Probabilistic Streaming Tensor Decomposition with Side Information

Yimin Zheng
University of Utah

UUCS-19-007

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

27 November 2019

Abstract

Tensor decomposition is an essential tool to analyze high-order interactions in multiway data. While most tensor decomposition approaches are developed for static data, many real-world applications generate tensor elements in a streaming fashion. On the other hand, the side information, such as a variety of the features for the entities and interactions, are produced in the mean time, which can greatly relieve data sparsity and potentially help find factors of better quality. In this thesis, we develop a Bayesian streaming tensor decomposition algorithm that can incrementally update the latent factors with streaming tensor elements in an arbitrary order, and meanwhile integrate the side information to enhance the factor quality. Experiments on four real-world datasets show that our method can improve upon existing streaming decomposition methods that do not exploit side information, and obtain at least comparable prediction accuracy to the stare-of-the-art static tensor decomposition approaches.

PROBABILISTIC STREAMING TENSOR DECOMPOSITION WITH SIDE INFORMATION

by

Yimin Zheng

A Senior Thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Bachelor of Computer Science

School of Computing

The University of Utah

November 2019

Copyright © Yimin Zheng 2019
All Rights Reserved

ABSTRACT

Tensor decomposition is an essential tool to analyze high-order interactions in multiway data. While most tensor decomposition approaches are developed for static data, many real-world applications generate tensor elements in a streaming fashion. On the other hand, the side information, such as a variety of the features for the entities and interactions, are produced in the mean time, which can greatly relieve data sparsity and potentially help find factors of better quality. In this thesis, we develop a Bayesian streaming tensor decomposition algorithm that can incrementally update the latent factors with streaming tensor elements in an arbitrary order, and meanwhile integrate the side information to enhance the factor quality. Experiments on four real-world datasets show that our method can improve upon existing streaming decomposition methods that do not exploit side information, and obtain at least comparable prediction accuracy to the stare-of-the-art static tensor decomposition approaches.

CONTENTS

ABSTRACT . . . . . **ii**

CHAPTERS

1. INTRODUCTION **1**

1.1 Background and preliminaries 4
1.1.1 CANDECOMP/PARAFAC (CP) Decomposition 4
1.1.2 Streaming Variational Inference 7
1.1.3 Probabilistic Streaming Tensor Decomposition 8
1.1.4 Side Information 8
1.1.5 Preliminaries 8
1.2 POSTsi 10
1.2.1 Bayesian CP Decomposition with Side Information model 10
1.2.2 Probabilistic Streaming Tensor Decomposition with Side Information algorithm 11
1.2.3 Computational Cost 14
1.3 Related Work 16
1.4 Experiments 17
1.4.1 Data 17
1.4.2 Baseline Methods 18
1.4.3 Experimental Setup and Metric 19
1.4.4 Evaluation on arbitrary-order streaming 20
1.4.5 Evaluation on Multi-aspect Streaming 21
1.5 Conclusion 26

REFERENCES **27**

CHAPTER 1

INTRODUCTION

Tensors, also known as a multidimensional arrays, are commonly used to represent the interactions among multiple entities. For example, the movie rating process can be considered as three-way interactions between *users*, *movies* and *time*, and hence the data can be represented by a three-mode tensor *(user, motive, time)*. Tensor decomposition is a fundamental tool to analyze these interactions. It decomposes the entire tensor into a simple form in terms of latent factors [13], which are feature representations of the entities that participate in the interactions. With the factors, we can discover the structures hidden in the entities, such as communities and anomalies, and predict various quantities of interests, such as click-through-rates, and recommendation accuracy.

While a variety of excellent tensor decomposition algorithms have been proposed [1, 10, 17, 18, 23, 25, 26], they only work on static tensors. However, in many real-world applications, the tensor elements are produced in a streaming manner, and after being accessed once, they are not allowed to be visited anymore, e.g., Snapchat and Instagram. This brings in challenges for the traditional batch decomposition algorithms.

To address this problem, recently a few incremental decomposition approaches were proposed to adapt to the dynamic growth of tensors. For example, MAST [20] allows the tensors to expand in all the modes simultaneously. However, MAST put constraints on the order of the streamed tensor elements — the new elements must reside in the incremental part of the tensor, rather than belong to the previous tensor (i.e., missing entries). To handle streams in an arbitrary order, [6] proposed Probabilistic Streaming Tensor Decomposition (POST) that uses streaming variational Bayes framework [4] to update the posterior distribution of the latent factors after every a few batches. It is based on a probabilistic model and the entry values are conditional independent given the latent factors. The order of tensor elements do not change the data likelihood, and hence the

factorization does not count on the order.

On the other hand, the side information, such as various features of the interactions and profiles of the entities, are collected simultaneously with the tensor elements. These information are valuable, can enrich the sparse tensor data, and potentially improve the estimation of the latent factors. Despite the promising prospect of the side information, existing approaches have rarely incorporated the side information into the decomposition process, especially in the streaming setting. We argue that the side information can be particularly useful for streaming tensor data, because the data is even sparser and more incomplete. Furthermore, since after going through the data once, we cannot access them again to correct the potential learning bias, the side information can help guide the real-time updates of the latent factors and prevent the learning from saturating into inferior results.

To bridge the gap, in this paper, we propose Probabilistic Streaming Tensor Decomposition with Side Information (POSTsi), a Bayesian streaming tensor decomposition algorithm that can (1) integrate the side information to improve the quality of incremental decomposition, (2) handle tensor elements that stream in an arbitrary order, and (3) quantify uncertainty of the latent factors and the predictions in real time. Specifically, we introduce a linear model that combines the output of a Bayesian CP decomposition model and the side features to predict the observed tensor entry values. We then develop an efficient streaming variational Bayes algorithm to update the posterior distribution of the latent factors and side feature weights every time upon a new batch of tensor elements are received. Following the incremental version of Bayes' rule, our algorithm uses the current posterior distribution as the prior, and integrates the likelihood of new tensor elements to calculate the updated posterior, which is in turn served as the prior for the next updates.

For evaluation, we compared with the state-of-the-art streaming tensor decomposition approaches, including POST, MAST and MASTsi — a variant of MAST that incorporates the side information. We also compared with state-of-the-art static tensor decomposition algorithms. On one real-world datasets which is a set of streaming tensor entries from a fixed-size tensor, our extended approach outperforms its predecessor. On four real-world datasets with the dynamic mode extension setting, our approach in most cases outperforms the competing streaming decomposition approaches, and is better than or

comparable to the static decomposition approaches.

1.1 Background and preliminaries

In this section, we first introduce a few key concepts used to develop the POSTsi. Then we introduce our notations.

Here, let's first discuss what Probabilistic Streaming Tensor Decomposition with Side Information means term by term. 1.) A *tensor decomposition* algorithm aims to express a tensor as a collection of operations acting on simpler tensors. CANDECOMP/PARAFAC (CP) and Tucker are two classical and widely used model for tensor decomposition algorithm. POSTsi is a tensor decomposition algorithm which builds on the CP decomposition model. 2.) POSTsi is a *probabilistic* algorithm. We design and use a probabilistic version of the CP decomposition model, such that POSTsi can provides uncertainty quantification on latent factors and predictions. 3.) POSTsi is a *streaming* algorithm. Upon a new batch of tensor elements, POSTsi updates the posterior distribution of latent factors with the likelihood of new tensor elements. POSTsi updates the posterior distribution of latent factors upon a new batch of tensor elements. 4.) *Side information* is any information that does not belong to the streaming tensor data but includes useful information to learn. One major aspect of POSTsi is to improve the quality of incremental decomposition by exploiting the side information coupled with the tensor.

1.1.1 CANDECOMP/PARAFAC (CP) Decomposition

A CP Decomposition is one of the most widely used and one of the oldest tensor decomposition models [13]. An exact CP Decomposition, a.k.a. tensor rank decomposition, is a linear higher-order extension of a famous matrix decomposition, the singular value decomposition. A CP Decomposition is an approximated version of exact CP Decomposition. The idea of exact CP Decomposition was first proposed along with the concept of tensor decomposition by Hitchcock in 1927 [8, 9, 13]. Kolda and Bader's survey *Tensor Decomposition and Applications* [13] has a comprehensive introduction on CP Decomposition, including its history, properties, its algorithms and applications.

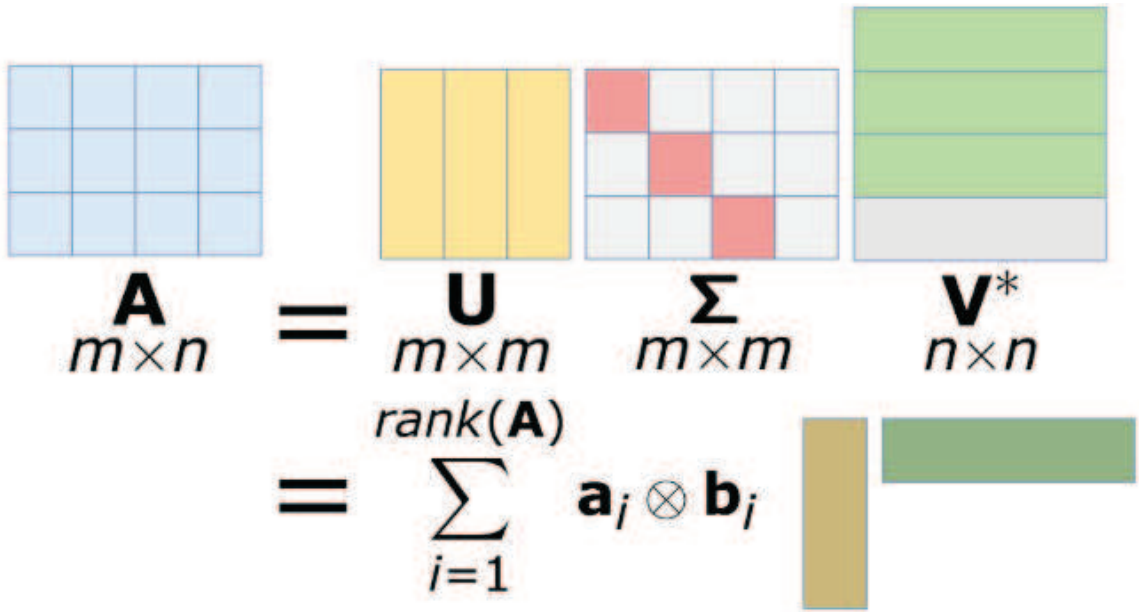

Figure 1.1: Singular Value Decomposition, A Matrix Decomposition

Assume $\underset{d_1 \times \cdots \times d_N}{\mathcal{A}}$ is an N-modes tensor, where d_i denotes the size of i-th mode of the tensor $\mathcal{A}$. With an exact CP decomposition algorithm, $\mathcal{A}$ can always be decomposed as the sum of R N-modes rank one tensor. Each N-modes rank one tensor can be further decomposed as an outer-product $\otimes$ of N vectors.

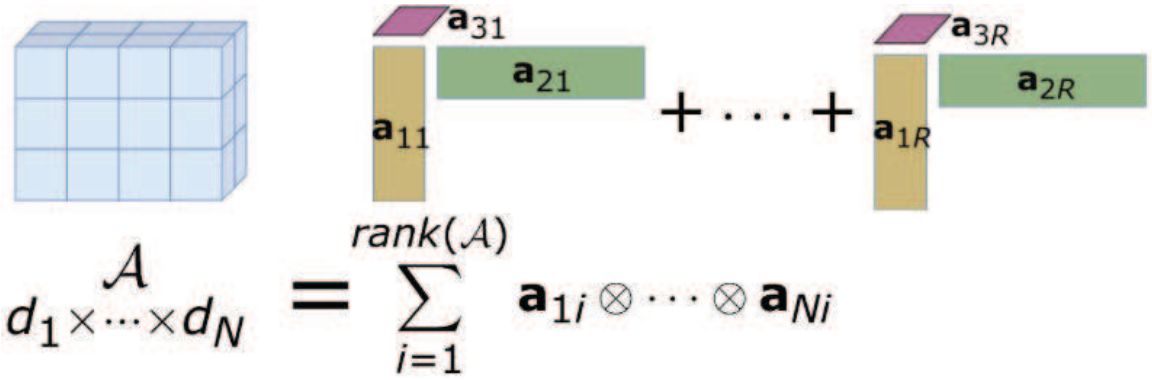

Figure 1.2: Exact CP Decomposition, a.k.a. tensor rank decomposition

However, finding the rank of a specific given tensor is an NP-hard problem. We have not found a straightforward algorithm to decide the rank of a specific given tensor nowadays. Thus, we also interested in CP Decomposition, an approximated version of exact CP Decomposition. The only differences between them is that we needs to manually supply a

parameter R to replace the rank of the given tensor. And the R is not necessarily equals to the rank of the given tensor. Thus, given an N-mode tensor $\underset{d_1 \times \cdots \times d_N}{\mathcal{A}}$ and a hyperparameter R, with CP Decomposition, we have the following form:

$$\mathcal{A} \approx \sum_{r=1}^{R} \mathbf{a}_{1i} \otimes \cdots \otimes \mathbf{a}_{Ni}$$

Here we call each decomposed vector $\mathbf{a}_{ni}$ a latent vector, or a latent factor, of $\mathcal{A}$. If we arrange all the latent vectors for the same mode of $\mathcal{A}$ together, we get a matrix which is called a factor matrix. Then, an element-wise CP Decomposition form can be written if we view each factor matrix as a collection of row vectors instead of a collection of column vectors, as shown in Figure 1.3.

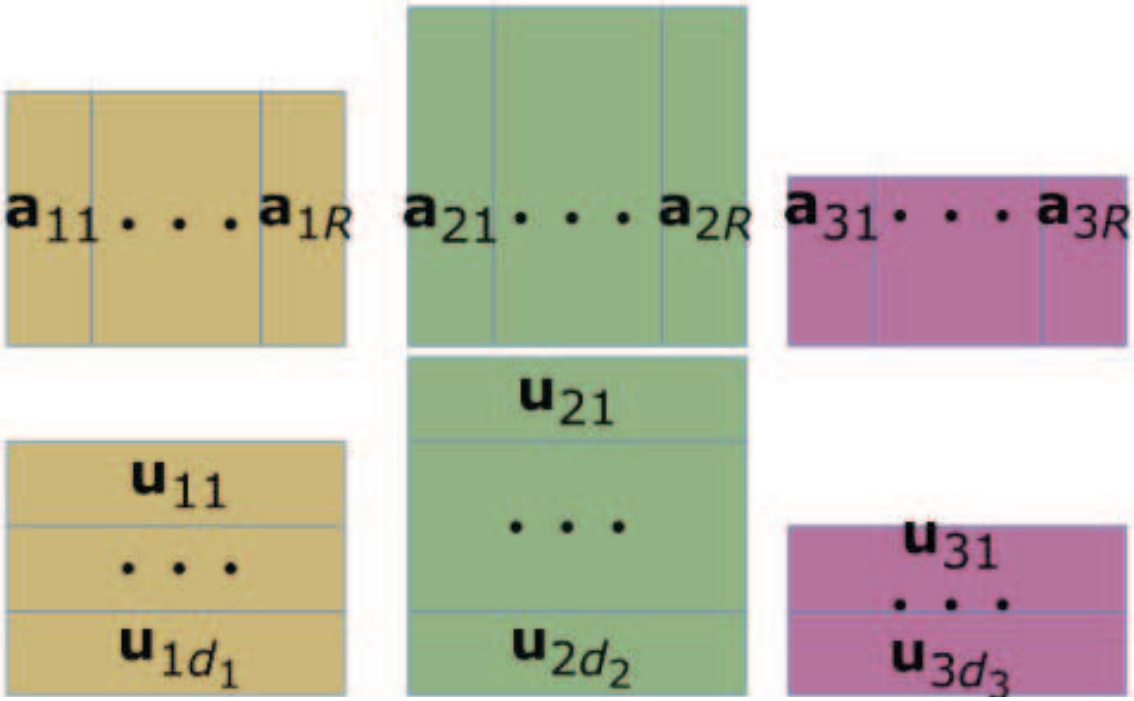

Figure 1.3: Factor matrix in two perspectives

Then, the following is a form of an element-wise CP Decomposition:

$$y_{\mathbf{i}} \approx \mathbf{1}^T (\mathbf{u}_{1i_1} \circ \cdots \circ \mathbf{u}_{Ni_N})$$

where $\mathbf{i} = [i_1, \cdots, i_N]$ denotes an index of a tensor entry of an N-mode tensor; the operator $\circ$ denotes the Hadamard product which performs element-wise multiplication.

1.1.2 Streaming Variational Inference

Given batches of data, the latent factor's and a linear classifier's prior, and the likelihood of latent factors and linear classifier, theoretically we can compute the exact posterior of the latent factors and linear classifier using Bayesian inference. However, we do not know the dependency among factors and linear classifier; and often the computation of the exact posterior with real-world prior models is intractable. So it is infeasible to learn the exact posterior of each latent factor and linear classifier with a straightforward Bayesian inference.

To overcome these problems, we can resort to Variational Inference, an approximate Bayesian inference method, to estimate the posterior distributions of latent factors and linear classifier using well-studied distributions. Furthermore, we use the streaming Variational Inference framework from [4] to do real-time estimation and updating on the posterior of each latent factors and linear classifier.

The main idea of Variational Inference is to approximate an intractable posterior $p(\theta|D)$ with a distribution $q(\theta)$ from a well-studied distribution family Q. It achieves this by minimising a KL divergence of the two distributions or equivalently maximising its evidence lower bound. Variational Inference turns the approximation problem into an optimisation problem. As for the streaming Variational Inference framework, it demonstrates a framework to allow real-time updates of the estimated posterior upon streaming data. Specifically, whenever a batch of streaming data arrives, we estimate the posterior of parameters using Variational Inference and use it as the prior parameters for processing the next batch of streaming data.

Variational Inference and streaming variational framework are essential concepts for our algorithm. Blei, Kucukelbir and McAuliffe's recent survey *Variational Inference: A Review for Statisticians* [3] provided an overview of the history, mathematical logic and research status of Variational Inference. Bishop's book *Pattern Recognition and Machine Learning* [2] also has a good overview of the mathematical logic and properties of Variational Inference. Broderick, Boyd, Wibisono, Wilson and Jordan's paper *Streaming Variational Bayes* [4] has a full description of the streaming variational inference framework.

1.1.3 Probabilistic Streaming Tensor Decomposition

Probabilistic Streaming Tensor Decomposition (POST) [6] is a recently published Probabilistic Streaming Dynamic Tensor Decomposition algorithm [27]. To the best of our knowledge, it is the first and only Bayesian inference algorithm that adapts the recently proposed multi-aspect streaming setting [20]. This paper first designs a probabilistic version of the CP decomposition model by providing prior to each latent parameters and likelihood of a single tensor entry. Then, it applies the Streaming Variational Inference on these priors and likelihood to achieve real-time update of latent parameters estimation. Since POST uses a Bayesian model, it naturally provides uncertainty quantification for all latent parameters.

1.1.4 Side Information

Side information is also known as auxiliary information. It is defined as "data that are neither from the input space nor from the output space of the function, but include useful information for learning it" [12]. In POSTsi, side information is data that does not belong to target tensor but is relevant to at least one mode of target tensor. Here is a concrete example. The MovieLens dataset [7] contains a three-mode (user, movie, week) target tensor describing "whether x user rated y movie in z week". Dataset also contains additional binary and categorical features describing the user's gender, occupation, age, movie's genre, etc. These extra features are considered as side information to POSTsi and can be exploited to improve the quality of tensor decomposition.

1.1.5 Preliminaries

In this paper, we use notations similar to ones in Probabilistic Streaming Tensor Decomposition [6]. Scalars (0-mode tensor) are denoted by lowercase or upper case letters (e.g., v or K). Vectors (1-mode tensor) are denoted by boldface lowercase (e.g., $\mathbf{i}$). Matrices (2-modes tensor) are denoted by boldface uppercase (e.g., $\mathbf{U}$). Higher-order tensors (3-modes tensor or higher) are denoted by calligraphic letters (e.g., $\mathcal{Y}$). The value of a tensor entry, of which

is an N-mode tensor, located at index $\mathbf{i} = [i_1, \cdots, 1_N]$ is denoted as $y_{\mathbf{i}}$

We assume that each tensor entry $y_{\mathbf{i}}$ is coupled with an external feature vector $\mathbf{j_i} = [j_1, \cdots, j_F]_{\mathbf{i}}$. These external feature vector is from the side information which we choose from the raw dataset. Index of a tensor entry, value of the tensor entry and the external feature $\mathbf{j_i}$ coupled to the tensor entry compose one data unit in a streaming batch. Then, we can denote one data unit as $\{[\mathbf{i}, \mathbf{j_i}, y_{\mathbf{i}}]\}$ and denote all $\{[\mathbf{i}, \mathbf{j_i}, y_{\mathbf{i}}]\}$ observed at time t as S_t. Latest observed entries at time instance t are denoted as $T = \{[\mathbf{i}, \mathbf{j_i}, y_{\mathbf{i}}]\}_{\mathbf{i} \in S_t}$. The value of all the observed tensor entries before time t are denoted as $D_t = \{[\mathbf{i}, \mathbf{j_i}, y_{\mathbf{i}}]\}_{\mathbf{i} \in S_1, \cdots, S_{t-1}}$. All observed entries are denoted as $D = \{[\mathbf{i}, \mathbf{j_i}, y_{\mathbf{i}}]\}_{\mathbf{i} \in S_1, \cdots, S_t} = T \cup D_t$

1.2 POSTsi

In this section, we present the Probabilistic Streaming Tensor Decomposition with Side Information algorithm (POSTsi). This extension enables the joint exploitation of side information and preserves uncertainty quantification on the learned linear classifier. In other words, POSTsi can 1.) utilize the side information which comes along with streaming tensor data to improve the quality of tensor decomposition and 2.) provide the uncertainty quantification of every latent factor and linear classifier.

Let $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a N-mode tensor. With CP Decomposition, each tensor entry $y_{\mathbf{i}}$ can be decompose into $y_{\mathbf{i}} \approx \boldsymbol{\lambda}_{\mathbf{i}}^T (\mathbf{u}_{\mathbf{i}_1}^1 \circ \cdots \circ \mathbf{u}_{\mathbf{i}_K}^K)$, a set of latent vectors and scalars. Then, a few recent tensor analysis works discover that side information could improve the quality of tensor decomposition. To exploit the potential of side information, we add a linear classifier component into the CP Decomposition model. Such that linear classifier can cooperation with side information. We call it CPDsi model and is defined as:

$$y_{\mathbf{i}} \approx \boldsymbol{\lambda}_{\mathbf{i}}^T (\mathbf{u}_{\mathbf{i}_1}^1 \circ \cdots \circ \mathbf{u}_{\mathbf{i}_K}^K) + \mathbf{w}^T \mathbf{j}_{\mathbf{i}}$$

In this model, the location of tensor entry $\mathbf{i}$, value of tensor entry $y_{\mathbf{i}}$ and side information j_i are provided as input data. Ultimately we want to develop an algorithm to learn the $\boldsymbol{\lambda}_{\mathbf{i}}, \mathbf{u}_{\mathbf{i}_k}^k$ and $\mathbf{w}$.

Yet, we can only learn point estimation of $\boldsymbol{\lambda}_{\mathbf{i}}, \mathbf{u}_{\mathbf{i}_k}^k$ and $\mathbf{w}$ with this model. Providing probabilistic distribution of the learned parameters are helpful in many real-world scenarios. For example, it can provide uncertainty quantification on the prediction of missing entries. So we further develop this model by converting it to a Bayesian model.

1.2.1 Bayesian CP Decomposition with Side Information model

To build the Bayesian version of the CP Decomposition with Side Information model, we design a Bayesian generative model. Assume that we generate latent vectors from a Gaussian prior $p(\mathcal{U}) = \prod_{k=1}^{K} \prod_{s=1}^{d_k} \mathcal{N}(\mathbf{u}_s^k | \mathbf{m}_s^k, v_1 \mathbf{I})$, the weight vector $\boldsymbol{\lambda}$ from a Multivariate Gaussian prior $p(\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda} | \boldsymbol{\mu}_\lambda, v_2 \mathbf{I})$, and weight scalars from a Gaussian prior $p(\mathbf{W}) = \prod_{f=1}^{F} \prod_{s=1}^{d_F} \mathcal{N}(w_s^f | \mu_s^f, v_3)$, where K denotes the number of modes the target tensor has, F denotes the number of features available as side information, d_f denotes the size of

f-th mode, d_F denotes the largest size of all side information features. Then, we define the likelihood of each observed tensor entry $y_{\mathbf{i}}$ given all the latent factors as a Probit likelihood $p(y_{\mathbf{i}}|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) = \Phi((2y_i - 1)(\boldsymbol{\lambda}^T t_{\mathbf{i}} + w_{\mathbf{j}}^T \mathbf{j}))$. We embed multiplier v_1, v_2 to control the variance/covariance of priors. The smaller the value of v_1, v_2, the more concentrate the samplings will be.

1.2.2 Probabilistic Streaming Tensor Decomposition with Side Information algorithm

Given the Bayesian CP Decomposition with Side Information model, we propose the Probabilistic Streaming Tensor Decomposition with Side Information (POSTsi) algorithm.

We use the streaming variational inference framework [4] and Mean Field Variational Inference [24] to achieve real-time update of posterior approximation upon each observed streaming batch S_t. In the Bayesian model described hereinabove, we used a Probit likelihood for processing binary tensor entry value. To derive a closed-form update, we want it to contain exponential family distribution. So we expand the likelihood by introducing a random variable $z_{\mathbf{i}}$ which has Gaussian distribution:

$$
\begin{aligned}
p(y_{\mathbf{i}}|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) &= \int_{z_{\mathbf{i}}} p(y_{\mathbf{i}}, z_{\mathbf{i}}|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) dz_{\mathbf{i}} \\
&= \int_{z_{\mathbf{i}}} p(z_{\mathbf{i}}|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}, \mathbf{D}) p(y_{\mathbf{i}}|z_{\mathbf{i}}, \mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}, \mathbf{D}) dz_{\mathbf{i}} \\
&= \int_{z_{\mathbf{i}}} \mathcal{N}(z_{\mathbf{i}}|\boldsymbol{\lambda}^T t_{\mathbf{i}} + w_{\mathbf{j}}^T \mathbf{j}), 1) p(y_{\mathbf{i}}|z_{\mathbf{i}}, \mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}, \mathbf{D}) dz_{\mathbf{i}} \\
&= \int_{z_{\mathbf{i}}} \mathcal{N}(z_{\mathbf{i}}|\boldsymbol{\lambda}^T t_{\mathbf{i}} + w_{\mathbf{j}}^T \mathbf{j}), 1) \mathbb{1}((2y_{\mathbf{i}} - 1) \geq 0) dz_{\mathbf{i}} \\
&= \Phi((2y_i - 1)(\boldsymbol{\lambda}^T t_{\mathbf{i}} + w_{\mathbf{j}}^T \mathbf{j}))
\end{aligned}
$$

Now we have a likelihood function in a new form that contains an exponential family distribution.

Then, we derive the joint probability of all latent parameters,

$$
\begin{aligned}
p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}, D) &= p(D|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) \cdot p(\mathcal{U}) \cdot p(\boldsymbol{\lambda}) \cdot p(\mathbf{W}) \\
&= \int_{z_\mathbf{i}} \mathcal{N}(z_\mathbf{i}|\boldsymbol{\lambda}^T t_\mathbf{i} + w_\mathbf{j}^T \mathbf{j}), 1) \mathbb{1}((2y_\mathbf{i} - 1) \geq 0) dz_\mathbf{i} \\
&\cdot (\prod_{k=1}^{K} \prod_{s=1}^{d_k} \mathcal{N}(\mathbf{u}_s^k|\mathbf{m}_s^k, v_1\mathbf{I})) \cdot \mathcal{N}(\boldsymbol{\lambda}|\boldsymbol{\mu}_\lambda, v_2\mathbf{I}) \\
&\cdot \prod_{f=1}^{F} \prod_{s=1}^{d_F} \mathcal{N}(w_s^f|\mu_s^f, v_3)
\end{aligned}
$$

On the other hand, we can further derive an update equation for posterior

$$
\begin{aligned}
p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}|D) &= \frac{p(D|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})}{p(D)} \\
&\propto \frac{p(D|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})}{p(D_t)} \\
&= p(T|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) \frac{p(D_t|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) p(\mathcal{U}) p(\boldsymbol{\lambda}) p(\mathbf{W})}{p(D_t)} \\
&= p(T|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}|D_t)
\end{aligned}
$$

This equation means that we can recursively update the optimized latent parameters upon each observed streaming batch T. However, our goal is to find the optimized posterior of each latent parameters. This recursive update only provides us with one joint posterior. Therefore, with Mean-Field Approximation, we approximate the joint posterior with a factorized posterior

$$
q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}) = (\prod_{k=1}^{K} \prod_{s=1}^{d_k} \mathcal{N} q(\mathbf{u}_s^k)) \cdot q(\boldsymbol{\lambda}) \cdot \prod_{f=1}^{F} \prod_{s=1}^{d_F} q(w_s^f)
$$

Now, we have a framework for POSTsi. To summarize, we first initialize factorized posterior $q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})$ with defined priors $p(\mathcal{U}), p(\boldsymbol{\lambda}), p(\mathbf{W})$. Next, upon each latest streaming batch T, use $q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})$ as $p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}|D_t)$ to calculate the optimized factorized posteriors $q^*(\mathcal{U}), q^*(\boldsymbol{\lambda}), q^*(\mathbf{W})$, or say $q^*(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})$. Then assign the result $q^*(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})$ back to $p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}|D_t, T)$ and wait for next batch of streaming data.

With this framework, we use variational inference [24] to calculate the optimized posterior $q^*(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})$ given latest streaming batch T and the previous posterior $p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}|D_t)$. Kullback Leibler (KL) divergence converts this inference problem into an optimization problem. The optimization problem is to find the posterior of each type of factorized latent

parameters, $q^*(\mathbf{u}_s^k), q^*(\boldsymbol{\lambda}), q^*(w_s^f)$, which together minimize the KL divergence between $p(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}|D)$ and $p(T|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})$, i.e.,

$$q^* = \underset{q \in Q}{\operatorname{argmin}}\, KL(q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})||p(T|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}))$$

However, the KL divergence

$$\begin{aligned} &KL(q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})||p(T|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})) \\ &\quad = -(\mathbb{E}_{q^*}[\log \frac{p(\{y_\mathbf{i}\}_{\mathbf{i} \in S_t}|\mathcal{U}, \boldsymbol{\lambda}, \mathcal{W})q(\mathcal{U}, \boldsymbol{\lambda}, \mathcal{W})}{q^*(\mathcal{U}, \boldsymbol{\lambda}, \mathcal{W})}]) + log p(\{y_\mathbf{i}\}_{\mathbf{i} \in S_t}) \\ &\quad = -(\mathcal{L}) + \log p(D) \end{aligned}$$

contains the model evidence $p(D)$ which in often time is very hard to compute. Luckily, we can convert the problem of minimizing the KL Divergence to a problem of maximizing the corresponding Evidence Lower Bound $\mathcal{L}$ [24]. Because the model evidence $p(D)$ is a constant, maximizing $KL(q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})||p(T|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W}))$ is equivalent to minimizing $\mathcal{L}$. The following is the expansion of Evidence Lower Bound $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q^*}[\log \frac{p(\{y_\mathbf{i}, z_\mathbf{i}\}_{\mathbf{i} \in S_t}|\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})q(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})}{q^*(\mathcal{U}, \boldsymbol{\lambda}, \mathbf{W})q(\mathbf{z})}]$$

To solve the minimization problem, we first take the functional derivative of $\mathcal{L}$ w.r.t. each factorized posterior $q^*(\mathbf{u}_s^k), q^*(\boldsymbol{\lambda}), q^*(w_s^f)$. Then set the derivative to zero and solve the equation. We obtain closed-form update functions for the posterior of each latent parameter,

$$q^*(\mathbf{u}_{\mathbf{i}_k}^k) = \mathcal{N}(\mathbf{u}_{\mathbf{i}_k}^k|{\boldsymbol{\mu}_{\mathbf{i}_k}^k}^*, {\boldsymbol{\Sigma}_{\mathbf{i}_k}^k}^*) \tag{1.1}$$

$$q^*(\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda}|{\boldsymbol{\mu}_\lambda}^*, \boldsymbol{\Sigma}_\lambda^*) \tag{1.2}$$

$$q^*(\mathbf{w}_{\mathbf{j}_f}^f) = \mathcal{N}(\mathbf{w}_{\mathbf{j}_f}^f|{\boldsymbol{\mu}_{\mathbf{j}_f}^f}^*, {\boldsymbol{\Sigma}_{\mathbf{j}_f}^f}^*) \tag{1.3}$$

and the closed-form calculation of $q(z_\mathbf{i})$ as a truncated Gaussian distribution,

$$q(z_\mathbf{i}) \propto \mathbb{1}((2y_\mathbf{i} - 1)z_\mathbf{i} \geq 0)\mathcal{N}(z_\mathbf{i}| < \boldsymbol{\lambda}^T \mathbf{t}_\mathbf{i} + w_\mathbf{i}^T \mathbf{j} >, 1)$$

The following is the supplement of the close-form update we get by solving the minimization problem:

$$
\begin{aligned}
\boldsymbol{\Sigma}_{\mathbf{i}_k}^{k*} &= [\boldsymbol{\Sigma}_{\mathbf{i}_k}^{k\ -1} + \sum_{\{\mathbf{g},\mathbf{j}\}\in S_t, \mathbf{g}_k=\mathbf{i}_k} \langle (\mathbf{t}_{\mathbf{g},\neg k} \circ \boldsymbol{\lambda})(\mathbf{t}_{\mathbf{g},\neg k} \circ \boldsymbol{\lambda})^T \rangle]^{-1} \\
\boldsymbol{\mu}_{\mathbf{i}_k}^{k*} &= \boldsymbol{\Sigma}_{\mathbf{i}_k}^{k*} [\boldsymbol{\Sigma}_{\mathbf{i}_k}^{k\ -1} \boldsymbol{\mu} + \sum_{\{\mathbf{g},\mathbf{j}\}\in S_t, \mathbf{g}_k=\mathbf{i}_k} (\langle z_{\mathbf{i}} \rangle - w_{\mathbf{j}}^T \mathbf{j}) \langle \mathbf{t}_{\mathbf{g},\neg k} \circ \boldsymbol{\lambda} \rangle] \\
\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}^{*} &= [\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}^{-1} + \sum_{\{\mathbf{i},\mathbf{j}\}\in S_t} \langle \mathbf{t}_{\mathbf{i}} \mathbf{t}_{\mathbf{i}}^T \rangle]^{-1} \\
\boldsymbol{\mu}_{\boldsymbol{\lambda}}^{*} &= \boldsymbol{\Sigma}_{\boldsymbol{\lambda}}^{*} [\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}^{-1} \boldsymbol{\mu}_{\boldsymbol{\lambda}} + \sum_{\{\mathbf{i},\mathbf{j}\}\in S_t} [\langle \mathbf{t}_{\mathbf{i}} \rangle (\langle z_{\mathbf{i}} \rangle - \langle w_{\mathbf{j}} \rangle^T \mathbf{j})]] \\
\Sigma_{\mathbf{j}_f}^{f*} &= [\Sigma_{\mathbf{j}_f}^{f\ -1} + \sum_{\{\mathbf{i},\mathbf{g}\}\in S_t, \mathbf{g}_f=\mathbf{j}_f} \langle \mathbf{g}_f \mathbf{g}_f^T \rangle]^{-1} \\
\mu_{\mathbf{j}_f}^{f*} &= \Sigma_{\mathbf{j}_f}^{f*} [\Sigma_{\mathbf{j}_f}^{f\ -1} \mu_{\mathbf{j}_f}^{f} \\
&\quad + \sum_{\{\mathbf{i},\mathbf{g}\}\in S_t, \mathbf{g}_f=\mathbf{j}_f} \mathbf{g}_f (\langle z_{\mathbf{i}} \rangle - \langle \boldsymbol{\lambda} \rangle^T \langle \mathbf{t}_{\mathbf{i}} \rangle - \langle \mathbf{w}_{\mathbf{g}_{\neg f}} \rangle^T \mathbf{g}_{\neg f})] \\
\langle z_{\mathbf{i}} \rangle &= \langle \boldsymbol{\lambda} \rangle^T \langle \mathbf{t}_{\mathbf{i}} \rangle + \langle w_{\mathbf{j}} \rangle^T \mathbf{j} + \frac{(2y_{\mathbf{i}} - 1)\phi(\langle \boldsymbol{\lambda} \rangle^T \langle \mathbf{t}_{\mathbf{i}} \rangle + \langle w_{\mathbf{j}} \rangle^T \mathbf{j})}{\Phi((2y_{\mathbf{i}} - 1)(\langle \boldsymbol{\lambda} \rangle^T \langle \mathbf{t}_{\mathbf{i}} \rangle + \langle w_{\mathbf{j}} \rangle^T \mathbf{j}))}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{t}_{\mathbf{i}} &= \mathbf{u}_{\mathbf{i}_1}^1 \circ \cdots \circ \mathbf{u}_{\mathbf{i}_K}^K \\
\mathbf{t}_{\mathbf{j},\neg k} &= \mathbf{u}_{\mathbf{j}_1}^1 \circ \cdots \circ \mathbf{u}_{\mathbf{j}_{k-1}}^{k-1} \circ \mathbf{u}_{\mathbf{j}_{k+1}}^{k+1} \circ \cdots \circ \mathbf{u}_{\mathbf{j}_K}^K \\
\langle w_{\mathbf{j}_{\neg f}} \rangle &= [\langle w_{\mathbf{j}_1} \rangle, \cdots, \langle w_{\mathbf{j}_{f-1}} \rangle, \langle w_{\mathbf{j}_{f+1}} \rangle, \cdots, \langle w_{\mathbf{j}_F} \rangle] \\
\langle \mathbf{t}_{\neg k} \circ \boldsymbol{\lambda} \rangle &= \boldsymbol{\mu}_{\mathbf{j}_1}^{1\ *} \circ \cdots \circ \boldsymbol{\mu}_{\mathbf{j}_{k-1}}^{k-1\ *} \circ \boldsymbol{\mu}_{\mathbf{j}_{k+1}}^{k+1\ *} \circ \cdots \circ \boldsymbol{\mu}_{\mathbf{j}_K}^{K\ *} \circ \boldsymbol{\mu}_{\boldsymbol{\lambda}}^{*} \\
\langle (\mathbf{t}_{\neg k} \circ \boldsymbol{\lambda})(\mathbf{t}_{\neg k} \circ \boldsymbol{\lambda})^T \rangle &= \langle \mathbf{u}_{\mathbf{j}_1}^1 \mathbf{u}_{\mathbf{j}_1}^{1\ T} \rangle \circ \cdots \circ \langle \mathbf{u}_{\mathbf{j}_{k-1}}^{k-1} \mathbf{u}_{\mathbf{j}_{k-1}}^{k-1\ T} \rangle \\
&\quad \circ \langle \mathbf{u}_{\mathbf{j}_{k+1}}^{k+1} \mathbf{u}_{\mathbf{j}_{k+1}}^{k+1\ T} \rangle \circ \cdots \circ \langle \mathbf{u}_{\mathbf{j}_K}^{K} \mathbf{u}_{\mathbf{j}_K}^{K\ T} \rangle \circ \langle \boldsymbol{\lambda} \boldsymbol{\lambda}^T \rangle \\
\langle \mathbf{u}_s^l \mathbf{u}_s^{l\ T} \rangle &= \boldsymbol{\Sigma}_s^{l\ *} + \boldsymbol{\mu}_s^{l\ *} \boldsymbol{\mu}_s^{l\ *T} \\
\langle \boldsymbol{\lambda} \boldsymbol{\lambda}^T \rangle &= \boldsymbol{\Sigma}_{\boldsymbol{\lambda}}{}^{*} + \boldsymbol{\mu}_{\boldsymbol{\lambda}}{}^{*} \boldsymbol{\mu}_{\boldsymbol{\lambda}}{}^{*T} \\
\langle \mathbf{t}_{\mathbf{i}} \mathbf{t}_{\mathbf{i}}^T \rangle &= \langle \mathbf{u}_{\mathbf{j}_1}^1 \mathbf{u}_{\mathbf{j}_1}^{1\ T} \rangle \circ \cdots \circ \langle \mathbf{u}_{\mathbf{j}_K}^K \mathbf{u}_{\mathbf{j}_K}^{K\ T} \rangle \\
\langle \mathbf{j}_f \mathbf{j}_f^T \rangle &= \mathbf{j}_f^2
\end{aligned}
$$

1.2.3 Computational Cost

Assume we do not perform any parallelization. Also assume that matrix inversion, multiplication, division cost $\mathcal{O}(n^3), \mathcal{O}(1), \mathcal{O}(1)$ correspondingly. Then, the time complexity to optimize all factorized posteriors, including $\{q^*(\mathbf{u}_{\mathbf{i}_k}^k)\}_{\{\mathbf{i},\mathbf{j}\}\in S_t, k\in[1,\cdots,K]}, q^*(\boldsymbol{\lambda}), q^*(w_{\mathbf{j}_f}^f)_{\{\mathbf{i},\mathbf{j}\}\in S_t, f\in[1,\cdots,F]}$,

$q^*(z_{\mathbf{i}})_{\{\mathbf{i},\mathbf{j}\}\in S_t}$ at time instance t is $\mathcal{O}(|S_t|KR^3 + |S_t|KR^2) + \mathcal{O}(|S_t|KR^3 + |S_t|K(F + KR)) + \mathcal{O}(R^3) + \mathcal{O}(R^3 + |S_t|F) + \mathcal{O}(|S_t|F) + \mathcal{O}(|S_t|F) + \mathcal{O}(|S_t|F) = \mathcal{O}(|S_t|(KR^3 + KF + K^R))$. On the other hand, the space complexity necessary to perform POSTsi is $\mathcal{O}((\sum_{k=1}^{K} \mathbf{d}_k R^2) + (\sum_{f=1}^{K} \mathbf{d}_F + |S_t|K))$, which includes the space for all latent parameters information and streaming data with side information D_t.

1.3 Related Work

In the past two decades, several tensor decomposition works have been proposed under the setting that a target tensor expands along one dimension, typically the time mode. For example, [21,22] proposed the Incremental Tensor Analysis (ITA), which is considered as pioneer works in the Dynamic Tensor Decomposition topic [16,27]. More recent, tensor decomposition under the setting that a target tensor can expands simultaneously along multiple dimensions (multi-aspect streaming) emerged. The Multi-aspect Streaming Tensor (MAST) algorithm proposed in [20] is a pioneer work on this more advance streaming setting. It utilizes existing latent factors of observed target tensor to estimate the latent factors of expanded target tensor. The Probabilistic Streaming Tensor Decomposition (POST) algorithm [6] uses a Bayesian approach to solve the multi-aspect streaming tensor decomposition problem, which enables quantification measurement on latent parameters and the prediction of latent factors. And because POST is built on element-wise CP Decomposition model, it can process arbitrary order streaming data, which is even more flexible than the Multi-aspect Streaming and might enable more applications. One the other hand, the Side Information infused Incremental Tensor Analysis (SIITA) [16] is proposed more recently. It is a new framework that developed under the multi-aspect streaming setting and incorporated side information to improve the prediction accuracy of missing tensor entries.

1.4 Experiments

In this section, we conduct experiments to evaluate the effectiveness of proposed POSTsi algorithm on recovering missing-tensor-entries. Specifically, we aim to answer the following questions:

- How does POSTsi's prediction accuracy compare with POST and TNCP under arbitrary-order streaming setting?
- How does POSTsi's prediction accuracy compare with state-of-the-art algorithms under Multi-aspect Streaming [16, 20] settings?

All training and testing were run with MATLAB 2019R on Intel i7-4790 (3.6GHz), GeForce GTX 970, 32GB memory, Linux.

1.4.1 Data

Here we introduce the four real-world datasets used in our experiments.

Click-Through Rate Prediction[1] is a Kaggle contest dataset sponsored by Avazu Inc. It is structured as a target tensor $7(\text{banner_pos}) \times 2075(\text{site_id}) \times 2309(\text{app_id}) \times 4581(\text{device_model})$ where each tensor entry coupled with an anonymized categorical vector $[7(\text{C1}) \times 606(\text{C14}) \times 8(\text{C15}) \times 9(\text{C16}) \times 162(\text{C17}) \times 4(\text{C18}) \times 41(\text{C19}) \times 161(\text{C20}) \times 35(\text{C21})]$. Tensor entry value indicates whether an advertisement with given properties was clicked or not. Notice that the side information is already anonymized and coupled with target tensor, it is unclear which modes did the side information coupled on. So we are not able to structure a complete tensor with side information coupled. Thus we used this dataset in the arbitrary-order streaming setting which does not need a complete tensor coupled with side information.

YELP [16][2] is structured as a target tensor $1000(\text{user}) \times 992(\text{business}) \times 93(\text{year-month})$ coupled with a side information matrix $992(\text{businesses}) \times 56(\text{cities})$ on business mode. Tensor entry value indicates whether a user rated a business in a given time or not. This

[1] https://www.kaggle.com/c/avazu-ctr-prediction/data

[2] https://github.com/madhavcsa/SIITA/blob/master/datasets

dataset is used in the multi-aspect streaming setting. The initial size is $200 \times 200 \times 93$. Then we added 40 new users and 40 new businesses in each time step until maxed out.

Retail [5][3] is an online transaction dataset. It is structured as a target tensor $4070(\text{product}) \times 38(\text{country}) \times 54(\text{week})$ coupled with one side information matrix on each mode. So we have three side information matrices in total: 1.) $4070(\text{product}) \times 99(\text{product_quantity})$ represents the total quantity of each distinct product. 2.) $38(\text{country}) \times 14(\text{country_quantity})$ represents the total quantity of products sold to each country. 3.) $54(\text{week}) \times 19(\text{week_quantity})$ represents the total quantity of products sold in each week.

MovieLens [7]is a movie recommendation dataset which is downsampled to simulate the MovieLens dataset used in [6]. To create this dataset, we identified the 400 most active users and the 400 most rated businesses in the first 31 weeks out of the origin MovieLens-1M dataset [7][4]. Then structure a target tensor $400(\text{user}) \times 400(\text{movie}) \times 31(\text{week})$ coupled with a side information tensor $400(\text{user}) \times 2(\text{gender}) \times 7(\text{age}) \times 21(\text{occupation}) \times 170(\text{country}) \times 49(\text{state})$ on user mode. Tensor entry value indicates whether a user rated a movie in a given time or not. This dataset is used in the multi-aspect streaming setting. The initial size is $50 \times 50 \times 31$. Then we add 10 new users and 10 new movies in each time step.

1.4.2 Baseline Methods

Here we introduce the four state-of-the-art baseline algorithms used in our experiments.

POST [6] is POSTsi without the ability of processing side information, in terms of functionality. It uses Streaming Variational Inference to solve a probabilistic CP decomposition model of target tensor.

SIITA [16] is an multi-aspect streaming tensor extension of the inductive framework from the matrix completion with side information [11,15,19]. To the best of our knowledge, SIITA is the only dynamic tensor completion algorithm that incorporates side information to improve accuracy performance.

TNCP [14] is a state-of-the-art static tensor decomposition algorithm. It optimizes a

[3] `https://archive.ics.uci.edu/ml/datasets/online+retail`

[4] `https://grouplens.org/datasets/movielens/`

factor matrix trace norm problem with the ADMM algorithm.

MAST [20], as its name would suggest, is a tensor completion algorithm built for Multi-aspect Streaming Tensor setting.

For the arbitrary-order streaming experiment, only POST and POSTsi are compared because other baselines TNCP, MAST, SIITA are not capable of processing this type of streaming data.

1.4.3 Experimental Setup and Metric

To evaluate the effectiveness of POSTsi and baseline algorithms, we followed a commonly used approach [6, 16, 20] to generate training and test datasets. We first decided a fixed missing percentage of whole target tensor, and randomly marked this many tensor entries as missing. Then we used the remaining tensor and corresponding side information as training data, and used the missing tensor entries as testing data for performance evaluation. To further investigate the impact of sparsity, we tested various tensor missing percentage $\{50\%, 80\%, 90\%\}$ in Multi-aspect Streaming setting. Similarly, we tested various rank $\{3, 5, 8, 10\}$ in both streaming settings to investigate the impact of rank.

Because all the tasks are binary classification, we used the Area Under Curve (AUC) based evaluation metric in all testing. Specifically, in arbitrary-order streaming setting, for each predefined rank, we ran involved algorithms on five random training/test split to predict the missing tensor entries and reported the AUCs from missed tensor entries/ground truth. In Multi-aspect Streaming setting, we followed the widely used routine [6, 16, 20] for this setting. We first did the train/test split and decide an initial tensor size, e.g. $\{200 \times 200 \times 93\}$ in YELP, to do a warm-start for all involved algorithms. Then upon each time step, we trained the algorithms with newly observed incremental data, e.g. 40 new users and 40 new businesses in YELP, and evaluated the average of AUCs on all incremental data up to current time step. The AUC evaluation approach we used here is called the running-average Area Under Curve (RA-AUC) [6, 20].

General parameter settings: POSTsi, POST, TNCP and MAST are CP-based algorithms. We initialized each element of each latent vector $\mathbf{u}$ by randomly sampling from the standard uniform distribution unif$(0, 1)$. Furthermore, because POSTsi and POST are Bayesian

CP-based algorithm, we also set the covariance of each $\mathbf{u}$ to be an identity matrix. Note that in the Multi-aspect Streaming setting, this initialization was done before the warm-up phase. On the other side, because SIITA is a Tucker-based algorithm, it cannot use the CP-based initialization. The original paper [16] suggests to randomly initialize the Tucker-based latent parameters $\mathcal{U}$ and $\mathcal{G}$. We used their implementation[5] to initialize $\mathcal{U}, \mathcal{G}$ by randomly sampling from uniform distribution.

For POSTsi, we initialized each weight scalar w_s^f to a standard normal distribution $\mathcal{N}(0,1)$ and $\boldsymbol{\lambda}$ to a Multivariate normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$. Unless specified in the experiment section, we followed [6], [16], [20], [14] to set the optimal parameters for POST, SIITA, MAST and TNCP, respectively.

1.4.4 Evaluation on arbitrary-order streaming

After laying out the general settings of our experiments, including what the baseline algorithms are, how we create train-test data, and how we train and evaluate the algorithms; what the general parameter settings are, we now list the detail of each experiment, including customized parameter setting and experiment results.

In the arbitrary-order streaming setting, we ran POSTsi, POST and TNCP on our Click-Through Rate Prediction dataset. MAST and SIITA, as Multi-aspect Streaming algorithms, theoretically do not support arbitrary-order streaming, so we skipped them in this experiment. The available TNCP implementation[6] requires a complete tensor to run. In this experiment, reconstruct a complete tensor requires over 100GB of memory, so we skip TNCP too. Here we set maxiter $= 500$ for all involved algorithms; tol $= 50$ for POST and POSTsi; tol $=$ for TNCP.

The 5-trial AUC results are shown in Fig. 1.4. Both POSTsi and POST have stable performance, while POSTsi consistently outperforms POST by about 1%.

---

[5] `https://github.com/madhavcsa/SIITA/blob/master/proposed/run_mast_si.m\#L90`

[6] `https://github.com/yishuaidu/POST/blob/f7ee2fd3fe21b8046f852dfb9f54bb8dac8ff802/code/MovieLen/NNCP_code/NNCP.m\#L1`

1.4.5 Evaluation on Multi-aspect Streaming

Here we test a various types of side information relations: 1.) One side information matrix coupled target tensor on one mode 2.) One side information tensor coupled target tensor on one mode 2.) Multiple side information matrices coupled target tensor on multiple modes

Coupled Side Information Matrix On Single Mode: In this experiment, we tested POSTsi, POST, MAST and TNCP on YELP dataset. We skipped SIITA in this one because, based on their available SIITA implementation[7], we were not able to finish SIITA within a reasonable time given its optimal parameter setting. YELP, among all datasets, contains the largest target tensor. We set maxiter $= 800$ for all invovled algorithms; tol $= 10^{-1}$ for POST and POSTsi; tol $= 10^{-3}$ for MAST; tol $= 10^{-2}$ for TNCP.

The result of each (missing percentage, rank) combination is shown in 1.5. Each subfigure shows the RA-AUC of all involved algorithms against each incremental time instance.

Coupled Side Information Tensor On Single Mode: In this experiment, we tested all algorithms on Retail dataset. We set maxiter $= 800$ for all involved algorithms; tol $= 10^{-2}$ for POST and POSTsi; tol $= 10^{-3}$ for MAST and TNCP. For SIITA, we set rank $= (r, r, r)$, K $= 1.5 * 10^4$, $\gamma = 1e-5$, alpha_step $= 1$, where r is picked from $\{3, 5, 8, 10\}$ the various ranks we want to test.

The result of each (missing percentage, rank) combination is shown in 1.6. Each subfigure shows the RA-AUC of all involved algorithms against each incremental time instance.

Coupled Side Information Matrix On Multiple Mode: In this experiment, we tested all algorithms on MovieLens dataset. We set maxiter $= 800$ for all involved algorithms; tol $= 10^{-2}$ for POST and POSTsi; tol $= 10^{-3}$ for MAST and TNCP. For SIITA, we set rank $= (r, r, r)$, K $= 1.5 * 10^4$, $\gamma = 8e-5$, alpha_step $= 1.06$.

The result of each (missing percentage, rank) combination is shown in 1.7. Each subfigure shows the RA-AUC of all involved algorithms against each incremental time instance.

In all cases, POSTsi's performance is similar to POST in the beginning but surpasses POST very quickly. POSTsi and POST, comparing with other algorithms, have outstand-

[7] `https://github.com/madhavcsa/SIITA`

ing and stable performance since the very beginning. On Retail and MovieLens, SIITA is sensitive to the change of missing percentage and rank; Though TNCP outperforms POSTsi occasionally, it is sensitive to the change of rank; MAST's performance generally has a positive correlation with rank and amount of available information, and is worse than POSTsi mots of the time.

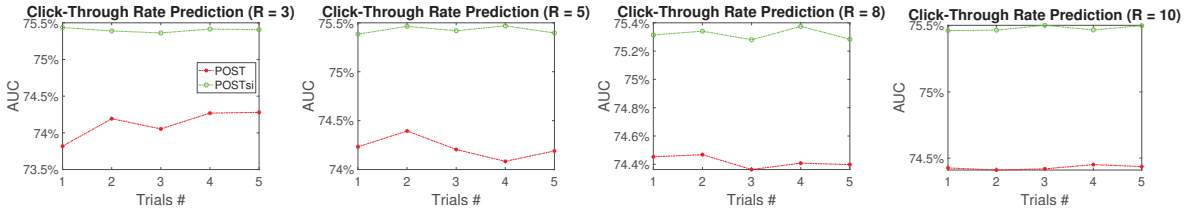

Figure 1.4: Click-Through Rate Prediction Dataset

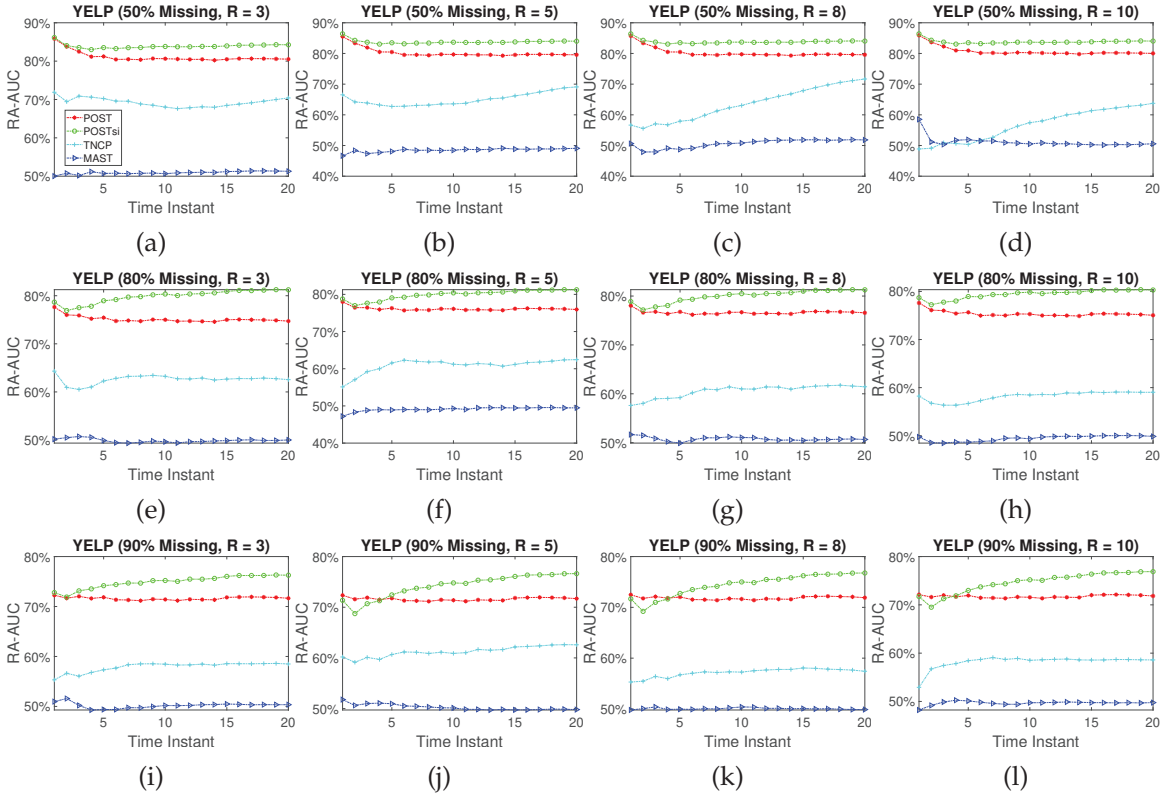

Figure 1.5: YELP

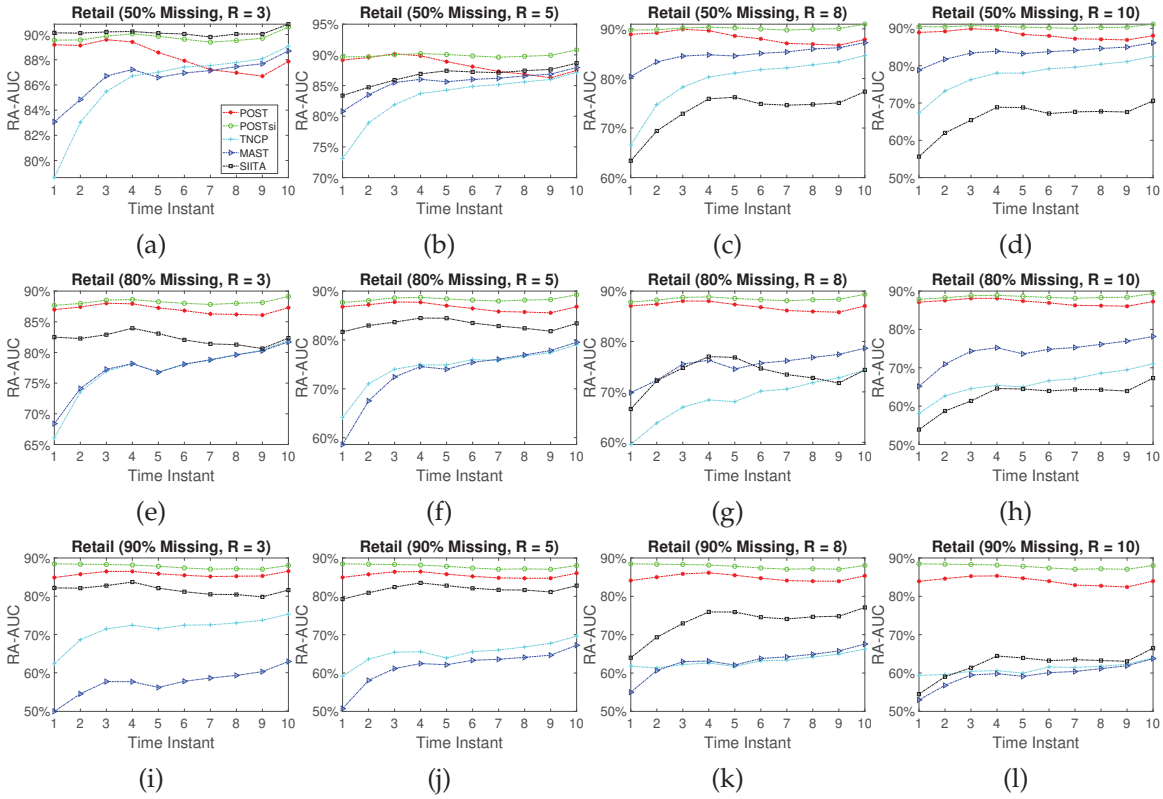

Figure 1.6: Retail

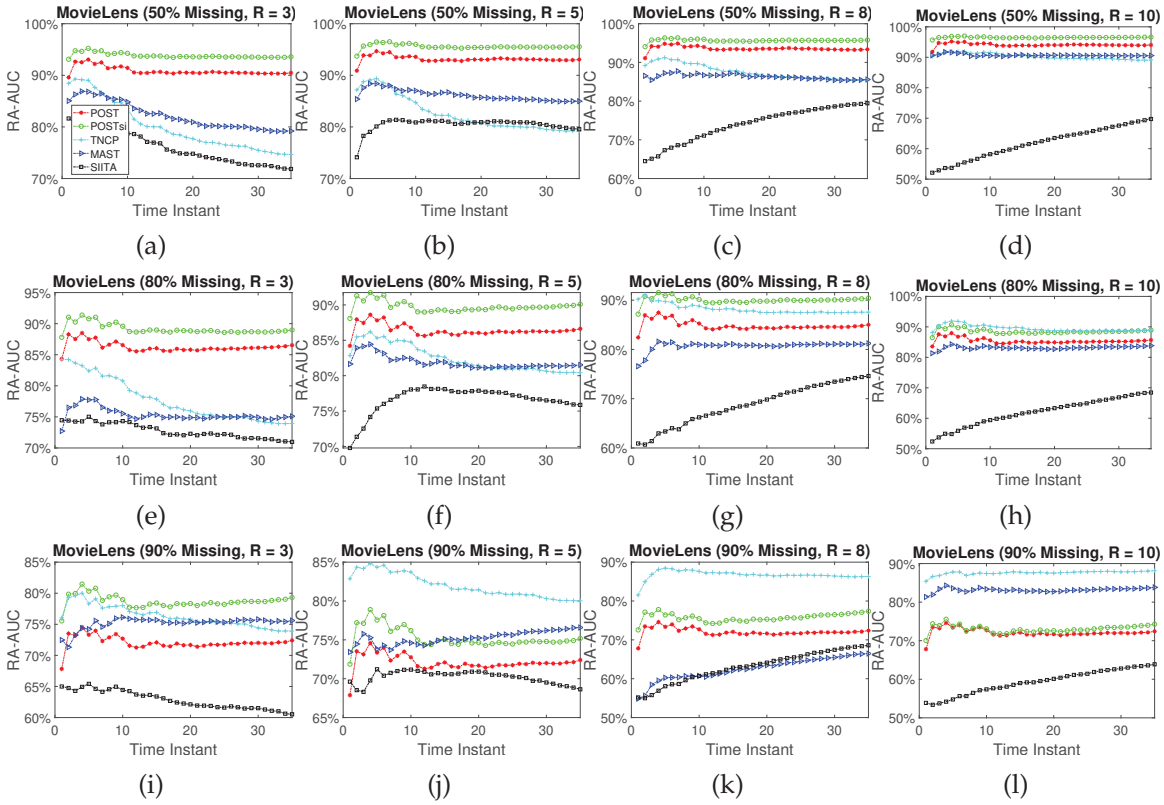

Figure 1.7: MovieLens

1.5 Conclusion

We have developed POSTsi, a streaming probabilistic tensor decomposition algorithm. Assume we have a tensor that simultaneously grows in multiple dimensions. Also, assume the tensor coupled with side information. The POSTsi can process the tensor entries stream in arbitrary order, provide uncertainty quantification on the result of decomposition and recovered tensor entry, and exploit the side information to enhance the quality of the result of decomposition. The experiments on real-world datasets show the encouraging potential of exploiting side information in streaming tensor decomposition algorithms. In future work, we can explore the possible applications of the decomposed tensor, which contains uncertainty information; we might also explore the possibility of automatically choosing the best tensor rank for POSTsi.

REFERENCES

[1] Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten M**o**rup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, March 2011.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, NY, corrected at 8th printing 2009 edition, 2009.

[3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017.

[4] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1727–1735. Curran Associates, Inc., 2013.

[5] Daqing Chen, Sai Laing Sain, and Kun Guo. Data mining for the online retail industry: a case study of rfm model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 19(3):197–208, September 2012.

[6] Y. Du, Y. Zheng, K. Lee, and S. Zhe. Probabilistic streaming tensor decomposition. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 99–108, November 2018.

[7] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: history and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[8] Frank L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[9] Frank L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.

[10] Peter D. Hoff. Hierarchical multilinear models for multiway data. *Computational Statistics & Data Analysis*, 55(1):530–543, January 2011.

[11] Prateek Jain and Inderjit S. Dhillon. Provable inductive matrix completion. *arXiv:1306.0626 [cs, math, stat]*, June 2013.

[12] Rico Jonschkowski, Sebastian Höfer, and Oliver Brock. Patterns for learning with side information. *arXiv:1511.06429 [cs, stat]*, November 2015.

[13] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, August 2009.

[14] Yuanyuan Liu, Fanhua Shang, Licheng Jiao, James Cheng, and Hong Cheng. Trace norm regularized candecomp/parafac decomposition with missing data. *IEEE Transactions on Cybernetics*, 45(11):2437–2448, November 2015.

[15] Nagarajan Natarajan and Inderjit S. Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12):i60–i68, June 2014.

[16] Madhav Nimishakavi, Bamdev Mishra, Manish Gupta, and Partha Talukdar. Inductive framework for multi-aspect streaming tensor completion with side information. *arXiv:1802.06371 [cs]*, February 2018.

[17] Piyush Rai, Changwei Hu, Matthew Harding, and Lawrence Carin. Scalable probabilistic tensor factorization for binary and count data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, page 7, 2015.

[18] Piyush Rai, Yingjian Wang, Shengbo Guo, Gary Chen, David Dunson, and Lawrence Carin. Scalable bayesian low-rank decomposition of incomplete multiway tensors. In *International Conference on Machine Learning*, pages 1800–1808, 2014.

[19] Si Si, Kai-Yang Chiang, Cho-Jui Hsieh, Nikhil Rao, and Inderjit S. Dhillon. Goal-directed inductive matrix completion. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1165–1174, New York, NY, USA, 2016. ACM.

[20] Qingquan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. Multi-aspect streaming tensor completion. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, pages 435–443, Halifax, NS, Canada, 2017. ACM Press.

[21] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383. ACM, 2006.

[22] Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. Incremental tensor analysis: theory and applications. *ACM Trans. Knowl. Discov. Data*, 2(3):11:1–11:37, October 2008.

[23] Ilya Sutskever, Joshua B. Tenenbaum, and Ruslan R Salakhutdinov. Modelling relational data using bayesian clustered tensor factorization. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1821–1828. Curran Associates, Inc., 2009.

[24] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2007.

[25] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 211–222. Society for Industrial and Applied Mathematics, April 2010.

[26] Yun Yang and David B. Dunson. Bayesian conditional tensor factorizations for high-dimensional classification. *Journal of the American Statistical Association*, 111(514):656–669, April 2016.

[27] Shuo Zhou. *On dynamic tensor decompositions*. PhD thesis, The University of Melbourne, Parkville, Melbourne, Australia, May 2019.