{X}_E^l\boldsymbol{W}_V)$$

Empirically, the one-sided normalization approach can balance the trade off between degree bias and representation power [52]. Our experiments suggested that the one-sided normalization has a better performance than normalizing both $\boldsymbol{X}_V$ and $\boldsymbol{X}_E$ or no normalization .

2.2.2 Hyperedge-specific node structural features

We first visualize the edge-level ambiguity by taking GNN and the hypergraph in Figure 2.4A as an example. Without distinguishing node features, GNN learns the embedding of node v by retrieving its relationship with every node while traversing the whole hypergraph ($v_1 - v_6$). Owing to isomorphic invariant property of GNN, it is easy to de-

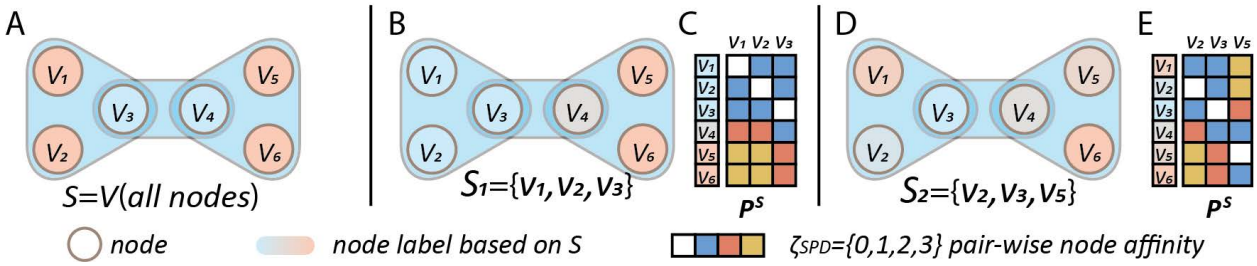

Figure 2.4. Edge-level ambiguity is caused by isomorphic nodes

rive that $f_{\text{GNN}}(v_1, \boldsymbol{H}) = f_{\text{GNN}}(v_2, \boldsymbol{H}) = f_{\text{GNN}}(v_5, \boldsymbol{H}) = f_{\text{GNN}}(v_6, \boldsymbol{H})$ and $f_{\text{GNN}}(v_3, \boldsymbol{H}) = f_{\text{GNN}}(v_4, \boldsymbol{H})$ as $v_2, v_5, v_6 \in \Pi_I(v_1)$ and $v_4 \in \Pi_I(v_3)$. Following lemma 2.1.3, by aggregating the node embeddings, GNN incurs edge-level ambiguity that recognise two different hyperedges $S_1 = \{v_1, v_2, v_3\}$ (Figure 2.4B) and $S_2 = \{v_2, v_3, v_5\}$ (Figure 2.4D) as the same when $AGG(f_{\text{GNN}}(v_1), f_{\text{GNN}}(v_2), f_{\text{GNN}}(v_3)) = AGG(f_{\text{GNN}}(v_2), f_{\text{GNN}}(v_3), f_{\text{GNN}}(v_5))$.

This example shed light on edge-level ambiguity (Figure 2.4A,B,D). Essentially, the structure specificity of S_1 and S_2 are reflected upon the relations between the nodes in the hypergraph and the nodes within the hyperedge. Here we reflect this relationship with hyperedge-specific affinity matrix as $\boldsymbol{P}^S \in \mathbb{R}^{\|V\| \times \|S\|}$ (Specifically, $\boldsymbol{P}^{S_1}$ and $\boldsymbol{P}^{S_2}$ in Figure 2.4C, 2.4E). GNN does not incorporate such specificity by only considering the node relations of whole hypergraph. Thus, it wrongly identified v_1, v_5 with same embedding w.r.t S_1 and S_2, while $f(v_1, \boldsymbol{H} \| S_1)$ and $f(v_5, \boldsymbol{H} \| S_2)$ should not be considered as equal since $\boldsymbol{P}^{S_1}_{v_1:} \neq \boldsymbol{P}^{S_2}_{v_5:}$ (Figure 2.4D, 2.4F). As consequence, the edge-level ambiguity is caused while ignoring the **dependency** between nodes v_1, v_5 and the specific hyperedge S_1, S_2.

Definition 2.2.1. ***Hyperedge-specific node structural features.*** *Given the hypergraph $\mathcal{H}$, we define hyperedge-specific matrix $\boldsymbol{P}^S \in \mathbb{R}^{\|V\| \times \|S\|}$ as hyepredge-specific node structure feature of a hyperedge (or a target nodes set of interest) S, where $\boldsymbol{P}^S_{\text{ij}} = \zeta(v_{\text{i}}, v_{\text{j}})$ and ζ is an nodes pair affinity measure function. $\boldsymbol{P}^S$ reveals the structural relationship between the nodes within the hyperedge and nodes from the hypergraph. For ease of illustration, in this chapter, we use shortest path distance (smallest number of edges that link two nodes) as the affinity measure function, i.e., ζ_{SPD}, whereas other affinity functions could be similarly applied.*

To encode such dependency, here we define the hyperedge-specific affinity matrix as hyperedge-specific node structure feature, where: 1) row of the matrix $\boldsymbol{P}^S_{v_{\mathrm{i}}:}$ can be regarded as the structure feature for node v_{i} w.r.t S, and moreover 2). $\boldsymbol{P}^S$ as whole can be considered as a feature of hyperedge S. Considering the information in $\boldsymbol{P}^S$ ease **edge-level ambiguity** as any permutation invariant f is more likely to distinguish hyperedges. For instance, since $\boldsymbol{P}^{S_1}_{v_1:} \neq \boldsymbol{P}^{S_2}_{v_5:}$, by adding such information, GNN can easily differentiate S_1 and S_2. Also because of $\boldsymbol{P}^S$, even the same node would have different effect towards specific hyperedges, (e.g., $\boldsymbol{P}^{S_1}_{v_2:} \neq \boldsymbol{P}^{S_2}_{v_2:}$ and $\boldsymbol{P}^{S_1}_{v_3:} \neq \boldsymbol{P}^{S_2}_{v_3:}$) (Figure 2.4). Noted, $\boldsymbol{P}^S$ will still be the same for isomorphic hyperedges. If $S_1 = \boldsymbol{\pi}(S_2)$, it is easy to derive $\boldsymbol{P}^{S_1} = \boldsymbol{\pi}(\boldsymbol{P}^{S_2})$.

Definition 2.2.2. ***Hyperedge-specific local representation.*** *Given a hyperedge (or a target nodes set) S, its q-hop neighbor nodes set $V^S_{(q)}$, edges set $E^S_{(q)}$, incidence matrix $\boldsymbol{H}^S_{(q)}$ and hyperedge-specific local node structure feature $\boldsymbol{P}^S_{(q)}$ are defined as follows:* $V^S_{(q)} = \{v_{\mathrm{j}} \| \zeta_{SPD}(v_{\mathrm{i}}, v_{\mathrm{j}}) \leq q, \forall v_{\mathrm{j}} \in V, \forall v_{\mathrm{i}} \in S\}$. $E^S_{(q)} = \{\mathrm{e}_{\mathrm{i}} \| \mathrm{e}_{\mathrm{i}} \subseteq V^S_{(q)}, \forall \mathrm{e}_{\mathrm{i}} \in E\}$. $\boldsymbol{H}^S_{(q)} \in \{0,1\}^{\|V^S_{(q)}\| \times \|E^S_{(q)}\|}$, *where* $\boldsymbol{H}^S_{(q)_{\mathrm{ij}}} = 1$ *if* $V^S_{(q)_{\mathrm{i}}} \in E^S_{(q)_{\mathrm{j}}}$, *otherwise* $\boldsymbol{H}^S_{(q)_{\mathrm{ij}}} = 0$. $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times \|S\|}$, *where* $\boldsymbol{P}^S_{(q)_{\mathrm{ij}}} = \zeta_{SPD}(v_{\mathrm{i}}, v_{\mathrm{j}} \| v_{\mathrm{i}} \in V^S_{(q)}, v_{\mathrm{j}} \in S)$.

Encoding the structural information of affinity matrix causes additional computation. To ensure the computational feasibility, instead of the representation with entire graphs, hyperedge local representation only requires q-hop enclosing subgraph $\boldsymbol{H}^S_{(q)}$ around S. Practically, $q \leq 2$ is sufficient for a good prediction performance. We argue that $q \leq 2$ is a practical setting in real-world analysis, because (1) exact isomorphic nodes are rare in real-world hypergraphs, utilizing local information to determine similar nodes can increase the inductive power of the model; (2) For the task of hyperedge prediction, the nodes beyond 2-hop are less deterministic for the existence of a hyperedge; (3) $q \leq 2$ bounds the size of $\boldsymbol{H}^S_{(q)}$ and $\boldsymbol{P}^S_{(q)}$ that dramatically reduce the computational cost and can be directly implemented in the message passing neural network. We present our model rooted in $f(S, \boldsymbol{P}^S_{(q)}, \boldsymbol{H}^S_{(q)})$ for the hypergraph edge representation/prediction task in the next section.

2.3 HIGNN for hyperedge prediction

In this section, we represent our model **HIGNN** to solve the ambiguity issues in hyperedge representation and prediction problems. HIGNN integrates two isomorphic invariant functions: 1) **S**tructural representing **N**eural network $f_{SN}(S, \boldsymbol{P}^S_{(q)}, \boldsymbol{H}^S_{(q)})$ that awares the hyperedge size differences (node-level ambiguity) by utilizing bipartite graph neural network and offsets the aggregating impact in lemma 2.1.3 with structural features as node features (edge-level ambiguity). 2) hyperedge **L**ocal **S**pectrum $f_{LS}(S, \boldsymbol{P}^S_{(q)})$ that reflects the low-rank property of $\boldsymbol{P}^S$ indicating the joint interactions between hyperedge and its local environment.

2.3.1 Structural representing neural network.

A series of work has explored the fixed k-node edge representation with affinity matrix $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times k}$, which grants new labels for each node in the presentation of edge S by imposing a permutation invariant function on every row of $\boldsymbol{P}^S_{(q)}$ (figure 2.4). For instance, to predict the link in *graph*, i.e., $k = 2$, SEAL considered a specific type of node labeling by tracking distances of a node to the target two nodes and showed superior performance over existing methods [47], [53]. Li et al, further generalized such a definition to the case with S of arbitrary sizes but they still work on graphs instead of hypergraphs [54]. Mathematically, Li et al. characterized the expressive power of the obtained GNNs which solve the edge-level ambiguity issue previously observed in graphs [55]. Motivated by these works, we propose f_{SN}, which integrates structural feature $\boldsymbol{P}^S_{(q)}$ by using a bipartite graph neural network. Unlike k-size edge representation, the affinity matrix $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times \|S\|}$ has varied dimension depending on the size of hyperedge, i.e. $\|S\|$. To construct a uniformed input, we first process $\boldsymbol{P}^S_{(q)}$ by using a set neural network (setNN) developed in precisely deepsets [56]. Specifically, by treating each row of $\boldsymbol{P}^S_{(q)}$ as an individual set vector, setNN acted as a permutation invariant function to standardize the node-wise feature into a feature matrix of a fixed dimension d. This feature matrix is then served as the input node features to initiate the message passing in the bipartite neural network, i.e.,

$$\boldsymbol{X}^0_{V^S_{(q)}} = f_{setNN}(\boldsymbol{P}^S_{(q)}), \boldsymbol{X}^0_{V^S_{(q)}} \in \mathbb{R}^{\|V^S_{(q)}\| \times d}$$

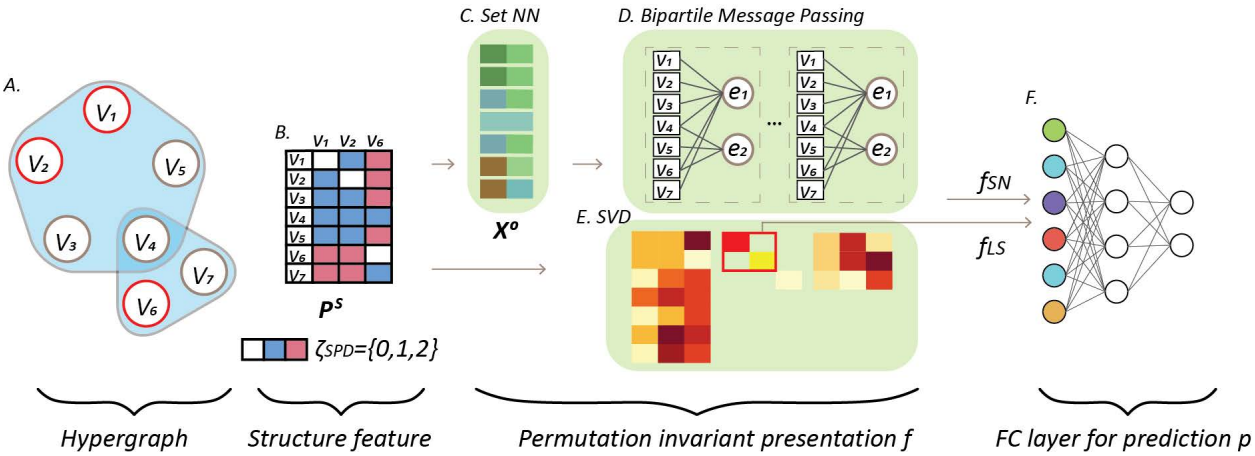

Figure 2.5. The overall framework of HIGNN

$$\boldsymbol{X}^{l+1}_{E^S_{(q)}} = \sigma((\boldsymbol{H}^S_{(q)})^T \boldsymbol{X}^l_{V^S_{(q)}} \boldsymbol{D}^{-1}_{E^S_{(q)}} \boldsymbol{W}^l_E)$$

$$\boldsymbol{X}^{l+1}_{V^S_{(q)}} = \sigma(\boldsymbol{H}^S_{(q)} \boldsymbol{X}^l_{E^S_{(q)}} \boldsymbol{W}^l_V),$$

Such that, $f_{SN}(v \| v \in S) = \boldsymbol{X}_{V^S_{(q)}v:}$ and $f_{SN}(S)$ also follows the aggregation form, i.e.

$$f_{SN}(S) = AGG(f_{SN}(v \| v \in S)).$$

2.3.2 Spectrum of the structure feature matrix

The utilization of $\boldsymbol{P}^S$ in f_{SN} alleviates the edge-level ambiguity in representing hyperedge with aggregating based graph neural network. However, such integration omits the matrix structure of $\boldsymbol{P}^S$ and does not represent it in full. In sight of this, we propose f_{LS}, a function based on the singular values $\boldsymbol{P}^S_{(q)}$, i.e., the spectrum of the subgraph $\boldsymbol{H}^S_{(q)}$. The rationale is that singular values reflects the low rank property of $\boldsymbol{P}^S_{(q)}$, i.e., the topological structure of $\boldsymbol{H}^S_{(q)}$. Intuitively, the affinity matrix with a higher low-rankness suggests the nodes in S are of higher topological similarity. As the singular value decomposition is invariant to row and column wise shuffles, f_{LS} based on the singular values of $\boldsymbol{P}^S_{(q)}$ is also isomorphic invariant. However, f_{LS} does not follow the aggregation form. Also, to cope with hyperedge of varied

sizes, i.e., $\|S\| \geq 2$, currently f_{LS} only takes the two largest singular value into account, the ratio between is sufficient to characterize the level of low-rankness of $\boldsymbol{P}^S_{(q)}$,

$$f_{LS}(S, \boldsymbol{P}^S_{(q)}) = f(\boldsymbol{\Sigma}_{11}, \boldsymbol{\Sigma}_{22}), \quad \boldsymbol{P}^S_{(q)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

Together, we present the **HIGNN** framework (Figure 2.5) that integrates two permutation invariant functions, the Structure Neural network f_{SN} and Local Spectrum information of structural feature matrix f_{LS}. Follows by a dense layer as prediction function p, HIGNN determines the existence of the hyperedges by

$$p(concat(f_{SN}(S, \boldsymbol{P}^S_{(q)}, \boldsymbol{H}^S_{(q)}), f_{LS}(S, \boldsymbol{P}^S_{(q)})))$$

2.4 Experiments

In this section, we briefly introduce the experimental setup and evaluate the performance of HIGNN with state-of-the-art GNN-based and structural heuristic based models.

2.4.1 Datasets

Following [35], twelve hypergraph datasets[1] were utilized in our evaluation.

- **DAWN**: Patient drug use in emergency room visits.
- **Email-Eu**: Emails with multiple email addresses.
- **Email-Enron**: Emails with multiple email addresses.
- **NDC-classes**: Drugs with multiple classification labels.
- **NDC-substances**: Drugs consist of multiple substances.
- **Threads-ask**: Users Q&A on askubuntu.com
- **Threads-math**: Users Q&A on math.stackexchange.com.

[1]↑All data are retrieved from www.cs.cornell.edu/∼arb/data/

- **Tags-ask**: Questions with tags on askubuntu.com.
- **Tags-math**: Questions with tags on math.stackexchange.com.
- **Contact-High**: People in contact at high school.
- **Contact-Primary**: People in contact at primary school.
- **Congress**: Legislative bill with the sponsor representative.

Table 2.1. Dataset statistics, for edge and node degree, we report the mean value along with its standard derivation.

Statistic \ Data	DAWN	email-Eu	email-Enron	NDC-classes	NDC-substances	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
No. Edge	138742	24399	1457	1047	6264	115987	535323	145053	169259	7818	12704	83105
No. Node	2290	979	143	1149	3438	90054	153806	3031	1627	327	242	1718
Edge degree	3.987(2.207)	3.488(2.849)	3.085(1.942)	6.115(4.839)	7.964(5.910)	2.309(0.635)	2.610(0.933)	3.427(0.992)	3.497(0.945)	2.327(0.531)	2.419(0.550)	8.812(6.853)
Node degree	241.554(1055.753)	86.935(114.531)	31.434(24.058)	5.572(15.708)	14.510(42.724)	2.974(21.754)	9.087(91.405)	164.558(606.784)	363.801(1040.086)	55.633(27.063)	126.979(55.148)	426.261(475.654)

For each dataset, we only keep the hyperedges containing at least two nodes. Detailed statistics of the twelve datasets are summarized in table 2.1. We also report the mean value of edge and node degree along with its standard derivation. We argue these datasets represent different scenarios in hypergraphs, including sparse (NDC-classes, threads-ask, threads-math), medium (NDC-substance, email-Enron, contact-high), and dense (DAWN, email-Eu, tags-ask, tags-math, contact-primary, congress) hypergraphs. We believe these datasets form a comprehensive benchmark set to will evaluate the performance and robustness of each model. Following [35], for each dataset, we generate 5 times negative hyperedges to the real ones as negative training data.

2.4.2 Benchmark with GNN-based models

Our first baseline method HGNN [32], [34] utilizing the incidence matrix $\boldsymbol{H}$ to replace $\boldsymbol{A}$ in the representation learning function , i.e.

$$\boldsymbol{X}^{l+1} = \sigma(\boldsymbol{D}_v^{-1/2}\boldsymbol{H}\boldsymbol{W}_E^l\boldsymbol{D}_{\mathrm{e}}^{-1}\boldsymbol{H}^T D_v^{-1/2}\boldsymbol{X}^l\boldsymbol{W}_V^l)$$

. This approach along with its variants could be regarded as a weighted clique expansion, i.e, $\boldsymbol{H}\boldsymbol{W}_E\boldsymbol{D}_{\mathrm{e}}^{-1}\boldsymbol{H}^T \approx \boldsymbol{W'}\boldsymbol{A}$, where $\boldsymbol{W'}$ is a weighting matrix, are expected to be affected by

both ambiguities. As introduced previously, the second baseline method employes relational graph neural network on node-edge bipartite expansion ($\boldsymbol{H}$RGCN) to offset node-level ambiguity [35], but not edge-level ambiguity. The third baseline, NHP, also takes the general form bipartite graph neural network but applies an additional scoring layer to preserve the higher order properties for hyperedge prediction [44]. Most recently, as our fourth baseline, Srinivasan et al developed a hyperedge family (FamilySet) based representation learning function. In addition to above methods, FamilySet updates node-/edge-wise embeddings with their nearby nodes/edges, whose representation function follows [35]

$$\boldsymbol{X}_E^{l+1} = \sigma(concat(\boldsymbol{A}_E\boldsymbol{X}_E^l, \boldsymbol{H}^T\boldsymbol{X}_V^l)\boldsymbol{W}_E^l)$$

$$\boldsymbol{X}_V^{l+1} = \sigma(concat(\boldsymbol{A}_V\boldsymbol{X}_V^l, \boldsymbol{H}\boldsymbol{X}_E^l)\boldsymbol{W}_V^l)$$

, where $\boldsymbol{A}_V$ is the clique expansion that records nearby nodes information and *concat* represents concatenation. $\boldsymbol{A}_E$ is the line graph for connected hyperedges [57], where $\boldsymbol{A}_{E_{\text{ij}}} = 1$ if $\exists v \in V$ s.t $v \in S_{\text{i}}, v \in S_{\text{j}}$, otherwise $\boldsymbol{A}_{E_{\text{ij}}} = 0$. The adaptation of clique expansion and line graph is expected to avoid node-level ambiguity and partially edge-level ambiguity. For our fifth baseline, we access method that deal with edge-level ambiguity but are affected by node-level ambiguity. A series of work try to amend such differences in edge presentation by adding additional affinity labels to each nodes[47], [53]–[55]. Among them, SEAL is the SOTA algorithm in utilizing structure feature as node label for link prediction[47]. To adopt SEAL in hyperedge prediction, we propose setSEAL that also take the output of deepsets, i.e., we change the labeling method in SEAL with $\boldsymbol{X}_{(q)}^0$ as node feature for graph neural network. SetSEAL is expected to be affected by node-level ambiguity. For our model HIGNN, we also compared the performance with or without f_{LS} to illustrate the necessity to add the spectrum information.

Following [35], using 5-fold cross validation, we report the mean and standard deviation of the F1-scores for GNN-based methods in Table 2.2. SetSEAL showed better performance than other baseline methods, indicating the more profound impact of edge-level ambiguity over node-level ambiguity. By tackling both node- and edge-level ambiguity, HIGNN on

Table 2.2. Model performance comparison on F1 score for the comparison of GNN-based methods

Methods \ Data	DAWN	email-Eu	email-Enron	NDC-classes	NDC-substances	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
HGNN	0.624(0.010)	0.664(0.003)	0.618(0.032)	0.614(0.005)	0.421(0.014)	0.425(0.007)	0.453(0.007)	0.545(0.005)	0.599(0.009)	0.759(0.030)	0.645(0.031)	0.412(0.003)
HRGCN	0.634(0.003)	0.661(0.006)	0.599(0.040)	0.676(0.049)	0.525(0.006)	0.464(0.010)	0.487(0.006)	0.545(0.006)	0.572(0.003)	0.739(0.012)	0.669(0.012)	0.544(0.004)
NHP	0.667(0.000)	0.668(0.002)	0.668(0.001)	0.669(0.003)	0.669(0.002)	0.670(0.003)	**0.669(0.002)**	0.669(0.002)	0.668(0.002)	0.671(0.003)	0.668(0.001)	0.667(0.001)
FamilySet	0.677(0.004)	0.687(0.002)	0.685(0.016)	0.768(0.004)	0.512(0.032)	0.605(0.002)	0.586(0.002)	0.605(0.002)	0.642(0.006)	0.786(0.033)	0.716(0.034)	0.566(0.011)
SetSEAL	0.814(0.013)	0.758(0.011)	0.667(0.032)	0.822(0.015)	0.868(0.019)	0.581(0.015)	0.483(0.021)	0.798(0.018)	0.833(0.015)	0.783(0.023)	0.772(0.016)	0.777(0.073)
HIGNN(f_{SN})	**0.840(0.012)**	0.780(0.010)	**0.793(0.007)**	0.880(0.021)	0.914(0.007)	0.623(0.024)	0.627(0.018)	**0.823(0.009)**	**0.869(0.006)**	**0.832(0.003)**	0.823(0.003)	**0.893(0.010)**
HIGNN(f_{SN}, f_{LS})	**0.838(0.010)**	**0.785(0.011)**	**0.793(0.016)**	**0.896(0.020)**	**0.918(0.006)**	**0.714(0.019)**	**0.654(0.027)**	**0.822(0.012)**	**0.869(0.006)**	**0.832(0.009)**	**0.834(0.002)**	**0.893(0.008)**

Table 2.3. Model performance comparison on AUC score in predicting 2-nodes hyperedges.

Methods \ Data	DAWN	email-Eu	email-Enron	NDC-classes	NDC-substances	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
GM	0.557(0.007)	0.630(0.010)	0.769(0.013)	0.606(0.015)	0.660(0.010)	0.500(0.000)	0.500(0.000)	0.540(0.008)	0.554(0.012)	0.616(0.003)	0.656(0.006)	0.734(0.015)
HM	0.609(0.011)	0.700(0.015)	0.830(0.011)	0.692(0.028)	0.720(0.012)	0.501(0.001)	0.501(0.000)	0.594(0.013)	0.629(0.019)	0.706(0.004)	0.730(0.006)	0.811(0.014)
AM	0.609(0.011)	0.700(0.015)	0.830(0.011)	0.692(0.028)	0.720(0.012)	0.501(0.001)	0.501(0.000)	0.594(0.013)	0.629(0.019)	0.706(0.004)	0.730(0.006)	0.811(0.014)
CN	0.783(0.023)	0.866(0.011)	0.855(0.017)	0.675(0.025)	0.714(0.018)	0.504(0.002)	0.504(0.002)	0.813(0.026)	0.884(0.012)	0.930(0.005)	0.864(0.004)	0.904(0.009)
JC	0.759(0.024)	0.867(0.011)	0.882(0.015)	0.682(0.026)	0.732(0.022)	0.496(0.002)	0.496(0.002)	0.770(0.027)	0.841(0.010)	0.930(0.004)	0.891(0.001)	0.915(0.007)
AA	0.786(0.023)	0.870(0.011)	0.869(0.015)	0.678(0.026)	0.728(0.020)	0.504(0.002)	0.504(0.002)	0.816(0.026)	0.886(0.012)	0.932(0.005)	0.871(0.004)	0.908(0.008)
HIGNN	0.838(0.017)	0.887(0.008)	0.902(0.014)	0.828(0.015)	0.920(0.015)	0.850(0.031)	0.828(0.045)	0.873(0.015)	0.887(0.015)	0.995(0.001)	0.897(0.002)	0.911(0.009)

Table 2.4. Model performance comparison on AUC score in predicting 3-nodes hyperedges.

Methods \ Data	DAWN	email-Eu	email-Enron	NDC-classes	NDC-substances	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
GM	0.896(0.008)	0.935(0.008)	0.969(0.010)	0.714(0.032)	0.887(0.017)	0.500(0.001)	0.504(0.004)	0.788(0.010)	0.877(0.006)	0.977(0.004)	**0.977(0.003)**	0.891(0.007)
HM	0.924(0.010)	0.951(0.004)	0.793(0.014)	0.794(0.026)	0.933(0.011)	0.521(0.005)	0.524(0.005)	0.877(0.006)	0.927(0.003)	0.834(0.009)	0.729(0.010)	0.875(0.008)
AM	0.943(0.010)	0.970(0.004)	**0.973(0.006)**	0.795(0.025)	0.940(0.012)	0.521(0.005)	0.524(0.005)	0.883(0.005)	0.940(0.003)	**0.989(0.002)**	**0.977(0.003)**	0.938(0.011)
CN	0.935(0.010)	0.948(0.005)	0.944(0.008)	0.716(0.029)	0.863(0.015)	0.524(0.008)	0.527(0.004)	0.875(0.008)	0.927(0.005)	0.973(0.005)	0.924(0.003)	0.967(0.009)
JC	0.918(0.011)	0.942(0.006)	0.949(0.007)	0.716(0.029)	0.866(0.017)	0.524(0.008)	0.527(0.004)	0.834(0.008)	**0.969(0.004)**	0.970(0.006)	0.941(0.002)	**0.969(0.008)**
AA	0.937(0.010)	0.949(0.005)	0.949(0.007)	0.716(0.029)	0.872(0.015)	0.524(0.008)	0.527(0.004)	0.877(0.008)	0.929(0.005)	0.974(0.005)	0.928(0.003)	**0.969(0.009)**
HIGNN	**0.971(0.004)**	**0.984(0.006)**	**0.982(0.006)**	**0.869(0.006)**	**0.973(0.009)**	**0.881(0.036)**	**0.889(0.011)**	**0.960(0.007)**	**0.978(0.004)**	**0.995(0.001)**	**0.980(0.001)**	**0.971(0.006)**

Table 2.5. Model performance comparison on AUC score in predicting 4-nodes hyperedges.

Methods \ Data	DAWN	email-Eu	email-Enron	NDC-classes	NDC-substances	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
GM	$0.977(0.006)_3$	$0.977(0.009)_3$	$0.982(0.016)_4$	$0.880(0.053)_2$	$0.916(0.016)_2$	$0.519(0.011)_2$	$0.514(0.005)_2$	$0.919(0.012)_3$	$0.969(0.005)_3$	$1.000(0.000)_2$	$0.988(0.001)_2$	$0.928(0.021)_3$
HM	$0.919(0.009)_2$	$0.881(0.026)_4$	$0.943(0.027)_3$	$0.933(0.029)_3$	$0.944(0.012)_2$	$0.565(0.029)_3$	$0.551(0.013)_3$	$0.952(0.004)_3$	$0.961(0.002)_3$	$0.980(0.017)_3$	$0.965(0.008)_4$	$0.742(0.038)_4$
AM	$0.987(0.005)_2$	$0.991(0.007)_3$	$0.983(0.012)_4$	$0.939(0.028)_3$	$0.952(0.014)_2$	$0.565(0.029)_2$	$0.551(0.013)_2$	$0.970(0.003)_3$	$0.989(0.003)_3$	$1.000(0.000)_3$	$0.998(0.001)_3$	$0.963(0.008)_3$
CN	$0.967(0.006)_3$	$0.962(0.010)_4$	$0.949(0.021)_3$	$0.725(0.054)_3$	$0.877(0.014)_2$	$0.540(0.005)_2$	$0.533(0.008)_2$	$0.902(0.013)_3$	$0.934(0.011)_3$	$0.988(0.004)_3$	$0.946(0.014)_3$	$0.971(0.010)_3$
JC	$0.955(0.006)_4$	$0.958(0.010)_4$	$0.947(0.017)_4$	$0.725(0.054)_4$	$0.875(0.018)_2$	$0.540(0.005)_2$	$0.533(0.008)_2$	$0.862(0.010)_3$	$0.905(0.014)_3$	$0.984(0.984)_3$	$0.942(0.013)_3$	$0.974(0.008)_4$
AA	$0.968(0.007)_3$	$0.963(0.010)_3$	$0.953(0.021)_3$	$0.726(0.054)_4$	$0.882(0.015)_2$	$0.540(0.005)_2$	$0.533(0.008)_2$	$0.904(0.013)_3$	$0.936(0.011)_3$	$0.987(0.004)_3$	$0.947(0.013)_3$	$0.972(0.010)_3$
HIGNN	0.988(0.004)	0.992(0.005)	0.994(0.004)	0.942(0.028)	0.984(0.005)	0.885(0.030)	0.916(0.013)	0.984(0.003)	0.994(0.001)	1.000(0.000)	0.994(0.005)	0.984(0.005)

average achieves better performance increase over all baseline methods. Compared with HIGNN(f_{SN}), adding the local spectrum information, namely, HIGNN(f_{SN}, f_{LS}), further increases model performance in some datasets while maintained the same performance in others.

2.4.3 Benchmark with heuristic models

Structural heuristic based methods also showed comparative performance in hyperedge prediction. Whereas not following the aggregation form in lemma 2.1.3, most of the methods are free from edge-level ambiguity. Shown in a recent study [43], node-level ambiguity could also be alleviated by involving higher order heuristics. However, the introduction of higher order information restrict heuristic methods to predict specific k-size hyperedges. Following [43], we benchmark HIGNN with heuristic methods (Geometric Mean (GM), Harmonic Mean (HM), Arithmetic mean (AM), Common neighbors (CN), Jaccard coefficient (JC), and ∘ Adamic-Adar index (AA)) in [43] on 2-, 3- and 4-nodes hyperedges tasks, and report the mean and standard variance of AUC of ROC in table 2.3, 2.4 and 2.5. Though in some cases, heuristic methods showed comparable performance, HIGNN still manage to deliver a stable and better results in all three prediction scenarioes.

2.5 Ablation study

We proposed the theoretical driven HIGNN framework that tackles the ambiguities issues in hyperedge prediction task. Thus, our key ablation design was to test the performance differences between our method that consider the ambiguities issue and baseline methods without such considerations. Specially, the baseline methods HGNN, setSEAL and structural heuristic methods suffer node-level ambiguity while HRGCN, NHP, and familyset suffer edge-level ambiguity. The performance increase of HIGNN over the baseline methods were listed in table 2.2, 2.3, 2.4 and 2.5, which validated our theoretical analysis and advocated the necessity in considering the ambiguities. We also evaluate the impact of spectrum information that illustrates its functionality for the task (table 2.2). Nonetheless, HIGNN is a complicated framework that utilized many components each serving different purposes.

Beside the general comparison, in the rest of the section, we evaluate the other main components in the HIGNN framework.

2.5.1 Set neural network frameworks

The hyperedge-specific node structure feature $\boldsymbol{P}^S_{(q)}$ characterizes the within hypergraph dependency of each node. By implementing structure feature within the bipartite graph neural network, f_{SN} collectively tackles both edge- and node-level ambiguities. One key component or requirement for f_{SN} is that $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V_{(q)}\| \times \|S\|}$ have different dimensions for different hyperedge S. However, the bipartite graph neural network requires a fixed feature matrix $\boldsymbol{X}^0_{V^S_{(q)}}$ as the input. To fill this gap, we utilize set neural network (setNN), precisely deepsets [56] to standardize $\boldsymbol{P}^S_{(q)} \in \mathbb{R}^{\|V^S_{(q)}\| \times \|S\|}$ into $\boldsymbol{X}^0_{V^S_{(q)}} \in \mathbb{R}^{\|V^S_{(q)}\| \times d}$, i.e., for each row of $\boldsymbol{P}^S_{(q)_{\text{i:}}}$, we transfer it to a uniformed vector $\boldsymbol{X}^0_{V^S_{(q)_{\text{i:}}}}$.

We also noted such information is in overlap with the spectrum information (f_{LS}). Since the singular values represent the low rank property of the affinity matrix, f_{LS} characterizes the local topological characteristics of a node set when predicting if they form a hyperedge. As one of our contribution, we integrated the spectrum information in HIGNN in predicting hyperedges. Showed in previous section, our experiments demonstrated introducing f_{LS} consistently improved the model performance on all benchmark datasets, especially when GNN-based models did not performed well.

To thoroughly investigate the power of set neural network and also justify the necessity of f_{LS}, we ask if the spectrum information could be directly learned by the set neural network models, i.e., if we change the setting of our set neural network model in f_{SN}, would the enhanced f_{SN} alone achieves the same performance as HIGNN($f_{SN} + f_{LS}$). To this end, we replaced the deepsets module with set transformer [58], which is capable to learn higher order interactions within the set. Such property corresponds well with the joint interaction between hyperedge with its local environment. To evaluate, we constructed four scenarios of different setNN with and without f_{LS}: 1) deepsets, 2) deepsets f_{LS}, 3) settransformer, 4) settransformer f_{LS}.

Here we report the AUC results of these scenarios in predicting hyperedge in table 2.6. For both setNN, the inclusion of f_{LS} maintains or strengthens the overall performance. Esepecially in the case that f_{SN} alone could not deliver a satisfactory performance (threads-ask, threads-math). We next focus on the comparison between settransformer and deepsets f_{LS}. We regard the former one as learned higher order interactions and the later one as retrieved higher order interactions. In most of the cases, deepsets plus f_{LS} outperforms settransformer. This result suggest that to directly learn the higher order interaction by changing neural network structure is unlikely to match the performance of the integrating the spectrum information f_{LS}. We then integrate set transformer with f_{LS} to test whether the combined effect could achieve better performance. Surprisingly, in many cases, these two information contradict with each other, which resulted a poor performance compared with settransformer or deepsets f_{LS} (DAWN, email-Eu, threads-ask, threads-math). Also settransformer has higher computational cost compared with deepsets. Subsequently, we fail to finish training the model for NDC-class dataset in three days and did not report results here. Overall, this experiment revealed that higher order information is a nontrivial task for f_{SN}. Even we include more advanced framework, like set transformer, the information learned may not necessarily reflect the true property of the hyperedge. This result further advocate the necessity of f_{LS} for the representation of hyperedge.

Table 2.6. F1 results of different set neural network scenarios.

Method \ Data	DAWN	email-Eu	email-Enron	NDC-class	NDC-substance	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
deepsets	0.972(0.004)	0.952(0.004)	0.956(0.003)	0.973(0.008)	0.988(0.002)	0.850(0.026)	0.876(0.007)	0.967(0.001)	0.981(0.002)	0.965(0.002)	0.940(0.002)	0.988(0.002)
deepsets f_{LS}	0.971(0.03)	0.954(0.005)	0.954(0.006)	0.973(0.009)	0.989(0.002)	0.895(0.033)	0.886(0.014)	0.966(0.003)	0.981(0.002)	0.964(0.003)	0.940(0.002)	0.988(0.002)
settransformer	0.970(0.003)	0.951(0.006)	0.957(0.004)	0.979(0.005)	NA	0.866(0.019)	0.827(0.044)	0.966(0.001)	0.981(0.002)	0.964(0.003)	0.940(0.002)	0.988(0.001)
settransformer f_{LS}	0.968(0.004)	0.950(0.005)	0.960(0.003)	0.978(0.006)	NA	0.863(0.021)	0.805(0.040)	0.966(0.002)	0.980(0.001)	0.965(0.002)	0.940(0.002)	0.988(0.001)

2.5.2 Pooling methods in set neural network

To deal with different size of hyperedge local structure, we introduce set neural network to compress and standardize such information for each nodes in the hyperedge local environment. Because of permutation invariant property of S, any row-wise operation on $\boldsymbol{P}^S_{(q)}$ should also be permutation invariant. SetNN fits this property perfectly as it regards $\boldsymbol{P}^S_{(q)_{\mathrm{i:}}}$ as a set rather than an ordered vector. Moreover, most setNN models like deepsets are very efficient

Table 2.7. AUC results of different pooling methods for set neural network.

Method \ Data	DAWN	email-Eu	email-Enron	NDC-class	NDC-substance	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
max	0.888(0.008)	0.862(0.019)	0.952(0.004)	0.865(0.029)	0.819(0.047)	0.875(0.010)	0.893(0.013)	0.885(0.024)	0.943(0.016)	0.961(0.003)	0.937(0.002)	0.985(0.002)
mean	0.936(0.021)	0.930(0.013)	0.940(0.004)	0.930(0.013)	0.947(0.015)	0.844(0.051)	0.892(0.008)	0.877(0.025)	0.932(0.019)	0.964(0.002)	0.939(0.002)	0.986(0.002)
sum	0.956(0.011)	0.940(0.004)	0.956(0.003)	0.957(0.010)	0.964(0.012)	0.906(0.013)	0.888(0.010)	0.941(0.014)	0.969(0.007)	0.965(0.002)	0.940(0.002)	0.988(0.002)

Table 2.8. AUC results of different normalization scenarios for different hypergraph data.

Method \ Data	DAWN	email-Eu	email-Enron	NDC-class	NDC-substance	threads-ask	threads-math	tags-ask	tags-math	contact-high	contact-primary	congress
Scenario 1	0.945(0.006)	0.930(0.013)	0.952(0.005)	0.933(0.013)	0.967(0.007)	0.920(0.008)	0.893(0.007)	0.945(0.011)	0.959(0.013)	0.964(0.001)	0.939(0.002)	0.988(0.002)
Scenario 2	0.956(0.011)	0.940(0.004)	0.956(0.003)	0.957(0.010)	0.964(0.012)	0.906(0.013)	0.888(0.010)	0.941(0.014)	0.969(0.007)	0.965(0.002)	0.940(0.002)	0.988(0.002)
Scenario 3	0.941(0.006)	0.939(0.005)	0.952(0.005)	0.936(0.014)	0.956(0.012)	0.867(0.041)	0.880(0.013)	0.914(0.026)	0.943(0.026)	0.964(0.002)	0.939(0.002)	0.988(0.002)
Scenario 4	0.950(0.006)	0.939(0.005)	0.946(0.005)	0.933(0.019)	0.960(0.019)	0.857(0.049)	0.874(0.017)	0.928(0.017)	0.967(0.008)	0.965(0.002)	0.939(0.002)	0.988(0.003)

to train and apply. One important parameter of setNN is the choice of pooling methods. Theoretically, any permutation invariant pooling methods (max-/mean-/sum-pooling) would maintain the permutation invariant property of setNN [56]. As for the case of hyperedge prediction, we recommend using sum-pooling which could reflect the edge–size information better than max or mean pooling. We also report their differences in table 2.7, as expected, sum-pooling enjoys better and stable performance compared with max and mean pooling.

2.5.3 Normalization on bipartite graph neural network

The non-linear activation function in bipatite graph neural network captures the non-linear dependency of hyperedge with different edge-degrees, which introduce additional flexibility to the edge-embedding $\boldsymbol{X}_E$ than the clique expansion based GNNs. Bipartite graph neural network is capable for representing hyperedge with different edge size. One important step in the bipartite graph neural network is to normalize node and edge embedding by their degree or size. Essentially, such normalizations balance the local topological characteristics and degree bias in embedding a single node or edge. Noted, an over-normalization could eliminate contextual meaningful topological characteristics while none or less normalization causes a degree or size bias, i.e., the difference of embedding of nodes and edges is not in agreement with its topological characteristics but heavily influenced by its node degree or edge size. To test the impact of different levels of normalization on the model performance, we test the following four normalization scenarios:

Scenario 1: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T \boldsymbol{X}_V^l \boldsymbol{W}_E^l)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H} \boldsymbol{X}_E^l \boldsymbol{W}_V^l)$

Scenario 2: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T \boldsymbol{X}_V^l \boldsymbol{D}_E^{-1} \boldsymbol{W}_E)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{H} \boldsymbol{X}_E^l \boldsymbol{W}_V)$

Scenario 3: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T \boldsymbol{X}_V^l \boldsymbol{W}_E)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{D}_V^{-1/2} \boldsymbol{H} \boldsymbol{X}_E^l \boldsymbol{D}_V^{-1/2} \boldsymbol{W}_V)$

Scenario 4: $\boldsymbol{X}_E^{l+1} = \sigma(\boldsymbol{H}^T \boldsymbol{X}_V^l \boldsymbol{D}_E^{-1} \boldsymbol{W}_E)$, $\boldsymbol{X}_V^{l+1} = \sigma(\boldsymbol{D}_V^{-1/2} \boldsymbol{H} \boldsymbol{X}_E^l \boldsymbol{D}_V^{-1/2} \boldsymbol{W}_V)$

Specifically, scenario 1 corresponds to none normalization on both node and edge, which relies on $\boldsymbol{W}_E$ and $\boldsymbol{W}_V$ to compensate the degree impact. Scenario 2 and 3 that correspond to conducting the normalization on only edge-side or node-side, respectively. And scenario 4 normalizes both edge- and node-side. We compared the impact of the four normalization scenarios on HIGNN on the benchmark datasets by fixing all other parameters.

We report the AUC results of different normalization scenarios for different hypergraph data in table 2.8. Compared with none (scenario 1), node-side (scenario 3) and two-side normalization (scenario 4), edge-side normalization (scenario 2) consistently shows a better performance in all the eight benchmark datasets. Empirically, we argue that the one-side normalization would better balance the information loss and degree bias, such that it outperforms scenario 1 and 4. For the better performance of scenario 2 than scenario 3, we speculate a major reason is that we utilize the node-embedding rather than edge embedding to predict hyperedge. By omitting the normalization on node-side, the pipeline would take advantage of node embedding difference for a better prediction. Such that, in HIGNN, we utilize the edge-size normalization scheme for the updating of node and edge embedding. We also noticed other works that introduce latent parameters to control the normalization [52]. This framework could certainly integrated in further improvement of HIGNN.

2.5.4 Complexity

To tackle the ambiguities problems in the hyperedge prediction task, HIGNN utilized different components for specific task. Such integration adds more computational burdens to the framework, wherein the extraction of structure features plays the key role. Theoretically, retrieving the structure feature of a hyperedge requiring the transverse of whole graph, which adds an order of computation and makes it not feasible for large datasets. Though the extra computation is inevitable, we could reduce the cost in several ways. First, not all the information in the structure feature is important for the representation of hyperedge,

especially for the nodes that are far away from the hyperedge. Restraining the structure information within local neighbors of the hyperedge, i.e., local structure information, is sufficient to restore most of the information for the model generalization and also save great computation by limiting the transverse within q-hop. Empirically, without loss of generalization power [49], for dense hypergraphs where each hyperedge has a large number of nodes within their q-hop neighborhood, sampling the neighborhood is also a feasible way to reduce the computational cost. Noted, the added computation does not prevent the application of HIGNN on very large hypergraph (exceed the memory of GPU) as the framework fits very well with mini-batch training scheme (extracts subgraphs for target nodes set).

2.6 Predicting DNA interactions

In mammalian cells, the 3D genome organization is proven to function in many biological processes, and the higher-order chromatin organizations are frequently linked to long-distance gene regulation that could control development and cell fate commitment [59], [60]. As introduced earlier, genetic interactions are higher-order connections that involve multiple entities, such as gene, enhancer, promoter, et al [5], [36], [37]. Current methods for analyzing the genome organization data are still limited to pair-wise connections, while efficient tools/methods are lacking for the exploration of higher order interactions in 3D genome data [18], [19], [61].

As a proof of concept study, here we utilize HIGNN to predict the genome higher-order interactions (hyperedge) of mouse embryonic cells. We retrieve the 3D genome connection data from [18]. Similarly, we only keep the hyperedges that have at least two nodes, and constructed the negative training data by generating five negative hyperedges for each hyperedge observed. We first test whether our model could achieve consistent performance across different chromosomes. For each of the 17 autosomals in mouse genome, we randomly selected 5 autosomals to study the interactions, resulting in 85×85 pair-wise cross validations. We compared HIGNN with the strongest baseline method setSEAL and report the Area under ROC curve (AUC) in figure 2.6A. In general, HIGNN outperforms setSEAL across all the test conditions. More importantly, the performance of HIGNN is very stable, since it

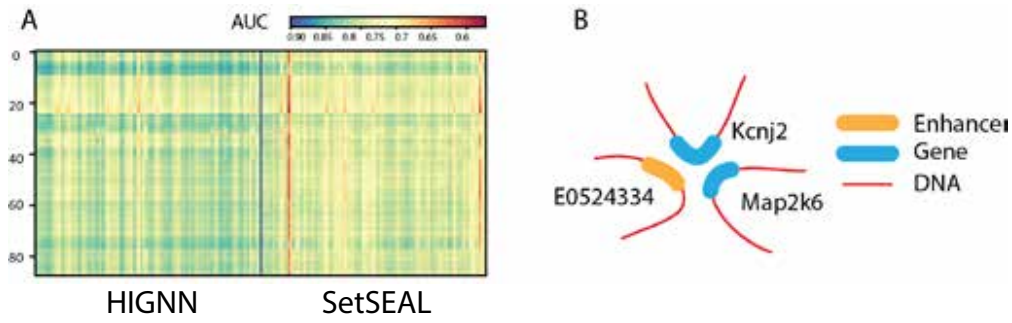

Figure 2.6. HIGNN gives plausible prediction of higher order genetic interaction.

did not show any bias towards specific chromosomes, unlike setSEAL on chromosome 2 and 17.

We then apply HIGNN to predict the 4-way genetic interactions in chromosome 11. One hyperedge corresponding to the interactions of the bin elements 5521, 5589, 5602, 5630, is predicted by HIGNN that is not captured by original assay. These bins are located within the same topological associated domain (TAD) [62]. Furthermore, we find genes Map2k6, Kcnj2 and enhancer E0524334 are located within 5521, 5589, 5602, respectively (figure 2.6B). The co-regulation in expression of Map2k6 and Kcnj2 has been experimentally reported in [46]. Together, these indicate that the co-regulation may be a result of the same enhancer. In summary, we demonstrated the reliability of HIGNN in predicting genetic higher-order interactions, as well as the potential of using hyperedge prediction to fully evaluate the effect of higher-order genetic interactions on gene expression.

2.7 Discussion

In this chapter, we formulate the genome interactions prediction task as hyperedge prediction task. Specifically, we mathematically discussed the ambiguity issues for current model in hyperedge structural representing tasks, i.e., node- and edge-level ambiguities. Motivated by such derivation and previous works we present HIGNN framework to predict higher-order interactions in hypergraph. In doing so, HIGNN utilizes bipartite graph neural network to avoid node-ambiguity caused by different arrangment of hyperedges and applying structure features to alleviate edge ambiguity introduced by aggregating based methods. Moreover,

HIGNN retrieved the spectrum information of the structure features, which reflects the joint interaction between the hyperedge and its local environment. Such information could not be easily learned under the framework of current graph neural network. As a result, HIGNN achieved better performance over most recent models.

Though HIGNN provides a plausible solution for hyperedge represenation tasks. There are still rooms to achieve the exact partition, i.e., $\Pi_f(S) \approx \Pi_I(S)$. For example, the structural features defined in this chapter is rooted from pair-wise affinity, the explainable power is in term limited for the clique expansion of hypergraph. Introducing higher-order structural features is likely to strength presentation as combining higher order heuristics improves prediction accuracy for empirical methods [43], [63]. We propose f_{LS} to reflect the low-rank property of $\boldsymbol{X}_S$ that we argue it captures (or partially) the joint interactions of hyperedge with its local environment. Currently, due to the hyperedge size difference, we only use top 2 singular value for f_{LS}. Such information retrieval comes with limited power resulting selective performance increase in certain datasets. Utilizing more singular value or a better way to retrieve the low rank information of $\boldsymbol{X}_S$ would also help the representation. Furthermore, we could also improves the explainable power by integrating advanced components like attention mechanism that already showed empirical success in hypergraph related tasks like node classification and recommendation tasks [45], [64], [65].

In the next chapter, we will shift our gear to scRNA-seq data, where we proposed a sophistically designed statistical model to identify the discrete gene expression states.

3. LTMG: A NOVEL STATISTICAL MODELING OF GENE EXPRESSION STATES IN SCRNA-SEQ DATA

Single-cell RNA sequencing (scRNA-seq) has gained extensive utilities in many fields, among which, the most important one is to investigate the heterogeneity and/or plasticity of cells within a complex tissue micro-environment and/or development process [24], [66], [67]. This has stimulated the design of a variety of methods specifically for single cells: modeling the expression distribution [68]–[70], differential expression analysis [71]–[76], cell clustering [77], [78], non-linear embedding based visualization [79], [80] and gene co-expression analysis [78], [81]. etc. Gene expression in a single cell is determined by the activation status of the gene's transcriptional regulators and the rate of metabolism of the mRNA molecule. In single cells, owing to the dynamic transcriptional regulatory signals, the observed expressions could span a wider spectrum, and exhibit a more distinct cellular modalities, compared with those observed on bulk cells [78]. In addition, the limited experimental resolution often results in a large number of expression values under detected, i.e. zero or lowly observed expressions, which are generally noted as 'dropout' events. How to decipher the gene expression multimodality hidden among the cells, and unravel them from the highly noisy background, forms a key challenge in accurate modeling and analyses of scRNA-seq data.

Clearly, all the analysis techniques for single cells RNA-Seq data including differential expression, cell clustering, dimension reduction, and gene co-expression, heavily depend on an accurate characterization of the single cell expression distribution. Currently, multiple statistical distributions have been used to model scRNA-Seq data [68], [69], [73], [74]. All the formulations consider a fixed distribution for zero or low expressions disregarding the dynamics of mRNA metabolism, and only the mean of expression level and proportion of the rest is maintained as target of interest. These methods warrant further considerations: (i) the diversity of transcriptional regulatory states among cells, as shown by the single molecular in situ hybridization (smFISH) data [82]–[84], would be wiped off with a simple mean statistics derived from non-zero expression values; (ii) some of the observed non-zero expressions could be a result of mRNA incompletely degraded, rather than expressions under certain active regulatory input, thus they should not be accounted as true expressions; (iii) zero-inflated

unimodal model has an over-simplified assumption for mRNA dynamics, particularly, the error distribution of the zero or low expressions are caused by different reasons, negligence of this may eventually lead to a biased inference for the multi-modality encoded by the expressions on the higher end.

To account for the dynamics of mRNA metabolism, transcriptional regulatory states as well as technology bias contributing to single cell expressions, we developed a novel left truncated mixture Gaussian (LTMG) distribution that can effectively address the challenges above, from a systems biology point of view. The multiple left truncated Gaussian distributions correspond to heterogeneous gene expression states among cells, as an approximation of the gene's varied transcriptional regulation states. Truncation on the left of Gaussian distribution was introduced to specifically handle observed zero and low expressions in scRNA-seq data, caused by true zero expressions, 'dropout' events and low expressions resulted from incompletely metabolized mRNAs, respectively. Specifically, LTMG models the normalized expression profile (log CPM, count per million reads) of a gene across cells as a mixture Gaussian distribution with K peaks corresponding to suppressed expression (SE) state and active expression (AE) state(s). We introduced a latent cutoff to represent the lowest expression level that can be reliably detected under the current experimental resolution. Any observed expression values below the experimental resolution are modeled as left censored data in fitting the mixture Gaussian model. For each gene, LTMG conveniently assigns each single cell to one expression state by reducing the amount of discretization error to a level considered negligible, while the signal-to-noise ratio and the interpretability of the expression data are largely improved. Based on the LTMG model, a differential expression test, a co-regulation module detection and a cell clustering algorithm were further developed.

A systematic method validation was conducted with the following key results: (i) LTMG achieves the best goodness of fitting in 23 high quality data sets, compared with four commonly utilized multimodal models of scRNA-seq data; (ii) using a set of mRNA kinetic data, we confirmed the validity of treating a significant portion of the low but non-zero expressions as a result of incompletely degraded mRNA in LTMG, which should not be considered as true expressions under active regulations; (iii) on a cancer single cell RNA-seq data, we demonstrated that single cell groups defined by distinct gene expression states captured

by LTMG, are in good agreement with known sub cell types, i.e. exhausted CD8+T cell population and subclasses of fibroblast cells, in other words, the multi-modality setting in LTMG uncovers the heterogeneity among single cells; (iv) non-linear embedding and cell clustering based on LTMG discretized expression states produces more informative clusters. A user-friendly R package with all the key features of the LTMG model was released through https://github.com/clwan/LTMGSCA.

3.1 Methods

3.1.1 Mathematical model linking gene expression states in single cell to transcriptional regulation

A gene's expression in a mammalian cell is the result of the interactions between its DNA template and a collection of transcriptional regulatory inputs (TRIs) including: (i) transcriptional regulatory factors (TFs) (cis-regulation); (ii) miRNA or lncRNA; (iii) enhancer and super-enhancer and (iv) epigenetic regulatory signals [85], [86]. For a gene with N possible transcriptional regulation inputs, $[TRI_{\mathrm{i}}]\,, \mathrm{i} = 1, ..., N$, the probability of its promoter being bound by an RNA polymerase, P_b, which is proportional to the rate of its transcription, can be modeled by a Michaelis–Menten equation [87], [88].

$$
\begin{aligned}
P_b &= \frac{R_0 + \frac{R_1[TRI_1]}{K_1} + ... + \frac{R_N[TRI_N]}{K_N} + \frac{R_{1,2}[TRI_1][TRI_2]}{K_{1,2}} + ... + \frac{R_{1,2...,N}[TRI_1][TRI_2]...[TRI_N]}{K_{1,2,...,N}}}{1 + \frac{[TRI_1]}{K_1} + ... + \frac{[TRI_N]}{K_N} + \frac{[TRI_1][TRI_2]}{K_{1,2}} + ... + \frac{[TRI_1][TRI_2]...[TRI_N]}{K_{1,2,...,N}}} \\
&= \frac{\sum_{\Omega \in M\{1,...,N\}} \frac{R_\Omega}{K_\Omega} \prod_{\mathrm{i} \in \Omega [TRI_{\mathrm{i}}]}}{\sum_{\Omega \in M\{1,...,N\}} \frac{1}{K_\Omega} \prod_{\mathrm{i} \in \Omega [TRI_{\mathrm{i}}]}}
\end{aligned}
\tag{3.1}
$$

Where R_{i}, $[TRI_{\mathrm{i}}]$, K_{i} denote production rate, concentration and kinetic parameters associated with the ith TRI; $M\{1, ..., N\}$ is the power set of $\{1, ..., N\}$, R_Ω, K_Ω denote the production rate and kinetic parameters associated with the interactive effects of TRIs in Ω, where $\Omega \in M\{1, ..., N\}$. The set of active TRIs in a single cell fully determines the transcription rate of the gene, and thus its transcriptional regulatory state (TRS). Note that in a single cell each TRI can be rationally simplified to have two states: present or absent from

the DNA molecule, thus the TRI_{i} is a Boolean variable and equation 3.1 becomes a discrete function with at most $\|M(1,...,N)\| = 2^N$ values:

$$\begin{aligned} &P_b(Current\,TRS = \{TRI, \mathrm{i} \in \Omega\}) = \\ &P_b(\{[TRI_{\mathrm{i}}] \gg 0, [TRI_{\mathrm{j}}] = 0 \| \mathrm{i} \in \Omega,\, \mathrm{j} \notin \Omega, \Omega \in M\}) = R_\Omega \end{aligned} \tag{3.2}$$

Such discretization of gene's transcriptional rate greatly simplified the kinetic model and has achieved satisfactory performances in deriving the transcriptional regulatory dependency between the gene's expression state and its TRIs, which has been commonly utilized in thermodynamic modeling of transcriptional regulation [89]–[91]. For a mammalian cell, the total number of combinations of TRIs can be substantially large, especially considering the epi-genetic regulators [85]. However, the number of TRSs of a gene in a single cell RNA-seq experiment is always much smaller. The reason being: (i) the phenotypic diversity of the cells measured in one experiment is relatively small; (ii) local interactive effects among multiple TRIs are exerted on the same regulatory element [86] and (iii) some master repressors such as chromatin folding or certain TFs can dominate the regulation of the gene's expression [86].

Denote M^X as the set of all possible TRS of gene X and α_Ω^X as the probablity of sampling a cell with TRS Ω, $\Omega \in M^X$, from the cell population. By introducting a Gaussian error to the simplified model describe above, the probability density function of the transcriptional rate of X in a single cell can be modeled as a mixture Gaussian distribution:

$$f(P_b^X) = \sum_{\Omega \in M^X} \alpha_\Omega^X \frac{1}{\sqrt{2\pi\sigma_\Omega^X}} \mathrm{e}^{-\frac{(P_b^X - R_\Omega^X)^2}{2(\sigma_\Omega^X)^2}} \quad s.t. \sum_{\Omega \in M^X} \sigma_\Omega^X = 1 \tag{3.3}$$

where the mixing probability, mean and standard deviation, α_Ω^X, R_Ω^X and σ_Ω^X correspond to the frequency, transcription rate, and variance of the TRS Ω. Single cell RNA-seq measures the abundance of mature mRNA in cytosol, which is determined by the transcription and degradation rate of the mRNA. The gene expression pattern we eventually observe is mainly shaped by the (i) cytosol mRNA abundance, compounded with (ii) observation errors and (iii) experimental resolution. Based on several common transcriptional regulation models,

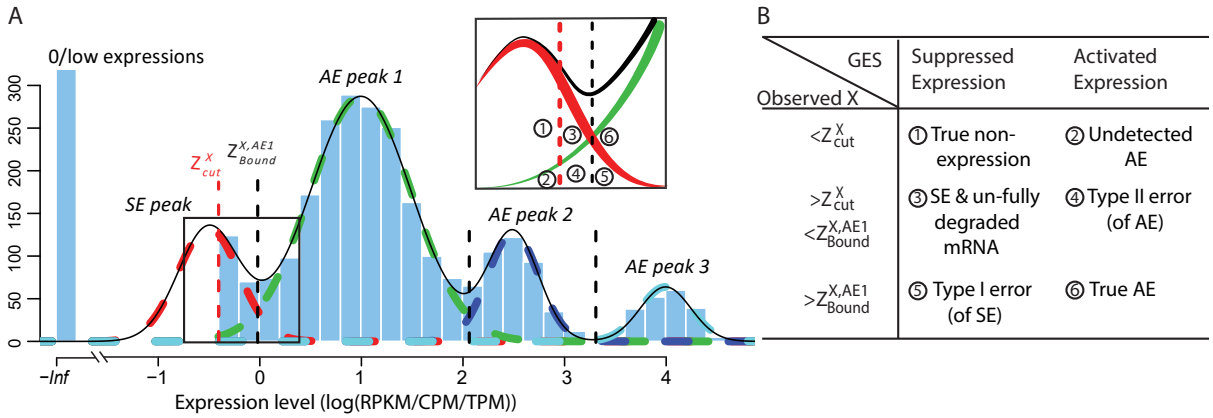

Figure 3.1. LTMG model for gene expression from scRNA-seq

including constant transcriptional regulatory input and transcriptional bursting [92], we extend the multimodality of transcription inputs and rates defined in (3.2) and (3.3) to the multimodality of observed mRNA abundance.

Denote $\hat{x}_{\mathrm{j}}, \mathrm{j} = 1, ..., N$ as the normalized gene expression (such as log CPM or TPM) of gene X in a scRNA-seq experiment with individual library constructed for N cells and measured with high sequencing depth. Based on the derivations above, we illustrated the relationship between the repertoire of the TRSs of X, multi-modality of mRNA abundance, and its observed gene expression profile in Figure 3.1A. A mixture Gaussian model is utilized to characterize the distribution of observed normalized gene expression level of X through multiple cells. Gene expressions falling into a same peak are considered to have the same gene expression state (GES), that share the same TRS or different TRS with a similar mean pattern; while the expressions falling into different peaks are more likely to have different TRSs. We index the Gaussian peaks by their means and denote the one with smallest mean as peak 1, and define $Z^{X,GES_{\mathrm{i}}}_{Bound}$ as the boundary for the $(\mathrm{i}+1)$th and ith peak, which can be easily obtained by maximum likelihood.

For robust characterization of the single cell expression distribution, a key challenge is to address the observed zero and low expressions. These low expressions could be a result of multiple factors, such as technical errors, incompletely degraded mRNAs and varied experimental resolutions. We introduced a latent threshold Z^X_{cut} where when $\hat{x}_{\mathrm{j}} > Z^X_{cut}$, $\hat{x}_{\mathrm{j}}$ is modeled by mixture Gaussian distribution. Otherwise, we conclude that $\hat{x}_{\mathrm{j}}$ cannot be reliably

quantified under the current experimental resolution. Correspondingly, peaks with mean smaller or larger than Z^X_{cut} were defined as suppressed expression (SE) or active expression (AE) peaks. Z^X_{cut} differentiates the large expression values that are more likely to be under active expression state, from those low expression values that are not reliably quantifiable. In scRNA-seq data, other than a small number of housekeeping genes, an SE peak generally exists for most genes.

Figure 3.1A and B illustrates the relationship between the expression states of X, observed expression level $\hat{x}_{\mathrm{j}}$, and Z^X_{cut}. Specifically, when $\hat{x}_{\mathrm{j}}$ is observed to be lower than Z^X_{cut}, it can be

(1) true non-expression or expressions under an suppressed expression state.

(2) true active expression with low observed values, i.e. 'drop-outs'.

(3) true non expression but observed to have non-zero expression value, probably due to sequencing error, or a delay in mRNA degradation.

(4) true active expression state but falsely observed to have low expression, called Type II error.

(5) true suppressed expression state but falsely observed to have high expression, called Type I error.

(6) true active expression state.

Based on the derivations above, we could model a single cell's gene expression profile as a multimodal distribution, with observations smaller than Z^X_{cut} left truncated. Hence, active expression states, i.e. the AE peaks, can be robustly inferred as mixture Gaussian is highly sensitive to outliers; and the unquantifiable non-zero low expressions, i.e. the SE peak(s), can be effectively handled.

3.1.2 Left Truncated Mixture Gaussian (LTMG) distribution for gene expression modeling

To accurately and robustly model the gene expression profile of scRNA-seq data, we developed a Left Truncated Mixture Gaussian model, namely LTMG, to fit the log transformed normalized gene expression measures of gene X, such as TPM, CPM or RPKM, over N cells as $X = (x_1, x_2, , x_N)$. We assume that x_{i} follows a mixture Gaussian distribution with K Gaussian peaks corresponding to different SE and AE peaks. We introduce a parameter Z_{cut}^X and consider the log transformed zero and low expression values smaller than Z_{cut} as left censored data. With the left truncation assumption, X is divided into reliably measured expressions ($x_{\mathrm{j}} \geq Z_{cut}^X$) and left-censored gene expressions ($x_{\mathrm{j}} < Z_{cut}^X$). The density function of X can be written as:

$$\begin{aligned} p(X\|\Theta) &= \prod_{\mathrm{j}=1}^{N} p(x_{\mathrm{j}}\|\Theta) = \prod_{\mathrm{j}=1}^{M}\sum_{\mathrm{i}=1}^{K} a_{\mathrm{i}} p_{\mathrm{i}}(x_{\mathrm{j}}\|\theta_{\mathrm{i}}, x_{\mathrm{j}} \geq Z_{cut}^X) \times \prod_{\mathrm{j}=M+1}^{N}\sum_{\mathrm{i}=1}^{K} a_{\mathrm{i}} p_{\mathrm{i}}(x_{\mathrm{j}}\|\theta_{\mathrm{i}}, x_{\mathrm{j}} < Z_{cut}^X) \\ &= \prod_{\mathrm{j}=1}^{M}\sum_{\mathrm{i}=1}^{K} a_{\mathrm{i}} \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{i}}} \mathrm{e}^{-\frac{(x_{\mathrm{j}}-\mu_{\mathrm{i}})^2}{2\sigma_{\mathrm{i}}^2}} \times \prod_{\mathrm{j}=M+1}^{N}\sum_{\mathrm{i}=1}^{K} a_{\mathrm{i}} p_{\mathrm{i}}(x_{\mathrm{j}}\|\theta_{\mathrm{i}}, x_{\mathrm{j}} < Z_{cut}^X) = L(\Theta\|X) \end{aligned} \tag{3.4}$$

where parameters $\Theta = \{a_{\mathrm{i}}, \mu_{\mathrm{i}}, \sigma_{\mathrm{i}}\|\mathrm{i} = 1, ..., K\}$ and a_{i}, μ_{i} and σ_{i} are the mixing probability, mean and standard deviation of the K Gaussian distributions, corresponding to K expression states, M is the number of observations x_{j} that are larger than Z_{cut}^X, N is the total number of observations. Θ can be estimated using EM algorithm given Z_{cut}^X and K. Here we first define the Q function as:

$$\begin{aligned} Q(\Theta^t, \Theta^{t-1}) &= \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=1}^{M} log(\alpha_{\mathrm{i}}^t p_{\mathrm{i}}(x_{\mathrm{j}}\|\mu_{\mathrm{i}}^t, \sigma_{\mathrm{i}}^t)) p(y_{\mathrm{j}} = \mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \\ &\sum_{\mathrm{j}=M+1}^{N} \int_{-\infty}^{Z_{cut}^X} \sum_{\mathrm{i}=1}^{K} log(\alpha_{\mathrm{i}}^t p_{\mathrm{i}}(Z_{\mathrm{j}}\|\mu_{\mathrm{i}}^t, \sigma_{\mathrm{i}}^t)) p(y_{\mathrm{j}} = \mathrm{i}\|Z_{\mathrm{j}}, \Theta^{t-1} p(Z_{\mathrm{j}}\|x_{\mathrm{j}}, \Theta^{t-1})) dZ_{\mathrm{j}} \end{aligned}$$

where $p_{\mathrm{i}}(x_{\mathrm{j}}\|\mu_{\mathrm{i}}^t, \sigma_{\mathrm{i}}^t) = \frac{1}{\sqrt{2\pi}\sigma_{\mathrm{i}}^t} \mathrm{e}^{-\frac{(x_{\mathrm{j}}-\mu_{\mathrm{i}}^t)^2}{2(\sigma_{\mathrm{i}}^t)^2}}$, Z_{j} is a latent variable for the true value of x_{j} if $x_{\mathrm{j}} < Z_{cut}^X$, i.e., left censored data. $y_{\mathrm{j}} = 1, ..., K$ are latent variables indicating that x_{j} comes from the jth

Gaussian peak, and t is the current iteration step. For the tth interaction of the algorithm, with computed parameter set Θ^{t-1}, the pdf of Z_{j} and probability of y_{j} are:

$$f(Z_{\mathrm{j}}\|\Theta^{t-1}) = \frac{\sum_{\mathrm{i}=1,...,K} \frac{1}{\sqrt{2\pi\sigma)_{\mathrm{i}}^{t-1}}} \mathrm{e}^{-\frac{(Z_{\mathrm{j}}-\mu_{\mathrm{i}}^{t-1})^2}{2(\sigma_{\mathrm{i}}^{t-1})^2}}}{\sum_{\mathrm{i}=1,...,K} \int_{-\infty}^{Z_{cut}^X} \frac{1}{\sqrt{2\pi\sigma)_{\mathrm{i}}^{t-1}}} \mathrm{e}^{-\frac{(Z_{\mathrm{j}}-\mu_{\mathrm{i}}^{t-1})^2}{2(\sigma_{\mathrm{i}}^{t-1})^2}} dy}$$

$$p(y_{\mathrm{j}} = k\|\Theta^{t-1}, x_{\mathrm{j}} \geq Z_{cut}^X) = \frac{\frac{1}{\sqrt{2\pi\sigma_k^{t-1}}} \mathrm{e}^{-\frac{(x_{\mathrm{j}}-\mu_k^{t-1})^2}{2(\sigma_k^{t-1})^2}}}{\sum_{\mathrm{i}=1,...,K} \frac{1}{\sqrt{2\pi\sigma_{\mathrm{i}}^{t-1}}} \mathrm{e}^{-\frac{(x_{\mathrm{j}}-\mu_{\mathrm{i}}^{t-1})^2}{2(\sigma_{\mathrm{i}}^{t-1})^2}}}, \; k = 1, ..., K$$

$$p(y_{\mathrm{j}} = k\|\Theta^{t-1}, x_{\mathrm{j}} < Z_{cut}^X) = \frac{\int_{-\infty}^{Z_{cut}^X} \frac{1}{\sqrt{2\pi\sigma_k^{t-1}}} \mathrm{e}^{-\frac{(y-\mu_k^{t-1})^2}{2(\sigma_k^{t-1})^2}} dy}{\sum_{\mathrm{i}=1,...,K} \int_{-\infty}^{Z_{cut}^X} \frac{1}{\sqrt{2\pi\sigma_{\mathrm{i}}^{t-1}}} \mathrm{e}^{-\frac{(y-\mu_{\mathrm{i}}^{t-1})^2}{2(\sigma_{\mathrm{i}}^{t-1})^2}} dy}, \; k = 1, ..., K$$

Then the Q function could be rewrite as:

$$\begin{aligned}
Q(\Theta, \Theta^{t-1}) &= \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=1}^{M} log(a_{\mathrm{i}}) p(y_{\mathrm{i}} = \mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=1}^{M} log(p_{\mathrm{i}}(x_{\mathrm{j}}\|\mu_{\mathrm{i}}, \sigma_{\mathrm{i}})) p(y_{\mathrm{i}} = \mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) \\
&+ \sum_{\mathrm{i}=1}^{}\sum_{\mathrm{j}=M+1}^{N} \int log(p_{\mathrm{i}}(x_{\mathrm{j}}\|\mu_{\mathrm{i}}, \sigma_{\mathrm{i}})) p(Z_{\mathrm{j}}\|x_{\mathrm{j}}, \Theta^{t-1}) dZ_{\mathrm{j}} p(y_{\mathrm{j}}\|x_{\mathrm{j}}, \Theta^{t-1}) \\
&= \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=1}^{M} log(a_{\mathrm{i}}) p(y_{\mathrm{i}} = \mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=1}^{M} log(p_{\mathrm{i}}(x_{\mathrm{j}}\|\mu_{\mathrm{i}}, \sigma_{\mathrm{i}})) p(y_{\mathrm{i}} = \mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) \\
&+ \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=M+1}^{N} \frac{1}{2\sigma_{\mathrm{i}}^2} p(y_{\mathrm{j}} = \mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) \begin{bmatrix} E(Z_{\mathrm{j}}^2\|\mu_{\mathrm{i}}^{t-1}, \sigma_{\mathrm{i}}^{t-1}, Z_{\mathrm{j}} < Z_{cut}) \\ - 2\mu_{\mathrm{i}} E(Z_{\mathrm{j}}\|\mu_{\mathrm{i}}^{t-1, \sigma_{\mathrm{i}}^{t-1}, Z_{\mathrm{j}} < Z_{cut}}) + \mu_{\mathrm{i}}^2 \end{bmatrix} \\
&+ \sum_{\mathrm{i}=1}^{K}\sum_{\mathrm{j}=1}^{N} log(a_{\mathrm{i}}) p(y_{\mathrm{i}} = \mathrm{i}\|Z_{\mathrm{j}}, \Theta^{t-1})
\end{aligned}$$

Let $H(x) = \frac{\phi(x)}{\Phi(x)}$, where $\phi(x)$ and $\Phi(x)$ are the pdf and cdf of standard normal distribution. Such that the M step could be written as:

$$\frac{\partial Q}{\partial a_{\mathrm{i}}} = 0 \to a_{\mathrm{i}}^t = \frac{1}{N}(\sum_{\mathrm{j}=1}^{M} P(\mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1})) + \sum_{\mathrm{j}=M+1}^{N} P(\mathrm{i}\|Z_{\mathrm{j}}, Z_{cut}, \Theta^{t-1})$$

$$\frac{\partial Q}{\partial \mu_{\mathrm{i}}} = 0 \to \mu_{\mathrm{i}}^t = \frac{\sum_{\mathrm{j}=1}^{M} x_{\mathrm{j}} P(\mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \sum_{\mathrm{j}=M+1}^{N} (\mu_{\mathrm{i}}^{t-1} - \sigma_{\mathrm{i}}^{t-1} H(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})) P(\mathrm{i}\|Z_{\mathrm{j}}, Z_{cut}, \Theta^{t-1})}{\sum_{\mathrm{j}=1}^{M} P(\mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \sum_{\mathrm{j}=M+1}^{N} P(\mathrm{i}\|Z_{\mathrm{j}}, Z_{cut}, \Theta^{t-1})}$$

$$\frac{\partial Q}{\partial \sigma_{\mathrm{i}}} = 0 \to (\sigma_{\mathrm{i}}^t)^2 = \frac{\sum_{\mathrm{j}=1}^{M} P(\mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1})(x_{\mathrm{j}} - \mu_{\mathrm{i}}^{t-1})^2}{\sum_{\mathrm{j}=1}^{M} P(\mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \sum_{\mathrm{j}=M+1}^{N} P(\mathrm{i}\|Z_{\mathrm{j}}, Z_{cut}, \Theta^{t-1})}$$
$$+ \frac{(\sigma_{\mathrm{i}}^{t-1})^2 \sum_{\mathrm{j}=M+1}^{N} (1 - \frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}} * H(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})) P(\mathrm{i}\|Z_{\mathrm{j}}, Z_{cut}, \Theta^{t-1})}{\sum_{\mathrm{j}=1}^{M} P(\mathrm{i}\|x_{\mathrm{j}}, \Theta^{t-1}) + \sum_{\mathrm{j}=M+1}^{N} P(\mathrm{i}\|Z_{\mathrm{j}}, Z_{cut}, \Theta^{t-1})}$$

And the E step is:

$$E(Z_{\mathrm{j}}\|\mu_{\mathrm{i}}^{t-1}, \sigma_{\mathrm{i}}^{t-1}, Z_{\mathrm{j}} < Z_{cut}) = \mu_{\mathrm{i}}^{t-1} + \sigma_{\mathrm{i}}^{t-1} E(\varepsilon_{\mathrm{j}}\|\varepsilon < \frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})$$
$$= \mu_{\mathrm{i}}^{t-1} + \frac{\sigma_{\mathrm{i}}^{t-1} \int_{-\infty}^{\frac{Q - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}}} w\phi(w)dw}{\Phi(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})} = \mu_{\mathrm{i}}^{t-1} + \frac{\sigma_{\mathrm{i}}^{t-1} \phi(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})}{\Phi(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})}$$
$$= \mu_{\mathrm{i}}^{t-1} + \sigma_{\mathrm{i}}^{t-1} H(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})$$

Similarly,

$$E(Z_{\mathrm{j}}^2\|\mu_{\mathrm{i}}^{t-1}, \sigma_{\mathrm{i}}^{t-1}, Z_{\mathrm{j}} < Z_{cut}) = (\mu_{\mathrm{i}}^{t-1})^2 + (\sigma_{\mathrm{i}}^{t-1})^2 - \sigma_{\mathrm{i}}^{t-1}(Z_{cut} + \mu_{\mathrm{i}}^{t-1}) H(\frac{Z_{cut} - \mu_{\mathrm{i}}^{t-1}}{\sigma_{\mathrm{i}}^{t-1}})$$

Θ can be estimated by iteratively running the E and M step in the above algorithm given Q, X and K.

3.2 Experiments

3.2.1 Dataset and existing methods

To conduct a comprehensive evaluation of our model, we collected 23 datasets totaling 66780 human and mouse cells across different cell extraction and sequencing platforms with varied experimental designs. It is noteworthy there are multiple scRNA-seq protocols that differ by cell capture, lyse and sequencing methods. These methods either construct individual libraries for each cell, or an overall library for thousands of cells at once, the latter of which is known as 'drop-seq' based method. Recent reviews suggested that the Smart-Seq2 protocols achieve best performance among the methods with individual libraries, and 10× Genomics Chromium is the most utilized commercialized pipeline [93]. Our data collection comprehensively covers human and mouse data generated by Smart-seq/Smart-Seq2, 10x Genomics and inDrops platforms from January 2016 to June 2018 in the GEO database. Hence, we consider this collection as unbiased testing data that can represent the general characteristics of the single cell data generated from the two types of protocols. Since each dataset has different levels of complexity, we reorganized the datasets into sub datasets with comparable levels of complexities. The sub datasets were generated to represent three different types of sample complexities: (i) pure condition, where each sub dataset contains cells of one type under a specific experimental condition; (ii) cell cluster, where each sub dataset belongs to a priori computationally clustered cells and (iii) complete data, where each sub dataset contains multiple mixed cell population, such as cells from one cancer tumor tissue. In total, sub datasets with 51 pure condition, 49 cell cluster and 78 complete data were extracted from the 23 large data sets. It is noteworthy that each sub data set consists of only cells from one of the 23 original data set, to avoid causing batch effect from single cell RNA sequencing.

We compared LTMG with Zero-inflated mixed Gaussian (ZIMG), MAST[68] and Beta Poisson (BPSC) [69]. We use MAST with default parameters, and for each gene, only non-zero values were used and fitted with Gaussian distribution. For BPSC, to achieve a reliable estimation, only genes with non-zero expressions in at least 25 single cells were kept. ZIMG was used with default parameters. Kolmogorov Statistic (KS) is used to measure

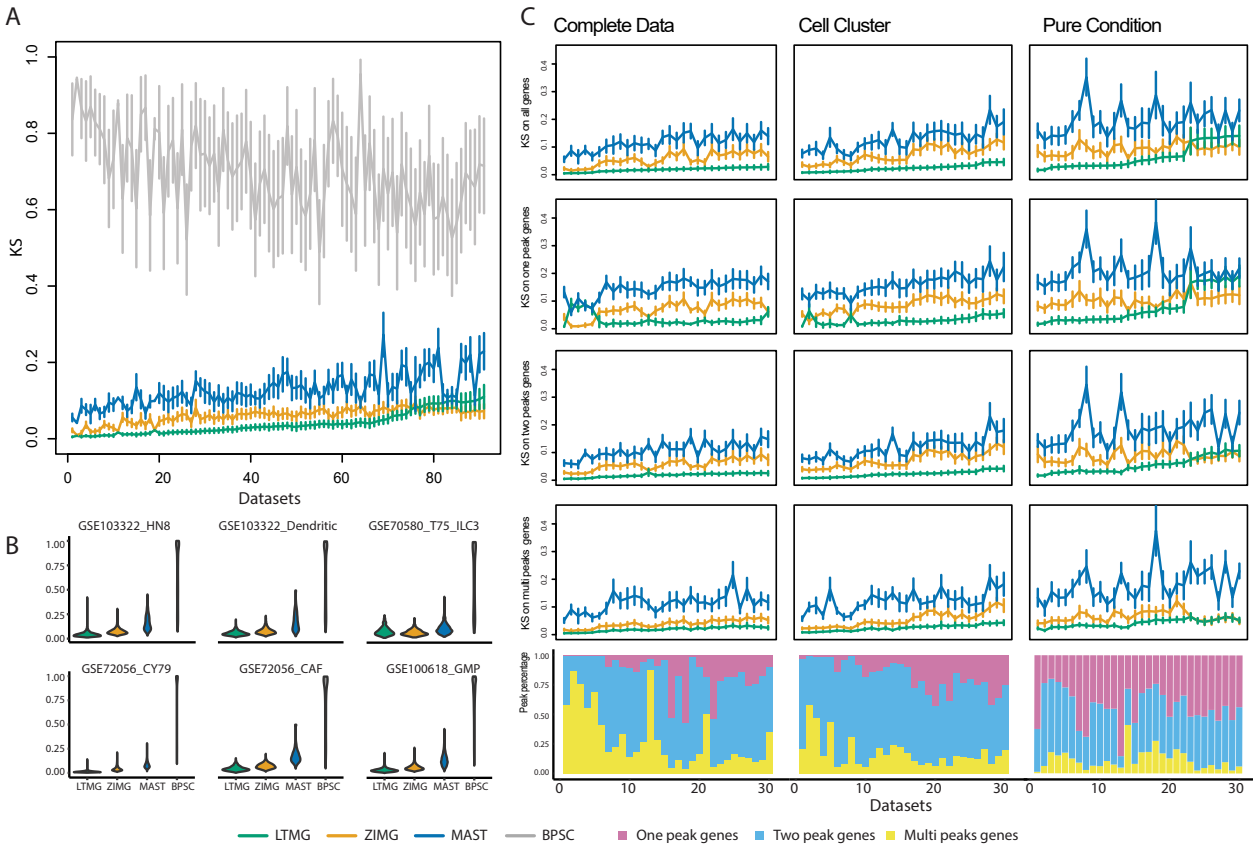

Figure 3.2. Model comparison of LTMG and existing methods

gene-wise goodness of fitting. For each gene, the KS score is assessed by using the none zero observations for ZIMG, MAST and BPSC models and normalized by dividing the KS score by the none zero proportions, due to their zero inflation assumption. Only genes kept for all four models are used for downstream evaluations. For each extracted sub dataset, we defined a goodness fitting score for each method using the mean and standard deviation of gene-wise KS values:

$$GF_{score} = \frac{1}{2}(\bar{KS} + \sigma(KS))$$

where $\bar{KS}$ is the mean value of gene-wise KS scores from a dataset and $\sigma(KS)$ is the standard deviation. The GF score evaluates each method on both overall accuracy (lower $\bar{KS}$ value) and stability (lower $\sigma(KS)$), and smaller GF indicates better goodness of fitting.

3.2.2 LTMG achieved better goodness of fitting

We first applied LTMG, ZIMG, MAST and BPSC to fit the expression profile of each gene in all the 178 sub data sets. Kolmogorov Statistics (KS) [94] was applied to evaluate the goodness of fitting of each gene, and for each dataset using each method. The mean and standard deviation of the KS values over all the genes for each dataset and method was calculated, and the 178 sub datasets were ordered in increasing order by the mean KS values calculated based on LTMG.And the comparisons on the top 91 datasets were shown in Figure 3.2A, which suggested: (a) LTMG has significantly better goodness of fitting compared with BPSC and MAST in all the analyzed data sets and outperforms ZIMG in most of the datasets (Figure 3.2A); (b) LTMG generally has a smaller number of outliers with poor fitting through all the datasets (Figure 3.2B), suggesting the higher robustness of LTMG comparing to others. Our analysis suggested that the average proportion of genes fitted with one, two, and more than two peaks are 42.5%, 44.9% and 12.6% in pure condition, 16.6%, 65.7% and 17.6% in cell cluster, and 25.4%, 51.5% and 23.1% in complete data sets, respectively.

In addition to investigating the goodness of fitting over all the genes, we focused on a more detailed comparison of gene groups that are fitted with different number of peaks under LTMG. We compared the goodness of fitting between LTMG and ZIMG, MAST, on all the genes, genes fitted with one, two and multiple peaks. Here, BPSC was dropped from the comparison, since it has much lower performance than other models. Figure 3.2C shows the top 30 sub datasets in each of the three cases: pure condition, cell cluster and complete data, that has the smallest KS values based on LTMG model respectively. Within the cell cluster and complete data sets, LTMG consistently outperformed ZIMG (120/127) and MAST (127/127), for genes fitted with different peaks. In the pure condition datasets, LTMG outperformed MAST in all the sub data sets (51/51), outperformed ZIMG (42/51) for the genes fitted with more than two Gaussian peaks, and have comparable performance as ZIMG (23/51) for the genes that are fitted with one or two peaks. A possible reason for the less significant performance of LTMG on the pure condition datasets could be that the sample size of the PC datasets is generally small (115 cells on average) compared to cell

cluster (388 cells) and complete (622 cells) data sets. A consequence is that the half bell shaped SE peak (Figure 3.1A) is not significantly different from a full Gaussian peak when the sample size is small. Notably, ZIMG tends to overfit, as the non-zero expression caused by incompletely degraded mRNA could inflate the number of AE peaks, while LTMG can effectively handle the non-zero low expressions by the left truncation assumption.

We also applied the LTMG model to three recent data sets of purified T cells collected from liver, lung and colon cancer tissues [95]–[97]. These data sets all consist of pure T cell with large sample sizes (5063, 11 138, and 12 346 cells). In these data sets, LTMG also achieved the best goodness of fitting comparing to ZIMG and MAST. LTMG identified more than 44.5% (4893/10 874), 69.73% (7093/10 172) and 69.95% (7551/10 794) of significantly expressed genes with at least one SE peak and two AE peaks in the three datasets, respectively. We further utilized a stringent criterion to select only the genes with at least two AE peaks, each of which covers significant proportion of the total cells and is distinct to other peaks. This results in 26.56% (2888/10 874), 22.67% (2306/10 172) and 24.56% (2651/10 794) of the genes with at least two distinct AE peaks in the three data sets, demonstrating the prevalence of multi-modality in gene expression states in large data sets, and the heterogeneity of single T cell expressions in tumor micro-environment.

3.2.3 LTMG handles zero and low expressions properly

The observed low expression depicted as ③ and ④ in Figure 3.1A are generally seen in all the analyzed data sets, which on average take 27.9%, 16.3% and 14.5% of non-zero values in the PC, CC and CD data. We hypothesized that one major contributor of the low expression is the incompletely degraded mRNA under the regulation of a TRS of suppressed state, which should be distinguished from those TRSs under active states, namely, ⑥ (Figure 3.3A). To validate this hypothesis, we collected a data set of experimentally measured mRNA kinetics of mouse fibroblast cells [98], and two scRNA-seq data set (GSE99235 and GSE98816) of mouse fibroblast cells [99], [100]. We examined the correlations between the mRNA half-lives and the estimated proportion of incompletely degraded mRNA. Specifically, positive correlations between (i) the proportions of uncensored observations in the SE

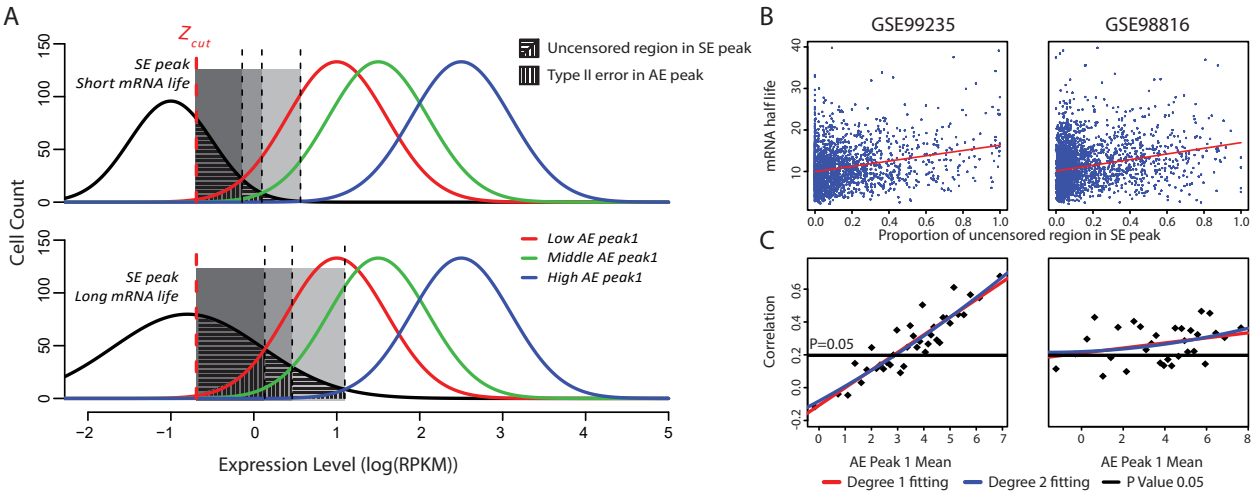

Figure 3.3. LTMG can handle experimental noise

peak, defined by $\frac{③+④}{①+②+③+④}$ in Figure 3.1A, and (ii) mRNA half-life, were consistently observed in both data sets (Figure 3.1B), suggesting that genes with more uncensored expressions regulated by suppressing regulators are probably a result of longer mRNA half-life. It is noteworthy the AE peaks for higher mean expression suffer less impact from the non-zero low expressions. To adjust for this bias, we examined the correlations of mRNA half-life with the proportion of uncensored observations conditional on the mean of AE peak. Significant positive correlations ($P < 0.05$) were observed for the genes with a relatively larger mean of AE peak, and the correlations tend to be stronger among the genes with larger AE peaks, in both of the analyzed data sets (Figure 3.3C), further validated the relationship between the observed low expression and incompletely degraded mRNA.

3.3 Biological applications

3.3.1 Modeling the transcriptomic heterogeneity among cells

The multi-modality characteristic of LTMG unravels the transcriptomic heterogeneity among a cell population. We then ask how cells behave with respect to our identified SE and AE peaks. For a gene, we denoted the cells with non-zero expression as 'Exp', the cells assigned to the AE peaks as 'AE' and the cells assigned to the SE peaks as 'SE'. We tested

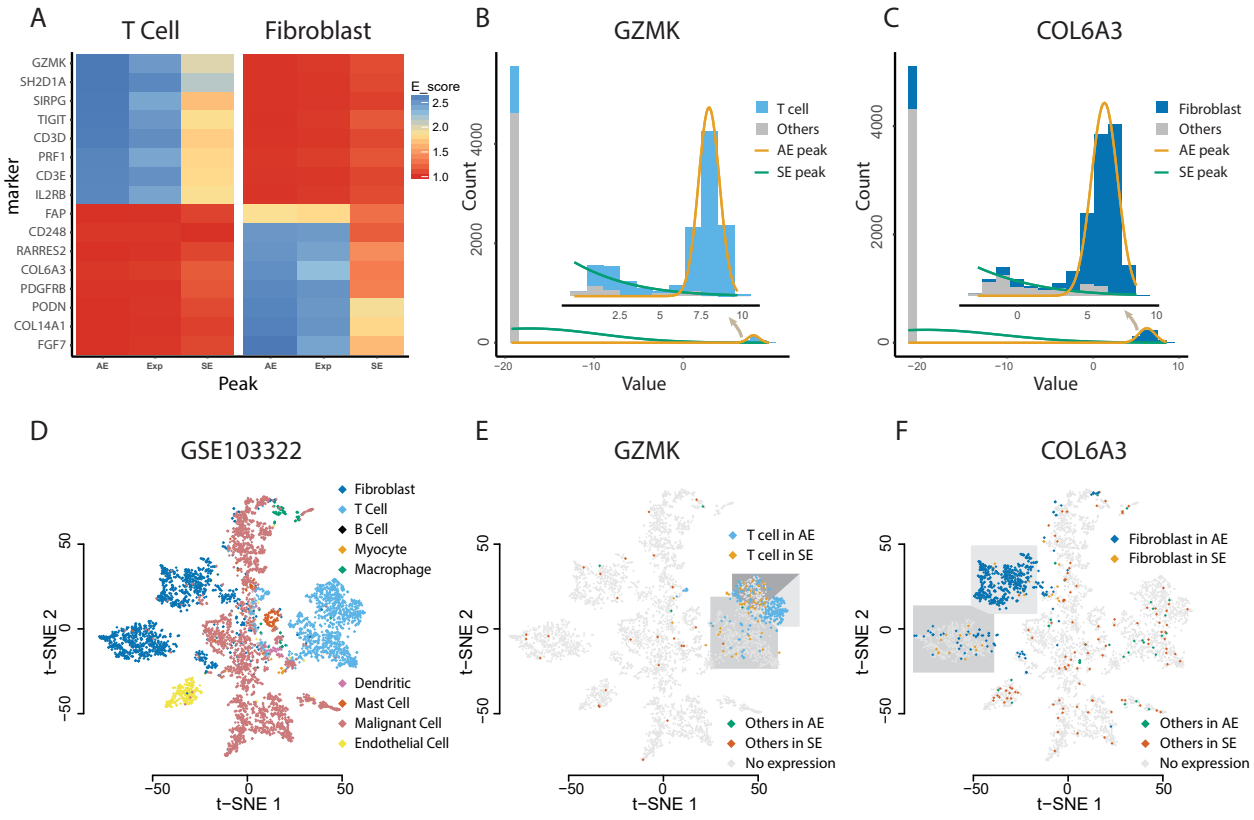

Figure 3.4. LTMG captures transcriptomic heterogeneity

for cell marker genes, how the cells of known cell type labels are distributed through the 'AE', 'Exp' and 'SE' cell groups, with regards to different marker genes. Our hypothesis is that for the cells with a certain identity such as cytotoxic T cells, they are expected to overly express specific cell marker genes like granzymes, such that their expression level is more likely to be in an AE peak rather than an SE peak. On the other hand, T cells are more likely to be enriched in certain AE peaks of granzymes but are excluded in SE peaks. In addition, since LTMG identifies certain low non-zero expressions to SE peak, we hypothesize that a cell type will be more strongly enriched to the AE peaks rather than all the cells with non-zero expression value of a marker gene.

We applied LTMG on a head and neck cancer (HNSC) data set (GSE103322) consisting of 5902 cells of nine cell types namely B cell, T cell, Myocyte, Macrophage, Endothelial, Dendritic and Mast cell, with pre-annotated cell labels and uniquely expressed maker genes [24]. We defined an enrichment score to evaluate the association between cell type and the

cell expression states, namely, 'AE', 'Exp' and 'SE', for each marker gene. Not surprisingly, our analysis showed that a cell type always significantly enriches the 'AE' expression state if the gene is specific to the cell type, suggesting that the AE state identified by LTMG is a good characterization of the true active expression state, comparing to other methods. Figure 3.4A shows the enrichment score of T and fibroblast cells associated with 'AE', 'Exp' and 'SE' states, for eight T cell marker genes (top eight rows) and eight fibroblast marker genes (bottom eight rows). Figure 3.4B and C illustrate the LTMG fitted curves of GZMK, a cytotoxic T cell marker, and COL6A3, a fibroblast marker. Figure 3.4D shows on a clustering visualization using 2D-tSNE plot of the nine cell types, the distribution of all the cells with the AE and uncensored SE states of these two genes. We observed that the CD8+ T cells with the AE expressions or uncensored SE expressions of GZMK were clearly separated to high cytotoxic and exhausted CD8+ T cells in the HNSC microenvironment [101]–[103] (Figure 3.4E). Similarly, the fibroblast cells with an AE or an uncensored SE expression of COL6A3 were differentially distributed as two sub fibroblast types (Figure 3.4F). Moreover, cells that expressed in SE peak are scattered outside T cell or Fibroblast cell region, validated that SE peak does not representing cell type identity and should be de-noised for further analysis.

3.3.2 Single-cell clustering based on inferred modality by LTMG

Our analysis suggested that the gene expression states inferred by LTMG can reflect the cell type specific gene expression characteristics by effectively removing the noise of the low but non-zero expressions. Here we show that this denoising approach can largely benefit the cell clustering analysis and visualization of the single cell data collected from complicated microenvironment such as cancer and peripheral blood samples.

Five dimension reduction and clustering methods including: (i) UMAP; (ii) t-SNE; (iii) UMAP on LTMG denoised data, called LTMG UMAP; (iv) t-SNE on LTMG denoised data, called LTMG t-SNE and (v) SIMLR, were compared on three datasets: GSE103322, GSE72056, and 10× PBMC with annotated cell types. We compared LTMG UMAP, LTMG t-SNE, UAMP, t-SNE and SIMLR by using the Silhouette width, the higher value of which

suggests a better consistency between predicted cell clusters and true cell labels. 2D visualization of cell clustering and the Silhouette width were shown in Figure 3.5. Our analysis suggested the cell clusters inferred from LTMG denoised data outperform the clusters identified by using original data, for both UMAP and t-SNE. In the GSE72056 and GSE103322 dataset, cell surface markers and predicted copy number variations were used to identify true malignant cells, which were composed by multiple subclasses of cells due to inter-tumor heterogeneity, as illustrated by the red colored cells in Figure 4. We observed the malignant cells, as well as other normal cells, are more spreaded over the 2D UAMP and t-SNE of the original data while the LTMG UMAP and LTMG t-SNE well manage the subclass of malignant cells from different patients (Figure 3.5). In addition, different types of immune and stromal cells were better distinguished from malignant cells and each other in the LTMG UMAP and LTMG t-SNE based embedding. A possible explanation is that the LTMG based transformation of gene expression states can better characterize the inter-cell type varied expression states via removing the intra-cell type gene expression variations that do not form varied expression states.

3.4 Discussion

We developed LTMG as a statistical model specifically for scRNA-Seq data. LTMG considers the heterogeneity of transcriptional regulatory and gene expression states, and in handling the low expressions, LTMG considers the metabolism rates of mRNA molecules, and experimental resolution in modeling scRNA-seq data, from a systems biology perspective. Our comprehensive model evaluations demonstrated that LTMG can accurately infer the multi-modality of genes expression states, better handle low expressions caused by suppressed regulation and incompletely degraded mRNA, and has a significantly improved goodness of fitting, compared to other existing models.

LTMG is designed for analysis of scRNA-seq with a comparable sequencing depth for each cell. Application of LTMG on drop-seq based data such as 10x Genomics data demonstrated that the model also outperforms other models in goodness of fitting and can successfully infer multimodality from single gene's expression profile. However, in cases where a wide span of

total reads among the cells in the drop-seq data exist, the distribution of the normalized gene expression may be severely affected by variations in total sequenced reads. We noticed that, the inference of varied expression states heavily relies on sample size. For the cells collected from a pure condition, on average, LTMG only identified 200–1500 genes with more than one distinct AE peaks when the sample size is several hundreds, while >2000 of such genes can be identified when the sample size is larger than 5000. SC2P introduced a cell wise sequencing resolution to account for the discrepancies in library sizes [74]. A possible future direction of LTMG is to incorporate a similar cell wise factor into the current model, so it will improve the characterization of varied expression resolution and SE peak for drop-seq based scRNA-Seq data. LTMG characterize the heterogenous gene expression states via a mixture Gaussian model on log normalized gene expression data. Log-normal assumption has been commonly utilized to model the active expressions, i.e. non-zero expressions, in MAST, scImpute, and SC2P. However, as derived in the above method, gene expression regulated by high frequency transcriptional bursting or highly dynamic regulatory signals, may unnecessarily follow distinct gene expression states that fits the mixture Gaussian assumption. High resolution data such as large scale smFISH data would be needed for inference of the gene expression states in this case, with more sophisticated model.

Our analysis also suggested that the cell clustering conducted on LTMG inferred gene expression states performs better than clustering on the raw expression data, either using the same or different clustering techniques. This indicates that to distinguish cell types, it suffices to use the distinct expression states of the genes, which forms a good characterization of the difference among cell types, and more importantly, the discretized expression states are more robust to noise and outliers. We believe that the cell type specifically expressed genes tend to form distinct gene expression states across a large cell population, compared with those non-specific genes, such as housekeeping genes, which could usually be fitted with one Gaussian peak of large variance. The flexibility in selecting the best number of peaks in LTMG can thus identify the genes with significantly varied expression states, that are more likely to be cell type specific markers. Actually, regulation of the cell type specific genes is more commonly seen through constant regulatory inputs, which best fits the assumption of LTMG model. Successfully distinguishing the cell type and phenotypic genes not only

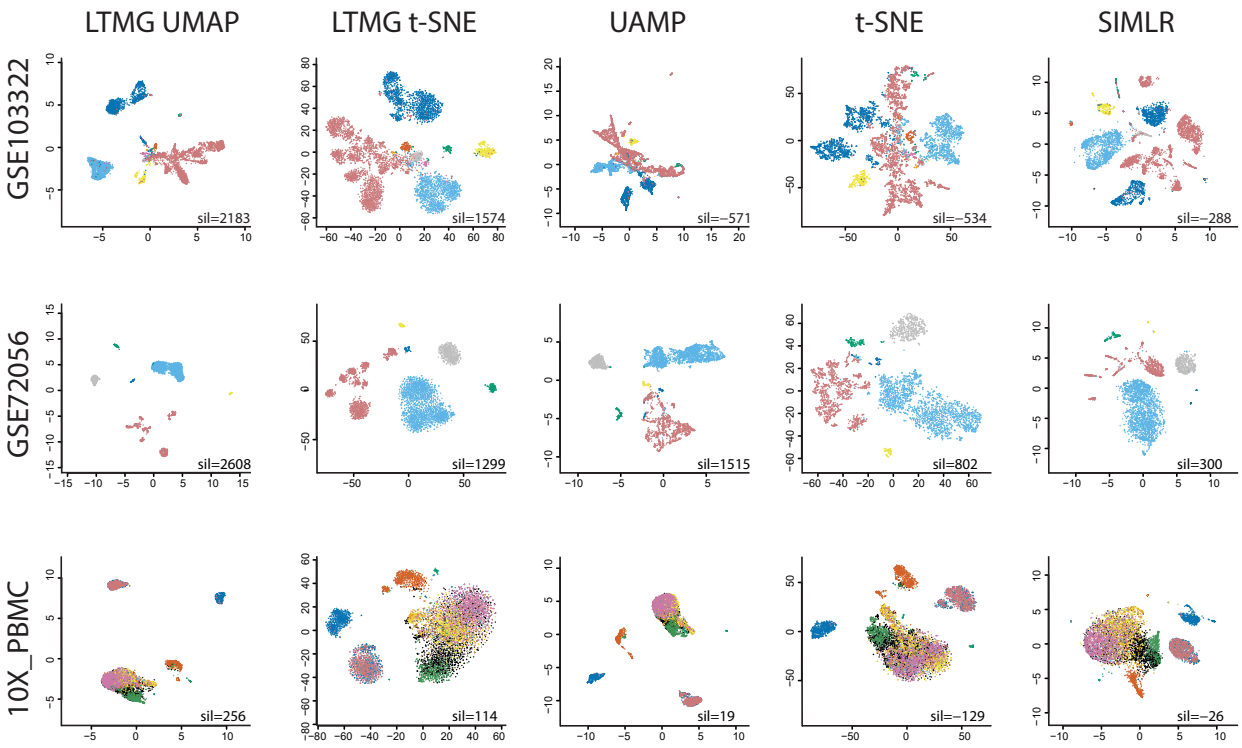

Figure 3.5. Clustering visualization of LTMG inferred states

increase the specificity of cell type clustering analysis, but also helps to extract the low rank structure in scRNA-seq data and provides more biologically meaningful visualization. LTMG model can also fit into cases with transcriptional bursting regulations, when considering the bi-state property observed from transcriptional bursting. A straightforward link between LTMG inferred peaks and the transcriptional bursting model is that the proportion and mean of each peak in LTMG directly corresponds to the frequency and expression level of each input signal [13]. Eventually, we hope the LTMG model based inference of gene expression states will shed new light on deducing the mechanisms transcriptional regulation by using scRNA-seq data.

LTMG transferred the continuous scRNA-deq data into discrete gene expression state data, which is a binary matrix, where every row is a gene expression state and every column is a cell. In the next chapter, we will introduce a machine learning approach, Boolean matrix factorization, that would be utilized to identify functional cell types from LTMG inferred discrete expression states.

4. FAST AND EFFICIENT BOOLEAN MATRIX FACTORIZATION BY GEOMETRIC SEGMENTATION

4.1 Overview

Binary data gains more and more attention during the transformation of modern living [104], [105]. It consists of a large domain of our everyday life, where the 1s or 0s in a binary matrix can physically mean whether or not an event of online shopping transaction, web browsing, medical record, journal submission, etc, has occurred or not. The scale of these datasets has increased exponentially over the years. Mining the patterns within binary data as well as adapting to the drastic increase of dimensionality is of prominent interests for nowadays data science research. Recent study also showed that some continuous data could benefit from binary pattern mining. For instance, the binarization of continuous single cell gene expression data to its on and off state, can better reflect the coordination patterns of genes in regulatory networks [13]. However, owing to its two value characteristics, the rank of a binary matrix under normal linear algebra can be very high due to certain spike rows or columns. This makes it infeasible to apply established methods such as SVD and PCA for BMF [106].

Boolean matrix factorization (BMF) has been developed particularly for binary pattern mining, and it factorizes a binary matrix into approximately the product of two low rank binary matrices following Boolean algebra, as shown in Figure 1. The decomposition of a binary matrix into low rank binary patterns is equivalent to locating submatrices that are dense in 1. Analyzing binary matrix with BMF shows its unique power. In the most optimal case, it significantly reduces the rank of the original matrix calculated in normal

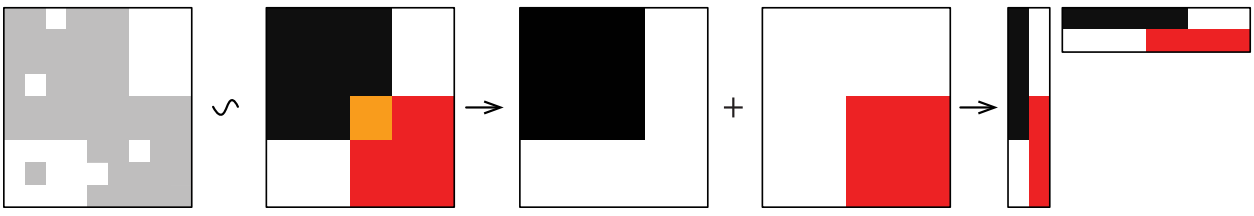

Figure 4.1. BMF, the addition of rank 1 binary matrices

linear algebra to its log scale [107]. Since the binary patterns are usually embedded within noisy and randomly arranged binary matrix, BMF is known to be an NP-hard problem [25].

4.2 Background

BMF was first introduced as a set basis problem in 1975 [108]. This area has received wide attention after a series of work by Mittenin et al [25], [109], [110]. Among them, the ASSO algorithm performs factorization by retrieving binary bases from row-wise correlation matrix in a heuristic manner [25]. Despite its popularity, the high computational cost of ASSO makes it impracticable when dealing with large scale data. Recently, an algorithm called Nassua was developed by the same group [110]. Nassua optimizes the initialization of the matrix factorization by locating dense seeds hidden within the matrix, and with improved performance comparing to ASSO. However, optimal parameter selection remains a challenge for Nassua. A second series of work called PANDA was developed by Claudio et al [111], [112]. PANDA aims to find the most significant patterns in the current binary matrix by discovering core patterns iteratively [111]. After each iteration, PANDA only retains a residual matrix with all the non-zero values covered by identified patterns removed. Later, PANDA+ was recently developed to reduce the noise level in core pattern detection and extension [112]. These two methods also suffer from inhibitory computational cost, as they need to recalculate a global loss function at each iteration. More algorithms and applications of BMF have been proposed in recent years. FastStep [113] relaxed BMF constraints to non-negativity by integrating non-negative matrix factorization (NMF) and Boolean thresholding. But interpreting derived non-negative bases could also be challenging. With prior network information, Kocayusufoglu et al, decompose binary matrix in a stepwise fashion with bases that are sampled from given network space [104]. Bayesian probability mapping has also been applied in this field . Ravanbakhsh et al proposed a probability graph model called “factor-graph” to characterize the embedded patterns, and developed a message passing approach, called MP [114]. On the other hand, Ormachine, proposed by Rukat et al, provided a probabilistic generative model for BMF [115]. Similarly, these Bayesian ap-

proaches suffer from low computational efficiency. In addition, Bayesian model fitting could be highly sensitive to noisy data.

4.2.1 Notations

A matrix is denoted by a uppercase character with a super script $n \times m$ indicating its dimension, such as $X^{n\times m}$, and with subscript $X_{\mathrm{i},:}$, $X_{:,\mathrm{j}}$, X_{ij} indicating ith row, jth column, or the (i,j)th element, respectively. A vector is denoted as a bold lowercase character, such as $\mathbf{a}$, and its subscript $\mathbf{a}_{\mathrm{i}}$ indicates the ith element. A scalar value is represented by a lowercase character, such as a, and $[a]$ as its integer part. $\|X\|$ and $\|\mathbf{x}\|$ represents the ℓ_1 norm of a matrix and a vector. Under the Boolean algebra, the basic operations include $\wedge(AND, 1\wedge 1=1, 1\wedge 0=0, 0\wedge 0=0)$, $\vee(OR, 1\vee 1=1, 0\vee 1=1, 0\vee 0=0)$, $\neg(NOT, \neg 1=0, \neg 0=1)$. Denote the Boolean element-wise sum, subtraction and product as $A\oplus B = A\vee B$, $A\ominus B = (A\wedge\neg B)\vee(\neg A\wedge B)$ and $A\circledast B = A\wedge B$, and the Boolean matrix product of two Boolean matrices as $X^{n\times m} = A^{n\times k}\otimes B^{k\times m}$, where $X_{\mathrm{ij}} = \vee_{l=1}^{k} A_{\mathrm{i}l}\wedge B_{l\mathrm{j}}$.

4.2.2 Problem statement

Given a binary matrix $X\in\{0,1\}^{n\times m}$ and a criteria parameter τ, the BMF problem is defined as identifying two binary matrices A^* and B^*, called pattern matrices, that minimize the cost function $\gamma(A,B;X)$ under criteria τ, i.e., $(A^*,B^*) = \mathrm{argmin}_{A,B}(\gamma(A,B;X)\|\tau)$. Here the criteria τ could vary with different problem assumptions. The criteria used in the current study is to identify A^* and B^* with at most k patterns, i.e., $A\in\{0,1\}^{n\times k}$, $B\in\{0,1\}^{k\times m}$, and the cost function is $\gamma(A,B;X) = \|X\ominus(A\otimes B)\|$. We call the lth column of matrix A and lth row of matrix B as the lth binary pattern, or the lth basis, $l=1,...,k$.

4.3 MEBF Algorithm Framework

4.3.1 Motivation of MEBF

BMF is equivalent to decomposing the matrix into the sum of multiple rank 1 binary matrices, each of which is also referred as a pattern or basis in the BMF literature [111].

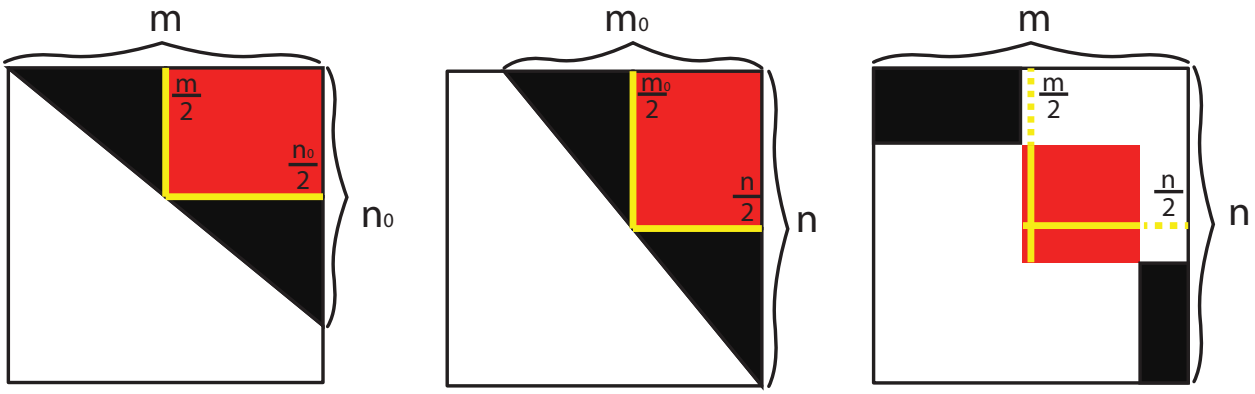

Figure 4.2. Three simplified scenarios for UTL matrices with direct SC1P.

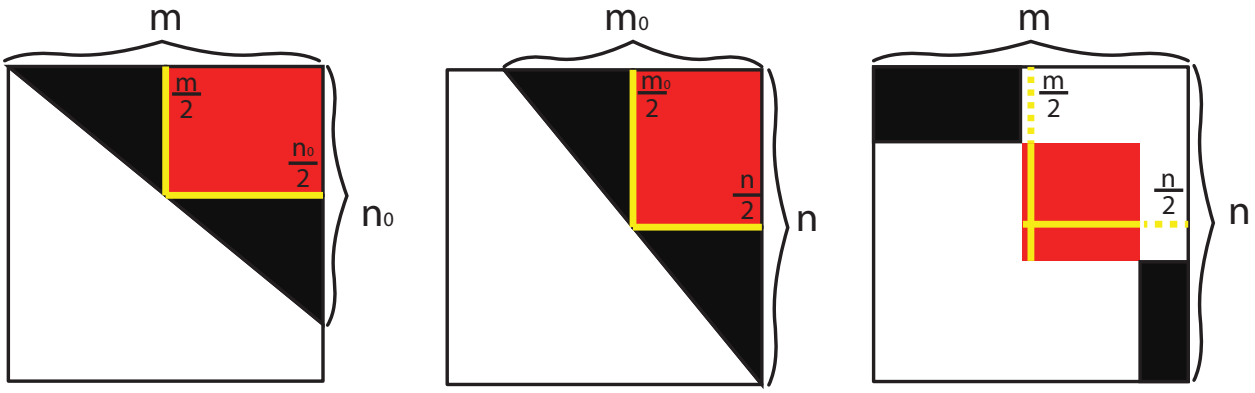

Figure 4.3. A schematic overview of the MEBF pipeline for three data scenarios where the matrix is roughly UTL with SC1P.

Lemma 4.3.1 (**Submatrix detection**). *Let* A^*,B^* *be the solution to*

$$\underset{A\in\{0,1\}^{n\times k},B\in\{0,1\}^{k\times m}}{\arg\min} \|X \ominus (A \otimes B)\|,$$

then the k *patterns identified in* A^*, B^* *correspond to* k *submatrices in* X *that are dense in 1's. In other words, finding* A^*, B^* *is equivalent to identify submatrices* X_{I_l,J_l}, $I_l \subset \{1,\ldots,n\}; J_l \subset \{1,\ldots,m\}$, $l = 1,...,k$, $s.t. \|X_{I_l,J_l}\| \geq t_0(\|I_l\| * \|J_l\|)$. *Here* $\|I_l\|$ *is the cardinality of the index set* I_l, t_0 *is a positive number between 0 and 1 that controls the noise level of* X_{I_l,J_l}.

Proof. $\forall l$, it suffices to let I_l be the indices of the lth column of A^* , such that $A^*_{:,l} = 1$; and let J_l be the indices of the lth row of B^* such that $B^*_{l,:} = 1$. □

Motivated by Lemma 4.3.1, instead of looking for patterns directly, we turn to identify large submatrices in X that are enriched by 1, such that each submatrix would correspond to one binary pattern or basis.

Definition 4.3.1 (**Direct consecutive-ones property, direct C1P**). *A binary matrix X has direct C1P if for each of its row vector, all 1's occur at consecutive indices.*

Definition 4.3.2 (**Simultaneous consecutive-ones property, SC1P**). *A binary matrix X has direct SC1P, if both X and X^T have direct C1P; and a binary matrix X has SC1P, if there exists a permutation of the rows and columns such that the permutated matrix has direct SC1P.*

Definition 4.3.3 (**Upper Triangular-Like matrix, UTL**). *A binary matrix $X^{m\times n}$ is called an **U**pper **T**riangular-**L**ike (**UTL**) matrix, if 1) $\sum_{\mathrm{i}=1}^{m} X_{\mathrm{i}1} \leq \sum_{\mathrm{i}=1}^{m} X_{\mathrm{i}2} \leq \cdots \leq \sum_{\mathrm{i}=1}^{m} X_{\mathrm{i}n}$; 2) $\sum_{\mathrm{j}=1}^{n} X_{1\mathrm{j}} \geq \sum_{\mathrm{j}=1}^{n} X_{2\mathrm{j}} \geq \cdots \geq \sum_{\mathrm{j}=2}^{n} X_{m\mathrm{j}}$. In other words, the matrix has non-increasing row sums from top down, and non-decreasing column sums from left to right.*

Lemma 4.3.2 (**UTL matrix with direct SC1P**). *Assume X has no all-zero rows or columns. If X is an UTL matrix and has direct SC1P, then an all 1 submatrix of the largest area in X is seeded where one of its column lies in the medium column of the matrix, or one of its row lies in the medium row of the matrix, as shown in Figure 4.2.*

Figure 4.2 presented three simplified scenarios of UTL matrix that has direct SC1P. In (a), (b), the 1's are organized in triangular shape, where certain rows in (a) and certain columns in (b) are all zero, and in (c), the 1's are shaped in block diagonal. After removing all-zero rows and columns, the upper triangular area of the shuffled matrix is dense in 1. It is easy to show that a rectangular with the largest area in a triangular is the one defined by the three midpoints of the three sides, together with the vertex of the right angle of the triangular, as colored by red in Figure 4.2. The width and height of the rectangular equal to half of the two legs of the triangular, i.e. $(\frac{m}{2}, \frac{n_0}{2})$, $(\frac{m_0}{2}, \frac{n}{2})$, $(\frac{m}{2}, \frac{n}{2})$ for the three scenarios in Figure 4.2 respectively. According to Lemma 4.3.2, this largest rectangular contains at least one row or one column (colored in yellow) of the largest all 1 submatrix in the matrix. Consequently, starting with one row or column, expansions with new rows or

columns could be done easily if they show strong similarity to the first row or column. After the expansion concludes, one could determine whether to retain the submatrix expanded row-wise or column-wise, whichever reduces more of the cost function.

It is common that the underlying SC1P pattern may not exist for a binary matrix, and we turn to find the matrix with closest SC1P.

Definition 4.3.4 (**Closest SC1P**). *Given a binary matrix X and a nonnegative weight matrix W, a matrix $\hat{X}$ that has SC1P and minimizes the distance $d_W(X, \hat{X})$ is the closest SC1P matrix of X.*

Based on Lemma 4.3.2, we could find all the submatrices in Lemma 4.3.1 by first permutating rows and columns of matrix X to be an UTL matrix with closest direct SC1P, locating the largest submatrix of all 1's, to be our first binary pattern. Then we are left with a residual matrix whose entries covered by existing patterns are set to zero. Repeat the process on the residual matrix until convergence. However, finding matrix of closest SC1P of matrix X is NP-hard [116], [117].

Lemma 4.3.3 (**Closest SC1P**). *Given a binary matrix X and a nonnegative weight matrix W, finding a matrix $\hat{X}$ that has SC1P and minimizes the distance $d_W(X, \hat{X})$ is an NP-hard problem.*

The NP-hardness of the closest SC1P problem has been shown in [116], [117]. Both exact and heuristic algorithms are known for the problem, and it has also been shown if the number of rows or columns is bounded, then solving closest SC1P requires only polynomial time [118]. In our MEBF algorithm, we attempt to address it by using heuristic methods and approximation algorithms.

4.3.2 MEBF algorithm

Overall, MEBF adopted a heuristic approach to locate submatrices that are dense in 1's iteratively. Starting with the original matrix as a residual matrix, at each iteration, MEBF permutates the rows and columns of the current residual matrix so that the 1's are gathered on entries of the upper triangular area. This step is to approximate the permutation

operation it takes to make a matrix UTL and direct SC1P. Then as illustrated in Figure 4.2 and Figure 4.3, the rectangular of the largest area in the upper triangular, and presumably, of the highest frequencies of 1's, will be captured. The pattern corresponding to this submatrix represents a good rank-1 approximation of the current residual matrix. Before the end of each iteration, the residual matrix will be updated by flipping all the 1's located in the identified submatrix in this step to be 0.

Algorithm 1 MEBF

Inputs: $X \in \{0,1\}^{n\times m}$, $t \in (0,1)$,τ
Outputs: $A^* \in \{0,1\}^{n\times k}$, $B^* \in \{0,1\}^{k\times m}$
$MEBF(X,t,\tau)$:
$X_{\text{residual}} \leftarrow X$, $\gamma_0 \leftarrow inf$
$A^* \leftarrow NULL$, $B^* \leftarrow NULL$
while $!\tau$ **do**
  $(\mathbf{a},\mathbf{b}) \leftarrow$ bidirectional_growth(X_{residual}, t)
  $A_{\text{tmp}} \leftarrow append(A^*, \mathbf{a})$
  $B_{\text{tmp}} \leftarrow append(B^*, \mathbf{b})$
  if $\gamma(A_{\text{tmp}}, B_{\text{tmp}}; X) > \gamma_0$ **then**
    $(\mathbf{a},\mathbf{b}) \leftarrow$ weak_signal_detection(X_{residual}, t) $A_{\text{tmp}} \leftarrow append(A^*, \mathbf{a})$
    $B_{\text{tmp}} \leftarrow append(B^*, \mathbf{b})$
  else if $\gamma(A_{\text{tmp}}, B_{\text{tmp}}; X) > \gamma_0$ **then**
    break
  end if
  $A^* \leftarrow append(A^*, \mathbf{a})$
  $B^* \leftarrow append(B^*, \mathbf{b})$
  $\gamma_0 \leftarrow \gamma(A^*, B^*; X)$
  $X_{\text{residual}_{\text{ij}}} \leftarrow 0$ when $(\mathbf{a} \otimes \mathbf{b})_{\text{ij}} = 1$
end while

Shown in Figure 4.3a, for an input Boolean matrix (a1), MEBF first rearranges the matrix to obtain an approximate UTL matrix with closest direct SC1P. This was achieved by reordering the rows so that the row norms are non-increasing, and the columns so that the column norms are non-decreasing (a2). Then, MEBF takes either the column or row with medium number of 1's as one basis or pattern (a3). As the name reveals, MEBF then adopts a median expansion step, where the medium column or row would propogate to other columns or rows with a bidirectional growth algorithm until certain stopping criteria is met. Whether to choose the pattern expanded row-wise or column-wise depends on which one minimizes the

cost function with regards to the current residual matrix. Before the end of each iteration, MEBF computes a residual matrix by doing a Boolean subtraction of the newly selected rank-1 pattern matrix from the current residual matrix (a4). This process continues until the convergence criteria was met. If the patterns identified by the bidirectional growth step stopped deceasing the cost function before the convergence criteria was met, another step called weak signal detection would be conducted (a6,a7). Figure 4.3b illustrated a special case, where the permutated matrix is roughly block diagonal (b1), which corresponds to the third scenario in Figure 4.2. The same procedure as shown in 3a could guarantee the accurate location of all the patterns. The computational complexity of bidirectional growth and weak signal detection algorithms are both O(nm) and the complexity of each iteration of MEBF is O(nm). The main algorithm of MEBF is illustrated below:

4.3.3 Bidirectional Growth

For an input binary (residual) matrix X, we first rearrange X by reordering the rows and columns so that the row norms are non-increasing, and the column norms are non-decreasing. The rearranged X, after removing its all-zero columns and rows, is denoted as X', the median column and median row of X' as $X'_{:,\text{med}}$ and $X'_{\text{med},:}$. Denote $X_{:,(\text{med})}$ and $X_{(\text{med}),:}$ as the column and row in X corresponding to $X'_{:,\text{med}}$ and $X'_{\text{med},:}$. The similarity between $X_{:,(\text{med})}$ and columns of X can be computed as a column wise correlation vector $\mathbf{m} \in (0,1)^m$, where $\mathbf{m}_\text{i} = \frac{<X_{:,\text{i}},X_{:,(\text{med})}>}{<X_{:,(\text{med})},X_{:,(\text{med})}>}$. Similarly, the similarity between $X_{(\text{med}),:}$ and rows of X can be computed as a vector $\mathbf{n} \in (0,1)^n$,$\mathbf{n}_\text{j} = \frac{<X_{\text{j},:},X_{(\text{med}),:}>}{<X_{(\text{med}),:},X_{(\text{med}),:}>}$. A pre-specified threshold $t \in (0,1)$ was further applied, and two vectors $\mathbf{e}$ and $\mathbf{f}$ indicating the similarity strength of the columns and rows of X with $X_{:,(\text{med})}$ and $X_{(\text{med}),:}$, are obtained, where $\mathbf{e}_\text{j} = (\mathbf{m}_\text{j} > t)$ and $\mathbf{f}_\text{i} = (\mathbf{n}_\text{i} > t)$. Here the binary vectors $\mathbf{e}$ and **$\mathbf{f}$ each represent one potential BMF pattern** . In each iteration, we select the row or column pattern whichever fits the current residual matrix better, i.e. the column pattern if $\gamma(X_{:,(\text{med})}, \mathbf{e}; X) < \gamma(\mathbf{f}, X_{(\text{med}),:}; X)$, or the row pattern otherwise. Here, the cost function is defined as $\gamma(\mathbf{a}, \mathbf{b}; X) = \|X \ominus (\mathbf{a} \otimes \mathbf{b})\|$. This is equivalent to selecting a pattern that achieves lower overall cost function at the current step. Obviously here, a smaller t could achieve higher coverage with less number of patterns,

while a larger t enables a more sparse decomposition of the input matrix with greater number of patterns. Patterns found by bidirectional growth do not guarantee a constant decrease of the cost function. In the case the cost function increases, we adopt a weak signal detection step before stopping the algorithm.

Algorithm 2 Bidirectional Growth

Inputs: $X \in \{0,1\}^{n\times m}$, $t \in (0,1]$
Outputs: $(\mathbf{a},\mathbf{b})$
bidirectional_growth(X,t) :
$X' \leftarrow$ **UTL operation on** X
$\mathbf{d} \leftarrow X_{:,(\text{med})}$, $\mathbf{e} \leftarrow \{(\frac{<X_{:,\text{j}},\mathbf{d}>}{<\mathbf{d},\mathbf{d}>} > t), \text{j} = 1,...,m\}$
$\mathbf{f} \leftarrow X_{(\text{med}),:}$, $\mathbf{g} \leftarrow \{(\frac{<X_{\text{i},:},\mathbf{f}>}{<\mathbf{f},\mathbf{f}>} > t), \text{i} = 1,...,n\}$
if $\gamma(\mathbf{d},\mathbf{e};X) > \gamma(\mathbf{g},\mathbf{f};X)$ **then**
  $\mathbf{a} \leftarrow \mathbf{g}$; $\mathbf{b} \leftarrow \mathbf{f}$
else
  $\mathbf{a} \leftarrow \mathbf{d}$; $\mathbf{b} \leftarrow \mathbf{e}$
end if

4.3.4 Weak Signal Detection Algorithm

Algorithm 3 Weak Signal Detection

Inputs: $X \in \{0,1\}^{n\times m}$, $t \in (0,1]$
Outputs: $(\mathbf{a},\mathbf{b})$
Weak_signal_detection(X,t)
$X' \leftarrow$ **UTL operation on** X
$\mathbf{d}^1 \leftarrow X'_{:,m} \wedge X'_{:,m-1}$
$\mathbf{e}^1 \leftarrow \{(\frac{<X_{:,\text{j}},\mathbf{d}^1>}{<\mathbf{d}^1,\mathbf{d}^1>} > t), \text{j} = 1,...,m\}$
$\mathbf{e}^2 \leftarrow X'_{1,:} \wedge X'_{2,:}$
$\mathbf{d}^2 \leftarrow \{(\frac{<X_{\text{i},:},\mathbf{e}^2>}{<\mathbf{e}^2,\mathbf{e}^2>} > t), \text{i} = 1,...,n\}$
$l \leftarrow \arg\min_{l=1,2} \gamma(\mathbf{d}^l,\mathbf{e}^l,X)$
$\mathbf{a} \leftarrow \mathbf{d}^l$; $\mathbf{b} \leftarrow \mathbf{e}^l$

The bidirectional growth steps do not guarantee a constant decrease of the cost function, especially when after the "large" patterns have been identified and the "small" patterns are easily confused with noise. To identify weak patterns from a residual matrix, we came up with a week signal detection algorithm to locate the regions that may still have small but

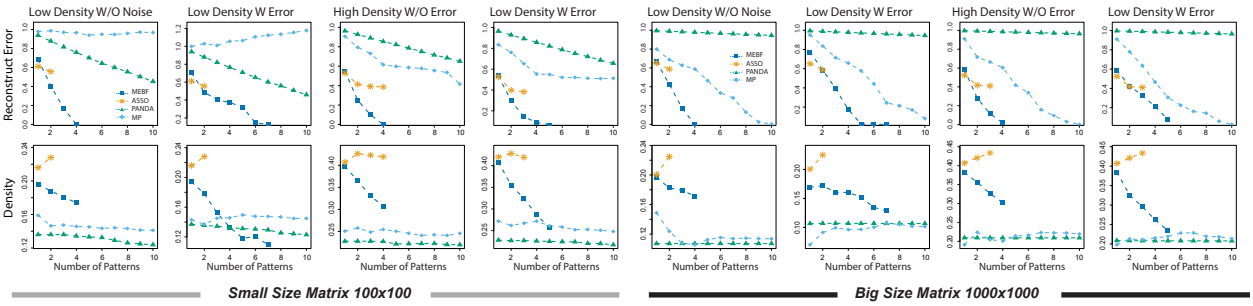

Figure 4.4. Performance comparisons of MEBF, ASSO, PANDA and MP on the accuracy of decomposition.

true patterns. Here, from the current residual matrix, we search the two columns with the most number of 1's and form a new column that is the intersection of the two columns; and the two rows with the most number of 1's and form a new row that is the intersection of the two rows. Starting from the new column and new row as a pattern, similar to bidirectional growth, we locate the rows or columns in the residual matrix that have high enough similarity to the pattern, thus expanding a single row or column into a submatrix. The one pattern among the two with the lowest cost function with regards to the residual matrix will be selected. And if addition of the pattern to existing patterns could decrease the cost function with regards to the original matrix, it will be retained. Otherwise, the algorithm will stop.

4.4 Experiment

4.4.1 Simulation data

We first compared MEBF[1] with three state-of-the-art approaches, ASSO, PANDA and Message Passing (MP), on simulated datasets.

A binary matrix $X^{n\times m}$ is simulated as

$$X^{n\times m} = U^{n\times k} \otimes V^{k\times m} +_f E$$

where

$$U_{\text{ij}}, V_{\text{ij}} \sim Bernoulli(p_0) \quad E_{\text{ij}} \sim Bernoulli(p)$$

[1]↑The code is available at https://github.com/clwan/MEBF

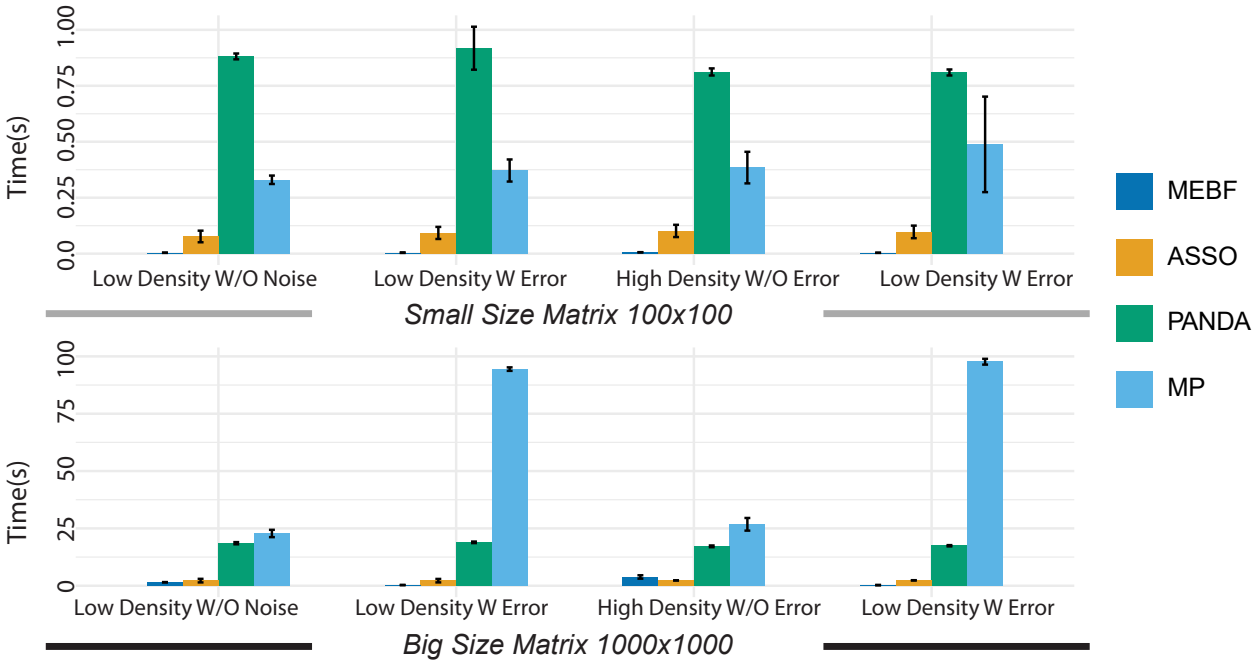

Figure 4.5. Performance comparison of MEBF, ASSO, PANDA and MP on computational cost.

" $+_f$ " is a flipping operation, s.t.

$$X_{\mathrm{ij}} = \begin{cases} \vee_{l=1}^{k} U_{\mathrm{il}} \wedge V_{\mathrm{lj}}, & E_{\mathrm{ij}} = 0 \\ \neg \vee_{l=1}^{k} U_{\mathrm{il}} \wedge V_{\mathrm{lj}}, & E_{\mathrm{ij}} = 1 \end{cases}$$

Here, p_0 controls the density levels of the true patterns, and E is introduced as noise that could flip the binary values, and the level of noise could be regulated by the parameter p. We simulated two data scales, a small one, $n = m = 100$, and a large one $n = m = 1000$. For each data scale, the number of patterns k, is set to 5, and we used two density levels, where $p_0 = 0.2, 0.4$, and two noise levels $p = 0, 0.01$. 50 simulation was done for each data scale at each scenario.

We evaluate the goodness of the algorithms by considering two metrics, namely the reconstruction error and density [115], [119], as defined below:

$$\text{Reconstruction error} := \frac{\|(U \otimes V) \ominus (A^* \otimes B^*)\|}{\|U \otimes V\|}$$

$$\text{Density} := \frac{\|A^{*n\times k}\| + \|B^{*k\times m}\|}{(n+m)\times k}$$

Here, U, V are the ground truth patterns while A^* and B^* are the decomposed patterns by each algorithm. The density metric is introduced to evaluate whether the decomposed patterns could reflect the sparsity/density levels of the true patterns. It is notable that with the same reconstruction error, patterns of lower density, i.e., higher sparsity are more desirable, as it leads to more parsimonious models.

In Figure 4.4 and 4.5, we show that, compared with ASSO, PANDA and MP, MEBF is the fastest and most robust algorithm. Here, the convergence criteria for the algorithms are set as: (1) 10 patterns were identified; (2) or for MEBF, PANDA and ASSO, they will also stop if a newly identified pattern does not decrease the cost function.

As shown in Figure 4.4, MEBF has the best performance on small and big sized matrices for all the four different scenarios, on 50 simulations each. It achieved the lowest reconstructed error with the least computation time compared with all other algorithms. The convergence rate of MEBF also outperforms PANDA and MP. Though ASSO converges early with the least number of patterns, its reconstruction error is considerably higher than MEBF, especially for high density matrices. In addition, ASSO derived patterns tend to be more dense than the true patterns, while those derived from the other three methods have similar density levels with the true patterns. By increasing the number of patterns, PANDA stably decreased reconstruction error, but it has a considerably slow convergence rate and high computation cost. MP suffered in fitting small size matrices, and in the case of low density matrix with noise, MP derived patterns would not converge. The standard deviations of reconstruction error and density across 50 simulations is quite low, and was demonstrated by the size of the shapes. The computational cost and its standard deviation for each algorithm is shown as bar plots in Figure 4.5, and clearly, MEBF has the best computational efficiency among all.

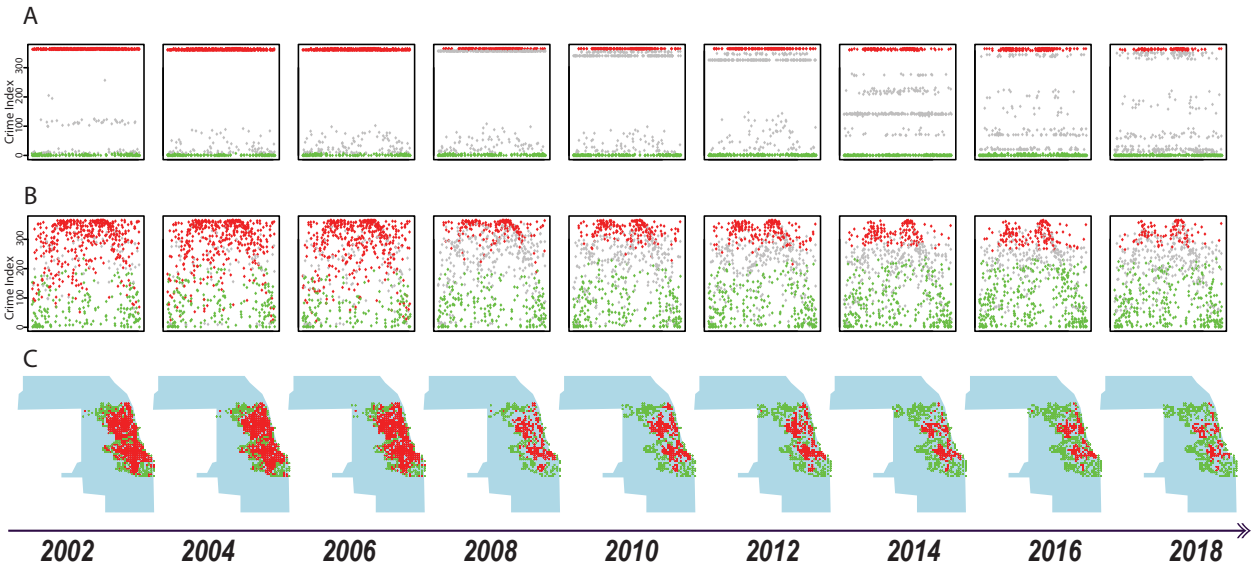

Figure 4.6. MEBF analysis of Chicago crime data over the years.

4.4.2 Real world data application

We next focus on the application of MEBF on two real world datasets, and its performance comparison with MP. Both datasets, *Chicago Crime records*[2] ($X \in \{0,1\}^{6787\times752}$) and *Head and Neck Cancer Single Cell RNA Sequencing data*[3] ($X \in \{0,1\}^{344\times5902}$), are publicly available. The computational cost of ASSO and PANDA are too inhibitive to be applied to datasets of such a large scale, so they were dropped from the comparisons. Due to a lack of ground-truth of the two low rank construction matrices and the possible high noise level in the real world datasets, it may not be reasonable to examine the reconstruction error with respect to the original matrix. Instead, we focused on two metrics, the density and coverage levels. Density metric was defined as in the simulation data, and coverage rate is defined as

$$\text{Coverage rate} := \frac{\|(X \cdot (A^* \otimes B^*))\|}{\|X\|}$$

With the same reconstruction error, binary patterns are more desirable to have high sparsity, meaning low density levels, as it makes further interpretation easier and avoids

[2]↑Chicago crime records downloaded on August 20, 2019 from https://data.cityofchicago.org/Public-Safety
[3]↑This head and neck sequencing data can be accessed at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE103322

Table 4.1. Comparison of MEBF and MP on real world data

	$\text{Coverage}_{(\text{MEBF/MP})}$	$\text{Density}_{(\text{MEBF/MP})}$	$\text{Time/s}_{(\text{MEBF/MP})}$
$\text{Crime}_{\text{k=5}}$	0.835/0.812	0.019/0.027	2.913/333.601
$\text{Crime}_{\text{k=20}}$	0.891/0.807	0.030/0.066	10.608/992.011
$\text{Single cell}_{\text{k=5}}$	0.496/0.463	2.06e-4/2.86e-4	1.846/137.623
$\text{Single cell}_{\text{k=20}}$	0.626/0.580	3.34e-4/7.22e-4	5.954/390.217

possible overfitting. On the other hand, in many real world data, 0 is more likely to be a false negative occurrence, compared with 1 being a false positive occurrence. In this regard, a higher coverage rate, meaning higher recovery of the 1's, would be a more reasonable metric than lower reconstruction error to the noisy original matrix, as 0's are more likely to be noisy observations than 1's.

We compared MEBF and MP for $k = 5$ and $k = 20$, and the density and coverage rate of the derived patterns and computation time of the two algorithms are presented in Table 4.1. Overall, as shown in Table 4.1, for both $k = 5$ and $k = 20$, MEBF outperforms MP in all three measures higher coverage rate, roughly half the density levels to MP, and the computation time of MEBF is approximately 1% to that of MP.

Next we discuss in detail the application of BMF on discrete data mining and continuous data mining, and present the findings on the two datasets using MEBF.

Discrete data mining

Chicago is the most populous city in the US Midwest, and it has one of the highest crime rates in the US. It has been well understood that the majority of crimes such as theft and robbery have strong date patterns. For example, crimes were committed for the need to repay regular debt like credit cards, which has a strong date pattern in each month. Here we apply MEBF in analyzing Chicago crime data from 2001 to 2019 to find crime patterns on time and date for different regions. The crime patterns is useful for the allocation of police force, and could also reflect the overall standard of living situation of regions in general.

We divided Chicago area into ~ 800 regions of roughly equal sizes. For each of the 19 years, a binary matrix $X_d^{n\times m}$ for the dth year is constructed, where n is the total dates in each year and m represents the total number of regions, and $X_{d_{\mathrm{i,j}}} = 1$ means that crime was reported at date i in region j in year d, $X_{d_{\mathrm{i,j}}} = 0$ otherwise. MEBF was then applied on each of the constructed binary matrices with parameters $(t = 0.7, k = 20)$ and outputs $A_d^{n\times k}$ and $B_d^{k\times m}$. The reconstructed binary matrix is accordingly calculated as $X_{d^*} = A_d^{n\times k} \otimes B_d^{k\times m}$. A crime index was defined as the total days with crime committed for regions j in year d, $C_{d_{\mathrm{j}}} = \sum_{\mathrm{i}=1}^{m} X_{d_{\mathrm{i,j}}}$.

Figure 4.6 shows the crime patterns over time, and only even years were shown due to space limit. In Figure 4.6A, from year 2002-2018, the crime index calculated from the reconstructed matrix, namely, $C_{d^*_{\mathrm{j}}} = \sum_{\mathrm{i}=1}^{m} X_{d^*_{\mathrm{i,j}}}$ was shown on the y-axis for all the regions on x-axis, In Figure 4.6A, points colored in red indicate those regions with crime index equal to total dates of the year, i.e., 365 or 366, meaning these regions are heavily plagued with crimes, such that there is no day without crime committed. Points colored in green shows vice versa, indicating those regions with no crimes committed over the year. Points are otherwise colored in gray. Figure 4.6B shows the crime index on the original matrix, and clearly, the green and red regions are distinctly separated, i.e. green on the bottom with low crime index, and red on the top with high crime index. This shows the consistency of the crime patterns between the reconstructed and original crime data, and thus, validate the effectiveness of MEBF pattern mining. Notably, the dramatic decrease in crime index starting from 2008 as shown in figure 4.6A and B correlates with the reported crime decrease in Chicago area since 2008. figure 4.6C shows the crime trend over the years on the map of Chicago. Clearly, from 2008 to now, there is a gradual increase in the green regions, and decrease in the red regions, indicating an overall good transformation for Chicago. This result also indicate that more police force could be allocated in between red and green regions when available.

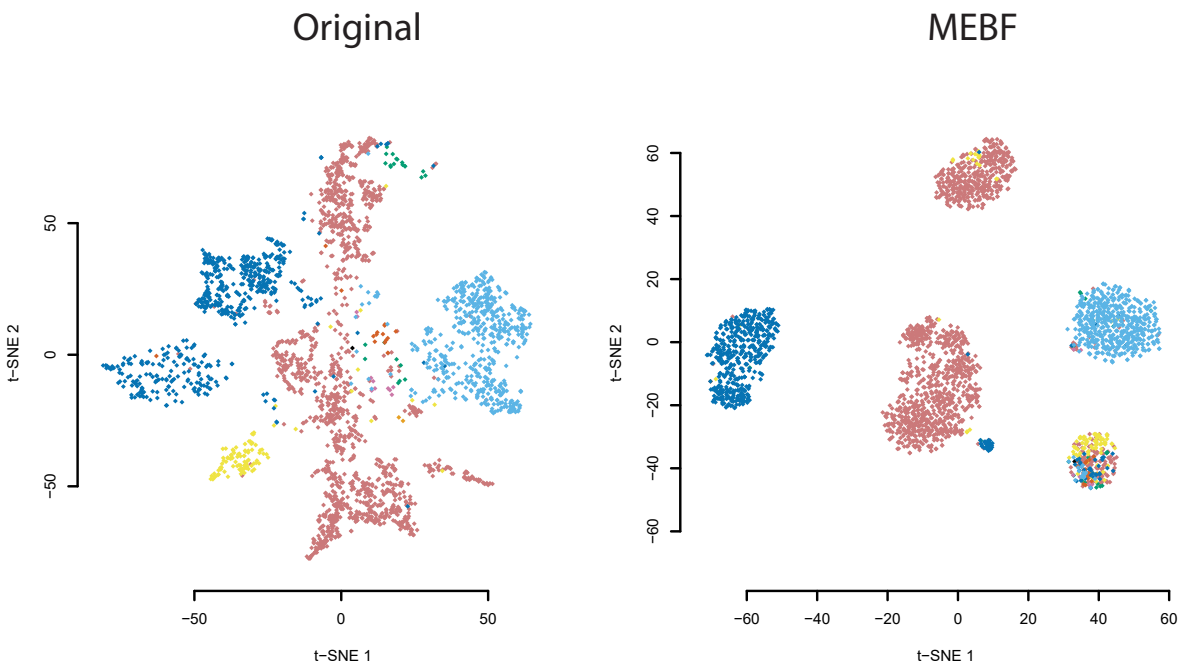

Figure 4.7. Visualization of single cell clustering before and after MEBF.

Continuous data denoising

Binary matrix factorization could also be helpful in continuous matrix factorization, as the Boolean rank of a matrix could be much smaller than the matrix rank under linear algebra. Recently, clustering of single cells using single-cell RNA sequencing data has gained extensive utilities in many fields, wherein the biggest challenge is that the high dimensional gene features, mostly noise features, makes the distance measure of single cells in clustering algorithm highly unreliable. Here we aim to use MEBF to denoise the continuous matrix while clustering.

We collected a single cell RNA sequencing (scRNAseq) data [24], that measured more than 300 gene expression features for over 5,000 single cells, i.e., $X^{5000\times 300}$. We first binarize original matrix X into X^*, s.t. $X^*_{ij} = 1$ where $X_{ij} > 0$,and $X^*_{ij} = 0$ otherwise. Then, applying MEBF on X^* with parameters $(t = 0.6, k = 5)$ outputs $A^{n\times k}$, $B^{k\times m}$. Let $X^{**} = A \otimes B$ and X_{use} be the inner product of X^* and X^{**}, namely, $X_{use} = X \circledast X^{**}$. X_{use} represents a denoised version of X, by retaining only the entries in X with true non-zero gene expressions. And this is reconstructed from the hidden patterns extracted by MEBF.

As shown in Figure 4.7, clustering on the denoised expression matrix, X_{use}, results in much tighter and well separated clusters (right) than that on the original expression matrix (left), as visualized by t-SNE plots shown in Figure 4.7. t-SNE is an non-linear dimensional reduction approach for the visualization of high dimensional data [120]. It is worth noting that, in generating Figure 4.7, Boolean rank of 5 was chosen for the factorization, indicating that the heterogeneity among cell types with such a high dimensional feature space could be well captured by matrices of Boolean rank equal to 5. Interestingly, we could see a small portion of fibroblast cell (dark blue) lies much closer to cancer cells (red) than to the majority of the fibroblast population, which could biologically indicate a strong cancer-fibroblast interaction in cancer micro-environment. Unfortunately, such interaction is not easily visible in the clustering plot using original noisy matrix.

4.5 Discussion

We sought to develop a fast and efficient algorithm for boolean matrix factorization, and adopted a heuristic approach to locate submatrices that are dense in 1's iteratively, where each such submatrix corresponds to one binary pattern in BMF. The submatrix identification was inspired by binary matrix permutation theory and geometric segmentation. Approximately, we permutate rows and columns of the input matrix so that the 1's could be "driven" to the upper triangular of the matrix as much as possible, and a dense submatrix could be more easily geometrically located. Compared with three state of the art methods, ASSO, PANDA and MP, MEBF achieved lower reconstruction error, better density and much higher computational efficiency on simulation data of an array of situations. Additionally, we demonstrated the use of MEBF on discrete data pattern mining and continuous data denoising, where in both case, MEBF generated consistent and robust findings.

In the next chapter, we will extend this framework for Binary tensor data, which could mimic complicated biological data scenarios.

5. GEOMETRIC ALL-WAY BOOLEAN TENSOR DECOMPOSITION

5.1 Overview

A tensor is a multi-dimensional array that can effectively capture the complex multidimensional features. A Boolean tensor is a tensor that assumes binary values endowed with the Boolean algebra. Boolean tensor has been widely adopted in many fields, including knowledge graph, recommendation system, spatial-temporal data etc [121]–[125]. Tensor decomposition is a powerful tool in extracting meaningful latent structures in the data, for which the popular CANDECOMP/PARAFAC (CP) decomposition is a generalization of the matrix singular value decomposition to tensor [126]. However, these algorithms are not directly usable for Boolean tensors. In this study, we focus on Boolean tensor decomposition (BTD) under similar framework to the CP decomposition.

As illustrated in Figure 5.1, BTD factorizes a binary tensor $\mathcal{X}$ as the Boolean sum of multiple rank 1 tensors. In cases when the error distribution of the tensor data is hard to model, BTD applied to binarized data can retrieve more desirable patterns with better interpretation than regular tensor decomposition [115], [127]. This is probably due to the robustness of logic representation of BTD. BTD is an NP-hard problem [127]. Existing BTD methods suffer from low efficiency due to high space/time complexity, and particularly, most BTD algorithms adopted a least square updating approach with substantially high computational cost [128], [129]. This has hindered their application to either large scale datasets, such as social network or genomics data, or tensors of high-order.

We proposed an efficient BTD algorithm motivated by the geometric underpinning of rank-1 tensor bases, namely GETF (Geometric Expansion for all-order Tensor Factorization). To the best of our knowledge, GETF is the first algorithm that can efficiently deal with all-order Boolean tensor decomposition with an $O(n)$ complexity, where n represents the total number of entries in a tensor. Supported by rigorous theoretical analysis, GETF solves the BTD problem via sequentially identifying the fibers that most likely coincides with a rank-1 tensor basis component. Our synthetic and real-world data based experiments validated the high accuracy of GETF and its drastically improved efficiency compared with existing

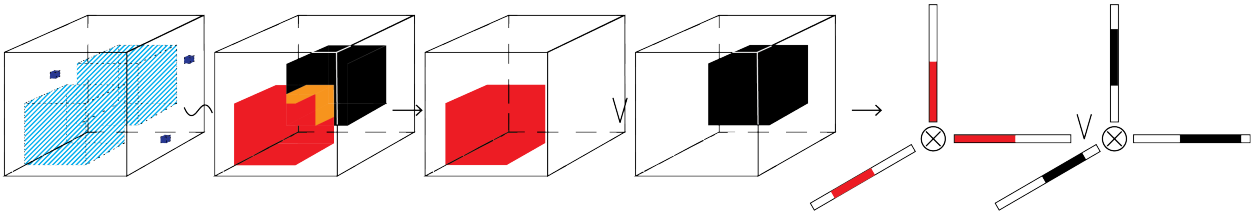

Figure 5.1. Boolean tensor decomposition

methods, in addition to its potential utilization on large scale or high order data, such as complex relational or spatial-temporal data. The key contributions of this study include: (1) Our proposed GETF is the first method capable of all-order Boolean tensor decomposition; (2) GETF has substantially increased accuracy in identifying true rank-1 patterns, with less than a tenth of the computational cost compared with state-of-the-art methods; (3) we provided thorough theoretical foundations for the geometric properties for the BTD problem.

5.2 Preliminaries

5.2.1 Notations

Notations in this study follow those in [130]. We denote the *order* of a tensor as k, which is also called *ways* or *modes*. Scalar value, vector, matrix, and higher order tensor are represented as lowercase character x, bold lowercase character $\mathbf{x}$, uppercase character X, and Euler script $\mathcal{X}$, respectively. Super script with mark $\times$ indicates the size and dimension of a vector, matrix or tensor while subscript specifies an entry. Specifically, a k-order tensor is denoted as $\mathcal{X}^{m_1 \times m_2 \ldots \times m_k}$ and the entry of position $\mathrm{i}_1, \mathrm{i}_2, , \mathrm{i}_k$ is represented as $\mathcal{X}_{\mathrm{i}_1 \mathrm{i}_2 \mathrm{i}_k}$. For a 3-order tensor, we denote its fibers as $\mathcal{X}_{:\mathrm{i}_2\mathrm{i}_3}$, $\mathcal{X}_{\mathrm{i}_1:\mathrm{i}_3}$ or $\mathcal{X}_{\mathrm{i}_1\mathrm{i}_2:}$ and its slices $\mathcal{X}_{\mathrm{i}_1::}$, $\mathcal{X}_{:\mathrm{i}_2:}$, $\mathcal{X}_{::\mathrm{i}_3}$. For a k-order tensor, we denote its mode-p fiber as $\mathcal{X}_{\mathrm{i}_1 \ldots \mathrm{i}_{p-1}:\mathrm{i}_{p+1} \ldots \mathrm{i}_k}$ with all indices fixed except for i_p. $\|\mathcal{X}\|$ represents the norm of a tensor, and $\|\mathcal{X}\|$ the L_1 norm in particular. The basic Boolean operations include $\wedge(and, 1 \wedge 1 = 1, 1 \wedge 0 = 0, 0 \wedge 0 = 0)$, $\vee(or, 1 \vee 1 = 1, 1 \vee 0 = 1, 0 \vee 0 = 0)$, and $\neg(not, \neg 1 = 0, \neg 0 = 1)$. Boolean entry-wise sum, subtraction and product of two matrices are denoted as $A \oplus B = A \vee B$, $A \ominus B = (A \wedge \neg B) \vee (\neg A \wedge B)$ and $A \circledast B = A \wedge B$. The outer Boolean product in this paper is considered as the addition of rank-1 tensors, which follows the scope of CP decomposition [126]. Specifically, a three-order Rank-1 tensor

can be represented as the Boolean outer product of three vectors, i.e. $\mathcal{X}^{m_1 \times m_2 \times m_3} = \mathbf{a}^{m_1} \otimes \mathbf{b}^{m_2} \otimes \mathbf{c}^{m_3}$. Similarly, for higher order tensor, $\mathcal{X}^{m_1 \times m_2 \ldots \times m_k}$ of rank l is the outer product of $A^{m_1 \times l,1}, A^{m_2 \times l,2}, ..., A^{m_k \times l,k}$, i.e. $\mathcal{X}^{m_1 \times m_2 \ldots \times m_k} = \vee_{\mathrm{j}=1}^{l}(A_{:\mathrm{j}}^{m_1 \times l,1} \otimes A_{:\mathrm{j}}^{m_2 \times l,2} \ldots \otimes A_{:\mathrm{j}}^{m_k \times l,k})$ and $\mathcal{X}_{\mathrm{i}_1 \mathrm{i}_2, \ldots, \mathrm{i}_k} = \vee_{\mathrm{j}=1}^{l}(A_{\mathrm{i}_1 \mathrm{j}}^{m_1 \times l,1} \wedge A_{\mathrm{i}_2 \mathrm{j}}^{m_2 \times l,2} \ldots A_{\mathrm{i}_k \mathrm{j}}^{m_k \times l,k})$, $\mathrm{j} = 1 \ldots l$ represents the rank-1 tensor components of a rank l CP decomposition of $\mathcal{X}$. In this paper, we denote $A^{m_\mathrm{i} \times l,\mathrm{i}}$, $\mathrm{i} = 1 \ldots k$ as the pattern matrix of the ith order of $\mathcal{X}$, its jth column $A_{:\mathrm{j}}^{m_\mathrm{i} \times l,\mathrm{i}}$ as the jth pattern fiber of the ith order, and $A_{:\mathrm{j}}^{m_1 \times l,1} \otimes A_{:\mathrm{j}}^{m_2 \times l,2} \ldots \otimes A_{:\mathrm{j}}^{m_k \times l,k}$ as the j-th rank-1 tensor pattern.

5.2.2 Problem statement

As illustrated in Figure 5.1, for a binary k-order tensor $\mathcal{X} \in \{0,1\}^{m_1 \times m_2 \ldots \times m_k}$ and a convergence criteria parameter τ, the Boolean tensor decomposition problem is to identify low rank binary pattern matrices $A^{m_1 \times l,1*}, A^{m_2 \times l,2*}, \ldots A^{m_k \times l,k*}$, the outer product of which best fit $\mathcal{X}$, where $A^{m_1 \times l,1*}, A^{m_2 \times l,2*}, ..., A^{m_k \times l,k*}$ are matrices of l columns. In other words, $(A^{m_1 \times l,1*}, A^{m_2 \times l,2*}, \ldots A^{m_k \times l,k*}) = argmin_{A^1, A^2, \ldots, A^k}(\gamma(A^{m_1 \times l,1}, A^{m_2 \times l,2}, ..., A^{m_k \times l,k}; \mathcal{X}) \| \tau)$ Here $\gamma(A^{m_1 \times l,1}, A^{m_2 \times l,2}, \ldots A^{m_k \times l,k}; \mathcal{X})$ is the cost function. In general, γ is defined to the reconstruction error $\gamma(A^{m_1 \times l,1*}, \ldots A^{m_k \times l,k*}; \mathcal{X}) = \| \mathcal{X} \ominus (A^{m_1 \times l,1*} \otimes \ldots \otimes A^{m_k \times l,k*}) \|_{L_p}$, and p is usually set to be 1.

5.2.3 Related work

In order of difficulty, Boolean tensor decomposition consists of three major tasks, Boolean matrix factorization (BMF, $k = 2$) [108], three-way Boolean tensor decomposition (BTD, $k = 3$) and higher order Boolean tensor decomposition (HBTD, $k > 3$) [131]. All of them are NP hard [127]. Numerous heuristic solutions for the BMF and BTD problems have been developed in the past two decades [25], [109]–[112], [128], [132], [133].

For BTD, Miettinen et al thoroughly defined the BTD problem ($k = 3$) in 2011 [132], and proposed the use of least square update as a heuristic solution. To solve the scalability issue, they further developed Walk'N'Merge, which applies random walk over a graph in identifying dense blocks as proxies of rank 1 tensors [133]. Despite the increase of scalability, Walk'N'Merge tends to pick up many small patterns, the addition of which doesn't necessarily

decrease the loss function by much. The DBTF algorithm introduced by Park et al. is a parallel distributed implementation of alternative least square update based on Khatri-Rao matrix product [129]. Though DBTF reduced the high computational cost, its space complexity increases exponentially with the increase of tensor orders due to the khatri-Rao product operation. Recently, Tammo et al. proposed a probabilistic solution to BTD, called Logistical Or Machine (LOM), with improved fitting accuracy, robustness to noises, and acceptable computational complexity [122]. However, the high number of iterations it takes to achieve convergence of the likelihood function makes LOM prohibitive to large data analysis. Most importantly, to the best of our knowledge, none of the existing algorithms are designed to handle the HBTD problem for higher order tensors.

5.3 GETF Algorithm and Analysis

GETF[1] identifies the rank-1 patterns sequentially: it first extracts one pattern from the tensor; and the subsequent patterns will be extracted sequentially from the residual tensor after removing the preceding patterns. We first derive the theoretical foundation of GETF. We show that the geometric property of the largest rank-1 pattern in a binary matrix developed in [121] can be naturally extended to higher order tensor. We demonstrated the true pattern fiber of the largest pattern can be effectively distinguished from fibers of overlapped patterns or errors by reordering the tensor to maximize its overlap with a left-triangular-like tensor. Based on this idea, the most likely pattern fibers can be directly identified by a newly develop geometric folding approach that circumvents heuristic greedy searching or alternative least square based optimization.

5.3.1 Theoretical analysis

We first give necessary definitions of the slice, re-order and sum operations on a k order tensor and an theoretical analysis of the property of a left-triangular-like (LTL) tensor.

Definition 5.3.1. *(p-order slice). The p-order slice of a tensor $\mathcal{X}^{m_1 \times \ldots \times m_k}$ indexed by $\mathbb{P}$ is defined by $\mathcal{X}_{\mathrm{i}_1,\ldots,\mathrm{i}_k}$, where i_k is a fixed value $\in \{1,\ldots,m_k\}$ if $k \in \bar{\mathbb{P}}$, and i_k is unfixed* ($\mathrm{i}_k =:$)

[1]↑Code can be accessed at https://github.com/clwan/GETF

if $k \in \mathbb{P}$, here $p = \|\mathbb{P}\|$ and $\bar{\mathbb{P}} = \{1, ..., k\} \setminus \mathbb{P}$. Specifically, we denote a $\|\mathbb{P}\|$ order slice of $\mathcal{X}^{m_1 \times ... \times m_k}$ with the index set $\|\mathbb{P}\|$ unfixed as $\mathcal{X}^{m_1 \times ... \times m_k}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}$ or $\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}$, in which $\mathbb{P}$ is the unfixed index set and $I_{\bar{\mathbb{P}}}$ are fixed indices.

Definition 5.3.2. *(Index Reordering Transformation, IRT). The index reordering transformation (IRT) transforms a tensor $\mathcal{X}^{m_1 \times, ... \times m_k}$ to $\tilde{\mathcal{X}} = \mathcal{X}_{P_1, P_2, ..., P_k}$, where $P_1, ..., P_k$ are any permutation of the index sets, $\{1, ..., m_1\}, ..., \{1, ..., m_k\}$.*

Definition 5.3.3. *(Tensor slice sum). The tensor slice sum of a k-order tensor $\mathcal{X}^{m_1 \times ... \times m_k}$ with respect to the index set $\mathbb{P}$ is defined as*

$$T_{sum}(\mathcal{X}, \mathbb{P}) \triangleq \sum_{\mathrm{i}_1=1}^{m_{\mathrm{i}_1}} ... \sum_{\mathrm{i}_{\|\mathbb{P}\|}=1}^{m_{\mathrm{i}_{\|\mathbb{P}\|}}} \mathcal{X}_{:...:\mathrm{i}_1:...:\mathrm{i}_{\|\mathbb{P}\|}:...:}, \mathrm{i}_1, ..., \mathrm{i}_{\|\mathbb{P}\|} \in \mathbb{P}.$$

$T_{sum}(\mathcal{X}, \mathbb{P})$ results in a $k - \|\mathbb{P}\|$ order tensor.

Definition 5.3.4. *(p-left-triangular-like, p-LTL). A k-order tensor $\mathcal{X}^{m_1 \times ... \times m_k}$ is called p-LTL, if any of its p-order slice, $\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}$, $\mathbb{P} \subset \{1, ..., k\}$ and $\|\mathbb{P}\| = p$, and $\forall \mathrm{j} \in \mathbb{P}, 1 \leq \mathrm{j}_1 < \mathrm{j}_2 \leq m_{\mathrm{j}}$, $T_{sum}(\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}, \mathbb{P} \setminus \{\mathrm{j}\})_{\mathrm{j}_1} \leq T_{sum}(\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}, \mathbb{P} \setminus \{\mathrm{j}\})_{\mathrm{j}_2}$.*

Definition 5.3.5. *(flat 2-LTL), A k-order 2-LTL tensor $\mathcal{X}^{m_1 \times ... \times m_k}$ is called flat 2-LTL within an error range ϵ, if any of its 2-order slice, $\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}$, $\mathbb{P} \subset \{1, ..., k\}$ and $\|\mathbb{P}\| = p$, and $\forall \mathrm{j} \in \mathbb{P}, 1 \leq \mathrm{j}_1 < \mathrm{j}_2 \leq m_{\mathrm{j}}$, $\|T_{sum}(\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}, \mathbb{P} \setminus \{\mathrm{j}\})_{\mathrm{j}_1} + T_{sum}(\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}, \mathbb{P} \setminus \{\mathrm{j}\})_{\mathrm{j}_2} - 2T_{sum}(\mathcal{X}_{\mathbb{P}, I_{\bar{\mathbb{P}}}}, \mathbb{P} \setminus \{\mathrm{j}\})_{(\mathrm{j}_1 + \mathrm{j}_2)/2}\| < \epsilon$*

The **Definition** 5.3.5 indicates the tensor sum of over any 2-order slice of a flat 2-LTL tensor is close enough to a linear function with the largest error less than ϵ. Figure 5.2A,C illustrate two examples of flat 2-LTL matrix and 2-LTL 3-order tensor. By the definition, the non-right angle side of a flat 2-LTL k-order tensor is close to a $k-1$ dimension plane, which is specifically called as the **k-1 dimension plane** of the flat 2-LTL tensor in the rest part of this paper.

Lemma 5.3.1 (Geometric segmenting of a flat 2-LTL tensor)**.** *Assume $\mathcal{X}$ is a k-order flat 2-LTL tensor and $\mathcal{X}$ has none zero fibers. Then the largest rank-1 subarray in $\mathcal{X}$ is seeded*

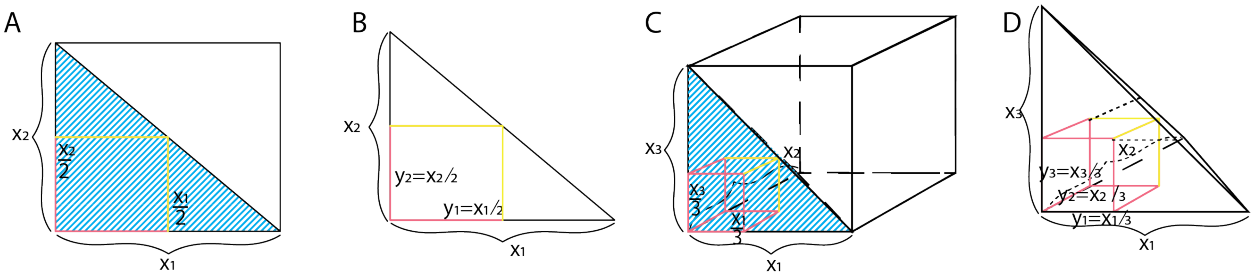

Figure 5.2. Optimal rank 1 subarray

where one of the pattern fibers is paralleled with the fiber that anchored on the $1/k$ *segmenting point (entry* $\{\|m_1\|/k, \|m_2\|/k, ..., \|m_k\|/k\}$*) along the sides of the right angle.*

Figure 5.2A,C illustrate flat 2-LTL matrix and 3-order tensor. Figure 5.2B,D illustrate the position (yellow dash lines) of the most likely pattern fibers in the flat 2-LTL matrix and 3-order tensor. We will first prove Lemma 5.3.1 holds for matrix and three-way tensor. Then we will generalize Lemma 5.3.1 to all-way tensor.

Proof. (Lemma 5.3.1 holds for k-order flat 2-LTL tensor, $k > 3$) Since lemma 5.3.1 holds for matrix and three-way tensor, the generality of lemma 5.3.1 for all-way flat 2-LTL tensor is introduced by mathematical induction. We assume lemma 5.3.1 holds for flat 2-LTL tensor with order of $k-1$. And the largest subarray has volume $f_{k-1}(y_1, y_2, ..., y_{k-1})$. For k-way flat 2-LTL tensor, $f(y_1, y_2, ..., y_{k-1}, y_k) = y_k(\frac{x_k - y_k}{x_k})^{k-1} f_{k-1}(y_1, y_2, ..., y_{k-1})$, where $f_{max} = \frac{\prod_{i=1}^{k} x_i}{k^k}$ is achieved when $y_k = \frac{x_k}{k}$. By induction, $y_i = \frac{k-1}{k} \cdot \frac{k-2}{k-1} \cdot ... \cdot \frac{1}{2} x_i = \frac{x_i}{k}$, $\forall i \in [1, k-1]$. In all, Lemma 5.3.1 holds for all-way flat 2-LTL tensor. □

Proof. (Lemma 5.3.1 holds for 3-way flat 2-LTL tensor) In figure 5.2D, 3-order flat 2-LTL tensor is depicted as right tetrahedron with three right-angle sides of length x_1, x_2 and x_3, respectively. We also let $f(y_1, y_2, y_3)$ represents the volume of the cuboid of interest. Integrating geometric constrain with Proof 1, $y_1 = \frac{x_1}{2}$, $y_2 = \frac{x_2}{2}, \frac{y_1}{x_1} = \frac{y_2}{x_2} = \frac{x_3 - y_3}{x_3}$. s.t., $f(y_1, y_2, y_3) = y_3 \cdot (\frac{1}{2} \cdot \frac{x_3 - y_3}{x_3} \cdot x_1) \cdot (\frac{1}{2} \cdot \frac{x_3 - y_3}{x_3} \cdot x_2) = y_3 \cdot (\frac{x_3 - y_3}{x_3})^2 \cdot \frac{1}{4} x_1 x_2$. When $y_3 = \frac{x_3}{3}$, f get the maximum value $f_{max} = \frac{1}{27} x_1 x_2 x_3$. Additionally, $y_1 = \frac{x_1}{3}$, $y_2 = \frac{x_2}{3}$. As indicated in Figure 5.2C, the optimal basis (yellow colored) is paralleled with lines (pink colored) anchored on the 1/3 segmenting point. □

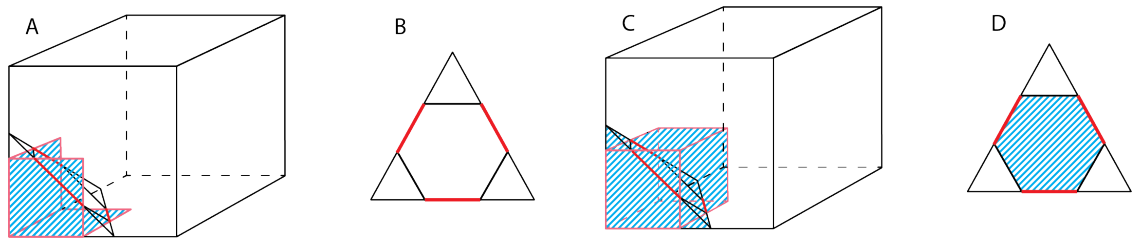

Figure 5.3. Suboptimal subarray for $k-1$ LTL tensor

Proof. (Lemma 5.3.1 holds for k-order flat 2-LTL tensor, $k > 3$) Since lemma 5.3.1 holds for matrix and three-way tensor, the generality of lemma 5.3.1 for all-way flat 2-LTL tensor is introduced by mathematical induction. We assume lemma 5.3.1 holds for flat 2-LTL tensor with order of $k-1$. And the largest subarray has volume $f_{k-1}(y_1, y_2, ..., y_{k-1})$. For k-way flat 2-LTL tensor, $f(y_1, y_2, ..., y_{k-1}, y_k) = y_k(\frac{x_k - y_k}{x_k})^{k-1} f_{k-1}(y_1, y_2, ..., y_{k-1})$, where $f_{max} = \frac{\prod_{\mathrm{i}=1}^{k} x_\mathrm{i}}{k^k}$ is achieved when $y_k = \frac{x_k}{k}$. By induction, $y_\mathrm{i} = \frac{k-1}{k} \cdot \frac{k-2}{k-1} \cdot ... \cdot \frac{1}{2} x_\mathrm{i} = \frac{x_\mathrm{i}}{k}$, $\forall \mathrm{i} \in [1, k-1]$. In all, Lemma 5.3.1 holds for all-way flat 2-LTL tensor. □

Lemma 5.3.2. *(Geometric perspective in seeding the largest rank-1 pattern) For a k order tensor $\mathcal{X}$ sparse enough and a given tensor size threshold λ, if its largest rank-1 pattern tensor is larger than λ, the IRT that reorders $\mathcal{X}$ into a (k-1)-LTL tensor reorders the largest rank-1 pattern to a consecutive block, which maximize the size of the connected solid shape overlapped with the $k-1$ dimension plane over a flat 2-LTL tensor larger than λ.*

Proof. The (k-1)-*LTL* IRT may reorder the indices of these overlapped patterns to the most bottom left position instead of the largest rank-1 pattern. However, if the tensor is sparse enough, i.e., the overlapped region among rank-1 patterns is relative small, the largest rank-1 patterns will be reordered to form a block in the (k-1)-*LTL* IRT. In addition, if the size of the overlapped pattern is significant enough, e.g. larger than a given threshold, the overlapped patterns can be identified as a distinct pattern. Otherwise, the largest rank-1 pattern has a distinct solid shape when intersecting with the k-1 dimension plane of the flat 2-*LTL* tensor that most cross it (Figure 5.3C,D), while the overlapped patterns in the (k-1)-*LTL* IRT will intersect with the k-1 dimension plane of the flat 2-*LTL* tensor most cross it in a ring shape (Figure 5.3A,B). □

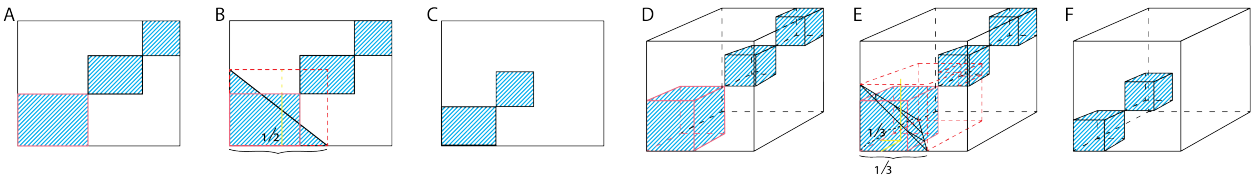

Figure 5.4. GETF sequentially decompose $k-1$ LTL tensor.

Lemma 5.3.3. *If a k-order tensor $\mathcal{X}^{m_1\times\ldots\times m_k}$ can be transformed into a p-LTL tensor by IRT, the p-LTL tensor is unique.*

Proof. If the indices of the p-LTL tensor is not unique there are two p-LTL tensor can be achieved by IRT of $\mathcal{X}^{m_1\times\ldots\times m_k}$, denoted as $\mathcal{X}_{A_1A_2\ldots A_k}$ and $\mathcal{X}_{B_1B_2\ldots B_k}$, where $A_1,\ldots,A_k$ and $B_1,\ldots,B_k$ are two permutations of the index sets $\{1,\ldots,m_1\},\ldots,\{1,\ldots,m_k\}$. Then any p order slice, $X_{\mathbb{P},I_{\bar{\mathbb{P}}}}$ of $\mathcal{X}_{A_1A_2,\ldots,A_k}$ has an identical slice in $\mathcal{X}_{B_1B_2,\ldots,B_k}$, which can be denoted as $\mathcal{X}_{\mathbb{P},I'_{\bar{\mathbb{P}}}}$. By the definition of p-LTL, $\forall \mathrm{j}\in\mathbb{P}, 1\le \mathrm{j}_1<\mathrm{j}_2\le m_{\mathrm{j}}$, $T_{sum}(X_{\mathbb{P},I_{\bar{\mathbb{P}}}},\mathbb{P}\setminus\{\mathrm{j}\})_{\mathrm{j}_1}\le T_{sum}(X_{\mathbb{P},I_{\bar{\mathbb{P}}}},\mathbb{P}\setminus\{\mathrm{j}\})_{\mathrm{j}_2}$ and $T_{sum}(X_{\mathbb{P},I'_{\bar{\mathbb{P}}}},\mathbb{P}\setminus\{\mathrm{j}\})_{\mathrm{j}_1}\le T_{sum}(X_{\mathbb{P},I'_{\bar{\mathbb{P}}}},\mathbb{P}\setminus\{\mathrm{j}\})_{\mathrm{j}_2}$. Hence, either both $T_{sum}(X_{\mathbb{P},I'_{\bar{\mathbb{P}}}},\mathbb{P}\setminus\{\mathrm{j}\})_{\mathrm{j}}$ and $T_{sum}(X_{\mathbb{P},I_{\bar{\mathbb{P}}}},\mathbb{P}\setminus\{\mathrm{j}\})_{\mathrm{j}}$ are identical with respect to j, or the index order of the jth order are identical in $X_{\mathbb{P},I_{\bar{\mathbb{P}}}}$ and $X_{\mathbb{P},I'_{\bar{\mathbb{P}}}}$, suggesting the uniqueness of the p-LTL tensor achieved by IRT of $\mathcal{X}$. □

Lemma 5.3.4. *If a k-order tensor is p-LTL, then it is x-LTL, for all the x=p,p+1,...,k.*

Proof. For any of its P+1 order slice of $\mathcal{X}$, denoted as $\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}}$ and $\forall \mathrm{j}\in\mathbb{P}+1, 1\le \mathrm{j}_1<\mathrm{j}_2\le m_{\mathrm{j}}$, $T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j}\})=\sum_{q=1}^{m_t}T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j},t\})_{:,q}$, where $\mathbb{P}+1$ represents a set of indices with $\|\mathbb{P}\|+1$ elements, $\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}}$ is a $\|\mathbb{P}\|+1$ order slice, and $\{t\}=\mathbb{P}+1\setminus\mathbb{P}$. Noting $T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j},t\})$ is a tensor slice sum that takes a $\|\mathbb{P}\|+1$ order slice as the input and outputs a matrix, which is equivalent to separately compute the tensor slice sum that takes a $\|\mathbb{P}\|$ order slice with fixed index on the tth order as the input and outputs a vector, i.e. $T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j},t\})_{:,q}=T_{sum}(\mathcal{X}_{\mathbb{P},I_{\mathbb{P}\bar{+}1}\cup\{\mathrm{i}_t=q\}},\mathbb{P}\setminus\{\mathrm{j}\})$. By the definition of $p-LTL$, $T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j}\},t)_{\mathrm{j}_1}=\sum_{q=1}^{m_t}T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j},t\})_{\mathrm{j}_1,q}\le\sum_{q=1}^{m_t}T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j},t\})_{\mathrm{j}_2,q}=T_{sum}(\mathcal{X}_{\mathbb{P}+1,I_{\mathbb{P}\bar{+}1}},\mathbb{P}+1\setminus\{\mathrm{j}\},t)_{\mathrm{j}_2}$ □

5.3.2 GETF algorithm

Based on the geometric property of the largest rank-1 pattern and its most likely pattern fibers, we developed an efficient BTD and HBTD algorithm—GETF, by iteratively reconstructing the to-be-decomposed tensor into a $k-1$ LTL tensor and identifying the largest rank-1 pattern. The main algorithm of GETF is formed by the iteration of the following five steps.

1. For a given tensor $\mathcal{X}^{m_1 \times m_2 \cdots \times m_k}$, in each iteration, GETF first reorders the indices of the current tensor into a (k-1)-LTL tensor by IRT (Figure 5.4A,D).

2. GETF utilizes **2_LTL_projection** algorithm to identify the flat 2-LTL tensor that maximizes the overlapped region between its $k-1$ dimension plane and current (k-1)-LTL tensor (Figure 5.4B,E).

3. A **Pattern_fiber_finding** algorithm is applied to identify the most likely pattern fiber of the overlap region of the 2-LTL tensor and the (k-1)-LTL tensor, i.e., the largest rank-1 pattern (Figure 5.5).

4. A **Geometric_folding** algorithm is applied to reconstruct the rank-1 tensor component from the identified pattern fiber that best fit the current to-be-decomposed tensor (Figure 5.6).

5. Remove the identified rank-1 tensor component from the current to-be-decomposed tensor (Figure 5.4C,F).

The inputs of GETF include the to-be-decomposed tensor $\mathcal{X}$, a noise tolerance threshold t parameter, a convergence criterion τ and a pattern fiber searching indicator $Exha$. In **Algorithm 4**, **o** represents a direction of geometric folding, which is a permutation of $\{1, ..., k\}$. The **2_LTL_projection** utilizes a project function and a scoring function to identify the flat 2-LTL tensor that maximizes the solid overlapped region between its $k-1$ dimension plane and a (k-1)-LTL tensor. The **Pattern_fiber_finding** and **Geometric_folding** algorithm are described below. Noted, there are k directions of pattern fibers and $k!$ combinations of the orders in identifying them from a k-order tensor or reconstructing a rank-1

pattern from them. Empirically, to avoid duplicated computations, we tested conducting k times of geometric folding is sufficient to identify the fibers and reconstruct the suboptimal rank-1 pattern. GETF also provides options to set the rounds and noise tolerance level of geometric folding in expanding a pattern fiber via adjusting the parameters $Exha$ and t.

Algorithm 4 GETF

Inputs: $\mathcal{X} \in \{0,1\}^{m_1 \times m_2 \ldots \times m_k}$, $t \in (0,1)$,τ, $Exha \in \{0,1\}$
Outputs: $A^{1*} \in \{0,1\}^{m_1 \times l}$, $A^{2*} \in \{0,1\}^{m_2 \times l}$, ... $A^{k*} \in \{0,1\}^{m_k \times l}$
$GETF(\mathcal{X}, t, \tau, Exha)$:
$\mathcal{X}^{\text{Residual}} \leftarrow \mathcal{X}$, $A^1 \leftarrow NULL$,..., $A^k \leftarrow NULL$
$\Omega \leftarrow$ Generate set of directions for geometric-folding$(k, Exha)$
while $!\tau$ **do**
  $\gamma_0 \leftarrow inf$, $\mathbf{a}^{1*} \leftarrow NULL$,..., $\mathbf{a}^{k*} \leftarrow NULL$
  for each direction $\mathbf{o}$ in Ω **do**
    $\mathcal{X}^{2-LTL} \leftarrow$ **2_LTL_projection**$(\mathcal{X})$
    $Pattern\ fiber^* \leftarrow$ **Pattern_fiber_finding**$(\mathcal{X}^{2-LTL}, \mathbf{o})$
    $(\mathbf{a}^1, ..., \mathbf{a}^k) \leftarrow$ **Geometric_folding**$(\mathcal{X}^{Residual}, Pattern fiber^*, \mathbf{o}, t)$
    if $\gamma(\mathbf{a}^1, ..., \mathbf{a}^k \| \mathcal{X}) < \gamma_0$ **then**
      $(\mathbf{a}^{1*}, ..., \mathbf{a}^{k*}) \leftarrow (\mathbf{a}^1, ..., \mathbf{a}^k)$; $\gamma_0 \leftarrow \gamma(\mathbf{a}^1, ..., \mathbf{a}^k \| \mathcal{X})$
    end if
  end for
  if $\gamma_0 \neq inf$ **then**
    $\mathcal{X}^{Residual}_{\mathrm{i}_1 \mathrm{i}_2 \ldots \mathrm{i}_k} \leftarrow 0$ when $(\mathbf{a}^{1*} \otimes \mathbf{a}^{2*} \ldots \otimes \mathbf{a}^{k*})_{\mathrm{i}_1 \mathrm{i}_2 \ldots \mathrm{i}_k} = 1$
$A^{\mathrm{j}*} \leftarrow append(A^{\mathrm{j}*}, \mathbf{a}^{\mathrm{j}*}), \mathrm{j} \in \{1, 2, ..., k\}$
  end if
end while

5.3.3 Auxiliary algorithms

Before we illustrate $2\_LTL\_projection$, $Pattern\_basis\_finding$ and $Geometric\_folding$, we will introduce some auxiliary algorithms first.

Direction generation There are in total $k!$ directions to construct a $k-1$ LTL tensor. The first step for GETF is to construct such directions set Ω, where $\mathbf{o} \in \Omega$ is the non repetitive combination of 1 to k. Empirically, most of the $k!$ direction will generate the duplicated output resulted from the same or closely related end $k-1$ LTL structural. Such that, normally, k directions are more than enough to generate the suboptimal rank 1 tensor. Still, we provide an $Exha$ Boolean parameter represents exhaustive searching, where $Exha = 1$,

$\Omega' = \Omega$ while $Exha = 0$, Ω' is the k sample of Ω. The final Ω' is the direction set to apply $Geometric\_folding$.

Find the segmenting coordinate As stated in $Pattern\_fiber\_finding$ algorithm, an essential step is to retrieve the $1/n$ coordinate point. This POS algorithm is designed for this task. The input for POS is a vector $\mathbf{d}$, on which to get the segmenting coordinate p, n is the denominator of the segmenting ratio and s is the noise level.

Algorithm 6 POS

Inputs: $\mathbf{d}, n, s$
Outputs: p
$POS(\mathbf{d}, n, s)$:
$\mathbf{m} \leftarrow the\ index\ of\ \mathbf{d} > s$
if $length(\mathbf{m}) < n$ **then**
    return 0 # no need to segment
else
    $q \leftarrow length(\mathbf{m})//n$
    $\mathbf{m}' \leftarrow order(\mathbf{d_m}, decreasing)$ return $\mathbf{m'}_q$
end if

Algorithm 7 TENS_FOLD

Inputs: a k-order tensor $\mathcal{X}^{m_1 \times m_2 \ldots \times m_k}$, the fiber $\mathbf{f}$ to be fold upon, and directions of this fiber $\mathbf{o}$
Outputs: the $(k-1)$ order tensor $\mathcal{H}$
$TENS\_FOLD(\mathcal{X}, \mathbf{o,f})$:
$d \leftarrow diff(range(m), \mathbf{o})$ #the fold dimension $\mathcal{H} \in \{0\}^{m_1 \times \ldots \times m_{d-1} \times m_{d+1} \ldots \times m_k}$ # initialization
$\mathcal{H}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_{d-1}\mathrm{i}_{d+1}\ldots\mathrm{i}_k} \leftarrow \mathcal{X}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_{d-1}:\mathrm{i}_{d+1}\ldots\mathrm{i}_k} \cdot \mathbf{f}$
return $\mathcal{H}$

Folding tensor based on fiber This $TENS\_FOLD$ algorithm allows an efficient folding of k order tensor into $k-1$ order tensor by one dimension on the basis of a specific fiber as mentioned in the main content.

5.3.4 2 LTL projection

From Lemma 5.3.1, the $1/k$ segmentation pinpoint the optimal pattern basis for the $flat\ 2-LTL\ tensor$. We could easily derive the (k-1)-LTL throught IRT. However, owing

to the impact of different levels of noise, lemma 5.3.1 does not hold on $(k-1)-LTL$ tensor in definite. The 2_LTL_projection algorithm is to fill this gap. 2_LTL_projection finds a proper searching space that maintains the $flat\ 2-LTL\ tensor$ property with limited noise. With 2_LTL_projection, we could find the biggest rank 1 tensor patterns in the first iteration, the second biggest patterns in the second iteration, so on and so forth. In practise, we omitted this process to further accelerate GETF. Without the 2_LTL_projection, the order of pattern size may vary with its sequential order. But it does not impact the overall decomposition efficiency. A similar study on Boolean matrix can be found in [121].

Algorithm 8 2_LTL_projection

Inputs: a k-order K-1 LTL tensor $\mathcal{X} \in \{0,1\}^{m_1 \times m_2 \ldots \times m_k}$ and the range of flat 2-LTL tensor $\{m_1^l, ..., m_1^h\} \otimes, ..., \otimes \{m_k^l, ..., m_k^h\}$
Outputs: the flat 2-LTL tensor maximizes the overlap between its k-1 dimension plane and $\mathcal{X}$
$S \leftarrow 0^{(m_1^h - m_1^1 + 1) \times ... \times (m_k^h - m_k^1 + 1)}$
for $(\mathrm{i}_1, \mathrm{i}_2, ...\mathrm{i}_k) \in \{1, ..., m_1^h - m_1^l + 1\} \otimes, ..., \otimes \{1, ..., m_k^h - m_k^l + 1\}$ **do**
  $\mathcal{X}^{projected} \leftarrow \mathrm{project}(\mathcal{X}, Plane(\mathrm{i}_1, \mathrm{i}_2, ...\mathrm{i}_k))$
  $S_{\mathrm{i}_1, \mathrm{i}_2, ...\mathrm{i}_k} \leftarrow sum(neighbor\_weighted\_scoring(\mathcal{X}^{projected}))$
end for
$\mathrm{i}_1^*, \mathrm{i}_2^*, ...\mathrm{i}_k^* \leftarrow argmax\{S\}$
return$(\mathrm{i}_1^*, \mathrm{i}_2^*, ...\mathrm{i}_k^*)$

5.3.5 Pattern fiber finding

The **Pattern_fiber_finding** algorithm is developed based on **Lemma** 5.3.1. Its input include a k-order tensor and a direction vector. Even the input is the entry-wise product of a flat 2-LTL tensor and the largest rank-1 pattern in a (k-1)-LTL tensor, it may still not be 2-LTL due to the existence of errors. We propose a recursive algorithm that recurrently rearrange an order of the input tensor and reports the coordinate of the pattern fiber on this order. The output is the position of the pattern fiber.

Assume we find the pattern fiber along the direction $\mathbf{o} \in \{1, ...k\}$. Derive from lemma 5.3.1. The candidate pattern fiber is revealed recursively. The recursive algorithm first computes tensor slice sum of $\mathcal{X}$ through mode $\mathbf{o}_1$ to $\mathbf{o}_{k-1}$, i.e., from $T_{sum}(\mathcal{X}, \{\mathbf{o}_1\})$ to

$T_{sum}(\mathcal{X}, \{\mathbf{o}_1, ..., \mathbf{o}_{k-1}\})$ and identify the first coordinate on the mode $\mathbf{o}_k$ as the $1/k$ segmenting point of the computed slice sum. Then in each recursive iteration n, there are $k-1$ computed coordinates for the mode of $\mathbf{o}_{k-n+1}$ to $\mathbf{o}_k$ and the $k-n$ order tensor of the slice sum of $\mathcal{X}$ through mode $\mathbf{o}_1$ to $\mathbf{o}_{k-n}$ computed previously, which together form a vector, on which the $1/(n+1)$ segmenting point is computed as the coordinate for the mode $\mathbf{o}_{k-n}$. Denote $(\mathrm{i}^0_1, ..., \mathrm{i}^0_{\mathbf{o}_1-1}, \mathrm{i}^0_{\mathbf{o}_1+1}, ..., \mathrm{i}^0_k)$ as the identified coordinates, the mode-$\mathbf{o}_1$ pattern fiber $\mathbf{a}^{m_{\mathbf{o}_1}\times 1, \mathbf{o}_1} = \mathcal{X}_{\mathrm{i}^0_1 ... \mathrm{i}^0_{\mathbf{o}_1-1} \mathrm{i}^0_{\mathbf{o}_1+1}, ... \mathrm{i}^0_k}$.

Algorithm 9 Pattern_fiber_finding

Inputs: a k-order tensor $\mathcal{X} \in \{0,1\}^{m_1 \times m_2 ... \times m_k}$, the finding direction $\mathbf{o}$ as defined in algorithm 4, a segmentation denominator n and the noise control prarmeter s.
Outputs: the coordinates $(\mathrm{i}_1, ..., \mathrm{i}_{\mathbf{o}_1-1}, \mathrm{i}_{\mathbf{o}_1+1}, ..., \mathrm{i}_k)$ of the mode-$\mathbf{o}_1$ basis fiber.
$Pattern\_fiber\_finding(\mathcal{X}, \mathbf{o}, s)$:
if is.vector($\mathcal{X}$) **then**
  return $POS(\mathcal{X}, n+1, s)$
else
  $n \leftarrow n+1$ #total order of the current $\mathcal{X}$
  $\mathcal{X} \leftarrow TENS\_FOLD(\mathcal{X}, \mathbf{o}_n)$ #compute the $T_{sum}(\mathcal{X}, \{\mathbf{o}_n\})$ for the current $\mathbf{o}_n$ mode.
  $Cor \leftarrow Pattern\_fiber\_finding(\mathcal{X}, \mathbf{o}, s)$ # recursion starts for folding next dimension and return coordinates.
  $\mathrm{i}_{\mathbf{o}_n - n} \leftarrow POS(\mathcal{X}_{:,...,:\mathrm{i}_{\mathbf{o}_n-n+1}:,...,:\mathrm{i}_{\mathbf{o}_n}:,...,:,n+1,s})*$
  $Cor \leftarrow append(Cor, \mathrm{i}_{\mathbf{o}_n - n})$ #integrate coordinates information
  return Cor
end if
$*\mathrm{i}_{\mathbf{o}_n-n+1}, ..., \mathrm{i}_{\mathbf{o}_n}$ are currently computed n coordinates of the mode $\mathbf{o}_n - n + 1$ to $\mathbf{o}_k$ of the pattern fiber.

The recursive algorithm first computes the tensor slice sum of $\mathcal{X}$ through mode $\{\mathbf{o}_1\}$ to $\{\mathbf{o}_{k-1}\}$ and identify the first coordinate on the mode $\mathbf{o}_k$ as the $1/k$ segmenting point of the computed tensor slice sum. Then in each recursive iteration n, there are $k-1$ computed coordinates for the mode of $\mathbf{o}_{k-n+1}$ to $\mathbf{o}_k$ and the $k-n$ order tensor of the slice sum of $\mathcal{X}$ through mode $\mathbf{o}_1$ to $\mathbf{o}_{k-n}$ that computed previously, which together form a vector, on which the $1/(n+1)$ segmenting point is computed as the coordinate for the mode $\mathbf{o}_{k-n}$. Denote $(\mathrm{i}^0_1, ..., \mathrm{i}^0_{\mathbf{o}_1-1}, \mathrm{i}^0_{\mathbf{o}_1+1}, ..., \mathrm{i}^0_k)$ as the identified coordinates, the mode-$\mathbf{o}_1$ pattern fiber $\mathbf{a}^{m_{\mathbf{o}_1}\times 1, \mathbf{o}_1} = \mathcal{X}_{\mathrm{i}^0_1 ... \mathrm{i}^0_{\mathbf{o}_1-1} \mathrm{i}^0_{\mathbf{o}_1+1}, ... \mathrm{i}^0_k}$.

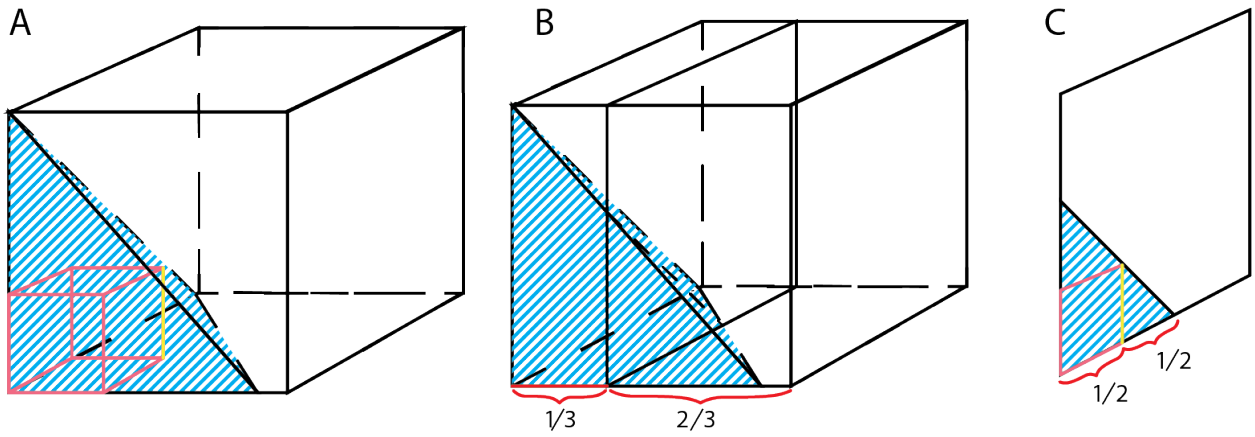

Figure 5.5. Illustration of finding pattern fiber in 3D

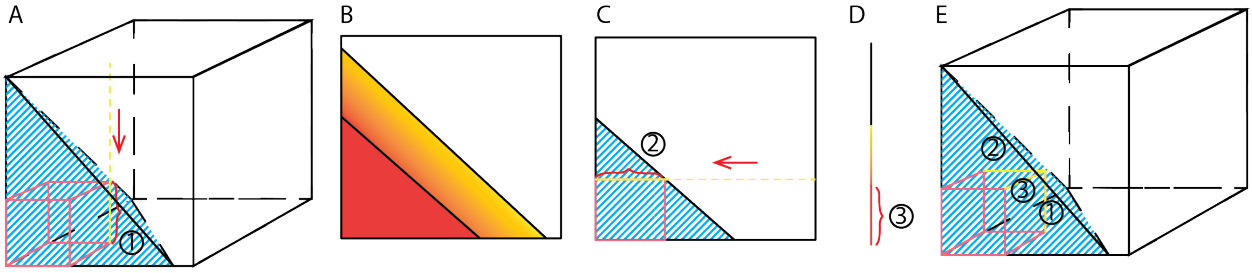

Figure 5.6. Illustration of Geometric_folding in 3D

Here in Figure 5.5, we illustrates the pattern fiber finding approach for a 3-order flat 2-LTL tensor $\mathcal{X}^{m_1 \times m_2 \times m_3}$. To identify the coordinates of the yellow colored pattern fiber with unfixed index of the 1st order $\mathcal{X}_{:\mathrm{i}_2\mathrm{i}_3}$ (Figure 5.5A), its coordinate of the 2nd order is anchored on the 1/3 segmenting point of $T_{sum}(\mathcal{X}, \{2\})$, denoted as i_2 (Figure 5.5B), and its coordinate of the 3rd order is on the 1/2 segmenting point of $T_{sum}(\mathcal{X}^{m_1 \times m_2 \times m_3}_{:\mathrm{i}_2:}, \{2\})$ (Figure 5.5C).

Owing to the recursive property of Pattern_fiber_finding, for the ith iteration of a tensor has the size of m^k, the computational cost is m^{i} for $TENS\_FOLD$ and $mlog(m)$ for POS. Such that, the overall computational cost for Pattern_fiber_finding is $\sum_{\mathrm{i}=1}^{k} m^{\mathrm{i}} + mlog(m) = \frac{m^{k+1}-m}{m-1} + kmlog(m)$.

5.3.6 Geometric folding

The geometric folding approach is to reconstruct the rank-1 tensor pattern best fit $\mathcal{X}$ from the pattern fiber identified by the **Pattern_fiber_finding** algorithm. For a k-order tensor $\mathcal{X}$ and the identified position of pattern fiber, the pattern fiber is denote as $\mathcal{X}_{:\mathrm{i}^0_2\ldots\mathrm{i}^0_k}$ (Figure 5.6A). The algorithm computes the inner product between $\mathcal{X}_{:\mathrm{i}^0_2\ldots\mathrm{i}^0_k}$ and each fiber $\mathcal{X}_{:\mathrm{i}_2\ldots\mathrm{i}_k}$ to generate a new $k-1$ order tensor $\mathcal{H}^{m_2\times\ldots\times m_k}$, $\mathcal{H}_{\mathrm{i}_2\ldots\mathrm{i}_m} = \sum_{\mathrm{j}=1}^{m_1} \mathcal{X}_{\mathrm{j}\mathrm{i}^0_2\ldots\mathrm{i}^0_k} \wedge \mathcal{X}_{\mathrm{j}\mathrm{i}_2\ldots\mathrm{i}_k}$ (Figure 5.6B). This new tensor is further discretized based on a user defined noise tolerance level and generates a new binary k-1 order tensor $\mathcal{X}'^{m_2\times m_3\ldots\times m_k}$ (Figure 5.6C). This approach is called as geometric folding of a k-order tensor into a k-1 order tensor based on the pattern fiber $\mathcal{X}_{:\mathrm{i}^0_2\ldots\mathrm{i}^0_k}$. This approach will be iteratively conducted to fold the k-way tensor into a 2 dimensional matrix with k-2 rounds of **Pattern_fiber_finding** and **Geometric_folding** and identifies k-2 pattern fibers. The pattern fibers of the last 2 dimensional will be identified as a BMF problem by using MEBF [121]. The output of **Geometric_folding** is the set of k pattern fibers of a rank-1 tensor (Figure 5.6E).

Algorithm 10 Geometric_folding

Inputs: A k-order tensor $\mathcal{X}^{m_1\times m_2\ldots\times m_k}$, the direction of pettern fiber finding and geometric folding $\mathbf{o}$ as defined in algorithm 1 and a noise tolerance parameter t
Outputs: Bases of the rank-1 tensor component $\mathbf{a}^1, \mathbf{a}^2, ..., \mathbf{a}^k$
$Geometric\_folding(\mathcal{X}, \mathbf{o}, t)$:
$\mathcal{X}^{original} \leftarrow \mathcal{X}$
$s \leftarrow NOISE\_CONTROL(\mathcal{X})$ # detect the noise level in $\mathcal{X}$
$\mathbf{o}^{original} \leftarrow \mathbf{o}$
for $\mathrm{i} = 1, ..., k-2$ **do**
  $\mathrm{no} \leftarrow \mathbf{o}^{original}_{\mathrm{i}\ldots k}$
  $\|\mathbf{o}\| = k - \mathrm{i} + 1$, $\mathbf{o}_\mathrm{j} = \mathbf{o}_\mathrm{j} - 1$, $\mathrm{i}f\ \mathbf{o}_\mathrm{j} > \mathbf{o}_{\mathrm{i}-1}$
  $(\mathrm{i}^0_1, ..., \mathrm{i}^0_{\mathbf{o}_1-1}, \mathrm{i}^0_{\mathbf{o}_1+1}, ..., \mathrm{i}^0_{\|\mathbf{o}\|}) \leftarrow Pattern\_fiber\_finding(\mathcal{X}, \mathbf{o}, s)$
  $\mathbf{a}^{m_{\mathbf{o}_1}\times 1, \mathbf{o}_1} = \mathcal{X}_{\mathrm{i}^0_1\ldots\mathrm{i}^0_{\mathbf{o}_1-1}\mathrm{i}^0_{\mathbf{o}_1+1}\ldots\mathrm{i}^0_{\|\mathbf{o}\|}}$
  $\mathcal{H}_{\mathrm{i}_1\mathrm{i}_2\ldots\mathrm{i}_{\|\mathbf{o}\|}} \leftarrow \sum_{\mathrm{j}=1}^{m_{\mathbf{o}_1}} \mathcal{X}_{\mathrm{i}^0_1\ldots\mathrm{i}^0_{\mathbf{o}_1-1}\mathrm{i}^0_{\mathbf{o}_1+1}\ldots\mathrm{i}^0_{\mathbf{o}_{\|\mathbf{o}\|}}} \wedge \mathcal{X}_{\mathrm{i}_1\ldots\mathrm{i}_{\mathbf{o}_1-1}\mathrm{i}_{\mathbf{o}_1+1}\ldots\mathrm{i}_{\mathbf{o}_{\|\mathbf{o}\|}}}$
  $\mathcal{X} \leftarrow \mathcal{H} \cdot 0$
  $\mathcal{X}_{\mathrm{i}_1\ldots\mathrm{i}_{\mathbf{o}_1-1}\mathrm{i}_{\mathbf{o}_1+1}\ldots\mathrm{i}_{\|\mathbf{o}\|}} \leftarrow 1$ if $\mathcal{H}_{\mathrm{i}_1\ldots\mathrm{i}_{\mathbf{o}_1-1}\mathrm{i}_{\mathbf{o}_1+1}\ldots\mathrm{i}_{\|\mathbf{o}\|}} \geq t \cdot \|\mathbf{a}\|$
end for
$(\mathbf{a}^{m_{\mathbf{o}_{k-1}}\times 1, \mathbf{o}_{k-1}}, \mathbf{a}^{m_{\mathbf{o}_k}\times 1, \mathbf{o}_k}) \leftarrow MEBF(X, t, s)$
$\mathbf{a}^{\mathbf{o}_\mathrm{i}} \leftarrow \mathbf{a}^{m_{\mathbf{o}_\mathrm{i}}\times 1, \mathbf{o}_\mathrm{i}}, \mathrm{i} = 1, ..., k$
return $\mathbf{a}^1, ..., \mathbf{a}^k$

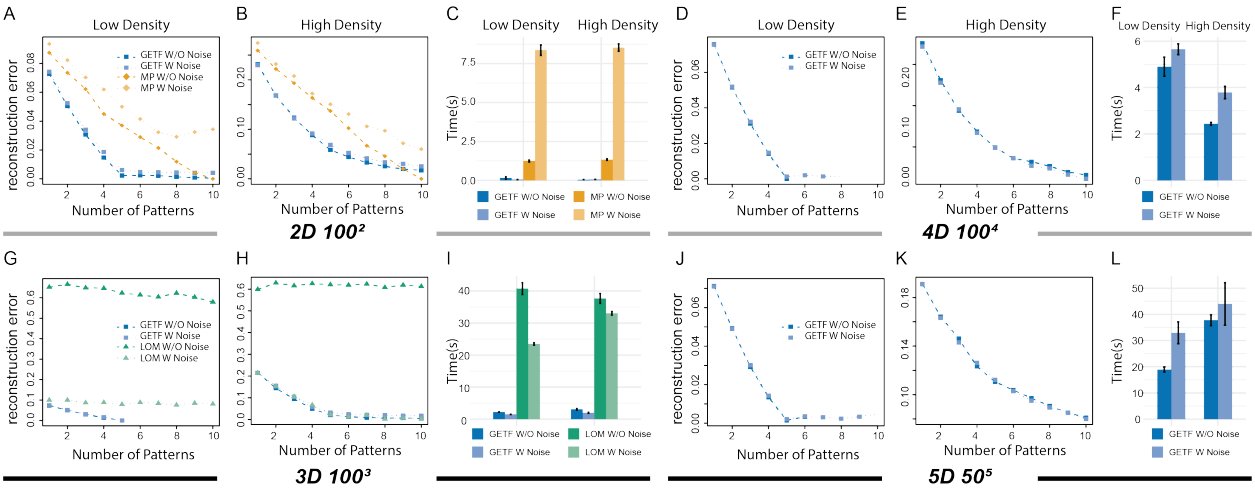

Figure 5.7. Performance analysis on simulated data

5.3.7 Complexity analysis

Assume k-order tensor has $n = m^k$ entries. The computation of **2_LTL_projection** is fixed based on its screening range, which is smaller than $O(m^k)$. To further accelerate GETF, we omitted the projection step as it does not affect the overall decomposition efficiency in practise. The computation of each **Pattern_fiber_finding** is $\frac{m^{k+1}-m}{m-1} + kmlog(m)$. **Geometric_folding** is a loop algorithm consisted of additions and **Pattern_fiber_finding**. The computation for **Geometric_folding** to fold a k-order tensor takes $\frac{2m^{k+2}-m^k}{(m-1)^2} + \frac{1-2m^2}{(m-1)^2} - \frac{km}{m-1} + \frac{k(k+1)}{2}mlog(m)$ computations. $GETF$ conducts k times $Geometric\_folding$ in each iteration to extract the suboptimal rank-1 tensor, by which, the overall computing cost on each iteration is $k(\frac{2m^{k+2}-m^k}{(m-1)^2} + \frac{1-2m^2}{(m-1)^2} - \frac{km}{m-1} + \frac{k(k+1)}{2}mlog(m)) \sim O(m^k)$. Hence $GETF$ is an $O(m^k) = O(n)$ complexity algorithm.

5.4 Experimental Results on Synthetic Datasets

We generated synthetic tensors with $k = 2, 3, 4, 5$ that correspond to the BMF, BTD, 4-order HBTD and 5-order HBTD problems, and for each order k, 4 scenarios are further created: (1) low density tensor without error, (2) low density tensor with error, (3) high density tensor without error and (4) high density tensor with error. Under each scenario, we fixed the number of true patterns as 5 and set the convergence criteria as 1) 10 patterns

have been identified, 2) the cost function stopped decreasing with newly identified patterns. We compared GETF with MP on BMF and LOM on BTD settings, which represent best performance for BMF and BTD problems respectively [114], [122]. The evaluation focuses on two metrics, time consumption and reconstruction error [114], [115]. For 4-order and 5-order HBTD, we only conducted GETF as other methods cannot handle or fail to converge in analyzing such high order tensors.

GETF significantly outperformed MP in reconstruction error (Figure 6.3A,B) and time consumption (Figure 6.3C) for all the four scenarios. Noted, the top five patterns identified by GETF coincided with the five true patterns in all scenarios. Similarly GETF outperformed LOM in all four scenarios, except for the high density with high noise case, where GETF and LOM performed comparatively in terms of reconstruction error (Figure 6.3G,H,I). GETF maintains the most favorable performance with over 10 times higher in computational efficiency. Figure 6.3 D-F,J-L show the capability of GETF on decomposing high order tensor data. Notably, the reconstruction error curve of GETF flattened after reaching the true number of components (Figure 6.3A,B,D,E,G,H,J,K), suggesting its high accuracy in identifying true number of patterns. The error bar stands for standard derivation of time consumption in Figure 6.3 C,F,I,L. Importantly, when the tensor order increases, its size increases exponentially. The high memory cost is regarded as an outstanding challenge for higher order tensor decomposition, for which an $O(n)$ algorithm like GETF is desirable. GETF showed consistent performance with respect to different noise levels. For a 5-way tensor with more than $3 * 10^8$ elements, GETF reached convergence in less than 1 minute. Overall, our experiments on synthetic datasets advocated the efficiency and robustness of GETF for the data with different tensor orders, data sizes, signal densities and noise levels.

5.5 Experimental Results on Real-world Datasets

We applied GETF on two real-world datasets, the Chicago crime record data[2], and a breast cancer spatial-transcriptomics data[3], which represents two scenarios with relatively lower and higher noise. We benchmarked GETF with LOM on the two data sets, and focused

[2]↑Chicago crime records downloaded on March 1st, 2020 from https://data.cityofchicago.org/Public-Safety
[3]↑Breast cancer data is retrieved from https://www.spatialresearch.org/resources-published-datasets

on comparing their reconstruction error and interpreting the pattern tensors identified by GETF.

We retrieved the crime records in Chicago from 2001 to 2019 and organized them into a 4D tensor, with four dimensions representing: 436 regions, 365 dates, 19 years and two crime categories (severe, and non-severe), respectively, i.e., $\mathcal{X}^{436\times365\times19\times2}$. An entry in the tensor has value 1 if a crime category was observed in the region for the day of the year. We first benchmark GETF with LOM on the 3D slice, $\mathcal{X}_{:::1} \in \{0,1\}^{436\times365\times19}$. GETF showed a clear advantage over LOM, including a less reconstruction error and higher interpretability. GETF converged after identifying two large patterns, while LOM identied more than eight patterns to achieve same level of reconstruciton error (Figure 5.8B). We only presented the results of GETF on the application of the 4-order tensor, on which LOM failed to converge, and used the top two patterns to reconstruct the original tensor, denoted as $\mathcal{X}^*$. To look for the crime date pattern, the crime index of a region defined as the total days of a year with crime occurrences in the region were associated with the identified low rank tensor patterns. We showed that $\mathcal{X}^*$ reconstructed from the CP decomposition is a denoised form of the origincal data. In Figure 5.8C, the high and low crime index were red and blue colored. Clearly, GETF reconstructed tensor is able to distinguish the two regions (Figure 5.8C). However, such a clear separation is less clear in the original data (Figure 5.8D). Next we examined the validity of the two regions with an outsider factor, regional crime counts, defined as the total number of crimes from 2001 to 2019 for that region. As shown in Figure 5.8E, the regions with higher crime index according to GETF consitently correspond to the regions of higher regional crime counts, and vice versa. In summary, we show that GETF is able to reveal the overall crime patterns by denoising the original tensor.

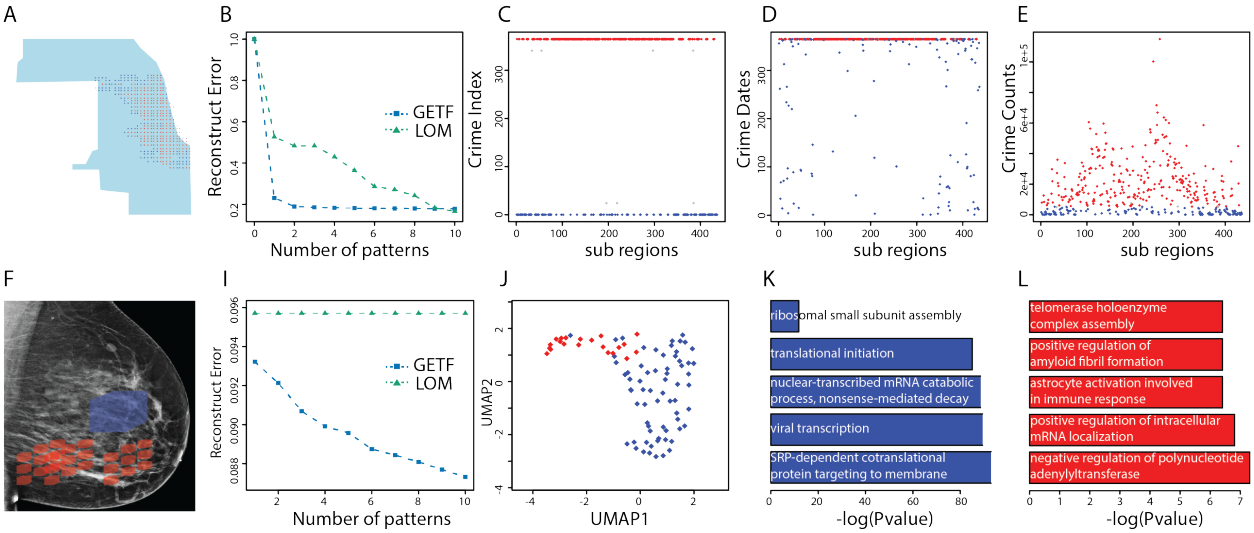

Figure 5.8. Real data benchmark and applications

The breast cancer spatial transcriptomics dataset [134], [135], as in Figure 5.8F, was collected on a 3D space with 1020 cell positions ($x \times y \times z = 15 \times 17 \times 4$), each of which has expression values of 13,360 genes, i.e., $\mathcal{X}^{13360\times15\times17\times4}$. The tensor was first binarized, and it takes value 1 if the expression of the gene is larger than zero. We again benchmarked the performance of GETF and LOM on a 3D slice, $\mathcal{X}_{:::1}$. LOM failed to generate any useful information seen from the non-decreasing reconstruction error, possibly because of the high noise of the transcriptomics data. On the other hand, GETF manage to derive patterns gradually (Figure 5.8I). We applied GETF only to the 4D tensor, and among the top 10 patterns, we analyzed two extremest patterns: one the most sparse (red) and the other the most dense (blue) (Figure 5.8F). The sparse pattern has 24 cell positions all expressing 232 genes ($232 \times 4 \times 4 \times 2$), the dense pattern has 90 cells positions expressing 40 genes ($40 \times 15 \times 3 \times 2$). A lower dimensional embedding of the 114 cells using UMAP [136] demonstrated them to be two distinct clusters (Figure 5.8J). We also conducted functional annotations using gene ontology enrichment analysis for the genes of the two patterns. Figure 5.8K,L showed the $-log(p)$ of the top 5 pathways enriched by the genes in each pattern, assessed by hypergeometric test. It implies that genes in the most dense pattern maintains the vibrancy of the cancer by showing strong activities in transcription and translation; while genes in the most sparse pattern maintains the tissue structure and suppress anti-tumor

immune effect. Our analysis demonstrated that the GETF is able to reveal the complicated but integrated spatial structure of breast cancer tissue with different functionalities.

5.6 Discussion

In this paper, we proposed GETF as the first efficient method for the all-way Boolean tensor decomposition problem. We provided rigorous theoretical analysis on the validity of GETF and conducted experiments on both synthetic and real-world datasets to demonstrate its effectiveness and computational efficiency. In the future, to enable the integration of prior knowledge, we plan to enhance GETF with constrained optimization techniques and we believe it can be beneficial for broader applications that desire a better geometric interpretation of the hidden structures.

In the next chapter, we will discuss the bias inside binary data for us to better find meaning patterns from data.

6. INDIVIDUAL BIAS IN BINARY DATA

6.1 Motivation

Binary matrix has been commonly utilized in multiple fields. Low rank pattern in a binary matrix is defined as rank-1 sub matrices formed by the product of two binary bases. Comparing to continuous data, recent studies demonstrated the rank-1 sub-matrices in binarized data is more robust for mechanism interpretation or sub-space representation [115], [121], because binary data in general bears reduced noise than continuous data. However, variations of the probability of 1s of rows or columns may lead to varied element-wise probability, causing a fairness issue in low rank representation of binary data [137].

An intuitive example is binary transaction records data (figure 6.1), in which 1s represent the purchase of items (each column) by users (each row). Different items or users are with varied activities in conducting purchasing. For example, super-users make more purchase, which can be independent to items, and popular items are more likely to be purchased. The transactions made between super users and popular items unnecessarily imply good recommendations since it can be simply caused by the high purchase chance. On the other hand, the group of items having a strong purchase preference within a certain group of users comparing to their background purchase rate is more valuable for recommendation. However,

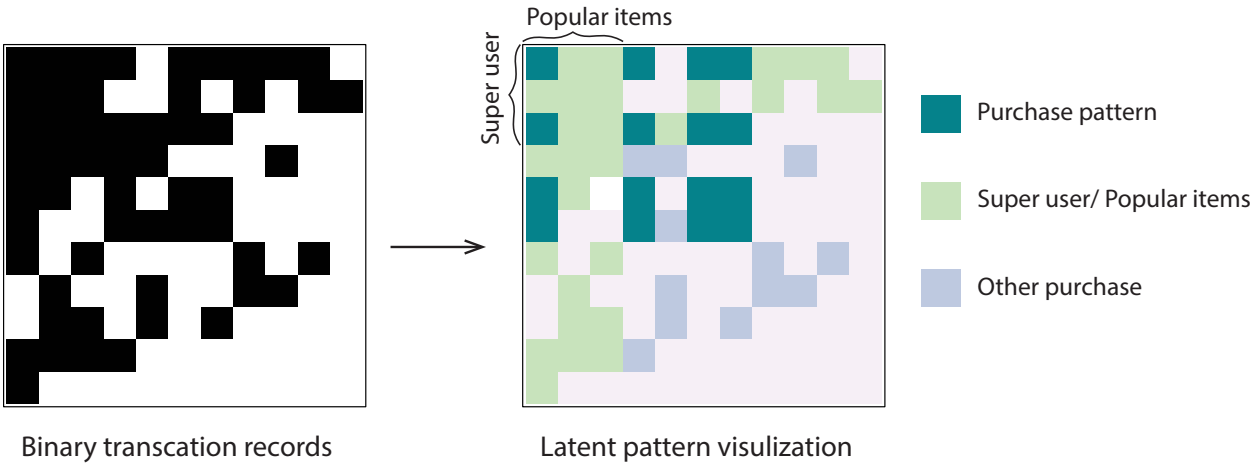

Figure 6.1. Individual bias in binary transaction records data

the fairness issue in the low rank representation of binary data due to varied element-wise background probability was rarely considered in existing formulations [138].

Here, we propose BIND, a BINary data Denoising method via considering that data is generated from the mixture of to-be-identified rank-1 patterns and an unknown background of element-wise probability, plus i.i.d. errors. BIND estimates the mixture distribution of the probabilities of 1s from rank-1 patterns and background in each row and column, by which the rows or columns that are more likely with true rank-1 patterns are distinguished by the over-represented 1s comparing to the background.

Key contributions of this work include:

1. BIND is the first of this kind of binary data denoising method via considering non-identical background distribution.

2. BIND can be easily implemented with state-of-the-arts BMF or CC methods for a fairer rank-1 pattern detection.

3. rigorous mathematical derivations are provided to characterize the property of disparate background distribution.

6.2 Background

6.2.1 Notations

We denote matrix, vector and scalar by uppercase, bold lowercase and lowercase character $X, \mathbf{x}, x$. Superscript with $\times$ indicates dimensions, while subscript implies index, such as $X_{\mathrm{ij}}^{m\times n}$ and $\mathbf{x}_{\mathrm{i}}^{m\times 1}$. $P_{\mathrm{ij}} \triangleq P(X_{\mathrm{ij}} = 1)$ denotes the element-wise probability of 1 at the element X_{ij}. $\|\mathbf{x}\|$ and $\|X\|$ represent the $l1$ norm of vector and matrix, and $\circ$ represents Hadamard product.

6.2.2 Related work

Existing methods of binary matrix low rank representation fall into two major categories, namely binary matrix decomposition (BMF) and co-clustering (CC). BMF aims to decom-

pose a binary matrix as the product of two low rank binary matrix by maximizing its overall fitting to the original matrix. The formulation of BMF is thus generalized as

$$X^{m\times n} = U^{m\times k}V^{k\times n} + E^{m\times n}$$

, where U and V are the low rank pattern matrices, and E is the flipping error with $p(1 \rightarrow 0) = p(0 \rightarrow 1) = p_0$. BMF problem is NP-hard, for which multiple heuristic algorithms have been developed. One representative method is ASSO, which retrieves candidate patterns by using row-/column-wise correlation [25]. More recently, Bayesian probability measure and geometrical identification largely improved the efficiency and accuracy of BMF [115], [121].

In contrast, the co-clustering (CC) method, also named as bi-clustering in statistics and computational biology, maximizes the enrichment of 1s in the detected patterns based on certain thresholds[139]. For given $X^{m\times n}$, most CC methods aim to identify the cardinality of index set $I_l \times J_l$, $l = 1, ..., k$, where $I_l \in \{1, ..., m\}$ and $J_l \in \{1, ..., n\}$,

$$s.t. \quad P_{\mathrm{ij}} = \begin{cases} p_l, \ \mathrm{if\ i, j} \in I_l \times J_l \\ p_0, \ \mathrm{if\ i, j} \notin I_l \times J_l \end{cases} \qquad \forall l = 1, ..., k$$

Noted, both BMF and CC methods assume the binary data is formed by the sum of to-be-identified rank-1 submatrices and an i.i.d error, where individuals bias has not been investigated.

6.2.3 Problem formulation

We consider the observed binary data with disparate element-wise background probability that is generated by:

$$X = U^{m\times k}V^{k\times n} + X^0 + E' + E \tag{6.1}$$

Compared with the formulation of BMF, X^0 is the background matrix. E' is the pattern wise observation error that each element from pattern l has a probability of $1 - p_l$ to be

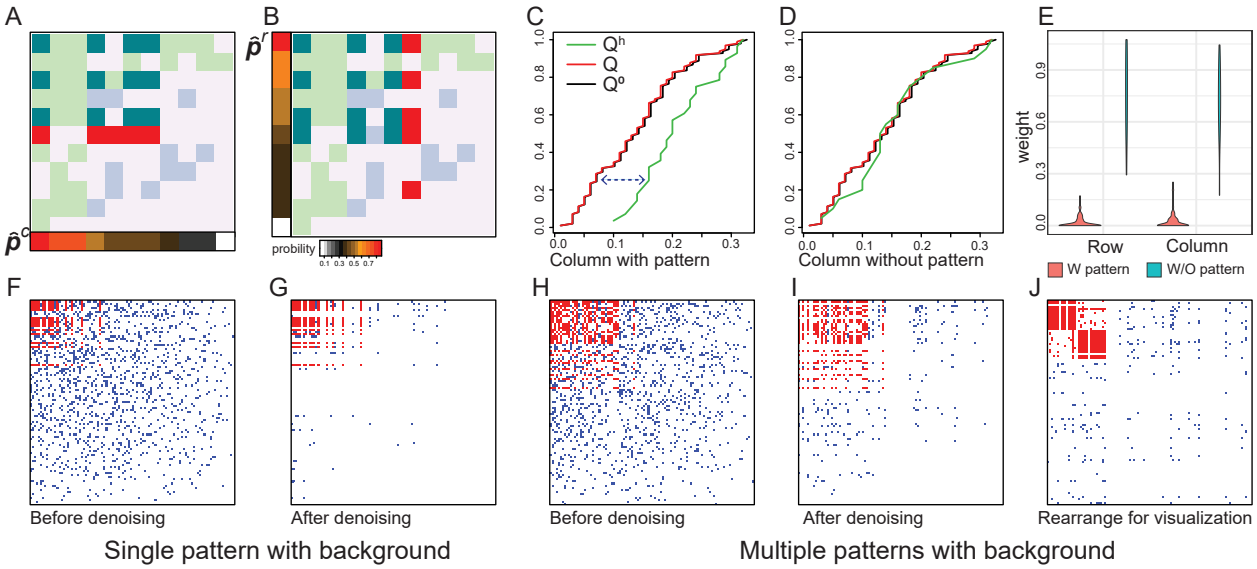

Figure 6.2. Quantile shift denoising

zero, while the elements outside patterns will not be impacted, i.e., $P_{ij}^{E'}(1 \to 0) = 1 - p_l$, if $i, j \in I_l \times J_l$, $P_{ij}^{E'}(1 \to 0) = 0$, if $i, j \notin I_l \times J_l$, $\forall l = 1, ..., k$.

Under this definition, by considering X^0 are 0, current BMF and CC described in 2.2 are special case of (6.1), and were designed to handle the pattern observation error E' and elment-wise flipping error E. Thus, the bottleneck of a fair binary submatrix detection lies in differentiating true patterns from the background X^0. We consider the assumption of $P(X_{ij}^0 = 1) \propto \mathbf{p}_i^{0,r} \cdot \mathbf{p}_j^{0,c}$ that can cover most of the binary data with disparate background, when X_{ij}^0 are conditionally independent with fixed row or column index, like the purchase transaction data in figure 6.1 with items of different popularity and users of different activity. We denote the row/column-wise background probability as $\mathbf{p}^{m\times 1,0,row}$ and $\mathbf{p}^{n\times 1,0,column}$, shorted as $\mathbf{p}^{0,r}$ and $\mathbf{p}^{0,c}$, where $\mathbf{p}_i^{0,r} \propto \hat{\mathbf{p}}_i^{0,r} = \frac{\|X_{i:}^0\|}{n}$ and $\mathbf{p}_j^{0,c} \propto \hat{\mathbf{p}}_j^{0,c} = \frac{\|X_{:j}^0\|}{m}$, and $P(X_{ij}^0 = 1)$ can be unbiasedly estimated as $\frac{\|X_{i:}^0\| \cdot \|X_{:j}^0\|}{\|X^0\|}$.

6.3 BIND framework

Here we propose the BIND[1] framework to identify the rank-1 patterns (U, V) from binary data X with disparate background X^0. Denoting $P(X^0_{\mathrm{ij}} = 1)$ as P^0_{ij}, the element-wise probability $P_{\mathrm{ij}} \triangleq P(X_{\mathrm{ij}} = 1)$ can be derived as:

$$P_{\mathrm{ij}} = \begin{cases} P^0_{\mathrm{ij}} \propto \mathbf{p}^{0,r}_{\mathrm{i}} \cdot \mathbf{p}^{0,c}_{\mathrm{j}}, \ \mathrm{ij} \notin any\, I_l \times J_l \\ 1 - (1 - P^0_{\mathrm{ij}})(1 - p_l) = p^0_{\mathrm{ij}} + (1 - p^0_{\mathrm{ij}})p_l, \ \mathrm{ij} \in I_l \times J_l \end{cases} \tag{6.2}$$

Specifically, the row and column probability $\mathbf{p}^r_{\mathrm{i}}$ and $\mathbf{p}^c_{\mathrm{j}}$ can be estimated by $\hat{\mathbf{p}}^r_{\mathrm{i}} = \frac{\|X_{\mathrm{i}:}\|}{n}$ and $\hat{\mathbf{p}}^c_{\mathrm{j}} = \frac{\|X_{:\mathrm{j}}\|}{m}$. Noted, $\mathbf{p}^r$ and $\mathbf{p}^c$ are formed by the mixture distribution of $\mathbf{p}^{0,r}, \mathbf{p}^{0,c}$ and p_l. Analogous to BMF and CC problem, direct inference of $\mathbf{p}^{0,r}, \mathbf{p}^{0,c}$ and p_l from $\mathbf{p}^r$ and $\mathbf{p}^c$ is NP-hard. As shown in Figure 6.2A-D, instead of computing $\mathbf{p}^{0,r}, \mathbf{p}^{0,c}$ and p_l, BIND identifies the rows and columns that are most likely conceiving patterns comparing to others. The elements of the intersection of the identified rows and columns more likely represent true rank-1 patterns (figure 6.2F-J). For this task, we introduce the quantile_shift algorithm with thorough mathematical proof.

Quantile_shift algorithm is designed to distinguish rows or columns that are more likely conceiving rank-1 patterns. First, we introduce the concept of empirical distribution of row-/column-wise probability, denoted as $\mathbf{F}^r$ and $\mathbf{F}^c$ (figure 6.2A,B), which are sampled from $\hat{\mathbf{p}}^r$ and $\hat{\mathbf{p}}^c$ with probability $P(\mathbf{F}^r = \hat{\mathbf{p}}^r_{\mathrm{i}}) \propto \hat{\mathbf{p}}^r_{\mathrm{i}}$ and $P(\mathbf{F}^c = \hat{\mathbf{p}}^c_{\mathrm{j}}) \propto \hat{\mathbf{p}}^c_{\mathrm{j}}$. The observed probability of hits $\mathbf{F}^h$ of any row i_0 or column j_0 is defined by $\mathbf{F}^{h,r,\mathrm{i}_0} = \{\hat{\mathbf{p}}^c_{\mathrm{j}} \| \mathrm{j}\ with\ X_{\mathrm{i}_0\mathrm{j}} = 1\}$ and $\mathbf{F}^{h,c,\mathrm{j}_0} = \{\hat{\mathbf{p}}^r_{\mathrm{i}} \| \mathrm{i}\ with\ X_{\mathrm{ij}_0} = 1\}$. Here $\mathbf{F}^r$ and $\mathbf{F}^c$ characterize the distribution of $\hat{\mathbf{p}}^r$ and $\hat{\mathbf{p}}^c$ of the 1s randomly drawn from $\hat{\mathbf{p}}^r$ and $\hat{\mathbf{p}}^c$. Intuitively, if a row or column conceives a distinct pattern, the quantile function Q^h of $\mathbf{F}^h$ will shift drastically from the quantile function Q^c of $\mathbf{F}^c$ or Q^r of $\mathbf{F}^r$ (figure 6.2C). On the other hand, Q^h will be similar to Q^c or Q^r if the row or column does not contain any pattern (figure 6.2D). Hence the shift between Q^h and Q^r or Q^c can serve as a weight s to differentiate the rows or columns more likely conceiving a pattern (figure 6.2E). Noted, here $\mathbf{F}^r$ and $\mathbf{F}^c$ serve as proxy of $\mathbf{F}^{0,r}$ and $\mathbf{F}^{0,c}$,

[1]↑Code and material can be access at https://github.com/clwan/BIND

which are the empirical distribution of the true background probability of $\mathbf{p}^{0,r}$ and $\mathbf{p}^{0,c}$. In the following content, we prove s approximates the pattern size within each row or column, i.e., $s \approx \|(UV + E')_{\mathrm{i}:}\|$ *or* $\|(UV + E')_{:\mathrm{j}}\|$ with certain bounds.

The input of *Quantile_shift* algorithm include a row or column index $\mathrm{i}_0/\mathrm{j}_0$, and $\hat{\mathbf{p}}^c$ or $\hat{\mathbf{p}}^r$, by which the empirical distribution $\mathbf{F}^c$ or $\mathbf{F}^r$ will be sampled, and the probability of hit of the row or column $\mathbf{F}^h$ will be computed. The output is weight s of the row or column. Without loss of generality, we illustrate the *Quantile_shift* algorithm for computing the weight of row i_0 below, and detailed mathematical proofs as follows:

Algorithm 11 Quantile_shift

Inputs: Row index i_0, Estimated column-wise probability $\hat{\mathbf{p}}^c$
Outputs: Estimated weight of significance of row i_0, $s^r_{\mathrm{i}_0}$
Quantile_shift$(\mathrm{i}_0, \hat{\mathbf{p}}^c)$:
$\mathbf{F}^c \leftarrow sampled\, from\, \hat{\mathbf{p}}^c\, with\, probability\, \hat{\mathbf{p}}^c$
$\mathbf{F}^h \leftarrow \{\hat{\mathbf{p}}^c_{\mathrm{j}} \| \mathrm{j}\ with\ X_{\mathrm{i}_0\mathrm{j}} = 1\}$
$\mathbf{F}^{(h)} \leftarrow sort(\mathbf{F}^h)$, $a \leftarrow length(\mathbf{F}^h)$
$Q^c(p) = sup(b)\ s.t.\ \frac{\|\mathbf{F}^c < b\|}{length(\mathbf{F}^c)} \leq p\ and\ \frac{\|\mathbf{F}^c > b\|+1}{length(\mathbf{F}^c)} > p$
for j=1...a **do**
  if $\mathbf{F}^{(h)}_{\mathrm{j}} > Q^c(\frac{\mathrm{j}}{a})$ **then**
    $t_{\mathrm{j}} \leftarrow$ *the column index s.t.* $\mathbf{F}^{(h)}_{\mathrm{j}} = \hat{\mathbf{p}}^c_{t_{\mathrm{j}}}$ & $X_{\mathrm{i}_0 t_{\mathrm{j}}} = 1$
    $s \leftarrow s + \frac{\mathbf{F}^{(h)}_{\mathrm{j}} - Q^c(\frac{\mathrm{j}}{a})}{1 - \hat{\mathbf{p}}^c_{\mathbf{t}_{\mathrm{j}}}}$
  end if
end for

Lemma 6.3.1. *If $\hat{\boldsymbol{p}}^r$ and $\hat{\boldsymbol{p}}^c$ are unbiased estimation of $\boldsymbol{p}^{0,r}$ and $\boldsymbol{p}^{0,c}$. The weight computed by quantile_shift is an unbiased estimation of the sum of $E(U^{m\times k}V^{k\times n} + E')$ with respect to that column or row.*

Proof. If $\hat{\mathbf{p}}^r$ and $\hat{\mathbf{p}}^c$ are unbiased estimation of $\mathbf{p}^{0,r}$ and $\mathbf{p}^{0,c}$, $\mathbf{F}^r$ or $\mathbf{F}^c$ generated from $\hat{\mathbf{p}}^r$ and $\hat{\mathbf{p}}^c$ form unbiased empirical distribution of row-/column-wise probability of 1s of X^0, i.e. $P(\mathbf{F}^{0,r} = \mathbf{p}^{0,r}_{\mathrm{i}}) \propto \mathbf{p}^{0,r}_{\mathrm{i}}$ and $P(\mathbf{F}^{0,c} = \mathbf{p}^{0,c}_{\mathrm{j}}) \propto \mathbf{p}^{0,c}_{\mathrm{j}}$. Without loss of generality, we prove the lemma for the computation of the weight of the i_0th row. Denote $\mathbf{t} = \{\mathrm{j} \| X_{\mathrm{i}_0\mathrm{j}} = 1\}$ and

$a = length(\mathbf{t})$, by **Algorithm** 11 and (6.2), $\forall\ \mathrm{j} \in \{1, ..., a\}$:

If $\mathrm{i}_0 t_{\mathrm{j}} \notin any\ I_l \times J_l$,

$$E(\mathbf{F}_{\mathrm{j}}^{(h)} - Q(\frac{\mathrm{j}}{a})) = E(\hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c} - sup(b \| \frac{\|\mathbf{F}^c < b\|}{length(\mathbf{F}^c)} \leq \frac{\mathrm{j}}{a})) = 0$$

Else, $\mathrm{i}_0 t_{\mathrm{j}} \in I_l \times J_l\ for\ certain\ l$,

$$E(\mathbf{F}_{\mathrm{j}}^{(h)} - Q(\frac{\mathrm{j}}{a})) = E(\hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c} + (1 - \hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c}) p_l - sup(b \| \frac{\|\mathbf{F}^c < b\|}{length(\mathbf{F}^c)} \leq \frac{\mathrm{j}}{a}))$$

$$= (1 - \hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c}) p_l$$

Such that

$$E(\sum_{\mathrm{j}=1}^{a} \frac{\mathbf{F}_{\mathrm{j}}^{(h)} - Q(\frac{\mathrm{j}}{a})}{1 - \hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c}}) = \sum_{l} \sum_{\mathrm{j}=1}^{a} p_l I = \|E(U^{m \times k} V^{k \times n} + E')_{\mathrm{i}_0:}\|$$

□

Lemma 6.3.2. *For X in (6.1), and $P_{\mathrm{ij}}^0 \triangleq P(X_{\mathrm{ij}}^0) \propto \boldsymbol{p}_{\mathrm{i}}^{0,r} \cdot \boldsymbol{p}_{\mathrm{j}}^{0,c}$, the probability estimated by $\hat{p}_{\mathrm{i}}^r = \frac{\|X_{\mathrm{i}:}\|}{n}$ and $\hat{p}_{\mathrm{j}}^c = \frac{\|X_{:\mathrm{j}}\|}{m}$ are bounded by $\|\hat{p}_{\mathrm{i}}^r - \boldsymbol{p}_{\mathrm{i}}^{0,r}\| \leq \frac{\sum_{l=1}^{k} \boldsymbol{1}(\mathrm{i} \in I_l) p_l \|J_l\|}{n}$, and $\|\hat{p}_{\mathrm{j}}^c - \boldsymbol{p}_{\mathrm{j}}^{0,c}\| \leq \frac{\sum_{l=1}^{k} \boldsymbol{1}(\mathrm{j} \in J_l) p_l \|I_l\|}{m}$.*

Lemma 6.3.2 can be derictly derived from (6.1) and (6.2).

Lemma 6.3.3. *The weight of the i_0th row (or similarly j_0th column) is with a bias led by the biasedly estimated $\hat{\boldsymbol{p}}^c$ and $\hat{\boldsymbol{p}}^r$, which is bounded by*

$$E(s - \|(UV + E')_{\mathrm{i}_0:}\|) \leq \frac{max(\boldsymbol{F^c}) + max(\frac{\|E(UV+E'):\mathrm{j}\|}{m})(\|\boldsymbol{F^c}\| + 1)}{min(1 - \boldsymbol{p}^h)\|\boldsymbol{F^c}\|}.$$

We still use the computation of the i_0th row to illustrate the proof. The case for columns can be similarly derived.

Proof. By Lemma 6.3.2, $\hat{\mathbf{p}}^c$ is a biased estimation of $\mathbf{p}^{0,c}$, where $\hat{\mathbf{p}}_{\mathrm{j}}^c = \frac{\|X_{:\mathrm{j}}\|}{m} \geq \mathbf{p}_{\mathrm{j}}^{0,c} = \frac{\|X_{:\mathrm{j}}^0\|}{m}$, $\mathrm{j} = 1, ..., m$. Hence $\mathbf{F}^{(h)} \geq \mathbf{F}^{0,(h)}$, suggesting $1 - \mathbf{F}^{0,(h)} \geq 1 - \mathbf{F}^{(h)}$ and $Q^c(\frac{\mathrm{j}}{a}) \geq Q^{0,c}(\frac{\mathrm{j}}{a})$, by which

$$\left\| \frac{\mathbf{F}_{\mathrm{j}}^{0,(h)} - Q^{0,c}(\frac{\mathrm{j}}{a})}{1 - \hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{0,c}} - \frac{\mathbf{F}_{\mathrm{j}}^{(h)} - Q^{c}(\frac{\mathrm{j}}{a})}{1 - \hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c}} \right\| \leq 2 \left\| \frac{\max\limits_{z \in (0,1)} \{Q^c(z) - Q^{0,c}(z)\}}{1 - \hat{\mathbf{p}}_{\mathbf{t}_{\mathrm{j}}}^{c}} \right\|$$

By lemma 6.3.2, the bias of $\|\hat{\mathbf{p}}_{\mathrm{j}}^{c} - \hat{\mathbf{p}}_{\mathrm{j}}^{c,0}\|$ is bounded by $\frac{\|E(UV+E')_{:\mathrm{j}}\|}{m}$. So the max shift caused in the quantile function $\max_{z\in(0,1)}\{Q^{c}(z) - Q^{0,c}(z)\}$ is bounded by $\frac{max(\hat{\mathbf{p}}^{c})+max(\frac{\|E(UV+E')_{:\mathrm{j}}\|}{m})}{\|\hat{\mathbf{p}}^{c}\|} + max(\frac{\|E(UV+E')_{:\mathrm{j}}\|}{m})$. Hence the cumulative bias is bounded by

$$E(s - \|E(UV+E')_{\mathrm{i}_0:}\|) \leq \frac{a(max(\hat{\mathbf{p}}^{c}) + max(\frac{\|E(UV+E')_{:\mathrm{j}}\|}{m})(\|\hat{\mathbf{p}}^{c}\|+1)))}{min(1-\hat{\mathbf{p}}^{c})\|\hat{\mathbf{p}}^{c}\|}$$

□

Lemma 6.3.1 suggests $\|Q^{h} - Q^{0}\|$ is an unbiased estimation of the expected number of 1s in the rank-1 patterns and Lemma 2-3 provide the bound of the bias of $\|Q^{h} - Q\|$ when Q^{0} is biasedly estimated as Q.

Theorem 6.3.4 (Quantile_shift)**.** *For a relative sparse binary matrix, the weight calculated by Quantile_shift sufficiently characterizes the indices of the patterns with largest $P_l\|I_l\|$ and $P_l\|J_l\|$.*

Proof. For i_0th row (or similarly for the j_0th column),

$$E(s - \|(UV+E')_{\mathrm{i}_0:}\|) \leq \frac{a(max(\hat{\mathbf{p}}^{c}) + max(\frac{\|E(UV+E')_{:\mathrm{j}}\|}{m})(\|\hat{\mathbf{p}}^{c}\|+1)))}{min(1-\hat{\mathbf{p}}^{c})\|\hat{\mathbf{p}}^{c}\|}$$

$$\approx \frac{a}{min(1-\hat{\mathbf{p}}^{c})} max\{\frac{max(\hat{\mathbf{p}}^{c})}{\|\hat{\mathbf{p}}^{c}\|}, max(\frac{\|E(UV+E')_{:\mathrm{j}}\|}{m})\}$$

, suggests that when the input matrix and rank-1 patterns are relatively sparse, the weight s approximates $(UV+E)_{\mathrm{i}_0:}$, i.e. largest values in $\mathbf{s}^{r}$ and $\mathbf{s}^{c}$ correspond to the rows and columns of the patterns with largest $P_l\|I_l\|$ and $P_l\|J_l\|$. □

BIND framework is developed to implement *Quantile_shift* algorithm with a BMF or CC method, denoted as $\mathcal{F}$, for a fairer rank-1 pattern identification under the formulation of (6.1). As illustrated in figure 6.2F-J, *Quantile_shift* denoises the majority of the background signal and enables a BMF or CC method better detects $U^{m\times k}$ and $V^{k\times n}$. A cutoff τ is needed to differentiated the weight of the rows or columns with true patterns (figure 6.2E). Empirically, τ could be set from 0.05 to 0.1 in BIND algorithm.

Table 6.1. Jaccard index before/after denoising synthetic data

p \ p_k	single pattern			Multiple pattern		
	0.8	0.9	1.0	0.8	0.9	1.0
0.5	0.17/0.67	0.18/0.79	0.20/0.88	0.28/0.59	0.31/0.73	0.34/0.84
0.6	0.13/0.48	0.14/0.61	0.16/0.73	0.23/0.47	0.26/0.59	0.28/0.69
0.7	0.11/0.29	0.11/0.37	0.13/0.47	0.19/0.34	0.21/0.40	0.22/0.52

BIND is capable for one direction denoising. The $Quantile\_shift$ algorithm is $O(n)$ or $O(m)$ for row or column weight computation and the BIND algorithm is $O(mn)$, which is smaller than most of current BMF and CC methods. The BIND algorithm is detailed below:

Algorithm 12 BIND

Inputs: Input data $X^{m\times n}$, Threshold τ, BMF/CC method $\mathcal{F}$
Outputs: Pattern matrices $U^{m\times k}$ and $V^{k\times n}$
$BIND(X, \tau, \mathcal{F})$:
$X_{use} \leftarrow 0 \cdot X$, $\mathbf{s}^r \leftarrow \mathbf{0}^{m\times 1}$, $\mathbf{s}^c \leftarrow \mathbf{0}^{n\times 1}$
$\hat{\mathbf{p}}^r_{\mathrm{i}} = \frac{\|X_{\mathrm{i}:}\|}{n}\ \forall \mathrm{i} = 1, ..., m$ and $\hat{\mathbf{p}}^c_{\mathrm{j}} = \frac{\|X_{:\mathrm{j}}\|}{m}\ \forall \mathrm{j} = 1, ..., n$
for i=1...m **do**
  $\mathbf{s}^r_{\mathrm{i}} = Quantile\_shift(\mathrm{i}, \hat{\mathbf{p}}^c)$
end for
for j=1...n **do**
  $\mathbf{s}^c_{\mathrm{j}} = Quantile\_shift(\mathrm{j}, \hat{\mathbf{p}}^r)$
end for
$I^r \leftarrow I(\mathbf{s}^r > \tau)$, $I^c \leftarrow I(\mathbf{s}^c > \tau)$, $X_{use} \leftarrow X \circ (I^r \cdot I^{cT})$
$U, V \leftarrow \mathcal{F}(X_{use}, ...)$

6.4 Experiment

We evaluate BIND on synthetic and real-world movie-lens data sets across different data scenarios. We demonstrate the implementation of BIND with different BMF and CC methods can significantly improve fairness in detecting rank-1 pattern from binary matrix with disparate background probability. We also highlight the application of BIND for better interpretation on real-world Movielens data.

We simulate synthetic data sets $X^{100\times 100}$ with fixed size by following (6.1): $X = U^{m\times k}V^{k\times n} + E' + X^0 + E$, with different pattern size $\in \{10, 15, 20\}$, pattern number $k \in \{1, 2\}$, obser-

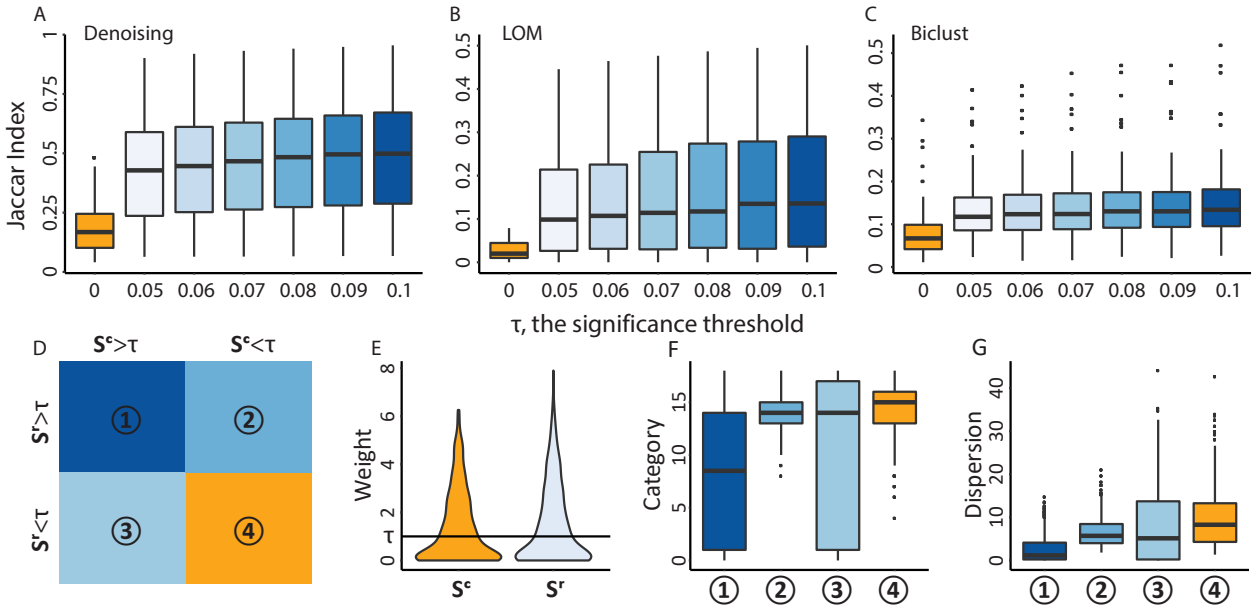

Figure 6.3. Performance on simulated and Movielens data

vation error $p_k \in \{0.8, 0.9, 1.0\}$, background probability $\mathbf{p}^{0,r}, \mathbf{p}^{0,c}$, and element-wise flipping error $p_0 \in \{0, 0.05\}$. Specifically, background probabilities were generated from uniform distribution $\mathbf{p}^{0,r}, \mathbf{p}^{0,c} \sim U[0.1, p]$, where $p \in \{0.5, 0.6, 0.7\}$ corresponds to different background probabilities. We deem 108 data scenarios from the above parameter settings and simulated 30 replicates for each scenario to form a test-bed. Jaccard index $\mathbf{D} = \frac{\|X \cap UV\|}{\|X \cup UV\|}$ ($X =$ *original or denoised data*) is used as the evaluation metric. For each data scenario, denoising performance is evaluated by the averaged Jaccard index on the 30 replicates. We first compare the performance with respect to different significance threshold $\tau = \{0, 0.05 - 1\}$, where $\tau = 0$ represents the data without denoising. As shown in figure 6.3A, the denoising process on average increased the Jaccard index by 2.6 fold and denoising efficiency is slightly increased with τ. Table 6.1 lists the denoising performance with respect to different number of patterns k, background probability p and observation probability p_k, where pattern size is set as 15 and $\tau = 0.1$.

We benchmark BIND by implementing with recently developed BMF method LOM and CC method Biclust, which showed top performance among similar state-of-the-arts methods [115], [139]. The implementation of BIND largely increased the accuracy in detecting true

patterns, which results in an averaged 7.5 (LOM) and 2.6 (Biclust) fold increase of the Jaccard index (figure 6.3B,C) .

We also demostrate that BIND increases the interpretation and denoising in real-world Movielens data, in which $X_{ij} = 1$ represents the interest of user i (row) in rating/watching movie j (column). Category label of each movie is provided. Intuitively, disparate background probablities naturally exist in this data due to different popularity of movies and activity of users. Data is divided into four regions by the I^r and I^c computed in **Algorithm 12** (figure 6.3D,E), where ① is the region most likely with patterns, and ②, ③ and ④ are denoised regions. Users in region ① watched more movies but less categories comparing to other regions (figure 6.3F), suggesting potential recommendation. In addition, region ① has smallest dispersion of the number of rated movies with respect to different categories, suggesting more stable rating preference of users towards their preferred movie types in this region (figure 6.3G).

6.5 Discussion

In this chapter, we focus on the individual bias of binary data. Specially we try to mitigate the "super user" effect in identifying the true patterns. Here we propose BIND, the first algorithm to quantify the bias effect of individual column or row. Though BIND gives the description of the individual bias. Currently, there is no method that could identify the bias and conduct matrix factorization simultaneously.

In the next chapter, we will propose the first method that identify bias and patterns simultaneously in a probabilistic manner.

7. BIAS AWARE PROBABILISTIC BOOLEAN MATRIX FACTORIZATION

7.1 Overview

Boolean matrix is a one type of data representation with binary entries that originates from a wide range of applications including recommendation system, network analysis, collaborative filtering, and biological gene expression [104], [105], [140]–[142]. The goal of **Boolean matrix factorization(BMF)** is to discover hidden patterns from binary data, where it finds a pair of low-rank binary matrices ($X \in \{0,1\}^{m\times k}$, $Y \in \{0,1\}^{k\times n}$) (Figure 7.1A,B,C), whose Boolean product approximates the original input matrix ($A \in \{0,1\}^{m\times n}$), i.e.,

$$A \sim X \otimes Y, \quad A_{\text{ij}} \sim \vee_{l=1}^{k} X_{\text{i}l} \wedge Y_{l\text{j}}.$$

Such low-rank decomposition can capture the local **dependency** between subsets of objects (row of A) and subsets of features (column of A). Specifically, in each rank-1 submatrix resulted from the decomposition into X, Y, i.e., $X_{:l} \otimes Y_{l:}$, it indicates a group of objects (i.e., nonzero entries in $X_{:l}$) sharing the same behavior on a set of features (i.e., nonzero entries in $Y_{l:}$). Here we denote the overall pattern matrix as $Z := X \otimes Y$. For the background error distribution, existing BMF methods tend to assume homoscedastic error distribution, or a universal flipping error with a flipping rate of $p_f = p(A_{\text{ij}} = 0 \| Z_{\text{ij}} = 1) = p(A_{\text{ij}} = 1 \| Z_{\text{ij}} = 0)$. In other words, the objective of BMF is to find the a decomposition of A such that

$$A = (Z + E)\, mod\, 2\,;\ s.t. Z = X \otimes Y, p(E_{\text{ij}} = 1) = p_f$$

where Z, E minimize a certain cost function $\tau(Z, A) = \|E\| = \|A \ominus (X \otimes Y)\|$ (Figure 7.1A,B,C). Here, $mod2$ represents the modulo operation with a quotient of 2, and $\|\cdot\|$ represents a certain norm measure defined by the cost function $\tau(\cdot)$.

Unfortunately, the assumption on homoscedastic error distribution is often violated when applied to complex real-world data, where the individual objects or features may have its

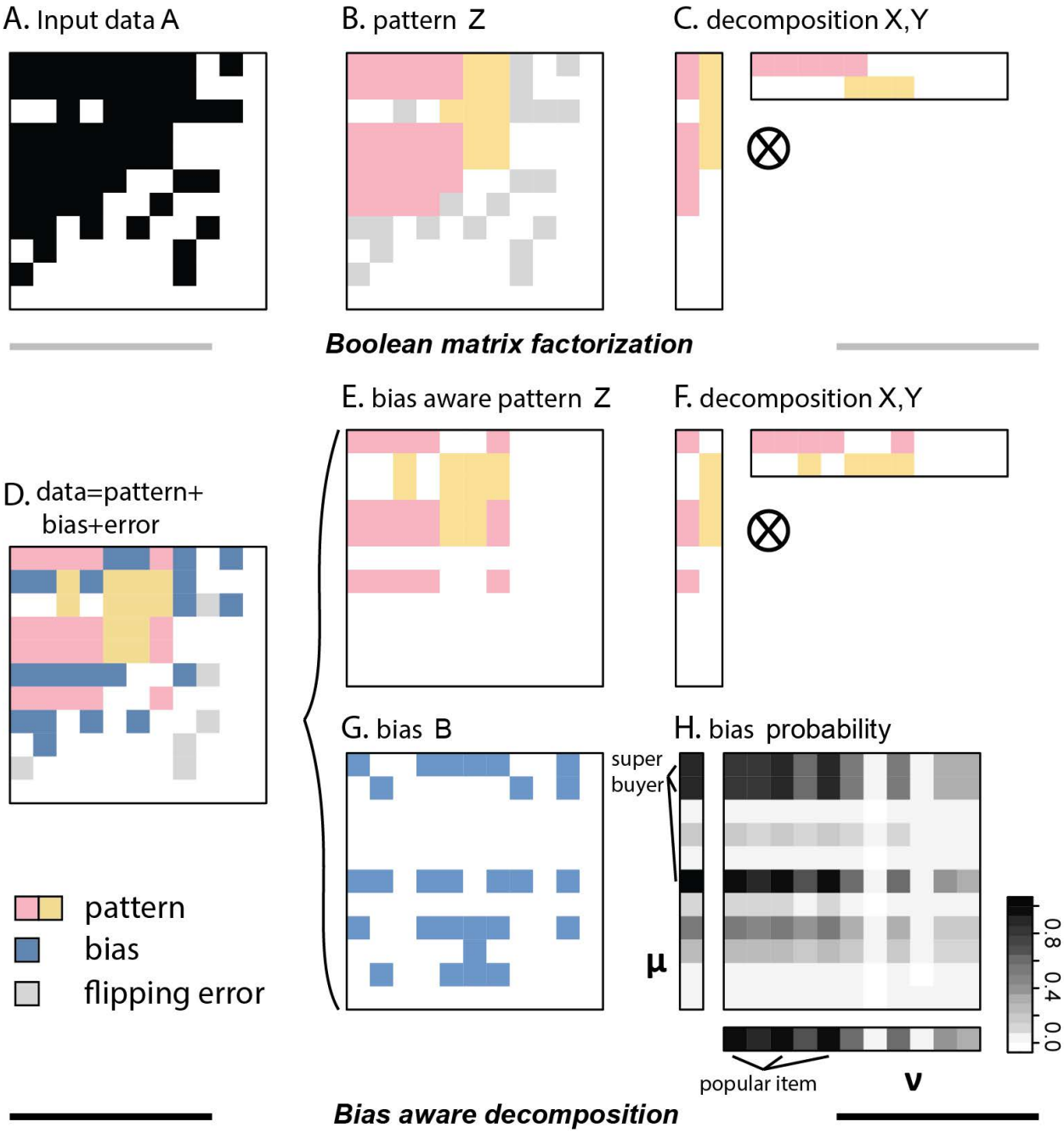

Figure 7.1. BMF with homoscedastic noise model and bias aware BMF with column- and row-specific bias.

specific bias pattern that result in **heteroscedastic error distribution**. Existing BMF methods fail to account for such object- or feature-specific bias, which could severly impact our ability to identify the true pattern Z, as the error matrix E may display row- or column-specific bias [143]. Take the online transaction records data as an example. The observed transaction records data from customers (row) and items (column) are constituted by three components: pattern, bias and flipping error (Figure 7.1D), meaning that aside from stochastic error, to determine whether or not a buyer would purchases a certain item, one should not only look at the purchase pattern that he/she belongs to (Figure 7.1E), but also his/her innate personal purchase preferences and the popularity of the item (Figure 7.1G). For example, a super-buyer, or someone with impulsive buying habits, is very likely to make a purchase regardless of the properties of the items; while a super-item, or a popular item, is also likely to be purchased by users with different characteristics (Figure 7.1B,H).

To mend the gap in binary data analysis, we propose **BABF** (**B**ias **A**ware **B**oolean matrix **F**actorization), the first tool to derive the latent binary pattern (Z), in the presence of individual row-wise and column-wise bias (Figure 7.1D-H), denoted as two real-valued probability vectors $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, with $\boldsymbol{\mu}_{\mathrm{i}} \in [0,1] \, \forall \mathrm{i} \in \{1, ..., m\}$ and $\boldsymbol{\nu}_{\mathrm{j}} \in [0,1] \, \forall \mathrm{j} \in \{1, ..., n\}$. These two vectors represent processes that are object- and feature-specific, and are independent from the pattern generation process, or the homoscedastic background error. In other words, they capture the individual bias generation process that can't be captured by the existing model.

In this work, our contribution is three-fold: 1) BABF is the first method that considers a heteroscedastic error model resulted from object- and feature-specific bias, which is more suitable for modeling real world data; 2) BABF is a highly efficient algorithm in capturing the low rank structures in binary matrix in the presence of individual bias, and showed robust performance in deriving the true patterns across different data scenarios; 3) As a byproduct of pattern discovery, BABF-derived individual bias patterns are highly consistent with the true bias pattern in simulated data and reasonable in real world data, which may lead to practical interpretations depending on different application scenarios.

7.2 Problem formulation

In this section, we formally address our objective to derive the latent Boolean patterns while considering the individual row- and column-wise bias in a probabilistic framework. We first introduce the notations used across this paper, then report the existing probabilistic BMF framework in [114], [115], and then our bias-aware BMF model, BABF.

7.2.1 notation

Matrix, vector and scalar values are denoted by uppercase (A), bold lowercase ($\mathbf{a}$) and lowercase character (a), respectively. The upper-script represents the dimension of the object (e.g. $A^{m\times n}$), while the lower-script indicates the element indices (e.g. i-th row: $A_{\mathrm{i}:}$, j-th column: $A_{:\mathrm{j}}$, and ij-th element: A_{ij}). $\|\cdot\|$ represents a certain type of norm measure. Under Boolean arithmetic, the *and*, *or*, and *not* operations are denoted by $\wedge$, $\vee$, and $\neg$. Subsequently, the Boolean element-wise sum and subtraction are defined as $X \oplus Y = X \vee Y$ and $X \ominus Y = (\neg X \vee Y) \wedge (X \vee \neg Y)$. The Boolean matrix product is defined as $Z = X \otimes Y$, where $Z_{\mathrm{ij}} = \vee_{l=1}^{k} X_{\mathrm{i}k} \wedge Y_{l\mathrm{j}}$.

7.2.2 Existing homoscedastic BMF model

Following [114], [115], each observed entry in a matrix A, i.e. $A_{\mathrm{ij}} \in \{0, 1\}$, is assumed to be generated from the latent pattern Z_{ij} with a homoscedastic error model with universal flipping probability p_f, where the likelihood function is defined as

$$p(A_{\mathrm{ij}} \| Z_{\mathrm{ij}}) = \begin{cases} 1 - p_f, \ \mathrm{i}f \ A_{\mathrm{ij}} = Z_{\mathrm{ij}} \\ p_f, \ \mathrm{i}f \ A_{\mathrm{ij}} \neq Z_{\mathrm{ij}} \end{cases}$$

$$p(A \| Z) = \prod_{\mathrm{ij}}^{m\times n} p(A_{\mathrm{ij}} \| Z_{\mathrm{ij}})$$

As $Z = X \otimes Y$, individual Bernoulli prior is applied on every element of X and Y, i.e.,

$$p(X) = \prod_{\mathrm{i}l}^{m\times k} p(X_{\mathrm{ij}}) \quad p(Y) = \prod_{l\mathrm{j}}^{k\times n} Y_{jl}$$

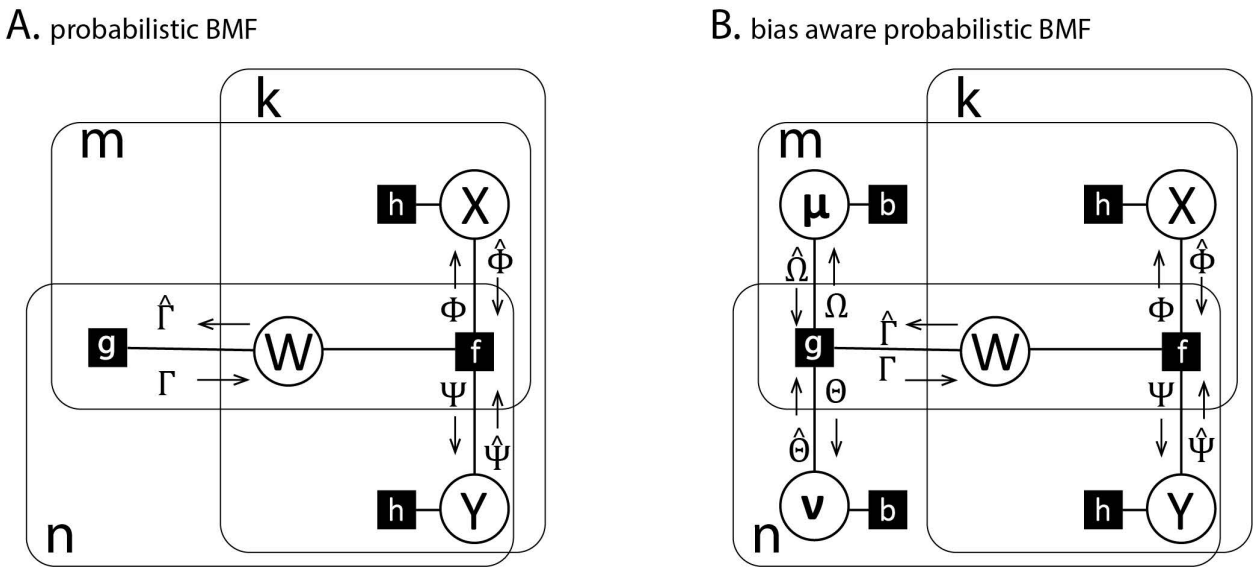

Figure 7.2. The factor graph representation of BMF and Bias-aware BMF. Noted, A is adopted from [114]

Under this formulation, BMF is equivalent to a *Maximum A posterior (**MAP**)* inference problem of X and Y that maximize the overall likelihood function

$$p(X, Y \| A) \propto p(X)p(Y)p(Z \| X, Y)p(A \| Z)$$

Following [114], we assume identical Bernoulli prior on X, Y, represented by factor h, e.g., $h(X_{il}) = log(p(X_{il}))$, $h(Y_{lj}) = log(p(Y_{lj}))$. Here, $p(Z \| X, Y)$ encodes the hard constraint that ensures the equality of the Boolean product, i.e., $Z = X \otimes Y$. By introducing an auxiliary tensor $W \in \{0, 1\}^{m \times n \times k}$, where $W_{ijl} = X_{il} \wedge Y_{lj}$, $Z_{ij} = \vee_{l=1}^{k} W_{ijl}$, this hard constraint is dispersed onto each element in W, and can be reformulated as an identity constraint as

$$p(W_{ijl} \| X_{il}, Y_{lj}) = \mathcal{I}(W_{ijl} = X_{il} \wedge Y_{lj})$$

where for $\mathcal{I}$, we have $\mathcal{I}(true) = 1$ and $\mathcal{I}(false) = 0$. Obviously, if $W_{ijl} \neq X_{il} \wedge Y_{lj}$, the factor $f(W_{ijl}, X_{il}, Y_{lj}) = log(p(W_{ijl} \| X_{il}, Y_{lj}))$ will be evaluated to be $-\infty$. Finally, factor $g(\{W_{ijl}\}, \forall l \in \{1, ..., k\}) = log(p(A_{ij} \| Z_{ij}))$ assess the likelihood of observed variable A_{ij} given

the latent pattern Z_{ij}. Overall, we have the factor graph representation of the log-likelihood $p(X, Y \| A)$ (Figure 7.2A, adopted from [114]) as

$$log(p(X, Y \| A)) = \sum_{\mathrm{ij}}^{m \times n} h(X_{\mathrm{ij}}) + \sum_{l\mathrm{j}}^{k \times n} h(Y_{l\mathrm{j}}) + \sum_{\mathrm{ij}l}^{m \times n \times k} f(W_{\mathrm{ij}l}, X_{\mathrm{i}l}, Y_{l\mathrm{j}}) + \sum_{\mathrm{ij}}^{m \times n} g(\{W_{\mathrm{ij}l}\}_l)$$

Owing to the NP-hard complexity of BMF [108], it is intractable to infer the MAP of the log-likelihood. Alternatively, focusing on the marginal-MAP often yield good empirical success [114], [115], e.g.,

$$\arg\max_{X_{\mathrm{i}l}} log(p(X_{\mathrm{i}l} \| A)) = \arg\max_{X_{\mathrm{i}l}} \sum_{X_{\mathrm{i}'l'} \setminus X_{\mathrm{i}l}, Y_{l'\mathrm{j}'}} log(p(X_{\mathrm{i}'l'}, Y_{l'\mathrm{j}'} \| A))$$

Max-sum belief propagation and Gibbs sampling have been reported to achieve good performance under such strategy [114], [115].

7.2.3 Proposed bias Aware BMF model

The probabilistic BMF model presented above provides a good framework for us to account for the feature- and object-wise bias. Comparing with the homoscedastic setting, the core advancement of our work is to consider the observed data as generated from a process that is more realistic: aside from stochastic error, or the homoscedastic error distribution as in [114], [115], we consider that the observed data is generated not only from the latent pattern $Z = X \otimes Y$, but also from independent object/feature behavior process governed by a bias matrix $B \in \{0, 1\}^{m \times n}$, where B is determined by a row- and column-wise bias vector $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ in such way that

$$p_{B_{\mathrm{ij}}} = p(B_{\mathrm{ij}} = 1) = \boldsymbol{\mu}_{\mathrm{i}} \boldsymbol{\nu}_{\mathrm{j}}$$

And the generation process of A is hence

$$A = B \oplus ((Z + E) \, mod \, 2).$$

The new likelihood of each observations can be characterized in the following four scenarios:

$$
\begin{aligned}
p(A_{\mathrm{ij}} = 1 \| Z_{\mathrm{ij}} = 0) &= 1 - (1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}) \\
p(A_{\mathrm{ij}} = 0 \| Z_{\mathrm{ij}} = 0) &= (1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}) \\
p(A_{\mathrm{ij}} = 1 \| Z_{\mathrm{ij}} = 1) &= 1 - p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}) \\
p(A_{\mathrm{ij}} = 0 \| Z_{\mathrm{ij}} = 1) &= p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})
\end{aligned}
$$

The new posterior probability could then be written as

$$
p(X, Y, \boldsymbol{\mu}, \boldsymbol{\nu} \| A) = p(X)p(Y)p(Z \| X, Y)p(\boldsymbol{\mu})p(\boldsymbol{\nu})p(A \| Z, \boldsymbol{\mu}, \boldsymbol{\nu})
$$

Factor graph representation of the new posterior is shown in Figure 7.2B. Comparing to the existing probabilistic BMF model introduced in 2.2, the new factor graph involves the row- and column-wise bias vectors $\boldsymbol{\mu}, \boldsymbol{\nu}$. Given no prior knowledge of the two variables, we assume a uniform prior on $\boldsymbol{\mu}, \boldsymbol{\nu}$, thus factor $b(\boldsymbol{\mu}_{\mathrm{i}}), b(\boldsymbol{\nu}_{\mathrm{j}})$ evaluate to 0 in the graph. And the likelihood factor g is also related to $\boldsymbol{\mu}, \boldsymbol{\nu}$ in the new formulation.

Same as BMF, the new bias aware formulation is still an NP-hard problem, and we also turn to find the marginal-MAP, which corresponds to optimally estimating individual variables, while the other variables are marginalized. In the next section, we introduce BABF algorithm to derive the decomposition.

7.3 The algorithm of BABF

While we assume our observations A is generated from two sources, latent pattern Z and Bias B; these two sources themselves can be considered as independent from each other. Such independence is also reflected on the factor graph (Figure 7.2B). Though the likelihood factor g and the auxiliary variables W are involved with both $\{X, Y\}$ and $\{\boldsymbol{\mu}, \boldsymbol{\nu}\}$, the direct message update of $\{X, Y\}$ and $\{\boldsymbol{\mu}, \boldsymbol{\nu}\}$ are independent with each other. Conveniently, $\{X, Y, W\}$ and $\{\boldsymbol{\mu}, \boldsymbol{\nu}, W\}$ can be considered as two individual systems to be treated separately.

Algorithm 13 BABF, Bias Aware BMF

Inputs: A, k, p_X, p_Y, p_f, t_{all}, t_{MF}, t_{BI}

BABF:

while $t \leq t_{all}$ and not converged messages **do**

$\boldsymbol{\mu}^{t+1}, \boldsymbol{\nu}^{t+1} \leftarrow$Bias_infer$(A, X^t, Y^t, t_{BI})$

$X^{t+1}, Y^{t+1} \leftarrow$ prob_BMF$(A, k, p_X, p_Y, p_f, \boldsymbol{\mu}^{t+1}, \boldsymbol{\nu}^{t+1}, t_{MF})$

end while

Bias_infer:

$Z := X \otimes Y$

while $t \leq t_{BI}$ and $error\_now < error\_all$ **do**

$error\_all := error\_now$

$\boldsymbol{\mu}_{\mathrm{i}}^{t+1} := \frac{\sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} A_{\mathrm{ij}} \boldsymbol{\nu}_{\mathrm{j}}^{t}}{\sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} \boldsymbol{\nu}_{\mathrm{j}}^{t}} \forall \mathrm{i} \in \{1, ..., m\}$

$\boldsymbol{\nu}_{\mathrm{j}}^{t+1} := \frac{\sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{m} A_{\mathrm{ij}} \boldsymbol{\mu}_{\mathrm{i}}^{t}}{\sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{m} \boldsymbol{\mu}_{\mathrm{i}}^{t}} \forall \mathrm{j} \in \{1, ..., n\}$

$error\_now := \sum_{\mathrm{ij}, Z_{\mathrm{ij}}=0} (A_{\mathrm{ij}} - \boldsymbol{\mu}_{\mathrm{i}} \boldsymbol{\nu}_{\mathrm{j}})^2$

end while

prob_BMF:

$p(A_{\mathrm{ij}} \| Z_{\mathrm{ij}}) \leftarrow$ calculate based on $\boldsymbol{\mu}, \boldsymbol{\nu}$.

Initialize $\Psi_{\mathrm{ij}l}^0$, $\hat{\Psi}_{\mathrm{ij}l}^0$, $\Phi_{\mathrm{ij}l}^0$, $\hat{\Phi}_{\mathrm{ij}l}^0$, $\Gamma_{\mathrm{ij}l}^0$, $\hat{\Gamma}_{\mathrm{ij}l}^0, \forall \mathrm{i}, \mathrm{j}, l$

while $t \leq t_{MF}$ and not converged messages **do**

$\Phi_{\mathrm{ij}l}^{t+1} := \max(\Gamma_{\mathrm{ij}l}^t + \hat{\Psi}_{\mathrm{ij}l}^t, 0) - \max(\Psi_{\mathrm{ij}l}^t, 0)$

$\Psi_{\mathrm{ij}l}^{t+1} := \max(\Gamma_{\mathrm{ij}l}^t + \hat{\Phi}_{\mathrm{ij}l}^t, 0) - \max(\Phi_{\mathrm{ij}l}^t, 0)$

$\hat{\Phi}_{\mathrm{ij}l}^{t+1} := log(\frac{1-p_f}{p_f}) + \sum_{\mathrm{j}' \neq \mathrm{j}} \Phi_{\mathrm{ij}'l}^t$

$\hat{\Psi}_{\mathrm{ij}l}^{t+1} := log(\frac{1-p_f}{p_f}) + \sum_{\mathrm{i}' \neq \mathrm{i}} \Psi_{\mathrm{i}'\mathrm{j}l}^t$

$\Gamma_{\mathrm{ij}l}^{t+1} := \min(log(\frac{p(A_{\mathrm{ij}} \| 1)}{p(A_{\mathrm{ij}} \| 0)}) + \sum_{l' \neq l} \max(\Gamma_{\mathrm{ij}l'}^t)), \max(0, -\max_{l' \neq l} \hat{\Gamma}_{\mathrm{ij}l'}^t))$

$\hat{\Gamma}_{\mathrm{ij}l}^{t+1} := \min(\hat{\Phi}_{\mathrm{ij}l}^t + \hat{\Psi}_{\mathrm{ij}l}^t, \hat{\Psi}_{\mathrm{ij}l}^k, \hat{\Phi}_{\mathrm{ij}l}^t)$

end while

$$X_{il} = \begin{cases} 1 & \text{if } log(\frac{1-p_f}{p_f}) + \sum_{\mathrm{i}=1}^{m} \Phi_{\mathrm{ij}l}^t > 0 \\ 0 & otherwise. \end{cases}$$

$$Y_{l\mathrm{j}} = \begin{cases} 1 & \text{if } log(\frac{1-p_f}{p_f}) + \sum_{\mathrm{i}=1}^{m} \Psi_{\mathrm{ij}l}^t > 0 \\ 0 & otherwise. \end{cases}$$

Outputs: $X, Y, \boldsymbol{\mu}, \boldsymbol{\nu}$

Here we introduce BABF in algorithm 13. BABF has two core components, prob_BMF and Bias_infer, corresponding to the derivations of $\{X, Y\}$ and $\{\boldsymbol{\mu}, \boldsymbol{\nu}\}$. Other than the input data A, BABF takes the pattern number parameters k, the Bernoulli prior of X, Y, filling error p_f and the maximum iterations for the overall algorithm as well as core components $(t_{all}, t_{MF}, t_{BI})$ as input, and outputs the decomposition X, Y and the bias vectors $\boldsymbol{\mu}, \boldsymbol{\nu}$.

When fixing bias vectors $\boldsymbol{\mu}, \boldsymbol{\nu}$, the only differences between bias aware BMF and the existing BMF is that each likelihood factor g_{ij} would evaluate to different probability assignments by referencing $\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\nu}_{\mathrm{j}}$. Following [114], we utilize the max-sum belief propagation (BP) strategy to approximate the overall likelihood in prob_BMF. Correspondingly, the message $\Gamma_{\mathrm{ij}l}$ that propagates the likelihood information to auxiliary variable W would be different from [114] with individualized probabilities.

The inference of the marginal-MAP of $\boldsymbol{\mu}, \boldsymbol{\nu}$ is a non-trivial task even with accurate pattern information Z, as for any bias variable $\boldsymbol{\mu}_{\mathrm{i}}$, any observation related to this variable is related to a different $\boldsymbol{\nu}_{\mathrm{i}}$, and vice versa. To circumvent this computational challenge, we adopted two modifications. 1) we only consider the observations that are not covered by pattern Z for bias inference. We argue the pattern related observations have marginal contribution to the bias inference and could be omitted. 2) Instead of deriving exact MAP of likelihood, we treat this as an optimization problem, where we could utilize conventional loss functions to achieve the same objective that optimize the difference between $\boldsymbol{\mu}, \boldsymbol{\nu}$ and background information. Inspired from [143], we apply a modified mean square loss. Take $\boldsymbol{\mu}_{\mathrm{i}}$ as an example, the loss function takes the form of

$$\Omega_{\mathrm{i}} = \sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} \boldsymbol{\nu}_{\mathrm{j}}^{t} (A_{\mathrm{ij}} - \boldsymbol{\mu}_{\mathrm{i}}^{t})^2$$

The most important benefit of this modified loss is that it ensures the probability of each $\boldsymbol{\mu}_{\mathrm{i}}$ would be from [0,1] and still consider the impact from $\boldsymbol{\nu}_{\mathrm{j}}$ for each observation A_{ij}. Moreover, it is with high computational feasibility as the updated $\boldsymbol{\mu}_{\mathrm{i}}^{t+1}$ could be easily derived as $\boldsymbol{\mu}_{\mathrm{i}}^{t+1} := \frac{\sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} A_{\mathrm{ij}} \boldsymbol{\nu}_{\mathrm{j}}^{t}}{\sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} \boldsymbol{\nu}_{\mathrm{j}}^{t}}$ Correspondingly, $\boldsymbol{\nu}_{\mathrm{j}}^{t+1} := \frac{\sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{m} A_{\mathrm{ij}} \boldsymbol{\mu}_{\mathrm{i}}^{t}}{\sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{n} \boldsymbol{\mu}_{\mathrm{i}}^{t}}$. Here, we implement this strategy in Bias_infer. Empirically, it is robust for the bias inference across different scenarios, which we will introduce in detail in the experiments section.

7.4 Message passing

Despite the dense connections in the factor graph, max-sum belief propagation achieved admirable performance in the case of approximate the MAP of Boolean matrix factorization [114]. Here we also utilize this strategy that not only derive the MAP of matrix decomposition X and Y, but also infer the background row- and column-wise bias $\boldsymbol{\mu}, \boldsymbol{\nu}$. Though the information of $\boldsymbol{\mu}, \boldsymbol{\nu}$ and X, Y communicates through likelihood factor g and auxiliary variable W, their independence of each other resulted in disconnected message update between $\boldsymbol{\mu}, \boldsymbol{\nu}$ and X, Y. Conveniently, $\{X, Y, W\}$ and $\boldsymbol{\mu}, \boldsymbol{\nu}, W\}$ can be considered as two separate systems. In this paper we focus on the message update of $\{\boldsymbol{\mu}, \boldsymbol{\nu}, W\}$, and adopt the algorithm in [114] for $\{X, Y, W\}$.

7.4.1 update X,Y,W

Variables to factor message.

Conveniently, all the variables in $\{X, Y, W\}$ are binary variables ($X_{\mathrm{i}l}$, $Y_{l\mathrm{j}}$, $W_{\mathrm{ij}l} \in \{0, 1\}$). Following the notation in [114], we denote the message between factors and variables as $\mathbf{m}$ (e.g., $\mathbf{m}_{X_{\mathrm{i}l} \to f_{\mathrm{ij}l}}(X_{\mathrm{ij}}) : \{0, 1\} \to \mathcal{R}$). Max-sum BP is utilized to calculate the outgoing message, while consideration all incoming messages from neighbor factors, despite the receiving one, e.g.,

$$\mathbf{m}_{X_{\mathrm{i}l} \to f_{\mathrm{ij}l}}(X_{\mathrm{ij}})^{t+1} = \mathbf{m}_{h_{\mathrm{i}l} \to X_{\mathrm{i}ll}}(X_{\mathrm{i}l})^t + \sum_{\mathrm{j}' \neq \mathrm{j}} \mathbf{m}_{f_{\mathrm{ij}'l} \to X_{\mathrm{i}l}}(X_{\mathrm{i}l})^t$$

Our objective is to achieve the maximum likelihood, which align with the difference between the message of $\mathbf{m}_{X_{\mathrm{i}l} \to f_{\mathrm{ij}l}}(X_{\mathrm{ij}} = 1)$ and $\mathbf{m}_{X_{\mathrm{i}l} \to f_{\mathrm{ij}l}}(X_{\mathrm{ij}} = 0)$, i.e.,

$$\hat{\Phi} = \mathbf{m}_{X_{\mathrm{i}l} \to f_{\mathrm{ij}l}}(1) - \mathbf{m}_{X_{\mathrm{i}l} \to f_{\mathrm{ij}l}}(0)$$

In the case of individual variable $X_{\mathrm{i}l}$ to the factor $f_{\mathrm{ij}l}$

$$\begin{aligned}
\hat{\Phi}_{\mathrm{ij}l}^{t+1} &= (\mathbf{m}_{h_{\mathrm{i}l} \to X_{\mathrm{i}ll}}(1)^t + \sum_{\mathrm{j}' \neq \mathrm{j}} \mathbf{m}_{f_{\mathrm{ij}'l} \to X_{\mathrm{i}l}}(1)^t) - (\mathbf{m}_{h_{\mathrm{i}l} \to X_{\mathrm{i}ll}}(0)^t + \sum_{\mathrm{j}' \neq \mathrm{j}} \mathbf{m}_{f_{\mathrm{ij}'l} \to X_{\mathrm{i}l}}(0)^t) \\
&= log(\frac{p(X_{\mathrm{i}l} = 1)}{p(X_{\mathrm{i}l} = 0)}) + \sum_{\mathrm{j}' \neq \mathrm{j}} \Phi_{\mathrm{ij}'l}^t
\end{aligned}$$

Similarly, the message $\hat{\Psi}$ can be derived as

$$\hat{\Psi}_{\mathrm{ij}l} = log(\frac{p(Y_{l\mathrm{j}} = 1)}{p(Y_{l\mathrm{j}} = 1)}) + \sum_{\mathrm{i}' \neq \mathrm{i}} \Psi^t_{\mathrm{i}'\mathrm{j}l}$$

For W, since each variable $W_{\mathrm{ij}l}$ has exact two factor neighbors g_{ij}, $f_{\mathrm{ij}l}$, the message from $W_{\mathrm{ij}k}$ to either factors is the message from the other factor, i.e.,

$$\mathbf{m}_{W_{\mathrm{ij}l} \to g_{\mathrm{ij}}}(W_{\mathrm{ij}l}) = \mathbf{m}_{f_{\mathrm{ij}l} \to W_{\mathrm{ij}l}}(W_{\mathrm{ij}l})$$

$$\mathbf{m}_{g_{\mathrm{ij}} \to W_{\mathrm{ij}l}}(W_{\mathrm{ij}l}) = \mathbf{m}_{W_{\mathrm{ij}l} \to f_{\mathrm{ij}l}}(W_{\mathrm{ij}l})$$

We will discuss in detail of the message involve factor g in next section.

factor to variable message

For factor h, it only connect to the single variable $X_{\mathrm{i}l}$ or $Y_{l\mathrm{j}}$, which works as prior knowledge for the sparsity of X and Y, where their information is passed through

$$h_{\mathrm{i}l}(X_{\mathrm{i}l} = 1) - h_{\mathrm{i}l}(X_{\mathrm{i}l} = 0) = log(\frac{p(X_{\mathrm{i}l} = 1)}{p(X_{\mathrm{i}l} = 0)})$$

$$h_{l\mathrm{j}}(Y_{l\mathrm{j}} = 1) - h_{l\mathrm{j}}(Y_{l\mathrm{j}} = 0) = log(\frac{p(Y_{l\mathrm{j}} = 1)}{p(Y_{l\mathrm{j}} = 0)})$$

Factor f links X, Y with the auxiliary variable W, that ensures $W_{\mathrm{ij}l} = X_{\mathrm{i}l} \wedge Y_{l\mathrm{j}}$, i.e.,

$$f(X_{\mathrm{i}l}, Y_{l\mathrm{j}}, W_{\mathrm{ij}l}) = log(\mathcal{I}(W_{\mathrm{ij}l} = X_{\mathrm{i}l} \wedge Y_{l\mathrm{j}}))$$

Notably, $f(X_{\mathrm{i}l}, Y_{l\mathrm{j}}, W_{\mathrm{ij}l}) \to -\infty$ if $W_{\mathrm{ij}l} \neq X_{\mathrm{i}l} \wedge Y_{l\mathrm{j}}$. Such that it restrict the message scenarios when passing the information from f to X, Y. Here, we use $\mathbf{m}_{f_{\mathrm{ij}l} \to X_{\mathrm{i}l}(X_{\mathrm{i}l})}$ as example, where $\mathbf{m}_{f_{\mathrm{ij}l} \to Y_{l\mathrm{j}}(Y_{l\mathrm{j}})}$ can be similarly derived. For $X_{\mathrm{i}l}$ to equal to 1, if $Y_{l\mathrm{j}} = 1$, restricted by f, $W_{\mathrm{ij}l} = 1$, and if $Y_{l\mathrm{j}} = 0$, $W_{\mathrm{ij}l} = 0$, thus,

$$\mathbf{m}_{f_{\mathrm{ij}l} \to X_{\mathrm{i}l}}(1)^{t+1} = max(\mathbf{m}_{Y_{l\mathrm{j}} \to f_{\mathrm{i}ll}}(1)^t + \mathbf{m}_{W_{\mathrm{ij}l} \to f_{\mathrm{ij}l}}(1)^t, \quad \mathbf{m}_{Y_{l\mathrm{j}} \to f_{\mathrm{i}ll}}(0)^t + \mathbf{m}_{W_{\mathrm{ij}l} \to f_{\mathrm{ij}l}}(0)^t)$$

While if $X_{\mathrm{i}l} = 0$, $W_{\mathrm{ij}l} = 0$ regardless the value of $Y_{l\mathrm{j}}$, i.e.,

$$\mathbf{m}_{f_{\mathrm{ij}l}\to X_{\mathrm{i}l}}(0)^{t+1} = max(\mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{i}ll}}(1)^t + \mathbf{m}_{W_{\mathrm{ij}l}\to f_{\mathrm{ij}l}}(0)^t, \quad \mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{i}ll}}(1)^t + \mathbf{m}_{W_{\mathrm{ij}l}\to f_{\mathrm{ij}l}}(0)^t)$$

Since $\hat{\Psi}_{\mathrm{ij}l} = \mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{ij}l}}(1) - \mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{ij}l}}(0)$, and $\Gamma_{\mathrm{ij}l} = \mathbf{m}_{W_{\mathrm{ij}l}\to f_{\mathrm{ij}l}}(1) - \mathbf{m}_{W_{\mathrm{ij}l}\to f_{\mathrm{ij}l}}(0)$ the message from f to X can be derived as

$$\Phi_{\mathrm{ij}l} = \mathbf{m}_{f_{\mathrm{ij}l}\to X_{\mathrm{i}l}}(1) - \mathbf{m}_{f_{\mathrm{ij}l}\to X_{\mathrm{i}l}}(0) = max(\Gamma_{\mathrm{ij}l} + \hat{\Psi}_{\mathrm{ij}l}, 0) - max(\hat{\Psi}_{\mathrm{ij}l}, 0)$$

Similarly, we have

$$\Psi_{\mathrm{ij}l} = \mathbf{m}_{f_{\mathrm{ij}l}\to Y_{\mathrm{i}l}}(1) - \mathbf{m}_{f_{\mathrm{ij}l}\to Y_{\mathrm{i}l}}(0) = max(\Gamma_{\mathrm{ij}l} + \hat{\Phi}_{\mathrm{ij}l}, 0) - max(\hat{\Phi}_{\mathrm{ij}l}, 0)$$

Following the same strategy, while considering the message from factor f to variable W, if $W_{\mathrm{ij}l} = 1$, $X_{\mathrm{i}l} = Y_{l\mathrm{j}} = 1$, whereas if $W_{\mathrm{ij}l} = 0$, either $X_{\mathrm{i}l}$ or $Y_{l\mathrm{j}}$ should equal to zero, i.e.,

$$\mathbf{m}_{f_{\mathrm{ij}l}\to W_{\mathrm{ij}l}}(1)^{t+1} = \mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{i}ll}}(1)^t + \mathbf{m}_{X_{\mathrm{i}l}\to f_{\mathrm{ij}l}}(1)^t$$

$$\mathbf{m}_{f_{\mathrm{ij}l}\to W_{\mathrm{ij}l}}(0)^{t+1} = max(\mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{i}ll}}(1)^t + \mathbf{m}_{X_{\mathrm{i}l}\to f_{\mathrm{ij}l}}(0)^t, \mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{i}ll}}(0)^t + \mathbf{m}_{X_{\mathrm{i}l}\to f_{\mathrm{ij}l}}(1)^t,$$
$$\mathbf{m}_{Y_{l\mathrm{j}}\to f_{\mathrm{i}ll}}(0)^t + \mathbf{m}_{X_{\mathrm{i}l}\to f_{\mathrm{ij}l}}(0)^t)$$

Such that

$$\hat{\Gamma}_{\mathrm{ij}k} = \mathbf{m}_{f_{\mathrm{ij}l}\to W_{\mathrm{ij}l}}(1) - \mathbf{m}_{f_{\mathrm{ij}l}\to W_{\mathrm{ij}l}}(0) = min(\hat{\Phi}_{\mathrm{ij}l} + \hat{\Psi}_{\mathrm{ij}l}, \hat{\Phi}_{\mathrm{ij}l}, \hat{\Psi}_{\mathrm{ij}l})$$

7.4.2 update μ, ν, W

In the previous section, we have derived the messages passing between the X, Y and W. In this section, we derive the message passing between μ, $\mathbf{v}$ and W, where they all related to the likelihood factor g. Also, different with binary variable W, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are Bernoulli variable, that the simplified singleton message does not applied for their message update.

We first reinstate the log likelihood function of each element (A_{ij}) that represent the factor g.

$$
\begin{aligned}
p(A_{\mathrm{ij}} = 1 \| Z_{\mathrm{ij}} = 0) &= 1 - (1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}) \\
p(A_{\mathrm{ij}} = 0 \| Z_{\mathrm{ij}} = 0) &= (1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}) \\
p(A_{\mathrm{ij}} = 1 \| Z_{\mathrm{ij}} = 1) &= 1 - p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}) \\
p(A_{\mathrm{ij}} = 0 \| Z_{\mathrm{ij}} = 1) &= p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})
\end{aligned}
$$

In the case of Bernoulli variables $\boldsymbol{\mu}_{\mathrm{i}}$, the incoming message from factor g to $\boldsymbol{\mu}_{\mathrm{i}}$ is certainly the likelihood information,

$$\Omega_{\mathrm{i}} = log(\prod_{\mathrm{j}=1}^{n} p(A_{\mathrm{ij}}))$$

while the message from $\boldsymbol{\mu}_{\mathrm{i}}$ to g would be the MAP of the posterior distribution, i.e.,

$$\hat{\Omega}_{\mathrm{i}} = \arg\max_{\boldsymbol{\mu}_{\mathrm{i}}} log(\prod_{\mathrm{j}=1}^{n} p(A_{\mathrm{ij}})p(\boldsymbol{\mu}_{\mathrm{i}})) = \arg\max_{\boldsymbol{\mu}_{\mathrm{i}}} (\sum_{\mathrm{j}=1}^{n} log(p(A_{\mathrm{ij}})) + b_{\mathrm{i}}(\boldsymbol{\mu}_{\mathrm{i}}))$$

Given no knowledge on the bias before hand, here we impose a uniform prior on the Bernoulli variable, such that $b_{\mathrm{i}}(\boldsymbol{\mu}_{\mathrm{i}}) = 0$. In addition, the log posterior is related to 4 situations,

$$
\begin{aligned}
\Omega_{\mathrm{i}} = &\sum_{\mathrm{j}=1, A_{\mathrm{ij}}=1, Z_{\mathrm{ij}}=0}^{n} log(1 - (1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})) + \sum_{\mathrm{j}=1, A_{\mathrm{ij}}=1, Z_{\mathrm{ij}}=1}^{n} log(1 - p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})) \\
&+ \sum_{\mathrm{j}=1, A_{\mathrm{ij}}=0, Z_{\mathrm{ij}}=0}^{n} log((1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})) + \sum_{\mathrm{j}=1, A_{\mathrm{ij}}=0, Z_{\mathrm{ij}}=1}^{n} log(p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})
\end{aligned}
$$

Here we assume $P_f \to 0$, such that $p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{v}_{\mathrm{i}}) \to 0$, and both $\sum_{\mathrm{j}=1, A_{\mathrm{ij}}=1, Z_{\mathrm{ij}}=1}^{n} log(1 - p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}))$ and $\sum_{\mathrm{j}=1, A_{\mathrm{ij}}=0, Z_{\mathrm{ij}}=1}^{n} log(p_f(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})$ can be approximate by a constant that does not contribute to the inference of $\hat{\Omega}_{\mathrm{i}}$. Also $(1 - p_f)(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})$ can be approximated by $(1 - \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}})$. It also has practical meanings, that for the inference of background bias, we only consider the values that are not covered by the latent pattern X, Y. While our objective is to infer $\boldsymbol{\mu}_{\mathrm{i}}$ that better reflect the background information of $A_{\mathrm{i}:}$. However, it is still non-trivial to derive $\hat{\Omega}_{\mathrm{i}}$ as every observation is related to a different $\boldsymbol{v}_{\mathrm{j}}$. Instead of deriving exact MAP

of likelihood, we treat this as an optimization problem, where we could utilize conventional loss function to achieve the same objective that optimize the difference between $\boldsymbol{\mu}_{\mathrm{i}}$ with $A_{\mathrm{i}:}$. Here, we apply a modified mean square loss, i.e.,

$$\Omega = \sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} \boldsymbol{v}_{\mathrm{j}} (A_{\mathrm{ij}} - \boldsymbol{\mu}_{\mathrm{i}})^2$$

The most important benefit of this modified loss is that it ensures the probability of each $\boldsymbol{\mu}_{\mathrm{i}}$ would be from $[0, 1]$ and still consider the impact from $\boldsymbol{v}_{\mathrm{j}}$ for each observations. Conveniently, $\hat{\Omega}_{\mathrm{i}}$ is inferred from the derivative of Ω, i.e.,

$$\hat{\Omega} = \arg\max_{\boldsymbol{\mu}_{\mathrm{i}}} \Omega = \frac{\sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} A_{\mathrm{ij}} v_{\mathrm{j}}}{\sum_{\mathrm{j}=1, Z_{\mathrm{ij}}=0}^{n} v_{\mathrm{j}}}$$

Similarly, we have

$$\Theta_{\mathrm{j}} = \sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{m} \boldsymbol{\mu}_{\mathrm{i}} (A_{\mathrm{ij}} - \boldsymbol{v}_{\mathrm{j}})^2$$

$$\hat{\Theta}_{\mathrm{j}} = \frac{\sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{m} A_{\mathrm{ij}} \boldsymbol{\mu}_{\mathrm{i}}}{\sum_{\mathrm{i}=1, Z_{\mathrm{ij}}=0}^{n} \boldsymbol{\mu}_{\mathrm{i}}}$$

Now we have derived all messages in the likelihood despite $\Gamma_{\mathrm{ij}l} : \mathbf{m}_{g_{\mathrm{ij}} \to W_{\mathrm{ij}l}}$ that passed the information from the likelihood factor to each of auxiliary variable $W_{\mathrm{ij}l}$. Overall, the message take the form of

$$\mathbf{m}_{g_{\mathrm{ij}} \to W_{\mathrm{ij}l}}(W_{\mathrm{ij}l})^{t+1} = \max_{W_{\mathrm{ij}l'}, l' \neq l} (g_{\mathrm{ij}}(Z_{\mathrm{ij}}, \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{v}_{\mathrm{j}}) \ + \sum_{l' \neq l} \mathbf{m}_{W_{\mathrm{ij}l'} \to g_{\mathrm{ij}}}(W_{\mathrm{ij}l'})^t)$$

When updating $W_{\mathrm{ij}l}$, we consider two scenarios: 1. $Z_{\mathrm{ij}} = \vee_{l=1}^{k} W_{\mathrm{ij}l} = 1$ with likelihood factor $p(A_{\mathrm{ij}} \| Z_{\mathrm{ij}} = 1)$ and 2. $\vee_{l=1}^{k} W_{\mathrm{ij}l} = 0$, $p(A_{\mathrm{ij}} \| Z_{\mathrm{ij}} = 0)$.

$W_{\mathrm{ij}l} = 1$ falls into the situation of scenarios 1, that no matter the value of $W_{\mathrm{ij}l'}$, $Z_{\mathrm{ij}} = \vee W_{\mathrm{ij}l} = 1$. The message for $W_{\mathrm{ij}l} = 1$ can be derived as

$$\begin{aligned}
\mathbf{m}_{g_{\mathrm{ij}} \to W_{\mathrm{ij}l}}(1) &= \max_{W_{\mathrm{ij}l'}, l' \neq l} (g_{\mathrm{ij}}(Z_{\mathrm{ij}}, \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{v}_{\mathrm{j}}) + \sum_{l' \neq l} \mathbf{m}_{W_{\mathrm{ij}l'} \to g_{\mathrm{ij}}}(W_{\mathrm{ij}l'})^t) \\
&= log(p_(A_{\mathrm{ij}} \| 1)) + \sum_{l' \neq l} max(\mathbf{m}_{W_{\mathrm{ij}l'} \to g_{\mathrm{ij}}}(1), \mathbf{m}_{W_{\mathrm{ij}l'} \to g_{\mathrm{ij}}}(0))
\end{aligned}$$

$W_{ijl} = 0$ could involve both cases. If $Z_{ij} = 0$, all $W_{ijl'} = 0$, i.e.,

$$\mathbf{m}_{g_{ij} \to W_{ijl}}(0) = log(p_(A_{ij} \| 0)) + \sum_{l' \neq l} \mathbf{m}_{W_{ijl'} \to g_{ij}}(0)$$

If $Z_{ij} = 1$, at least one of $W_{ijl'}$ equal to zero. To achieve the maximum likelihood, the $W_{ijl'}$ with the maximum likelihood difference on 0 or 1 should be set as 1, we denote it as W_{ijl^*}, where $l^* = \arg\max_{l' \neq l}(\mathbf{m}_{W_{ijl'} \to g_{ij}}(1) - \mathbf{m}_{W_{ijl'} \to g_{ij}}(0))$, such that we have

$$\mathbf{m}_{g_{ij} \to W_{ijl}}(0) = log(p_(A_{ij} \| 1)) + \sum_{l' \neq l} max(\mathbf{m}_{W_{ijl'} \to g_{ij}}(1), \mathbf{m}_{W_{ijl'} \to g_{ij}}(0)) - \mathbf{m}_{W_{ijl*} \to g_{ij}}(0)$$

Taken together,

$$\mathbf{m}_{g_{ij} \to W_{ijl}}(0) = max(log(p_(A_{ij} \| 0)) + \sum_{l' \neq l} \mathbf{m}_{W_{ijl'} \to g_{ij}}(0), log(p_(A_{ij} \| 1))$$
$$+ \sum_{l' \neq l} max(\mathbf{m}_{W_{ijl'} \to g_{ij}}(1), \mathbf{m}_{W_{ijl'} \to g_{ij}}(0)) - \mathbf{m}_{W_{ijl*} \to g_{ij}}(0))$$

Therefore

$$\Gamma_{ijl} = \mathbf{m}_{g_{ij} \to W_{ijl}}(1) - \mathbf{m}_{g_{ij} \to W_{ijl}}(0) = log(p_(A_{ij} \| 1)) + \sum_{l' \neq l} max(\mathbf{m}_{W_{ijl'} \to g_{ij}}(1), \mathbf{m}_{W_{ijl'} \to g_{ij}}(0))$$
$$- max(log(p_(A_{ij} \| 0)) + \sum_{l' \neq l} \mathbf{m}_{W_{ijl'} \to g_{ij}}(0), log(p_(A_{ij} \| 1)) + \sum_{l' \neq l} max(\mathbf{m}_{W_{ijl'} \to g_{ij}}(1), \mathbf{m}_{W_{ijl'} \to g_{ij}}(0))$$
$$- \mathbf{m}_{W_{ijl*} \to g_{ij}}(0)) = min(log(\frac{p(A_{ij} \| 1)}{p(A_{ij} \| 0)}) + \sum_{l' \neq l} max(0, \hat{\Gamma}^t_{ijl'}), max(0, -max_{l' \neq l} \hat{\Gamma}^t_{ijl'}))$$

7.5 Experiments

We evaluate the performance of our bias aware model on both synthetic and real world datasets. We first introduce related methods for BMF and report the benchmark performance across different simulated data scenarios. We then highlight the practical use of BABF in our analysis of a movielens and gene expression data.

In this section we illustrate the performance of BABF, the first tool to simultaneously derive both $\{X, Y\}$ and $\{\boldsymbol{\mu}, \boldsymbol{\nu}\}$.

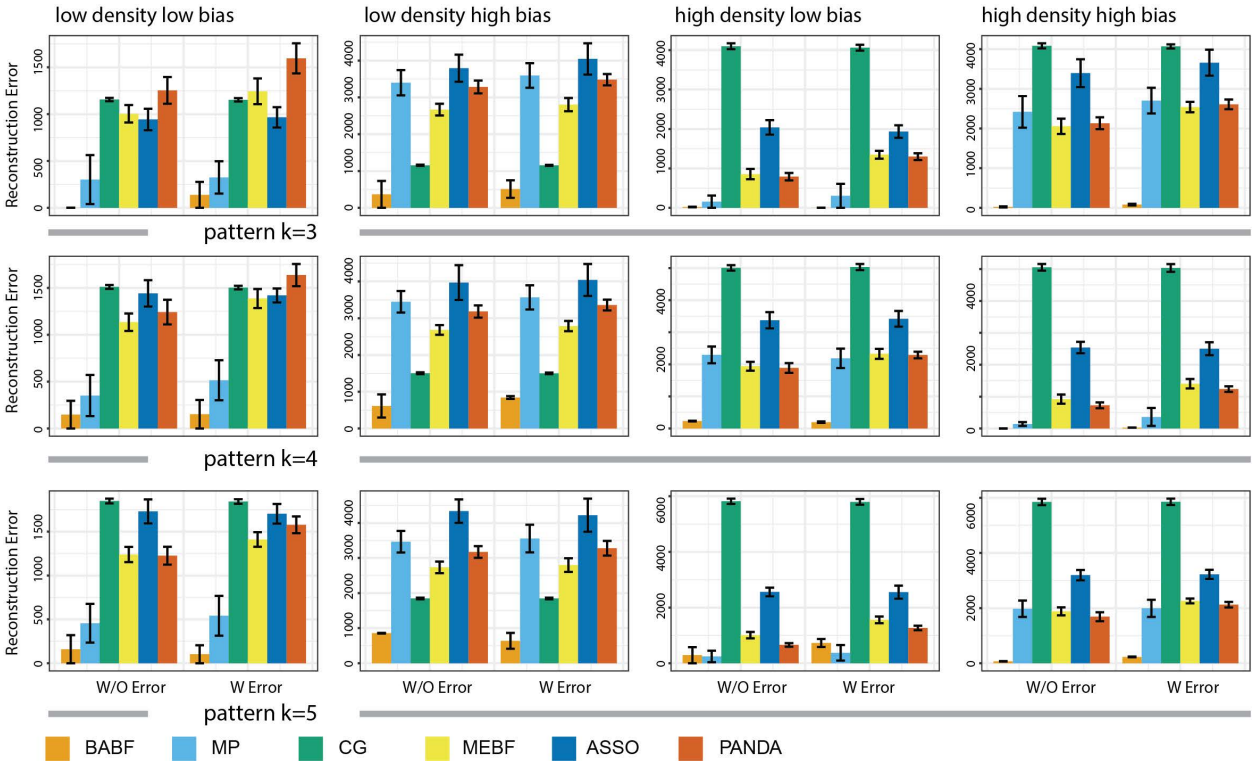

Figure 7.3. Performance comparison on simulated data

7.5.1 related work

In addition to the probabilistic methods introduced above [114], [115], different heuristic methods have been developed to solve the BMF problem. Previously [143] systematically discussed the bias issue in BMF, but their focus is to explore the identifiability of the patterns in the presence of bias in the noise model. For the rest of the methods, none of them considered the heteroscesdastic issue of the error distribution. Among these methods, ASSO represents a series of work from Miettinen et al [25], [110], [144], [145]. ASSO first generates a pool of column basis from row-wise correlation matrix, and iteratively search for the best column and row basis following a pre-defined cost function. PANDA is another series of heuristic methods that embed the cost function in the search of top_k core patterns [111], [112]. Formal Concept Analysis also showed empirical success in BMF [146]–[148]. More recently, [121] proposed a fast algorithm by formulating submatrix pattern identification in a geometric perspective. [149] formulates BMF as an integer program problem and utilize

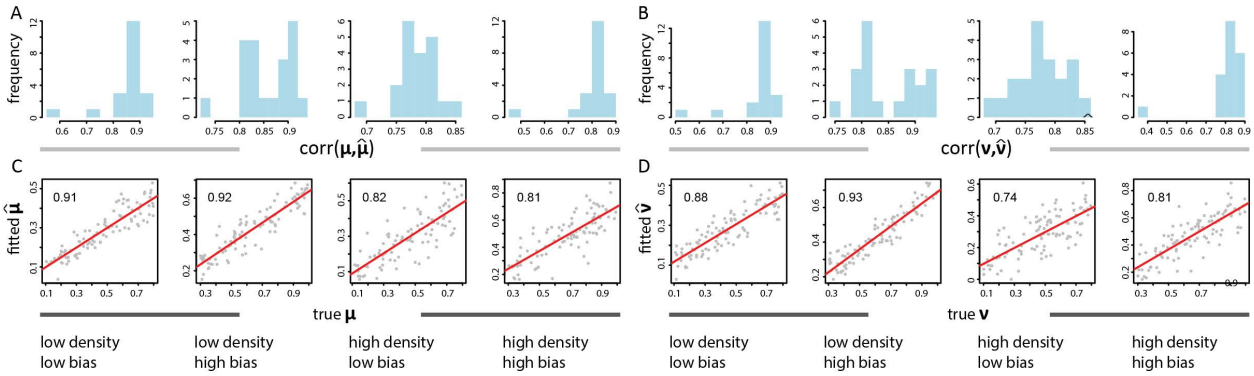

Figure 7.4. BABF inferred bias is highly correlated with ground truth bias

column generation framework to search for the best solutions. Here, we benchmark the performance of BABF with MP [114], CG [149], MEBF [121], ASSO [25] and PANDA [111] and believe that this set of methods represent the state-of-the-art performance of BMF in different perspectives.

7.5.2 benchmark on simulated data

We simulate an observed binary matrix A by the following model:

$$A = B \oplus ((Z + E)\, mod\, 2).$$

Here, B, Z, E represent the column-/row-wise bias matrix, pattern matrix and error matrix respectively. Each entry in $B, E \in \{0,1\}^{m\times n}$ is simulated to follow Bernoulli distribution with success probabilities $p(B_{\mathrm{ij}}) \propto \boldsymbol{\mu}_{\mathrm{i}}\boldsymbol{\nu}_{\mathrm{j}}$ and $p(E_{\mathrm{ij}}) = p_f$. The latent pattern matrix is generated by $Z = X \otimes Y$, where $X \in \{0,1\}^{m\times k}, Y \in \{0,1\}^{k\times n}$, and entries in X, Y also follow Bernoulli distributions with success probabilities $p(X_{il}) = p_X$ and $p(Y_{\mathrm{lj}}) = p_Y$. To comprehensively evaluate the methods, we generate varied data scenarios by considering different pattern numbers ($k \in \{3,4,5\}$), and flipping error ($p_f \in \{0, 0.05\}$). We also use different levels of p_X, p_Y to simulate pattern matrices of different density levels, where low density has $p_X = p_Y = 0.2$ while high density has $p_X = p_Y = 0.4$. The bias level is controlled by $\boldsymbol{\mu}, \boldsymbol{\nu}$. In case of low bias, we sample every $\boldsymbol{\mu}_{\mathrm{i}}$, $\boldsymbol{\nu}_{\mathrm{j}}$ uniformly from $[0.1, 0.8]$, which yields a overall bias level of $p_{\bar{B}_{\mathrm{ij}}} \sim 0.2$. For the high bias case, $\boldsymbol{\mu}_{\mathrm{i}}$, $\boldsymbol{\nu}_{\mathrm{j}}$ is sampled from $[0.3, 0.9]$ that

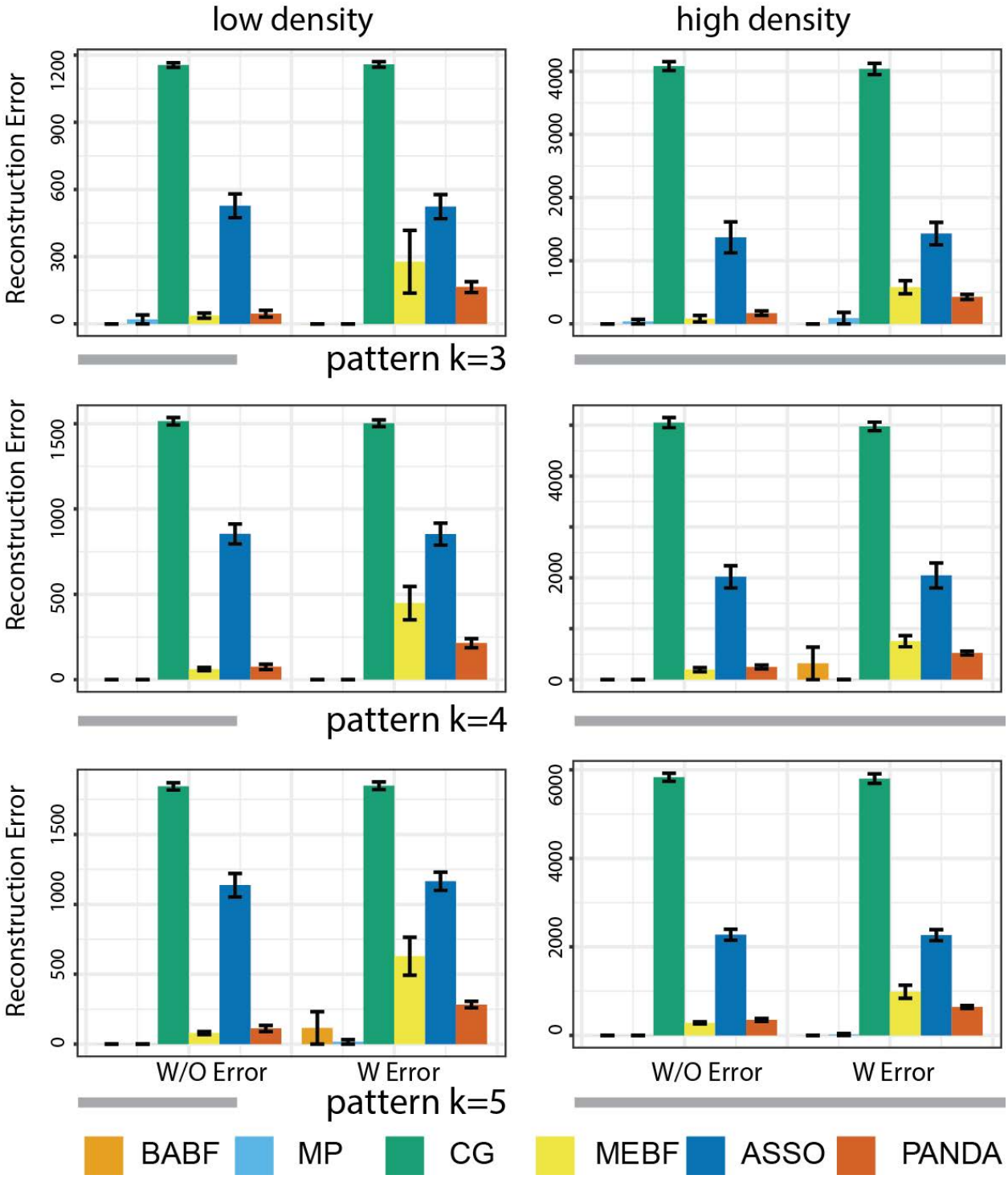

Figure 7.5. Performance comparison on simulated data without individual bias

result a bias probabilities of each variable at $p^{-}_{B_{ij}} \sim 0.36$. Altogether, we resulted in 24 data scenarios. For each scenarios, we set $m = n = 100$ and simulate 20 replicates.

Performance on reconstruction error

We report the benchmark results in Figure 7.3. We utilize default setting of the benchmarking methods in our analysis. As for BABF, we assume the prior of X, Y as Bernoulli distribution with $p_X = p_Y = 0.5$ and a flipping error of $p_f = 0.01$. The maximum iterations of t_{all}, t_{BI}, t_{MF} are set at 20, 5 and 50.

For each method, we compare their performance using reconstruction_error, i.e, $\|\hat{Z} - Z\|$ as evaluation metric. Here, $\|\cdot\|$ represents the L_1 norm, and $\hat{Z}$ denotes the derived pattern matrix by each method. Lower reconstruction error indicates a better performance. It is anticipated that heuristic approaches like CG, MEBF, ASSO, PANDA would show varied performances respect to different data scenarios as different bias level would result in different impact on their underlying heuristic assumptions. Probabilistic method MP showed an overall stable performance but still struggles with high bias level. As expected, BABF achieves the most desirable performance with different bias levels, which highlight the importance to consider individual bias. Additionally, BABF revealed its robustness towards different data scenarios.

Evaluate inferred bias

We explore whether BABF could reliably recover the bias levels $\boldsymbol{\mu}, \boldsymbol{\nu}$. Here, we denote BABF inferred row- and column-wise bias as $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\nu}}$. Since it is easy to find a scalar value r, s.t., $\boldsymbol{\mu}_i \cdot \boldsymbol{\nu}_j = r\hat{\boldsymbol{\mu}}_i \cdot \frac{1}{r}\hat{\boldsymbol{\nu}}_j$, we do not seek to directly compare the difference of values between $\boldsymbol{\mu}_i, \hat{\boldsymbol{\mu}}_i$, or $\boldsymbol{\nu}_j, \hat{\boldsymbol{\nu}}_j$, but instead analyze the correlation between the inferred bias and true bias for every input matrix A, i.e., $corr(\boldsymbol{\mu}, \hat{\boldsymbol{\mu}}), corr(\boldsymbol{\nu}, \hat{\boldsymbol{\nu}})$. We report the correlation results across different data scenarios with pattern number ($k = 4$) in Figure 7.4. Every scenario has 20 replications. Figure 7.4A,B show the row- and column-wise bias across different data scenarios. In most cases, BABF inferred bias achieved over 0.8 correlation with ground truth. Even in the worst case, the correlation is as high as 0.4. To give a more intuitive idea,

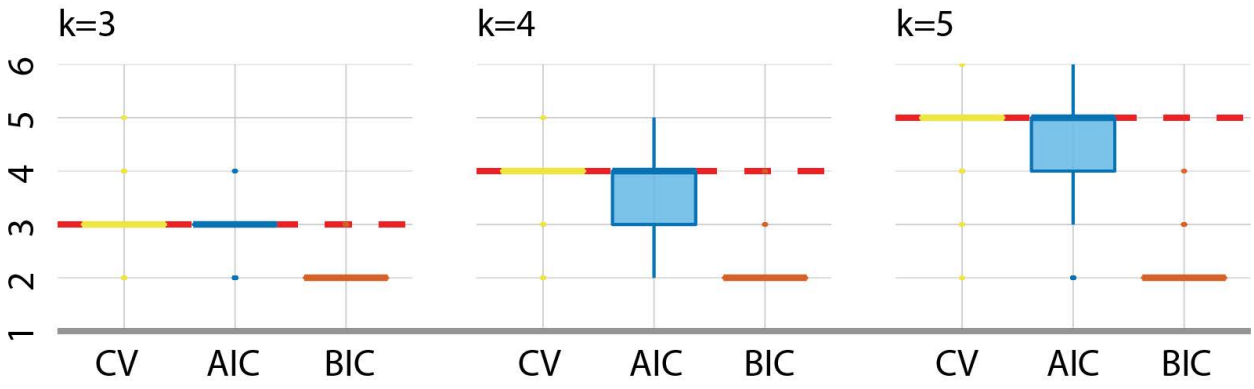

Figure 7.6. Model selection of pattern number k

we reveal the inferred bias and true bias of the first input matrix from each scenario as an example in Figure 7.4C,D. The high correlation suggests desirable performance of BABF to infer the individual bias associated with the objects and the features.

Performance on data without bias

Next, we wish to test how BABF performs on data without bias ($\boldsymbol{\mu}_{\mathrm{i}} = \boldsymbol{\nu}_{\mathrm{j}} = 0, \forall \mathrm{i}, \mathrm{j}$). In other words, we want to see if BABF is only suitable to model biased data scenarios. In Figure 7.3, we report the reconstruction error of the methods across 12 data scenarios all without background bias. In general, BABF and MP showed reliable performance. In some high density cases, BABF performs slightly worse than MP, but the difference is only marginal. Overall, BABF showed robust performance towards different data scenarios.

Selection of the pattern number

In our setting, the pattern number k is the most important hyper-parameter that directly determines the number of variables in the factor graph. Under the probabilistic framework, we could utilize different statistic metric to select the most optimal pattern numbers. Here we test three metrics, including cross validation (CV), Akaike information criterion (AIC) and Bayesian information criterion (BIC) for the model selection of k. For CV, we use 90% of the data for fitting and the rest 10% for testing [150]. For AIC and BIC, we utilize the formulation in [151]. For all the methods, we evaluate the metrics on $k = \{2, ..., 6\}$ and

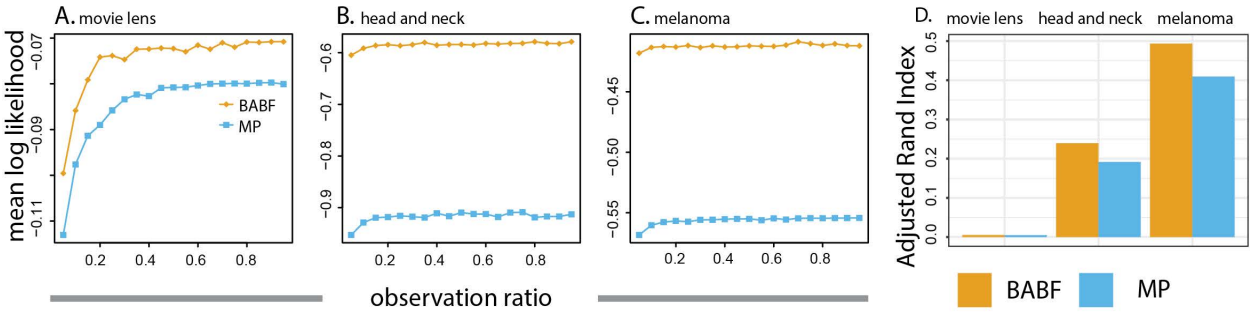

Figure 7.7. Goodness of fitting on the decomposition for real-world data

select the best k following their formulation. We tested above metrics across all 24 data scenarios and report the pattern number selections results in box plots (Figure 7.6). Here red dash marked the ground truth of k. Overall, CV showed consistently accurate selection of pattern number, with only marginal derivations for a small number of cases. AIC and BIC are impacted by the size of input data given a rather large number of variables in the model. Particular, BIC tends to always identify the smallest k for the model.

7.5.3 Analysis on real-world data

We tested the performance of BABF on three real world datasets, movie lens data from [152] and two biological gene expression datasets, head and neck cancer and melanoma single cell RNAseq (scRNA-seq) data from [21], [24]. The choice of the dataset as well as the pre-processing procedure follows previous works [115], [153]. In movie lens data we have 943 users, that rated/not rated 1682 films. In head and neck, and melanoma data, we have 5902 cells that express/not express 7954 genes, and 4486 cells that express/not express 8210 genes. For each dataset, we first identify the number of patterns $k \in \{2, 20\}$ through cross validation, which yield 5 patterns in movie lens, 3 patterns in melanoma and 6 patterns in head and neck. BABF is then applied to retrieve $\hat{X}, \hat{Y}$, $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\nu}}$ following the specific pattern number for each dataset. We mainly focus on addressing two questions: 1. Would the consideration of individual bias benefit our interpretation of real world data? 2. Does the inferred bias carry any practical meaning?

Data interpretation

Since the underlying true patterns of real world data is not accessible, instead of comparing decomposed pattern $\hat{Z}$ with input matrix A, where A is constituted of not only the true pattern matrix Z but also likely noise matrix E and bias matrix B, we use a likelihood metric. Specifically, we evaluate the goodness of model fitting as the overall likelihood, where a larger likelihood indicating a better fitting of the data. We compare the likelihood of BABF with the probabilistic BMF method MP. Similar to [114], [115], we investigate the methods' performance by only keeping a certain percentage of the observations, called observation level, while masking the rest of the observations. At every observation level, we replicate the analysis for five times and report the mean log-likelihood value. We report the likelihood results in Figure 7.7A,B,C. On all three datasets, BABF showed higher overall likelihood compared with MP, which suggests that the individual bias assumption is more realistic for real-world data, that movie viewers or cells could be vastly different from each other even in the same pattern group, and such bias is independent with the latent pattern. This advocates the necessity to consider the individual bias in the BMF problem.

To further test the interpretability of the patterns, we examined how the patterns coincide with cell type labels in the two expression datasets, and the movie genres in the movielens dataset, using adjusted rand index, where a higher value corresponds to a greater similarity [154]. Figure 7.7D shows the peformance of BABF and MP on three datasets. Though both BABF and MP perform poorly on movie lens data, the decomposition from BABF showed higher similarity with given labels in both biological data, which partially revealed a better decomposition of BABF compared with MP.

Practical meaning of inferred bias

The individual bias assumption allows BABF to outperform or have comparable performance with the existing BMF methods, whether such bias is present or not. Here, we want to understand whether the inferred bias could reflect certain practical meaning. Inspired by [143], in movie lens data, we want to explore the inferred bias on individual user with their taste on movie types. In our hypothesis, if a user only focus on certain genres of movie,

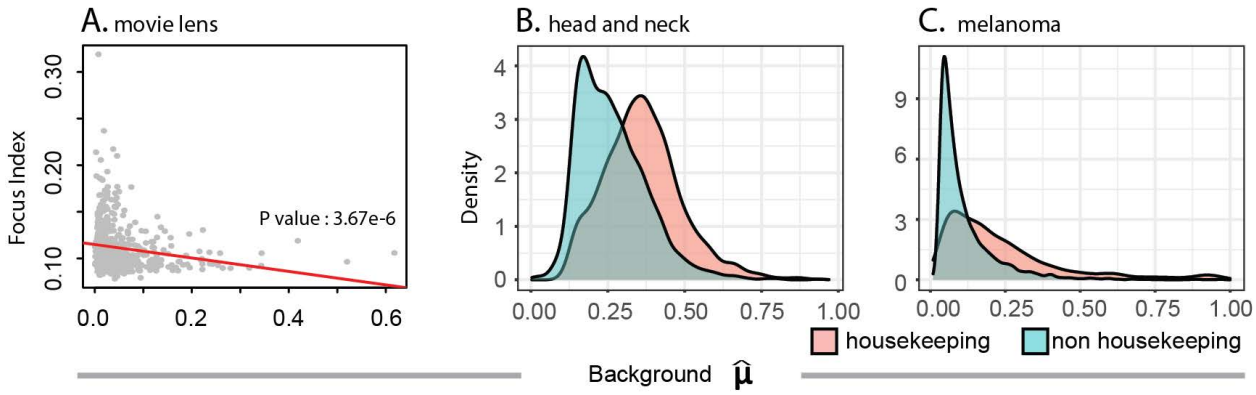

Figure 7.8. Interpretation analysis on inferred bias

then their behavior could be majorly explained by pattern information Z, while with less effect from B. Here we design the *focus index* to quantify such effect. Specifically if a user watched a numbers of movies in c categories, i.e., $a = b_1 + ... + b_c$, the focus index of this user is calculated as $focus\_index := \sum_{i=1}^{c}(\frac{b_i}{a})^2$. As anticipated, inferred bias is negatively correlated with the focus index (Figure 7.8A, $corr = -0.19,\ p = 3.67e-6$). This significant negative correlation revealed that the inferred bias partially revealed certain taste of the movie viewers.

In the case of gene expression, we focus on two group of genes, housekeeping genes and non-housekeeping genes [155]. As the name revealed, housekeeping genes are to maintain the basic activities of the cells, that each cell, regardless of their cell types, will all express these genes. On the other hand, non-housekeeping genes will be the ones that reflect the cell-type specificity. For example, T cells will express T cell marker genes like CD3D,CD3E [156], [157]. Figure 7.8B,C are the density plot visualization of the inferred bias on housekeeping and non-housekeeping genes. As expected, housekeeping genes have a much bigger effect from bias as their expression behavior is not related with any patterns. On the other hand, since the non-housekeeping genes revealed the specificity of the cell, its behavior is largely covered by the latent pattern, such that we witness a small bias in $\hat{\boldsymbol{\mu}}$ on both datasets.

7.6 Discussion

In this paper, we propose a bias aware model, BABF, which is the first algorithm to derive Boolean matrix decomposition in the presence of individual object- and feature-wise bias. Compared with other methods, BABF is a highly efficient approach, which not only results in good approximation of the true binary pattern with low reconstruction error, but also infers individual bias with high consistency with ground truth. The bias inference from BABF could lead to interesting interpretations depending on different data scenarios.

8. CONCLUSION AND BEYOND

In this research, we aim to reveal the discrete signals that regulate gene expression. Specifically, we focus on two types of data, 3C and scRNA-seq. For 3C, we try to regain the full elucidation of genome arrangement, which we regard as the very first step to understand gene expression regulation. Owing to the limited experimental resolution, we applied a novel hypergraph neural network to predict the unseen interactions in the genome, where our top predictions reveals potential mechanism of gene co-expression by sharing the same enhancer element. Next we shift our focus to identify gene expression state from scRNA-seq data. We proposed a novel statistical model LTMG that could identify the expression state but in the meantime mitigate the impact of data noise. LTMG is with better goodness of fitting compared with other methods. Moreover, the identified expression state has better reflection of cell type property. To utilize the discretized gene expression state to characterize the cell type functionalities, we referenced a machine learning approach, namely, Boolean matrix/tensor factorization. To alleviate its NP-hard complexity, we proposed the fast and efficient BMF and BTD methods, MEBF and GETF that revealed the functional linkage between a set of genes with a distinct cell type. We then take a step further to evaluate the row- and column-wise bias in binary data. And proposed the first bias aware BMF method. In doing so, we wish to eliminate the bias impact either from experiments or intrinsically from the data, and generate the patterns that of better biological meanings.

As revealed in Figure 1.1, gene expression is a complicated process that have different levels of regulations. In this research, we mainly focus on two levels, genome arrangement and transcription. We proposed a series of methods that aims to reveals the gene regulation utilizing limited data resources. However, they only characterize partial information at current stage. There is still a long way to connect every dots and achieve the full understanding of gene expression. In the future, we could focus different levels of data, like ATAC-seq, Chip-seq, etc. Moreover, we could reference causal learning approaches to identify the true causes of the regulation at different expression level. In the meantime, we could further embrace the technological development like spatial transcripts, multi-omics sequencing to integrate information from different dimensions.

REFERENCES

[1] M. Pertea, A. Shumate, G. Pertea, A. Varabyou, F. P. Breitwieser, Y.-C. Chang, A. K. Madugundu, A. Pandey, and S. L. Salzberg, "Chess: A new human gene catalog curated from thousands of large-scale rna sequencing experiments reveals extensive transcriptional noise," *Genome biology*, vol. 19, no. 1, pp. 1–14, 2018.

[2] E. Bianconi, A. Piovesan, F. Facchin, A. Beraudi, R. Casadei, F. Frabetti, L. Vitale, M. C. Pelleri, S. Tassani, F. Piva, *et al.*, "An estimation of the number of cells in the human body," *Annals of human biology*, vol. 40, no. 6, pp. 463–471, 2013.

[3] C. Buccitelli and M. Selbach, "Mrnas, proteins and the emerging principles of gene expression control," *Nature Reviews Genetics*, vol. 21, no. 10, pp. 630–644, 2020.

[4] F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, no. 5258, pp. 561–563, 1970.

[5] P. Cramer, "Organization and regulation of gene transcription," *Nature*, vol. 573, no. 7772, pp. 45–54, 2019.

[6] S. Hocine, R. H. Singer, and D. Grünwald, "Rna processing and export," *Cold Spring Harbor perspectives in biology*, vol. 2, no. 12, a000752, 2010.

[7] E. Li, "Chromatin modification and epigenetic reprogramming in mammalian development," *Nature Reviews Genetics*, vol. 3, no. 9, pp. 662–673, 2002.

[8] H. Cedar and Y. Bergman, "Linking dna methylation and histone modification: Patterns and paradigms," *Nature Reviews Genetics*, vol. 10, no. 5, pp. 295–304, 2009.

[9] V. W. Zhou, A. Goren, and B. E. Bernstein, "Charting histone modifications and the functional organization of mammalian genomes," *Nature Reviews Genetics*, vol. 12, no. 1, pp. 7–18, 2011.

[10] W. S. Dynan and R. Tjian, "The promoter-specific transcription factor sp1 binds to upstream sequences in the sv40 early promoter," *Cell*, vol. 35, no. 1, pp. 79–87, 1983.

[11] Y. W. Fong, C. Cattoglio, T. Yamaguchi, and R. Tjian, "Transcriptional regulation by coactivators in embryonic stem cells," *Trends in cell biology*, vol. 22, no. 6, pp. 292–298, 2012.

[12] N. Sonenberg and A. G. Hinnebusch, "Regulation of translation initiation in eukaryotes: Mechanisms and biological targets," *Cell*, vol. 136, no. 4, pp. 731–745, 2009.

[13] A. J. Larsson, P. Johnsson, M. Hagemann-Jensen, L. Hartmanis, O. R. Faridani, B. Reinius, Å. Segerstolpe, C. M. Rivera, B. Ren, and R. Sandberg, "Genomic encoding of transcriptional burst kinetics," *Nature*, vol. 565, no. 7738, pp. 251–254, 2019.

[14] T. Stuart and R. Satija, "Integrative single-cell analysis," *Nature reviews genetics*, vol. 20, no. 5, pp. 257–272, 2019.

[15] L. Fiorillo, F. Musella, M. Conte, R. Kempfer, A. M. Chiariello, S. Bianco, A. Kukalev, I. Irastorza-Azcarate, A. Esposito, A. Abraham, *et al.*, "Comparison of the hi-c, gam and sprite methods using polymer models of chromatin," *Nature methods*, vol. 18, no. 5, pp. 482–490, 2021.

[16] E. Shapiro, T. Biezuner, and S. Linnarsson, "Single-cell sequencing-based technologies will revolutionize whole-organism science," *Nature Reviews Genetics*, vol. 14, no. 9, pp. 618–630, 2013.

[17] R. Kempfer and A. Pombo, "Methods for mapping 3d chromosome architecture," *Nature Reviews Genetics*, vol. 21, no. 4, pp. 207–226, 2020.

[18] S. A. Quinodoz, N. Ollikainen, B. Tabak, A. Palla, J. M. Schmidt, E. Detmar, M. M. Lai, A. A. Shishkin, P. Bhat, Y. Takei, *et al.*, "Higher-order inter-chromosomal hubs shape 3d genome organization in the nucleus," *Cell*, vol. 174, no. 3, pp. 744–757, 2018.

[19] R. A. Beagrie, A. Scialdone, M. Schueler, D. C. Kraemer, M. Chotalia, S. Q. Xie, M. Barbieri, I. de Santiago, L.-M. Lavitas, M. R. Branco, *et al.*, "Complex multi-enhancer contacts captured by genome architecture mapping," *Nature*, vol. 543, no. 7646, pp. 519–524, 2017.

[20] J.-H. Su, P. Zheng, S. S. Kinrot, B. Bintu, and X. Zhuang, "Genome-scale imaging of the 3d organization and transcriptional activity of chromatin," *Cell*, vol. 182, no. 6, pp. 1641–1659, 2020.

[21] I. Tirosh, B. Izar, S. M. Prakadan, M. H. Wadsworth, D. Treacy, J. J. Trombetta, A. Rotem, C. Rodman, C. Lian, G. Murphy, *et al.*, "Dissecting the multicellular ecosystem of metastatic melanoma by single-cell rna-seq," *Science*, vol. 352, no. 6282, pp. 189–196, 2016.

[22] A. A. Kolodziejczyk, J. K. Kim, V. Svensson, J. C. Marioni, and S. A. Teichmann, "The technology and biology of single-cell rna sequencing," *Molecular cell*, vol. 58, no. 4, pp. 610–620, 2015.

[23] J. H. Levine, E. F. Simonds, S. C. Bendall, K. L. Davis, D. A. El-ad, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, *et al.*, "Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis," *Cell*, vol. 162, no. 1, pp. 184–197, 2015.

[24] S. V. Puram, I. Tirosh, A. S. Parikh, A. P. Patel, K. Yizhak, S. Gillespie, C. Rodman, C. L. Luo, E. A. Mroz, K. S. Emerick, *et al.*, "Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer," *Cell*, vol. 171, no. 7, pp. 1611–1624, 2017.

[25] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, "The discrete basis problem," *IEEE transactions on knowledge and data engineering*, vol. 20, no. 10, pp. 1348–1362, 2008.

[26] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in neural information processing systems*, vol. 19, pp. 1601–1608, 2006.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*, PMLR, 2017, pp. 1263–1272.

[28] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, *et al.*, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, pp. 688–702, 2020.

[29] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, 2007.

[30] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019, pp. 417–426.

[31] C.-A. Yu, C.-L. Tai, T.-S. Chan, and Y.-H. Yang, "Modeling multi-way relations with hypergraph embedding," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1707–1710.

[32] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *AAAI*, 2019.

[33] B. Fatemi, P. Taslakian, D. Vazquez, and D. Poole, "Knowledge hypergraphs: Prediction beyond binary relations," *arXiv preprint arXiv:1906.00137*, 2019.

[34] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, p. 107 637, 2021.

[35] B. Srinivasan, D. Zheng, and G. Karypis, "Learning over families of sets-hypergraph representation learning for higher order tasks," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, SIAM, 2021, pp. 756–764.

[36] H. Sutherland and W. A. Bickmore, "Transcription factories: Gene expression in unions?" *Nature Reviews Genetics*, vol. 10, no. 7, pp. 457–466, 2009.

[37] M. Yu and B. Ren, "The three-dimensional organization of mammalian genomes," *Annual review of cell and developmental biology*, vol. 33, pp. 265–289, 2017.

[38] J. Jiménez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 573–584, 2020.

[39] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 729–732.

[40] M. Zhang, Z. Cui, S. Jiang, and Y. Chen, "Beyond link prediction: Predicting hyperlinks in adjacency space," in *AAAI*, 2018.

[41] C. Berge, *Hypergraphs: combinatorics of finite sets*. Elsevier, 1984, vol. 45.

[42] H. Nassar, A. R. Benson, and D. F. Gleich, "Pairwise link prediction," in *IEEE ASONAM*, 2019.

[43] S.-e. Yoon, H. Song, K. Shin, and Y. Yi, "How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction," in *Proceedings of The Web Conference*, 2020.

[44] N. Yadati, V. Nitin, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar, "Nhp: Neural hypergraph link prediction," in *KDD*, 2020.

[45] R. Zhang, Y. Zou, and J. Ma, "Hyper-sagnn: A self-attention based graph neural network for hypergraphs," in *ICLR*, 2019.

[46] U. S. Melo, R. Schöpflin, R. Acuna-Hidalgo, M. A. Mensah, B. Fischer-Zirnsak, M. Holtgrewe, M.-K. Klever, S. Türkmen, V. Heinrich, I. D. Pluym, *et al.*, "Hi-c identifies complex genomic rearrangements and tad-shuffling in developmental diseases," *The American Journal of Human Genetics*, vol. 106, no. 6, pp. 872–884, 2020.

[47] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *NeurIPS*, 2018.

[48] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[49] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.

[50] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, "Hypergraph learning with line expansion," *arXiv preprint arXiv:2005.04843*, 2020.

[51] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "Hypersage: Generalizing inductive representation learning on hypergraphs," *arXiv preprint arXiv:2010.04558*, 2020.

[52] Y. Dong, W. Sawin, and Y. Bengio, "Hnhn: Hypergraph networks with hyperedge neurons," *arXiv preprint arXiv:2006.12278*, 2020.

[53] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Labeling trick: A theory of using graph neural networks for multi-node representation learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[54] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding: Design provably more powerful neural networks for graph representation learning," in *NeurIPS*, 2020.

[55] B. Srinivasan and B. Ribeiro, "On the equivalence between positional node embeddings and structural graph representations," in *ICLR*, 2019.

[56] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," *arXiv preprint arXiv:1703.06114*, 2017.

[57] R. Tyshkevich and V. E. Zverovich, "Line hypergraphs," *Discrete Mathematics*, vol. 161, no. 1-3, pp. 265–283, 1996.

[58] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 3744–3753.

[59] J. R. Dixon, I. Jung, S. Selvaraj, Y. Shen, J. E. Antosiewicz-Bourget, A. Y. Lee, Z. Ye, A. Kim, N. Rajagopal, W. Xie, *et al.*, "Chromatin architecture reorganization during stem cell differentiation," *Nature*, vol. 518, no. 7539, pp. 331–336, 2015.

[60] D.-S. Lee, C. Luo, J. Zhou, S. Chandran, A. Rivkin, A. Bartlett, J. R. Nery, C. Fitzpatrick, C. O'Connor, J. R. Dixon, *et al.*, "Simultaneous profiling of 3d genome structure and dna methylation in single human cells," *Nature methods*, vol. 16, no. 10, pp. 999–1006, 2019.

[61] F. Tavares-Cadete, D. Norouzi, B. Dekker, Y. Liu, and J. Dekker, "Multi-contact 3c reveals that the human genome during interphase is largely not entangled," *Nature Structural & Molecular Biology*, vol. 27, no. 12, pp. 1105–1114, 2020.

[62] J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, and B. Ren, "Topological domains in mammalian genomes identified by analysis of chromatin interactions," *Nature*, vol. 485, no. 7398, pp. 376–380, 2012.

[63] J. Huang, C. Chen, F. Ye, W. Hu, and Z. Zheng, "Nonuniform hyper-network embedding with dual mechanism," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 3, pp. 1–18, 2020.

[64] K. Ding, J. Wang, J. Li, D. Li, and H. Liu, "Be more with less: Hypergraph attention networks for inductive text classification," *arXiv preprint arXiv:2011.00387*, 2020.

[65] J. Wang, K. Ding, Z. Zhu, and J. Caverlee, "Session-based recommendation with hypergraph attention networks," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, SIAM, 2021, pp. 82–90.

[66] E. Azizi, A. J. Carr, G. Plitas, A. E. Cornish, C. Konopacki, S. Prabhakaran, J. Nainys, K. Wu, V. Kiseliovas, M. Setty, *et al.*, "Single-cell map of diverse immune phenotypes in the breast tumor microenvironment," *Cell*, vol. 174, no. 5, pp. 1293–1308, 2018.

[67] G. Zheng, J. Terry, P. Belgrader, P. Ryvkin, Z. Bent, R. Wilson, S. Ziraldo, T. Wheeler, G. McDermott, J. Zhu, *et al.*, *Massively parallel digital transcriptional profiling of single cells. nat. commun. 8, 1–12*, 2017.

[68] G. Finak, A. McDavid, M. Yajima, J. Deng, V. Gersuk, A. K. Shalek, C. K. Slichter, H. W. Miller, M. J. McElrath, M. Prlic, *et al.*, "Mast: A flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell rna sequencing data," *Genome biology*, vol. 16, no. 1, pp. 1–13, 2015.

[69] T. N. Vu, Q. F. Wills, K. R. Kalari, N. Niu, L. Wang, M. Rantalainen, and Y. Pawitan, "Beta-poisson model for single-cell rna-seq data analyses," *Bioinformatics*, vol. 32, no. 14, pp. 2128–2135, 2016.

[70] W. Li and J. Li, *An accurate and robust imputation method scimpute for single-cell rna-seq data. nat commun 9 (1): 997*, 2018.

[71] S. Anders and W. Huber, *Differential expression analysis for sequence count data. nat. preced*, 2010.

[72] T. Wang, B. Li, C. E. Nelson, and S. Nabavi, "Comparative analysis of differential gene expression analysis tools for single-cell rna sequencing data," *BMC bioinformatics*, vol. 20, no. 1, pp. 1–16, 2019.

[73] P. V. Kharchenko, L. Silberstein, and D. T. Scadden, "Bayesian approach to single-cell differential expression analysis," *Nature methods*, vol. 11, no. 7, pp. 740–742, 2014.

[74] Z. Wu, Y. Zhang, M. L. Stitzel, and H. Wu, "Two-phase differential expression analysis for single cell rna-seq," *Bioinformatics*, vol. 34, no. 19, pp. 3340–3348, 2018.

[75] D. J. McCarthy, Y. Chen, and G. K. Smyth, "Differential expression analysis of multifactor rna-seq experiments with respect to biological variation," *Nucleic acids research*, vol. 40, no. 10, pp. 4288–4297, 2012.

[76] M. Love, W. Huber, and S. Anders, *Moderated estimation of fold change and dispersion for rna-seq data with deseq2 genome biology 15 (12), 550*, 2014.

[77] V. Y. Kiselev, K. Kirschner, M. T. Schaub, T. Andrews, A. Yiu, T. Chandra, K. N. Natarajan, W. Reik, M. Barahona, A. R. Green, *et al.*, "Sc3: Consensus clustering of single-cell rna-seq data," *Nature methods*, vol. 14, no. 5, pp. 483–486, 2017.

[78] S. Aibar, C. B. González-Blas, T. Moerman, V. A. Huynh-Thu, H. Imrichova, G. Hulselmans, F. Rambow, J.-C. Marine, P. Geurts, J. Aerts, *et al.*, "Scenic: Single-cell regulatory network inference and clustering," *Nature methods*, vol. 14, no. 11, pp. 1083–1086, 2017.

[79] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using umap," *Nature biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.

[80] B. Wang, J. Zhu, E. Pierson, D. Ramazzotti, and S. Batzoglou, "Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning," *Nature methods*, vol. 14, no. 4, pp. 414–416, 2017.

[81] Y. Zhang, J. Xie, J. Yang, A. Fennell, C. Zhang, and Q. Ma, "Qubic: A bioconductor package for qualitative biclustering analysis of gene co-expression data," *Bioinformatics*, vol. 33, no. 3, pp. 450–452, 2017.

[82] K. H. Chen, A. N. Boettiger, J. R. Moffitt, S. Wang, and X. Zhuang, "Spatially resolved, highly multiplexed rna profiling in single cells," *Science*, vol. 348, no. 6233, aaa6090, 2015.

[83] E. Torre, H. Dueck, S. Shaffer, J. Gospocic, R. Gupte, R. Bonasio, J. Kim, J. Murray, and A. Raj, "Rare cell detection by single-cell rna sequencing as guided by single-molecule rna fish," *Cell systems*, vol. 6, no. 2, pp. 171–179, 2018.

[84] S. Shah, E. Lubeck, W. Zhou, and L. Cai, "In situ transcription profiling of single cells reveals spatial organization of cells in the mouse hippocampus," *Neuron*, vol. 92, no. 2, pp. 342–357, 2016.

[85] G. A. Maston, S. K. Evans, and M. R. Green, "Transcriptional regulatory elements in the human genome," *Annu. Rev. Genomics Hum. Genet.*, vol. 7, pp. 29–59, 2006.

[86] T. I. Lee and R. A. Young, "Transcriptional regulation and its misregulation in disease," *Cell*, vol. 152, no. 6, pp. 1237–1251, 2013.

[87] A. Ay and D. N. Arnosti, "Mathematical modeling of gene expression: A guide for the perplexed biologist," *Critical reviews in biochemistry and molecular biology*, vol. 46, no. 2, pp. 137–151, 2011.

[88] R. Khanin, V. Vinciotti, V. Mersinias, C. Smith, and E. Wit, "Statistical reconstruction of transcription factor activity using michaelis–menten kinetics," *Biometrics*, vol. 63, no. 3, pp. 816–823, 2007.

[89] Z. Duren, X. Chen, R. Jiang, Y. Wang, and W. H. Wong, "Modeling gene regulation from paired expression and chromatin accessibility data," *Proceedings of the National Academy of Sciences*, vol. 114, no. 25, E4914–E4923, 2017.

[90] S. A. van Hijum, M. H. Medema, and O. P. Kuipers, "Mechanisms and evolution of control logic in prokaryotic transcriptional regulation," *Microbiology and Molecular Biology Reviews*, vol. 73, no. 3, pp. 481–509, 2009.

[91] M. A. H. Samee, B. Lim, N. Samper, H. Lu, C. A. Rushlow, G. Jiménez, S. Y. Shvartsman, and S. Sinha, "A systematic ensemble approach to thermodynamic modeling of gene expression from sequence data," *Cell Systems*, vol. 1, no. 6, pp. 396–407, 2015.

[92] R. D. Dar, B. S. Razooky, A. Singh, T. V. Trimeloni, J. M. McCollum, C. D. Cox, M. L. Simpson, and L. S. Weinberger, "Transcriptional burst frequency and burst size are equally modulated across the human genome," *Proceedings of the National Academy of Sciences*, vol. 109, no. 43, pp. 17 454–17 459, 2012.

[93] X. Zhang, T. Li, F. Liu, Y. Chen, J. Yao, Z. Li, Y. Huang, and J. Wang, "Comparative analysis of droplet-based ultra-high-throughput single-cell rna-seq systems," *Molecular cell*, vol. 73, no. 1, pp. 130–142, 2019.

[94] G. Marsaglia, W. W. Tsang, and J. Wang, "Evaluating kolmogorov's distribution," *Journal of statistical software*, vol. 8, pp. 1–4, 2003.

[95] C. Zheng, L. Zheng, J.-K. Yoo, H. Guo, Y. Zhang, X. Guo, B. Kang, R. Hu, J. Y. Huang, Q. Zhang, *et al.*, "Landscape of infiltrating t cells in liver cancer revealed by single-cell sequencing," *Cell*, vol. 169, no. 7, pp. 1342–1356, 2017.

[96] L. Zhang, X. Yu, L. Zheng, Y. Zhang, Y. Li, Q. Fang, R. Gao, B. Kang, Q. Zhang, J. Y. Huang, *et al.*, "Lineage tracking reveals dynamic relationships of t cells in colorectal cancer," *Nature*, vol. 564, no. 7735, pp. 268–272, 2018.

[97] X. Guo, Y. Zhang, L. Zheng, C. Zheng, J. Song, Q. Zhang, B. Kang, Z. Liu, L. Jin, R. Xing, *et al.*, "Global characterization of t cells in non-small-cell lung cancer by single-cell sequencing," *Nature medicine*, vol. 24, no. 7, pp. 978–985, 2018.

[98] B. Schwanhäusser, D. Busse, N. Li, G. Dittmar, J. Schuchhardt, J. Wolf, W. Chen, and M. Selbach, "Global quantification of mammalian gene expression control," *Nature*, vol. 473, no. 7347, pp. 337–342, 2011.

[99] M. Vanlandewijck, L. He, M. A. Mäe, J. Andrae, K. Ando, F. Del Gaudio, K. Nahar, T. Lebouvier, B. Laviña, L. Gouveia, *et al.*, "A molecular atlas of cell types and zonation in the brain vasculature," *Nature*, vol. 554, no. 7693, pp. 475–480, 2018.

[100] L. He, M. Vanlandewijck, M. Mae, J. Andrae, K. Ando, F. Del Gaudio, K. Nahar, T. Lebouvier, B. Lavina, L. Gouveia, *et al.*, *Single-cell rna sequencing of mouse brain and lung vascular and vessel-associated cell types. sci data. 2018; 5: 180160*, 2018.

[101] M. Barry and R. C. Bleackley, "Cytotoxic t lymphocytes: All roads lead to death," *Nature Reviews Immunology*, vol. 2, no. 6, pp. 401–409, 2002.

[102] Y. Guo, J. Chen, T. Zhao, and Z. Fan, "Granzyme k degrades the redox/dna repair enzyme ape1 to trigger oxidative stress of target cells leading to cytotoxicity," *Molecular immunology*, vol. 45, no. 8, pp. 2225–2235, 2008.

[103] E. J. Wherry, "T cell exhaustion," *Nature immunology*, vol. 12, no. 6, pp. 492–499, 2011.

[104] F. Kocayusufoglu, M. X. Hoang, and A. K. Singh, "Summarizing network processes with network-constrained boolean matrix factorization," in *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018, pp. 237–246.

[105] T. Balasubramaniam, R. Nayak, and C. Yuen, "People to people recommendation using coupled nonnegative boolean matrix factorization," 2018.

[106] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular value decomposition and principal component analysis," in *A practical approach to microarray data analysis*, Springer, 2003, pp. 91–109.

[107] S. D. Monson, N. J. Pullman, and R. Rees, "A survey of clique and biclique coverings and factorizations of (0, 1)-matrices," *Bull. Inst. Combin. Appl*, vol. 14, pp. 17–86, 1995.

[108] L. J. Stockmeyer, *The set basis problem is NP-complete*. IBM Thomas J. Watson Research Division, 1975.

[109] P. Miettinen and J. Vreeken, "Mdl4bmf: Minimum description length for boolean matrix factorization," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 4, p. 18, 2014.

[110] S. Karaev, P. Miettinen, and J. Vreeken, "Getting to know the unknown unknowns: Destructive-noise resistant boolean matrix factorization," in *Proceedings of the 2015 SIAM International Conference on Data Mining*, SIAM, 2015, pp. 325–333.

[111] C. Lucchese, S. Orlando, and R. Perego, "Mining top-k patterns from binary datasets in presence of noise," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, SIAM, 2010, pp. 165–176.

[112] C. Lucchese, S. Orlando, and R. Perego, "A unifying framework for mining approximate top-k binary patterns," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 2900–2913, 2013.

[113] M. Araujo, P. Ribeiro, and C. Faloutsos, "Faststep: Scalable boolean matrix decomposition," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2016, pp. 461–473.

[114] S. Ravanbakhsh, B. Póczos, and R. Greiner, "Boolean matrix factorization and noisy completion via message passing.," in *ICML*, 2016, pp. 945–954.

[115] T. Rukat, C. C. Holmes, M. K. Titsias, and C. Yau, "Bayesian boolean matrix factorisation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 2969–2978.

[116] E. Junttila *et al.*, "Patterns in permuted binary matrices," 2011.

[117] M. Oswald and G. Reinelt, "The simultaneous consecutive ones problem," *Theoretical Computer Science*, vol. 410, no. 21-23, pp. 1986–1992, 2009.

[118] M. Oswald, "Weighted consecutive ones problems," Ph.D. dissertation, 2003.

[119] R. Belohlavek, J. Outrata, and M. Trnecka, "Toward quality assessment of boolean matrix factorizations," *Information Sciences*, vol. 459, pp. 71–85, 2018.

[120] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[121] C. Wan, W. Chang, T. Zhao, M. Li, S. Cao, and C. Zhang, "Fast and efficient boolean matrix factorization by geometric segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[122] T. Rukat, C. Holmes, and C. Yau, "Probabilistic boolean tensor decomposition," in *International conference on machine learning*, 2018, pp. 4413–4422.

[123] V. Hore, A. Viñuela, A. Buil, J. Knight, M. I. McCarthy, K. Small, and J. Marchini, "Tensor decomposition for multiple-tissue gene expression experiments," *Nature genetics*, vol. 48, no. 9, p. 1094, 2016.

[124] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.

[125] X. Bi, A. Qu, X. Shen, *et al.*, "Multilayer tensor factorization with applications to recommender systems," *The Annals of Statistics*, vol. 46, no. 6B, pp. 3308–3333, 2018.

[126] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[127] P. Miettinen *et al.*, "Matrix decomposition methods for data mining: Computational complexity and algorithms," 2009.

[128] P. Miettinen, "Sparse boolean matrix factorizations," in *2010 IEEE International Conference on Data Mining*, IEEE, 2010, pp. 935–940.

[129] N. Park, S. Oh, and U. Kang, "Fast and scalable distributed boolean tensor factorization," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, IEEE, 2017, pp. 1071–1082.

[130] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[131] I. Leenen, I. Van Mechelen, P. De Boeck, and S. Rosenberg, "Indclas: A three-way hierarchical classes model," *Psychometrika*, vol. 64, no. 1, pp. 9–24, 1999.

[132] P. Miettinen, "Boolean tensor factorizations," in *2011 IEEE 11th International Conference on Data Mining*, IEEE, 2011, pp. 447–456.

[133] D. Erdos and P. Miettinen, "Walk'n'merge: A scalable algorithm for boolean tensor factorization," in *2013 IEEE 13th International Conference on Data Mining*, IEEE, 2013, pp. 1037–1042.

[134] P. L. Ståhl, F. Salmén, S. Vickovic, A. Lundmark, J. F. Navarro, J. Magnusson, S. Giacomello, M. Asp, J. O. Westholm, M. Huss, *et al.*, "Visualization and analysis of gene expression in tissue sections by spatial transcriptomics," *Science*, vol. 353, no. 6294, pp. 78–82, 2016.

[135] N. Wu, J. Phang, J. Park, Y. Shen, Z. Huang, M. Zorin, S. Jastrzebski, T. Fevry, J. Katsnelson, E. Kim, *et al.*, "Deep neural networks improve radiologists' performance in breast cancer screening," *IEEE transactions on medical imaging*, 2019.

[136] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[137] Z. Zhu, X. Hu, and J. Caverlee, "Fairness-aware tensor-based recommendation," in *Proceedings of the 27th ACM CIKM*, 2018, pp. 1153–1162.

[138] S. Yao and B. Huang, "Beyond parity: Fairness objectives for collaborative filtering," in *Advances in Neural Information Processing Systems*, 2017, pp. 2921–2930.

[139] S. Kaiser and F. Leisch, "A toolbox for bicluster analysis in r," 2008.

[140] P. Miettinen and S. Neumann, "Recent developments in boolean matrix factorization," *arXiv preprint arXiv:2012.03127*, 2020.

[141] S. Zhao, G. Pan, J. Tao, Z. Luo, S. Li, and Z. Wu, "Understanding smartphone users from installed app lists using boolean matrix factorization," *IEEE Transactions on Cybernetics*, 2020.

[142] L. Liang, K. Zhu, and S. Lu, "Bem: Mining coregulation patterns in transcriptomics via boolean matrix factorization," *Bioinformatics*, vol. 36, no. 13, pp. 4030–4037, 2020.

[143] C. Wan, W. Chang, T. Zhao, S. Cao, and C. Zhang, "Denoising individual bias for fairer binary submatrix detection," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2245–2248.

[144] P. Miettinen and J. Vreeken, "Model order selection for boolean matrix factorization," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 51–59.

[145] N. Tatti and P. Miettinen, "Boolean matrix factorization meets consecutive ones property," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, SIAM, 2019, pp. 729–737.

[146] R. Belohlavek and M. Trnecka, "From-below approximations in boolean matrix factorization: Geometry and new algorithm," *Journal of Computer and System Sciences*, vol. 81, no. 8, pp. 1678–1697, 2015.

[147] R. Belohlavek and M. Trnecka, "A new algorithm for boolean matrix factorization which admits overcovering," *Discrete Applied Mathematics*, vol. 249, pp. 36–52, 2018.

[148] R. Belohlavek, J. Outrata, and M. Trnecka, "Factorizing boolean matrices using formal concepts and iterative usage of essential entries," *Information Sciences*, vol. 489, pp. 37–49, 2019.

[149] R. A. Kovacs, O. Gunluk, and R. A. Hauser, "Binary matrix factorisation via column generation," *arXiv preprint arXiv:2011.04457*, 2020.

[150] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, Montreal, Canada, vol. 14, 1995, pp. 1137–1145.

[151] P. Stoica and Y. Selen, "Model-order selection: A review of information criterion rules," *IEEE Signal Processing Magazine*, vol. 21, no. 4, pp. 36–47, 2004.

[152] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[153] C. Wan, D. Jia, Y. Zhao, W. Chang, S. Cao, X. Wang, and C. Zhang, "A data denoising approach to optimize functional clustering of single cell rna-sequencing data," in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2020, pp. 217–222.

[154] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[155] E. Eisenberg and E. Y. Levanon, "Human housekeeping genes, revisited," *TRENDS in Genetics*, vol. 29, no. 10, pp. 569–574, 2013.

[156] M. E. Call, J. Pyrdol, M. Wiedmann, and K. W. Wucherpfennig, "The organizing principle in the formation of the t cell receptor-cd3 complex," *Cell*, vol. 111, no. 7, pp. 967–979, 2002.

[157] C. Wan, W. Chang, Y. Zhang, F. Shah, X. Lu, Y. Zang, A. Zhang, S. Cao, M. L. Fishel, Q. Ma, *et al.*, "Ltmg: A novel statistical modeling of transcriptional expression states in single-cell rna-seq data," *Nucleic acids research*, vol. 47, no. 18, e111–e111, 2019.