NAIST-IS-DD0961018

Doctoral Dissertation

Generalization of Tensor Factorization and Applications

Kohei Hayashi

February 20, 2012

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Kohei Hayashi

Thesis Committee:

Professor Kazushi Ikeda	(Supervisor)
Professor Yuji Matsumoto	(Co-supervisor)
Assistant Professor Kazuho Watanabe	(Co-supervisor)

Generalization of Tensor Factorization and Applications*

Kohei Hayashi

Abstract

A multi-dimensional array or a *tensor* is a common representation of relational data such as the WWW network and protein-protein interactions. Given a data tensor, *tensor factorization* finds low-dimensional features of it that are useful for the data analysis.

As the first contribution of this thesis, we study a tensor factorization model for a heterogeneously attributed tensor such as that contains mixed discrete and continuous variables. The model can manage such heterogeneity by employing individual exponential-family distributions for each attribute of the tensor. The assumption of heterogeneity makes the Bayesian inference intractable, and we cast the EM algorithm approximated by the Laplace method and Gaussian process. We also extend the algorithm for online learning.

Next, we discuss a kernel-based tensor factorization framework for completion of a partially observed tensor. We use the observed elements as inputs for a kernel function. An efficient conjugate-gradient-based algorithm is developed, and it enables us to capture the given tensor on high (possibly infinite) dimensions with moderate computational cost. We also show that our framework includes the K-nearest neighbor and the GP-based models as special cases.

We apply both methods to real data tensors and demonstrate their performances with comparison to conventional methods.

Keywords:

Relational data, tensor factorization, exponential family, Bayesian inference, Gaussian process, kernel method.

* Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0961018, February 20, 2012.

テンソル分解の一般化とその応用*

林 浩平

内容梗概

WWW ネットワークや蛋白質相互作用といった関係データは多次元配列，あるいは**テンソル**として表現できる．**テンソル分解**は与えられたデータテンソルの低次元特徴量を抽出する手法であり，データ解析によく用いられる．

本論文では，まず第一の貢献として，例えば離散値と連続値をあわせ持つような，ヘテロな構造を持つテンソルに適したテンソル分解モデルを研究する．テンソルの各要素ごとに異なる指数型分布族を仮定することで，データが持つヘテロ性を適切に扱うことが可能となる．指数型分布族の採用によりベイズ推論が計算困難となるため，ラプラス近似とガウス過程により近似した EM アルゴリズムを新に導出する．またより大規模なデータを処理するためオンライン学習アルゴリズムへの拡張も行う．

次に，部分的に観測されたテンソルの要素補完のための，カーネル法に基づくデータテンソルのモデルを提案する．観測されたテンソルの要素そのものをカーネル関数の入力とし，データの低ランク性を仮定しないモデリングを可能とする．共役勾配法に基づく効率的なアルゴリズムも同時に導出する．さらに，K-近傍法，行列あるいはテンソル分解法，あるいはガウス過程に基づく手法が提案手法の特別な場合として解釈可能であることを示す．

それぞれの手法について複数の実データを応用し，既存のテンソル分解法と比較して提案手法の性能がどのような場合に優れているかを検証する．

キーワード

関係データ, テンソル分解, 指数型分布族, ベイズ推論, ガウス過程, カーネル法

* 奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DD0961018, 2012 年 2 月 20 日.

Contents

Chapter 1

Introduction

Multi-dimensional arrays or *tensors* are common representations of various data. As a network can be represented by an adjacency matrix (note that a matrix is a special case of a tensor), for example, a multi-dimensional network (Tang et al., 2009), having multiple linkages, can be seen as a *link types* $\times$ *nodes* $\times$ *nodes* tensor. Another example is a temporal sequence of measurements from various distributed sensors such as microphones, thermometers, and video cameras, which are represented by a tensor with dimensions *sensor types* $\times$ *locations* $\times$ *time*. Such kind of data are called relational data (Getoor and Taskar, 2007), which are a collection of relationships among objects.

1.1. Motivation and Contributions

Since tensor representations of relational data are high-dimensional and large-scale, feature extraction is necessary in general for data analysis. *Tensor factorization* is a method that transforms the original tensor into low-dimensional parameters with keeping the tensor structure, which has various applications in data mining such as face recognition, social network analysis, and EEG analysis (Kolda and Bader, 2009; Mørup, 2011).

Many different ways of decomposing a tensor have been proposed (Kolda and Bader, 2009; Acar and Yener, 2009). The *Tucker decomposition* (Tucker, 1966) is one of the general forms of tensor factorization, which can be seen as a direct extension of the singular value decomposition (SVD) from matrices to tensors. Recently, non-negative extensions have been developed especially in neuroscience fields (Shashua and Hazan, 2005; Cichocki et al., 2007). In the Bayesian community several authors have studied its probabilistic extensions (Liu et al., 2007; Tao et al., 2008; Mørup and Hansen, 2009; Chu and Ghahramani, 2009).

Simply speaking, the Tucker decomposition is based on two assumptions: (i) observation noises are following a Gaussian distribution and (ii) the given tensor is

low-rank[1]. These two assumptions are general and make the algorithm feasible. Of course, however, the Tucker decomposition is not effective when these assumptions do not hold. A typical scenario is that the data are *heterogeneously* attributed, i.e., when the statistical properties of the attributes are quite different from each other. For example, if the elements of the given tensor are mixed continuous and discrete variables, the Gaussian noise assumption is not valid. In some real data sets the low-rank assumption are also not satisfied. Apparently we cannot know whether the given tensor is truly low-rank or not in advance, and the low-rank constraint may make the performance worse, comparing to the usual data types such as vectors or matrices.

Improving the computational efficiency is another important issue in tensor factorization. An L-th order data tensor represents the all combinations among the L different objects as the elements. The combinatorial explosion then easily occurs with growing L and it rapidly consumes computational resources in both time and space.

In this thesis, we tackle these problems and propose two new tensor factorization frameworks, which independently relaxes the Tucker decomposition's assumptions. Firstly, we introduce the basics about tensors and the Tucker decomposition. Next we generalize the Tucker decomposition with employing the exponential-family distributions. A kernel extension of the Tucker decomposition is then discussed that allows us to model a full-rank tensor. In each model an efficient learning algorithm is developed and the performance is evaluated by using several real data sets.

Notation

Throughout this thesis, we call an L-dimensional arrays an *L-mode* or an *L-th order tensor*. As special cases, 1- and 2-mode tensors are equivalent to vectors and matrices, respectively. We use a lowercase bold letter as a vector, an uppercase bold letter as a matrix and an uppercase bold letter with underline such as $\underline{\mathbf{A}}$ as a tensor. We denote an l-th mode *unfolded* tensor $\underline{\mathbf{A}}$ by $\mathbf{A}^{(l)}$. Unfolding, or matricizing of a tensor, is done by reordering the elements of a tensor into a matrix with keeping the structure with respect to one arbitrary mode of the tensor. We denote a vectorization of $\underline{\mathbf{A}}$ by vec $\underline{\mathbf{A}}$ or simply $\vec{\mathbf{a}}$. For example, when we have an L-th order tensor $\underline{\mathbf{A}}$ whose dimensions are $D_1 \times \cdots \times D_L$, $\mathbf{A}^{(l)}$ is a $D_l \times D_{\backslash l}$ matrix with $D_{\backslash l} \equiv \prod_{k \neq l} D_k$, and $\vec{\mathbf{a}}$ is a D-dimensional vector with $D \equiv \prod_{l=1}^{L} D_l$. $\mathbb{I}(\cdot)$ denotes the indicator function.

1. We discuss the notion of the rank of tensors in Section 2.4

Chapter 2

Tensor Factorization

In this chapter, we shortly review the basics of a tensor and its factorization methods. Note that we only discuss the case that the order of a data tensor is three ($L = 3$). However, the formulations can easily be generalized to higher order tensors ($L > 3$).

2.1. Tucker Decomposition

Let $\underline{\mathbf{X}}$ be a $D_1 \times D_2 \times D_3$ observation tensor. *The Tucker decomposition* provides a way to factorize $\underline{\mathbf{X}}$ into a core tensor $\underline{\mathbf{Z}} \in \mathbb{R}^{K_1 \times K_2 \times K_3}$ and three factor matrices $\mathbf{U}_l \in \mathbb{R}^{D_l \times K_l}$ $(l = 1, 2, 3)$, and represents an (i, j, k)-th element x_{ijk} of $\underline{\mathbf{X}}$ as

$$x_{ijk} = y_{ijk} + \varepsilon_{ijk}, \tag{2.1}$$

$$y_{ijk} = \sum_{q=1}^{K_1} \sum_{r=1}^{K_2} \sum_{s=1}^{K_3} z_{qrs} u_{1,iq} u_{2,jr} u_{3,ks} \tag{2.2}$$

where z_{qrs} is a (q, r, s)-th element of $\underline{\mathbf{Z}}$, $u_{l,iq}$ is an (i, q)-th element of the factor matrix $\mathbf{U}_l$, and ε_{ijk} is an additive observation noise. Figure 2.1 illustrates how the Tucker decomposition factorizes the data tensor. The matrix $\mathbf{U}_l$ captures a structure of correlation on the l-th mode of $\underline{\mathbf{X}}$. To reduce the redundancy of the model, orthonormal constraints are commonly imposed to the factor matrices.

For later convenience, we rewrite equation (2.1) by a vector and a matrix form. Let $\vec{\mathbf{x}}$ be a $D \equiv D_1 D_2 D_3$-dimensional vector whose elements are given by these of $\underline{\mathbf{X}}$ with appropriate reordering. Then we can rewrite equation (2.1) as

$$\vec{\mathbf{x}} = \mathbf{W}\vec{\mathbf{z}} + \vec{\boldsymbol{\varepsilon}}, \quad \mathbf{W} \equiv \mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1 \tag{2.3}$$

where $\otimes$ is the Kronecker product of matrices and $\mathbf{W} \in \mathbb{R}^{D \times K}$ is a matrix which maps a vectorized $K \equiv K_1 K_2 K_3$-dimensional core tensor $\vec{\mathbf{z}}$ to $\mathbb{R}^D$. In this form, the Tucker decomposition is viewed as a standard linear model in which $\vec{\mathbf{z}}$ is a K-dimensional representation of the observation $\vec{\mathbf{x}}$ in the linear space spanned by K bases $\mathbf{w}_d = \mathbf{u}_{3,i} \otimes \mathbf{u}_{2,j} \otimes \mathbf{u}_{1,k}$, where $\mathbf{w}_d$ is the d-th row vector of $\mathbf{W}$, d is an index

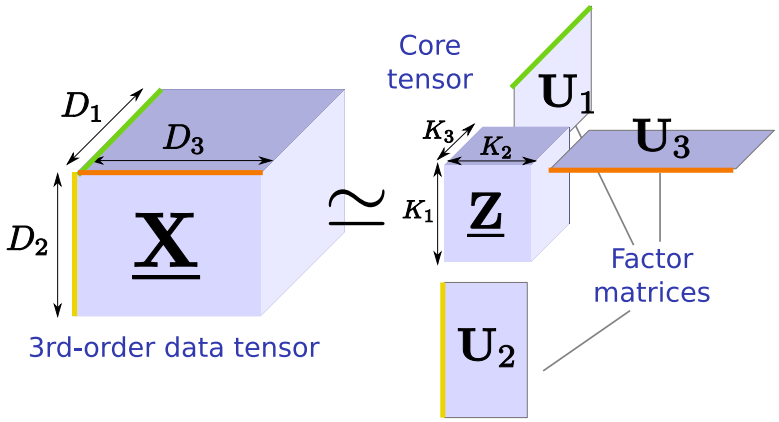

Figure 2.1: The Tucker decomposition.

of vectorized data associated with the index (i, j, k) of the tensor representation, and $\mathbf{u}_{l,i}$ is the i-th row vector of $\mathbf{U}_l$. The structure of the data tensor is embedded in $\mathbf{W}$ by the Kronecker products of $\{\mathbf{U}_l\}$.

The Tucker decomposition estimates the parameters $\underline{\mathbf{Z}}$ and $\{\mathbf{U}_l\}$ by minimizing the sum of square errors $\sum_{ijk} \varepsilon_{ijk}^2$. Higher-order singular value decomposition (HOSVD) (De Lathauwer et al., 2000b) is one of the methods to solve the Tucker decomposition. HOSVD estimates $\mathbf{U}_l$ as the top K_l leading left singular vectors of $\mathbf{X}^{(l)} \in \mathbb{R}^{D_l \times D_{\setminus l}}$. For more detail, see (Kolda and Bader, 2009; Cichocki et al., 2007).

2.2. PARAFAC

PARAFAC, also known as CANDECOMP, is another form of tensor factorization, introduced by Harshman (1970). PARAFAC can be seen as a restricted model of the Tucker decomposition, i.e., PARAFAC decomposes $\underline{\mathbf{X}}$ into three factor matrices $\mathbf{A}_1 \in \mathbb{R}^{D_1 \times Q}$, $\mathbf{A}_2 \in \mathbb{R}^{D_2 \times Q}$, and $\mathbf{A}_3 \in \mathbb{R}^{D_3 \times Q}$:

$$x_{ijk} = \sum_{j=1}^{J} a_{1,iq} a_{2,jq} a_{3,kq} + v_{ijk} \tag{2.4}$$

where v_{ijk} is a noise (residual). In comparison with the Tucker decomposition, the dimensions of the bases of PARAFAC are restricted to take the same value and each basis only interact with the corresponding one. Due to the simplicity, the interpretation of the estimated factor matrices is much easier than the Tucker decomposition.

2.3. pTucker

A probabilistic extension of the Tucker decomposition has been proposed by Chu and Ghahramani (2009). The model called *pTucker* employs the Gaussian likelihood and

the Gaussian prior to the parameters:

$$p(\underline{\mathbf{Z}}) = \prod_{qrs} N(z_{qrs} \mid 0, 1), \tag{2.5}$$

$$p(\mathbf{U}_l) = \prod_{iq} N(u_{l,iq} \mid 0, 1), \tag{2.6}$$

$$p(\underline{\mathbf{X}} \mid \underline{\mathbf{Z}}, \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3) = \prod_{ijk} N(x_{ijk} \mid y_{ijk}, \sigma^2) \tag{2.7}$$

where y_{ijk} is defined in Equation (2.2). pTucker estimates the factor matrix based on the marginal likelihood,

$$\hat{\mathbf{U}}_1 = \underset{\mathbf{U}_1}{\operatorname{argmax}}\, p(\mathbf{U}_1) \int p(\underline{\mathbf{X}} \mid \underline{\mathbf{Z}}, \mathbf{U}_1, \hat{\mathbf{U}}_2, \hat{\mathbf{U}}_3) p(\underline{\mathbf{Z}}) \mathrm{d}\underline{\mathbf{Z}}. \tag{2.8}$$

Note that if $\hat{\mathbf{U}}_2$ and $\hat{\mathbf{U}}_3$ are known, we obtain $\hat{\mathbf{U}}_1$ as a closed-form solution.

Due to the Gaussian prior and the marginalization of the core tensor, the solution of pTucker tends to be low-rank Chu and Ghahramani (2009) (next section introduces the notion of low-rank of a tensor.) This effect is caused by the model marginalization in the Bayesian inference (Nakajima et al., 2010, 2011).

2.4. Rank of Tensors

A rank-one tensor $\mathbf{A}$ is defined as the outer product of vectors, i.e.,

$$\mathbf{A} = \mathbf{u} \circ \mathbf{v} \circ \mathbf{w} \qquad \Longleftrightarrow \qquad a_{ijk} = u_i v_j w_k \quad \forall i, j, k. \tag{2.9}$$

If $\underline{\mathbf{X}}$ can exactly be represented by sum of K rank-one tensors, we say the *rank* of $\underline{\mathbf{X}}$ is K. Although the notion of the tensor rank is simple and intuitive, the computation of the tensor rank of a given tensor is not straightforward, which is known as an NP complete problem (Hastad, 1990). Computation of the tensor rank is an important problem in communication complexity (Pudlák et al., 1997) and its lower- and upper-bounds are well studied (Alexeev et al., 2011).

Here we introduce the notion of mode-k rank (De Lathauwer and De Moor, 1998; Tomioka et al., 2011), which is defined as the rank of the mode-k unfolded tensor, i.e., $\mathrm{rank}(\mathbf{X}^{(k)})$ for the given tensor $\underline{\mathbf{X}}$. Unlike the tensor rank, the mode-k rank is solved by standard linear algebra techniques such as SVD, which is computable in polynomial time. Note that the mode-k rank for matrices takes the same value of the rank of matrices, i.e., $\mathrm{rank}(\mathbf{X}) = \mathrm{rank}(\mathbf{X}^\top)$.

We say $\underline{\mathbf{X}}$ is rank-(K_1, K_2, K_3) tensor if the mode-k rank of $\underline{\mathbf{X}}$ is K_l for $l = 1, 2,$ and 3. Note that, if $\underline{\mathbf{X}}$ is rank-(K_1, K_2, K_3), $\mathbf{X}$ is exactly represented by the Tucker decomposition (2.1) with the $K_1 \times K_2 \times K_3$ core tensor, which is simply obtained by applying SVD for each unfolded tensor of $\underline{\mathbf{X}}$ in polynomial time. The best rank-$(\tilde{K}_1, \tilde{K}_2, \tilde{K}_3)$ approximation of $\tilde{K}_l < K_l$ for $l = 1, 2,$ and 3 does, however, not coincide to the Tucker decomposition constructed by the best $\tilde{K}$ approximation of SVDs.

Chapter 3

Factorization of a Heterogeneous Tensor

3.1. Introduction

The most basic and general problem setting in the network analysis is that of a single network. But in many cases there possibly exist other different yet related networks. Intuitively, if there exist correlations among the networks, it is effective to model the entire structure holding among them rather than to model them independently. When the data contain missing elements, such an integrative approach will be especially effective for data completion. This idea, borrowing information from across multiple data sources, can be seen as a typical data fusion (Hall and Llinas, 1997) or transfer learning (Pan and Yang, 2010) problem. The tensor factorization methods seem to be a desirable approach to deal with such multiple information.

However, the approach is unfeasible when the data are *heterogeneously* attributed, i.e., when the statistical properties of the attributes are quite different from each other. As discussed in the previous chapter, the Tucker decomposition estimates the parameters by the least squares approach, which is equivalent to the maximum likelihood estimate under the assumption of the isotropic Gaussian noise. From this statistical viewpoint, the Tucker decomposition is inappropriate to model non-Gaussian observations such as a heterogeneously attributed array with both real and discrete variables.

Another problem of tensor factorization is that the learning algorithm is not appropriate for processing data in real-time. Since the learning algorithm of the Tucker decomposition is mainly focused on the batch procedure, we need to re-compute the entire structure of the data tensor whenever additional samples are observed; it is difficult to analyze time-series data such as the multiple sensor measurements in an online manner.

To overcome these problems, we propose a new tensor factorization method called "Exponential family Tensor Factorization" (ETF), which generalizes the likelihood of the Tucker decomposition by using exponential-family distributions. The exponential

family is a class of distributions that is widely applicable for modeling various types of data. For real-valued observations, we can use the Gaussian distribution. The exponential distribution is especially useful for non-negative values. Discrete variables can be represented by distributions such as Poisson and Bernoulli. To deal with the heterogeneity, we assume an individual exponential-family distribution for each attribute on the array data. In addition, we introduce latent variables that capture noise-corrupted heterogeneous array data into a unified low-dimensional parameter space. Because there is no analytical solution when the exponential family is applied for Bayesian inference in general, we use the expectation-maximization (EM) algorithm for parameter estimation, in which the Gaussian process (Rasmussen and Williams, 2006) is employed for approximation. Our approximation scheme provides a computationally efficient algorithm for parameter estimation compared to a naïve sampling method, and also allows us to derive a Bayesian predictive distribution for missing elements in a consistent manner. We also propose an efficient online learning procedure of ETF. The online algorithm allows us to deal with a large but slender data tensor such as time-series data, which the batch algorithm cannot handle due to the computational complexity. In addition, it enables us to process a data tensor sequentially in real-time without keeping past observations. Finally we show that estimated parameters can be applied for missing-values prediction and anomaly detection.

3.2. Exponential Family Tensor Factorization

As mentioned before, the Tucker decomposition estimates the parameters $\{\mathbf{U}_l\}$ and $\vec{\mathbf{z}}$ by minimizing the sum of square errors. In a probabilistic perspective, we can interpret that the Tucker decomposition models the expectation of $\vec{\mathbf{x}}$ by $\mathbf{W}\vec{\mathbf{z}}$ and then estimates the maximum likelihood solution of $\{\mathbf{U}_l\}$ and $\vec{\mathbf{z}}$ under the assumption of a spherical Gaussian noise $\vec{\boldsymbol{\varepsilon}}$. However, this assumption is not appropriate when the data $\underline{\mathbf{X}}$ is heterogeneously distributed.

To overcome the heterogeneity, we employ the *exponential-family* distributions defined by

$$\text{Expon}(x \mid \boldsymbol{\omega}) \equiv \exp\left[\boldsymbol{\omega}^\top \mathbf{g}(x) - \psi(\boldsymbol{\omega}) + F(x)\right] \tag{3.1}$$

where $\mathbf{g}(x) = (x, \tilde{\mathbf{g}}(x)^\top)^\top$ is a sufficient statistic with nonlinear function $\tilde{\mathbf{g}}$, $\boldsymbol{\omega} = (\theta, \tilde{\boldsymbol{\omega}}^\top)^\top$ is a natural parameter, $\exp(F(x))$ is a base measure, and

$$\psi(\boldsymbol{\omega}) = \ln \int \exp[\boldsymbol{\omega}^\top \mathbf{g}(x) + F(x)] \mathrm{d}x \tag{3.2}$$

is the log-partition function. The exponential family (3.1) includes many distributions such as Gaussian, Poisson, and Binomial. If $g(x) = x$ and $\omega = \theta$, i.e., only x is a sufficient statistics, then the distribution (3.1) is specially called *natural exponential family.*

Table 3.1: Summary of popular distributions in the exponential family. Here we define a sigmoid function $f(\theta) \equiv 1/(1+\mathrm{e}^{-\theta})$. Note that a Bernoulli distribution is a special case of Binomial where n, the number of trials, is 1.

Distribution	$\boldsymbol{\omega}$	Domain of x	$\psi(\boldsymbol{\omega})$	$\psi'(\boldsymbol{\omega}) = \mathbb{E}[x\|\boldsymbol{\omega}]$	$\psi''(\boldsymbol{\omega}) = \mathrm{var}[x\|\boldsymbol{\omega}]$
Gaussian	$(\theta, \tilde{\omega})^\top$	$(-\infty, \infty)$	$\frac{1}{2}(\frac{\theta^2}{2\tilde{\omega}} - \ln 2\tilde{\omega})$	$\frac{\theta}{2\tilde{\omega}}$	$\frac{1}{2\tilde{\omega}}$
Bernoulli	θ	$\{0, 1\}$	$\ln(1+\mathrm{e}^{\theta})$	$f(\theta)$	$f(\theta)(1-f(\theta))$
Binomial	θ	$\{0, 1, \dots, n\}$	$n\ln(1+\mathrm{e}^{\theta})$	$nf(\theta)$	$nf(\theta)(1-f(\theta))$
Poisson	θ	$\{0, 1, \dots\}$	e^{θ}	e^{θ}	e^{θ}
Exponential	θ	$[0, \infty)$	$-\ln(-\theta)$	$-\frac{1}{\theta}$	$\frac{1}{\theta^2}$

For example, the Gaussian distribution $N(x|\mu, \sigma^2)$ with a mean μ and a variance σ^2 is written as

$$\begin{aligned}&\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x-\mu)^2\right] \\ &= \exp\left[\begin{pmatrix}\mu/\sigma^2 \\ 1/2\sigma^2\end{pmatrix}^\top \begin{pmatrix}x \\ -x^2\end{pmatrix} - \frac{1}{2}\left(\frac{\mu^2}{2\sigma^2} + \ln(2\pi\sigma^2)\right)\right].\end{aligned}$$

Then we have $\mathbf{g}(x) = (x, -x^2)^\top$, $\boldsymbol{\omega} = (\mu/\sigma^2, 1/2\sigma^2)^\top$, $\psi(\boldsymbol{\omega}) = (\theta^2/2\tilde{\omega} - \ln 2\tilde{\omega})/2$, and $F(x) = -\ln(2\pi)/2$. Note that some distributions (e.g., Poisson and Binomial) only have the first natural parameter θ. Since ψ is the cumulant generating function of the distribution (3.1), we see that the derivative $\psi' : \boldsymbol{\omega} \mapsto \partial\psi/\partial\theta|_{\boldsymbol{\omega}}$ is a mapping from the natural parameter $\boldsymbol{\omega}$ to the conditional expectation $\mathbb{E}[x|\boldsymbol{\omega}]$;

$$\begin{aligned}\psi'(\boldsymbol{\omega}) &= \frac{\partial}{\partial\theta} \ln \int \exp[\boldsymbol{\omega}^\top \mathbf{g}(x) + F(x)]\mathrm{d}x \\ &= \int x \exp[\boldsymbol{\omega}^\top \mathbf{g}(x) - \psi(\boldsymbol{\omega}) + F(x)]\mathrm{d}x = \mathbb{E}[x|\boldsymbol{\omega}]. \end{aligned} \tag{3.3}$$

Similarly, the variance $\mathrm{var}[x|\boldsymbol{\omega}]$ is given by using the second derivative $\psi'' : \boldsymbol{\omega} \mapsto \partial^2\psi/\partial\theta^2|_{\boldsymbol{\omega}} > 0$, which implies that the exponential-family distribution (3.1) is log-concave with respect to θ. Popular distributions in the exponential family are summarized in table 3.1.

Let us consider a probabilistic generative model of a heterogeneous vectorized data tensor $\vec{\mathbf{x}}$. We assume that d-th observation $\vec{x}_d$ follows the distribution $\mathrm{Expon}_{h(d)}$ where $h(d) \in \{\mathrm{Gaussian}, \mathrm{Poisson}, \dots\}$. The data tensor may contain missing entries and we denote an index set of the observed entries of $\vec{\mathbf{x}}$ by $\mathcal{I}$. Then we define the likelihood as

$$\vec{\mathbf{x}}_{\mathcal{I}} \sim \prod_{d\in\mathcal{I}} \mathrm{Expon}_{h(d)}(\vec{x}_d \mid \vec{\theta}_d = \mathbf{w}_d^\top \vec{\mathbf{z}}, \tilde{\boldsymbol{\omega}}_{h(d)}) \tag{3.4}$$

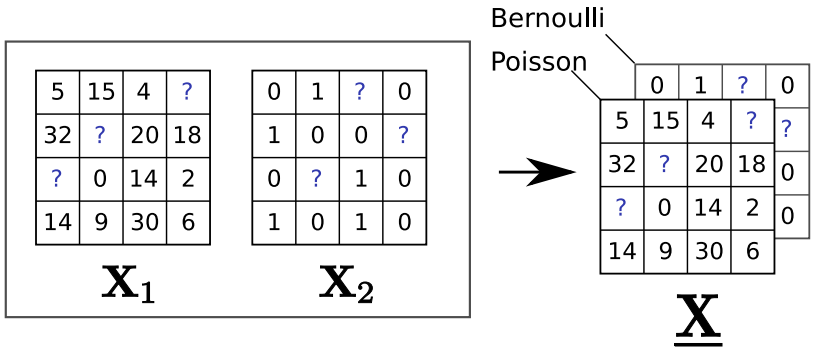

Figure 3.1: An example of heterogeneously attributed tensors.

where $\vec{\mathbf{x}}_{\mathcal{I}}$ is a vector collecting the observed entries. Note that the parameter $\tilde{\boldsymbol{\omega}}_{h(d)}$ is shared by observations having same distribution index $h(d)$ While the Tucker decomposition parametrizes its expectation as $\mathbb{E}[\vec{\mathbf{x}}] = \mathbf{W}\vec{\mathbf{z}}$, our model considers a representation of the natural parameter $\boldsymbol{\omega}$ as $\vec{\boldsymbol{\theta}} = \mathbf{W}\vec{\mathbf{z}}$. We treat the vector $\vec{\mathbf{z}}$ as a latent variable associated with $\vec{\mathbf{x}}$ and assume a spherical Gaussian prior. We also consider a standard Gaussian prior for each row of the factor matrices $\{\mathbf{U}_l\}$,

$$\vec{\mathbf{z}} \sim N(\vec{\mathbf{z}} \mid 0, \mathbf{I}), \quad \mathbf{U}_l \sim \prod_{i=1}^{D_l} N(\mathbf{u}_i^{(l)} \mid 0, \gamma_l^{-1}\mathbf{I}). \tag{3.5}$$

We call the above model ETF. The joint log-likelihood $\mathcal{L}$ is then written as

$$\mathcal{L} = \sum_{d \in \mathcal{I}} \left\{ \vec{x}_d \vec{\theta}_d - \psi_{h(d)}(\vec{\theta}_d) \right\} - \frac{||\vec{\mathbf{z}}||^2}{2} - \sum_{l=1}^{L} \frac{\gamma_l}{2} ||\mathbf{U}_l||^2 + \text{const.} \tag{3.6}$$

The *heterogeneity* of ETF is controlled by the distribution index $h(d)$. Unlike existing models of factorization such as PCA, ETF can independently choose the distributions from the exponential family (3.1) for each attribute. This allows us more flexible modeling of data. For example, let us consider two $D_2 \times D_3$ matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ (see Figure 3.1). We assume that the statistical natures of these matrices are quite different, e.g., the domain of $\mathbf{X}_1$ is positive integer and that of $\mathbf{X}_2$ is binary. A naïve approach would be to model $\mathbf{X}_1$ and $\mathbf{X}_2$ independently. In our framework, we combine $\mathbf{X}_1$ and $\mathbf{X}_2$ as a $2 \times D_2 \times D_3$ tensor $\underline{\mathbf{X}}$ and set Expon_h to the Poisson distribution for x_{1jk} and to Bernoulli for x_{2jk} $(j = 1, \ldots, D_2, k = 1, \ldots, D_3)$. When ETF is applied to $\underline{\mathbf{X}}$, $\mathbf{U}_1$ would extract the correlation information between $\mathbf{X}_1$ and $\mathbf{X}_2$ in the low-dimensional natural parameter space. In this sense, our approach has richer representation ability than the independent modeling of $\mathbf{X}_1$ and $\mathbf{X}_2$.

The likelihood function (3.4) has a strong connection to generalized linear models with canonical links (McCullagh and Nelder, 1989). For example, if we choose a Bernoulli distribution as $\text{Expon}(x|\boldsymbol{\omega})$ for binary data $x \in \{0, 1\}$, ψ' becomes a sigmoid function. In this case, ETF is equivalent to a logistic regression, where $\vec{\mathbf{z}}$, $\vec{\mathbf{x}}$, and $\mathbf{W}$

correspond the input, output, and regression coefficients, respectively. Note that when we set all Expon_d for $d = 1, \ldots, D$ as the isotropic Gaussian, the log-likelihood is equivalent to the loss function of the conventional Tucker decomposition.

3.3. Bayesian Inference

Given the observation $\vec{\mathbf{x}}$, we estimate the model parameters $\vec{\mathbf{z}}$, $\{\mathbf{U}_l\}$, and $\tilde{\boldsymbol{\omega}}$. We estimate $\{\mathbf{U}_l\}$ the marginal-MAP, which is the maximum of the marginal posterior

$$p(\{\mathbf{U}_l\} \mid \vec{\mathbf{x}}_{\mathcal{I}}) \propto \int p(\vec{\mathbf{x}}_{\mathcal{I}}, \{\mathbf{U}_l\} \mid \vec{\mathbf{z}}) p(\vec{\mathbf{z}}) \mathrm{d}\vec{\mathbf{z}}. \tag{3.7}$$

Since the integral operation is intractable, we employ the EM algorithm; however, there are two difficulties for applying the EM algorithm to our model:

- E-step: The normalization term $\int p(\vec{\mathbf{x}}_{\mathcal{I}}|\vec{\mathbf{z}}) p(\vec{\mathbf{z}}) \mathrm{d}\vec{\mathbf{z}}$ of the posterior is generally intractable because of the non-conjugacy of the prior and the likelihood, except in the case that the given likelihood is fully Gaussian.

- M-step: When we maximize the *expected* log-likelihood $\mathbb{E}_{\vec{\mathbf{z}}}[\mathcal{L}|\vec{\mathbf{x}}_{\mathcal{I}}]$ with respect to the parameters $\{\mathbf{U}_1, \ldots, \mathbf{U}_L\}$, the main difficulty for evaluating the expected log-likelihood resides in obtaining the expectation of ψ. Because the function ψ is nonlinear, its expectation by the posterior is generally intractable.

To tackle these problems, we propose a new framework for approximation of the EM algorithm with combining the two techniques: Laplace approximation and Gaussian process (GP). Laplace approximation gives a posterior by a Gaussian distribution. GP approximates ψ and ψ' by exponential-quadratic forms in which the expectation by the Gaussian distribution is analytically solvable. By combining the two, we compute the expected log-likelihood and it allows us to employ a gradient-based optimization in M-step. Furthermore, we can also derive the mean of the Bayesian predictive distribution in a consistent manner.

As an alternative, MAP estimators for $\{\vec{\mathbf{z}}, \mathbf{U}_1, \ldots, \mathbf{U}_L\}$ are also plausible. Since the MAP estimation only needs to maximize the joint log-likelihood (3.6), it does not require to take the expectation. However, the MAP estimator is unstable for noisy observations; we empirically show this fact in the experimental results at Section 3.7.1. Note that, in our case, the MAP estimation is equivalent to the zeroth order delta approximation (Blei and McAuliffe, 2007) of the marginal-MAP estimator.

3.3.1 Posterior Inference by Laplace's Method

In E-step, we approximate the posterior $p(\vec{\mathbf{z}}|\vec{\mathbf{x}})$ by Gaussian $q(\vec{\mathbf{z}}) \equiv N(\vec{\mathbf{z}}|\vec{\mathbf{z}}_0, \boldsymbol{\Sigma}_0)$ with Laplace approximation where $\vec{\mathbf{z}}_0$ is maximum a *posteriori* (MAP), i.e., the mode of the posterior distribution of $\vec{\mathbf{z}}$, and $\boldsymbol{\Sigma}_0$ is the negative inverse of the Hessian of $\mathcal{L}$

at $\vec{\mathbf{z}}_0$. We can use gradient-based method to find $\vec{\mathbf{z}}_0$ with the following gradient and Hessian,

$$\frac{\partial \mathcal{L}}{\partial \vec{\mathbf{z}}} = \mathbf{W}^\top(\vec{\mathbf{x}} - \vec{\boldsymbol{\psi}}') - \vec{\mathbf{z}}, \quad \frac{\partial^2 \mathcal{L}}{\partial \vec{\mathbf{z}}(\partial \vec{\mathbf{z}})^\top} = -\mathbf{W}^\top \boldsymbol{\Psi}'' \mathbf{W} - \mathbf{I}, \tag{3.8}$$

where we define $\vec{\boldsymbol{\psi}}' \equiv (\psi'_{h(1)}(\vec{\theta}_1), \ldots, \psi'_{h(D)}(\vec{\theta}_D))$ and $\boldsymbol{\Psi}'' \equiv \mathrm{diag}(\vec{\boldsymbol{\psi}}'')$.[1] Note that the maximization of $\mathcal{L}$ with respect to $\vec{z}$ is a concave problem, since the negative Hessian is positive definite. Thus, given $\mathbf{W}$, we can find the global maximum of $\vec{\mathbf{z}}$.

3.3.2 Approximation of Expectation with GP

In M-step, we consider the marginal-MAP estimation of $\mathbf{U}_l$. The expected log-likelihood is approximated with the Laplace approximation as

$$\begin{aligned}\bar{\mathcal{L}} &\equiv \int \mathcal{L}(\vec{\mathbf{z}}, \{\mathbf{U}_l\}) N(\vec{\mathbf{z}} | \vec{\mathbf{z}}_0, \boldsymbol{\Sigma}_0) \mathrm{d}\vec{\mathbf{z}} \\ &= \sum_{d \in \mathcal{I}} \{\vec{x}_d \mathbb{E}_q[\vec{\theta}_d] - \mathbb{E}_q[\psi_{h(d)}(\vec{\theta}_d)]\} - \sum_{l=1}^{L} \frac{\gamma_l}{2} ||\mathbf{U}_l||^2 + \text{const.}\end{aligned} \tag{3.9}$$

Here we introduce the unfolded core tensor $\mathbf{Z}^{(l)} \in \mathbb{R}^{K_l \times K_{\setminus l}}$ and natural parameter

$$\boldsymbol{\Theta}^{(l)} = \mathbf{U}_l \mathbf{Z}^{(l)} \mathbf{B}_l^\top \in \mathbb{R}^{D_l \times D_{\setminus l}}, \tag{3.10}$$

$$\mathbf{B}_l = \mathbf{U}_L \otimes \cdots \otimes \mathbf{U}_{l+1} \otimes \mathbf{U}_{l-1} \otimes \cdots \otimes \mathbf{U}_1 \tag{3.11}$$

By denoting by $\mathbf{U}_l^-$ the pseudo inverse of $\mathbf{U}_l$, we obtain the gradient of the expected log-likelihood (3.9) as

$$\begin{aligned}\frac{\partial \bar{\mathcal{L}}}{\partial \mathbf{U}_l} &= \mathbb{E}_q[(\mathbf{X}^{(l)} - \boldsymbol{\Psi}'^{(l)}) \mathbf{B}_l (\mathbf{Z}^{(l)})^\top] - \gamma_l \mathbf{U}_l \\ &= \mathbf{X}^{(l)} \mathbf{A}_l^\top - \mathbb{E}_q[\boldsymbol{\Psi}'^{(l)} \{\boldsymbol{\Theta}^{(l)}\}^\top] \mathbf{U}_l^- - \gamma_l \mathbf{U}_l.\end{aligned} \tag{3.12}$$

where $\boldsymbol{\Psi}'^{(l)}$ is the unfolded tensor of $\vec{\boldsymbol{\psi}}'$ and $\mathbf{A}_l \equiv \mathbb{E}_q[\mathbf{Z}^{(l)}] \mathbf{B}_l^\top$. Here we assume that differential and integral operators are commutative.

To solve the intractable expectations $\mathbb{E}_q[\psi_{h(d)}(\vec{\theta}_d)]$ in the expected log-likelihood (3.9)[2] and $\mathbb{E}_q[\boldsymbol{\Psi}'^{(l)} \{\boldsymbol{\Theta}^{(l)}\}^\top]$ in the gradient (3.12), we approximate ψ_h by a GP's predictive mean function m_h. We separately summarize the approximation framework in Appendix, which is mainly based on the following theorem.

Theorem 1 *Let $m^{(p)}$ be the p-th derivative of the predictive mean function of a GP whose covariance function is a Gaussian kernel. For arbitrary positive integers $p, q \geq 0$, $\mathbb{E}_{p(x_*)}[{x_*}^q m^{(p)}(x_*)]$ where $p(x_*)$ is the Gaussian distribution with a mean μ_* and a variance σ_*^2 is explicitly written as a function of p, q, μ_* and σ_*^2.*

1. Here we omit θ from the functions $\psi, \psi', \ldots$ for the sake of clarity.
2. The expected log-likelihood (3.9) is required to check the convergence of the EM algorithm.

A proof of theorem 1 is given in Section A.4. We observe that the expectations $\mathbb{E}_q[\psi_{h(d)}(\vec{\theta}_d)]$ and $\mathbb{E}_q[\mathbf{\Psi}'^{(l)}\{\mathbf{\Theta}^{(l)}\}^\top]$ are the cases of $q=0, p=0$ and $q=1, p=1$ in theorem 1, and the solutions are given in equation (A.32) and (A.34), respectively.[3]

Note that the expectations in equation (3.9) and (3.12) can independently be taken for each element of $\vec{\boldsymbol{\theta}}$, since we assume that each element of $\vec{\mathbf{x}}$ is independently distributed in (3.4). For example, by a change of variable, we observe that the expectation is rewritten as

$$\int \boldsymbol{\psi}(\vec{\theta}_d) q(\vec{\mathbf{z}}) \mathrm{d}\vec{\mathbf{z}} = \int \boldsymbol{\psi}(\vec{\theta}_d) q(\vec{\theta}_d) \mathrm{d}\vec{\theta}_d \tag{3.13}$$

where $q(\vec{\boldsymbol{\theta}})$ is a Gaussian distribution

$$q(\vec{\boldsymbol{\theta}}) = N(\mathbf{W}\mathbb{E}_q[\vec{\mathbf{z}}], \mathbf{W}\mathrm{cov}_q[\vec{\mathbf{z}}]\mathbf{W}^\top). \tag{3.14}$$

The result allows us to share the training inputs for each $\psi_h, h = 1, \ldots, H$; it reduces the total number of samples used in GP.

For updates of $\{\mathbf{U}_l | l = 1, \ldots, L\}$, we use an alternating optimization, or a block coordinate descent approach (Bertsekas and Bertsekas, 1999), i.e., maximizing $\bar{\mathcal{L}}$ with respect to $\mathbf{U}_l$ with fixed $\{\mathbf{U}_n | n \neq l\}$ iteratively by changing the index l. We use a quasi-Newton method to find the local optima with respect to $\{\mathbf{U}_l\}$.

3.3.3 Details of Implementation and Computational Complexity

We estimate $\tilde{\omega}$, which corresponds to a precision parameter of a Gaussian distribution (see Table 3.1), by using a maximum likelihood solution of a variance of Gaussian attributed elements. We initialize $\{\mathbf{U}_l\}$ by using HOSVD before starting the EM algorithm. We show a pseudo code of our inference algorithm for an L-th order data tensor $\underline{\mathbf{X}}$ in Figure 1. Note that the subroutine `PREPARE_GP` in Figure 1 is a prepossessing procedure for the GP approximation, that includes a sampling procedure of the GP's inputs. For more details, see Section A.6.

The dominating complexity in E-step is the computation of the covariance (inverse of the Hessian) in Laplace approximation, which needs $\mathcal{O}(K^3)$ where $K = \dim(\vec{\mathbf{z}})$. In the GP approximation of M-step, the inverse of kernel matrix $\mathbf{C}$ defined in (A.6) needs $\mathcal{O}(N^3)$, where N is the number of samples used as the GP's training inputs. N determines the accuracy of the approximation. We need this expensive computation for each EM iteration. Before the update of $\mathbf{U}_l$, we need to compute the pseudo inverse $\mathbf{U}_l^-$, which is typically $\mathcal{O}(D_l K_l^2)$. The computation of the gradient with respect to $\mathbf{U}_l$ needs $\mathcal{O}(DK_{\backslash l})$.

3. Here we consider that $\mathbf{W}$ is not a random variable. Since we approximate the posterior of $\vec{\mathbf{z}}$ as Gaussian in the previous section, the posterior of $\vec{\boldsymbol{\theta}} = \mathbf{W}\vec{\mathbf{z}}$ is also a Gaussian. This allows us to compute the above expectations as expectations with respect to not $\vec{\mathbf{z}}$ but $\vec{\boldsymbol{\theta}}$.

Algorithm 1 Batch algorithm for Bayesian inference of ETF, where $\underline{\mathbf{X}}$ is an input data tensor, $\{\gamma_l\}$ are the hyper-parameters of the prior distributions, α is a hyper-parameter of a kernel function of GP (see equation (A.1)), and N is the number of sample used for the GP approximation. `PREPARE_GP` is described in appendix A.6.

Input: $\underline{\mathbf{X}}$, $\{\gamma_l\}$, α, N
  Estimate $\tilde{\omega}$
  Initialize $\mathbf{U}_1, \ldots, \mathbf{U}_L$ by `HOSVD`
  repeat
    $\mathbf{W} = \mathbf{U}_L \otimes \cdots \otimes \mathbf{U}_2 \otimes \mathbf{U}_1$
    // E-step
    $\vec{\mathbf{z}}_0 = \mathrm{argmax}\, \mathcal{L}(\vec{\mathbf{z}})$
    $\mathbf{\Psi}_0'' = \mathrm{diag}(\psi''(\mathbf{w}_1^\top \vec{\mathbf{z}}_0), \ldots, \psi''(\mathbf{w}_D^\top \vec{\mathbf{z}}_0))$
    $\mathbf{\Sigma}_0 = (\mathbf{W}^\top \mathbf{\Psi}_0'' \mathbf{W} + \mathbf{I})^{-1}$
    // M-step
    `PREPARE_GP`$(\underline{\mathbf{\Theta}}, N)$
    for $l = 1, \ldots, L$ **do**
      $\mathbf{U}_l = \mathrm{argmax}_{\mathbf{U}_l}\, \bar{\mathcal{L}}(\mathbf{U}_1, \ldots, \mathbf{U}_L)$
    end for
  until Convergence
  Return $\vec{\mathbf{z}}, \mathbf{U}_1, \ldots, \mathbf{U}_L$

3.4. Online learning Algorithm

In this section, we extend the batch algorithm to an online procedure which divides a data tensor into multiple slices and sequentially estimates the parameters. This extension allows real-time data processing and reduces computational cost at the expense of the accuracy of the parameter estimation.

3.4.1 Sequential update of $\mathbf{U}_l$

We assume that the l-th mode's dimension of the data tensor is considerably large, and cannot cast the batch algorithm. For such a tensor, we first slice the data tensor $\underline{\mathbf{X}}$ along l-th mode and divide $\underline{\mathbf{X}}$ into $D_1 \times \ldots D_{l-1} \times \tilde{D}_l \times D_{l+1} \times \cdots \times D_L$ tensor $\underline{\tilde{\mathbf{X}}}$ and $D_1 \times \ldots D_{l-1} \times (D_l - \tilde{D}_l) \times D_{l+1} \times \cdots \times D_L$ tensor $\underline{\hat{\mathbf{X}}}$. We choose $\tilde{D}_l (< D_l)$ as sufficiently small so that the batch algorithm can handle $\underline{\tilde{\mathbf{X}}}$.

After the estimation of the parameters $\underline{\mathbf{Z}}$, $\{\mathbf{U}_n | n \neq l\}$, and $\{\mathbf{u}_{l,i} | i = 1, \ldots, \tilde{D}_l\}$ with the divided tensor $\underline{\tilde{\mathbf{X}}}$ by the batch algorithm, we estimate the remaining $\mathbf{U}_l$ with

Algorithm 2 Online algorithm of Bayesian inference of ETF.

Input: $\underline{\tilde{\mathbf{X}}}$, $\underline{\hat{\mathbf{X}}}$, $\{\gamma_l\}$, α, l, N
  Estimate $\tilde{\omega}$
  initialize $\{\vec{\mathbf{z}}_0, \mathbf{U}_1, \ldots, \mathbf{U}_L\}$ by the batch algorithm of ETF
  $\mathbf{W} = \mathbf{U}_L \otimes \cdots \otimes \mathbf{U}_2 \otimes \mathbf{U}_1$
  $\mathbf{\Psi}_0'' = \mathrm{diag}(\psi''(\mathbf{w}_1^\top \vec{\mathbf{z}}_0), \ldots, \psi''(\mathbf{w}_D^\top \vec{\mathbf{z}}_0))$
  $\mathbf{\Lambda}_0 = \mathbf{W}^\top \mathbf{\Psi}_0'' \mathbf{W} + \mathbf{I}$
  $\mathbf{B} = \mathbf{U}_L \otimes \cdots \otimes \mathbf{U}_{l+1} \otimes \mathbf{U}_{l-1} \otimes \cdots \otimes \mathbf{U}_2 \otimes \mathbf{U}_1$
  for $i = 1, \ldots, D_l$ **do**
    $\mathbf{\Sigma}_i = (\mathbf{\Lambda}_{i-1})^{-1}$
    PREPARE_GP($\underline{\mathbf{\Theta}}, N$)
    $\mathbf{u}_{l,i} = \mathrm{argmax}_{\mathbf{u}}\, \bar{\ell}_i^{(l)}$
    // Update posterior covariance
    $\mathbf{W}_i = (\mathbf{u}_{l,i})^\top \otimes \mathbf{B}$
    $\mathbf{\Psi}_i'' = \mathrm{diag}(\psi''(\theta_{id}^{(l)}))$, $\boldsymbol{\theta}_i^{(l)} = \mathbf{W}_i \mathrm{vec}\, \mathbf{Z}_0^{(l)}$
    $\mathbf{\Lambda}_i = \mathbf{\Lambda}_{i-1} + \mathbf{W}_i^\top \mathbf{\Psi}_i'' \mathbf{W}_i$
  end for
  Return $\vec{\mathbf{z}}, \mathbf{U}_1, \ldots, \mathbf{U}_L$

$\underline{\hat{\mathbf{X}}}$. The marginal likelihood $\bar{\mathcal{L}}$ can be decomposed as

$$\bar{\mathcal{L}} = \sum_{i=1}^{D_l} \bar{\ell}_{l,i}(\mathbf{u}_{l,i}) + \text{const.}, \tag{3.15}$$

$$\bar{\ell}_{l,i}(\mathbf{u}) = \sum_{d \in \mathcal{I}_l} \{x_{id}^{(l)} \mathbb{E}_q[\theta_{id}^{(l)}] - \mathbb{E}_q[\psi_{id}^{(l)}]\} - \frac{1}{2} ||\mathbf{u}||^2 . \tag{3.16}$$

Since the i-th row vector $\mathbf{u}_{l,i}$ of $\mathbf{U}_l$, depends only on the observation $\mathbf{x}_i^{(l)}$, we can sequentially estimate $\mathbf{u}_{l,i}$ for $i = 1, \ldots, D_l$. We simultaneously update the posterior covariance $\mathbf{\Sigma}$ as well as $\mathbf{u}_{l,i}$, since the Hessian of the posterior distribution varies with $\mathbf{u}_{l,i}$ even if $\vec{\mathbf{z}}$ is fixed (see equation (3.8)). The maximization of $\bar{\ell}_{l,i}$ and the update of $\mathbf{\Sigma}$ correspond to the M- and the E-steps of the batch algorithm, respectively. We summarize this algorithm in Figure 2.

An important advantage of the online algorithm is that the computational cost is much lower than the batch algorithm. Since it does not need alternating updates anymore, the convergence speed is considerably fast. Another advantage is the online possessing for time-series data. The algorithm enables us to estimate $\mathbf{u}_{l,i}$ in real-time when $\mathbf{x}_i^{(l)}$ is observed every i-th time. However, since we do not use the entire information of the observation in the online algorithm, prediction performance for missing values would be worse especially when the feature space of new observation $\mathbf{x}_i^{(l)}$ is substantially different from that of $\underline{\tilde{\mathbf{X}}}$. We investigate the performance of the online algorithm in a case at Section 3.7.1.

3.4.2 Computational complexity

Although it is difficult to compare the computational complexity, the online algorithm are significantly faster than the batch algorithm. Since the online algorithm has only one variable $\mathbf{U}_l$ to maximize, we no longer need to take a coordinate descent approach used in the batch algorithm; it dramatically reduces the computational cost for convergence. We will empirically show how the computational cost is improved in Section 3.7.2.

Note that, in the case of $K > D_{\backslash l}$, we efficiently update the posterior covariance $\mathbf{\Sigma}_i$ by using the matrix inversion lemma as

$$\begin{aligned}\mathbf{\Sigma}_i &= (\mathbf{\Sigma}_{i-1}^{-1} + \mathbf{W}_i^\top \mathbf{\Psi}_i'' \mathbf{W}_i)^{-1} \\ &= \mathbf{\Sigma}_{i-1} - \mathbf{\Sigma}_{i-1}\mathbf{W}_i^\top (\mathbf{\Psi}_i'' + \mathbf{W}_i \mathbf{\Sigma}_{i-1} \mathbf{W}_i^\top)^{-1} \mathbf{W}_i \mathbf{\Sigma}_{i-1}.\end{aligned}$$

It reduces the computational complexity from $\mathcal{O}(K^3)$ to $\mathcal{O}({D_{\backslash l}}^3)$.

3.5. Applications

After the convergence of the algorithm, we have estimated parameters $\vec{\mathbf{z}}$ and $\{\mathbf{U}_l\}$ under the observed entries $\vec{\mathbf{x}}_{\mathcal{I}}$. We use these parameters for mainly two purposes: missing-values prediction and anomaly detection. Given natural parameter $\vec{\boldsymbol{\theta}}$, we predict a missing element $\vec{x}_d, d \notin \mathcal{I}$ by the mean of the Bayesian predictive distribution $\mathbb{E}[\vec{x}_d \mid \vec{\mathbf{x}}_{\mathcal{I}}]$. The factor matrix $\mathbf{U}_l$ is used to find anomalies which lies on the l-th mode of the data.

3.5.1 Missing-values prediction

The marginalization of $\vec{\mathbf{z}}$ by the posterior is intractable, which is required in the computation of Bayesian predictive distribution. Instead, we approximate the nonlinear function $\psi'(\theta)$ by a first-order derivative of the GP's predictive mean m', and we obtain the approximated predictive mean given by

$$\begin{aligned}\mathbb{E}[\vec{x}_d \mid \vec{\mathbf{x}}_{\mathcal{I}}] &= \int \vec{x}_d \int p(\vec{x}_d \mid \vec{\mathbf{z}}, \mathbf{w}_d) p(\vec{\mathbf{z}} \mid \vec{\mathbf{x}}_{\mathcal{I}}) \mathrm{d}\vec{\mathbf{z}} \mathrm{d}\vec{x}_d \\ &= \int \psi'_{h(d)}(\vec{\theta}_d) p(\vec{\mathbf{z}} \mid \vec{\mathbf{x}}_{\mathcal{I}}) \mathrm{d}\vec{\mathbf{z}} \;\simeq\; \mathbb{E}_q[m'_{h(d)}(\vec{\theta}_d)]. \end{aligned} \tag{3.17}$$

For the transformation of second line, we use the relation described in equation (3.3). Theorem 1 gives the analytical form of the expectation $\mathbb{E}_q[m'_{h(d)}(\vec{\theta}_d)]$ in equation (A.33), which is the case of $p = 1$ and $q = 0$. We can also obtain the variance ($p = 2$) or any other higher-order moments ($p \geq 3$) of the predictive distribution of ETF by using theorem 1.

3.5.2 Anomaly detection

As described in Section 2.1, the factor matrix $\mathbf{U}_l$ is a low-dimensional feature of the l-th mode of the observation tensor. If we estimate parameters from the data tensor

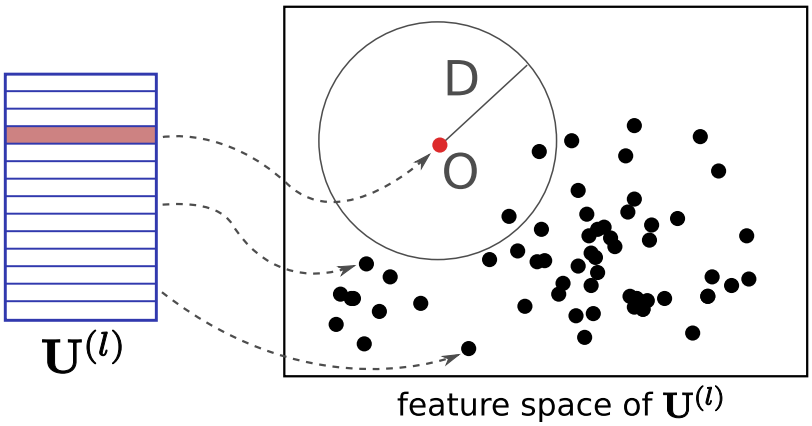

Figure 3.2: Anomaly detection as distance-based outlier detection on the estimated parameter space.

which contains anomalous values, then the corresponding parts of the factor matrices could be captured as *outliers* compared to other regular parts. By using the factor matrix as inputs of outlier detection, we can find intrinsic anomalies without the influence by the observation noise.

There are several methods for discovering outliers. Here we employ distance-based outlier proposed by Knorr et al. (2000).

Definition 2 *an object O in a data set T is a $DB(p, r)$ outlier if at least fraction p of the objects in T lies greater than distance r from O.*

Figure 3.2 illustrates the idea of distance-based outlier. If we set $p = 0.995$ and we have $D_l = 1000$, i.e. the number of point in Figure 3.2 is 1000, the point O is detected as an outlier when the hypersphere centered in O with radius r contains at most 5 other points. Note that in the Tucker decomposition, the scale among $\{\mathbf{U}_l\}$ and $\underline{\mathbf{Z}}$ is ill-posed, i.e., $\alpha\mathbf{U}_l$ and $\frac{1}{\alpha}\underline{\mathbf{Z}}$ produces same $\underline{\mathbf{\Theta}}$. Thus we need to normalize $\mathbf{U}_l$ in column-wise before applying the distanced-based outlier. For example, if we employ Euclidean distance, it becomes cosine similarity due to the normalization.

Note that we are only interested in the anomalies, of e.g., extra-ordinal nodes in a multiple network. In other words, we don't have interests for anomalies of individual elements of data tensor but for distinctive dimensions of the mode.

3.6. Related Works

Recently, the tensor factorization methods are widely applied to data mining problems (Mørup, 2011). PARAFAC (Harshman, 1970) is a tensor factorization method that can be seen as a special case of the Tucker method whose core tensor shares the number of dimensions ($K_1 = K_2 = \cdots = K_L$) and is restricted to diagonal.

PARAFAC is therefore more suitable for feature extraction and is applied to chemoinfomatics (Kolda and Bader, 2009). There are several algorithms for solving the Tucker decomposition such as HOSVD, higher-order orthogonal iteration (HOOI) (De Lathauwer et al., 2000a) which is an iterative extension of HOSVD, and alternating gradient methods. Since the objective function of the Tucker decomposition is non-convex, the solution obtained by these algorithms is not guaranteed to converge to a global optimum. Recently, several works that formulate the Tucker decomposition as a convex optimization problem are proposed (Liu et al., 2009; Tomioka et al., 2011); They handles a fully parametrized tensor whose dimensions are same as a data tensor with per-mode trace-norm regularization for the unfolding of the parameter tensors. Sun et al. (2006) propose an efficient online learning algorithm which solves a special case of the Tucker decomposition called Tucker 2.

Bayesian extensions of tensor factorization methods are also well studied. Chu and Ghahramani (2009) proposed a probabilistic extension of the Tucker method, known as pTucker. The model of pTucker can be seen as a special case of ETF whose attributes are specified solely by Gaussian distribution. Instead of a MAP estimate, pTucker marginalizes the core $\underline{\mathbf{Z}}$ out and estimates a expectation of the marginal posterior of $\{\mathbf{U}_l\}$. The solution of pTucker will be robust than the non-Bayesian tucker decomposition. In ETF, the marginalization of $\underline{\mathbf{Z}}$ is basically intractable due to the assumption of the heterogeneity. Even though the attributes are all Gaussian, we need to evaluate the determinant of the Hessian and take its gradient for every iteration, that is computationally demanding for the online optimization approach. (Mørup and Hansen, 2009) proposed a model that can be seen as a sparse extension of pTucker. Instead of marginalizing out the core tensor, the automatic relevance determination (ARD) priors are introduced to the core and the factor matrices. The ARD prior forces as the parameters to be sparse, and it is helpful for interpretation of data structure. Shashua and Hazan (2005) studied the non-negative PARAFAC with the latent variable. The parameter was inferred by the EM algorithm. A Bayesian extension of PARAFAC for time-series data is also studied (Xiong et al., 2010) .

In several matrix factorization studies, the non-Gaussian observation has been dealt with. Collins et al. (2002) proposed exponential family PCA (EPCA), which generalizes the likelihood of probabilistic PCA to the exponential family with no prior for the latent variable. A fully Bayesian extension of EPCA with Markov chain Monte Carlo was also proposed (Mohamed et al., 2009). Wedel and Kamakura (2001) proposed a similar model that also generalized the prior distribution of latent variable to the exponential family. Unlike EPCA, the study focused on the handling of the heterogeneous attributes. Similar approach are proposed in (Ma**v**zgut et al., 2010) in the context of multilinear PCA. Collective matrix factorization (Singh and Gordon, 2008) aims to improve the accuracy of the prediction by borrowing information across the multiple matrices (not tensor) with different sizes. The loss function is equivalent to the log-likelihood of the exponential-family distributions. Several studies on tensor factorization have approached the data fusion problem, using the Tucker de-

composition (Liu et al., 2010) or PARAFAC (Dunlavy et al., 2006) for homogeneous data.

There are some works tackling large-scale optimization problem of tensor modelings. For large-scale data tensors, Leskovec and Faloutsos (2007) proposed a fast and memory efficient algorithm of PARAFAC for a sparse tensor. This work empirically shows that a small number of elements are enough for reconstruction of the data tensor. It can compute a $1000 \times 1000 \times 1000$ data tensor in $\sim 10,000$ seconds with eliminating 99.5% of elements as missing. Pairwise interaction tensor factorization (PITF) (Rendle and Thieme, 2010) is another tensor factorization method for tag recommendation problem implemented by the stochastic gradient. Like PARAFAC, the core tensor of PITF is restricted to diagonal that improves the computational time. Stochastic gradient descent (Bottou and LeCun, 2004; Koren et al., 2009) would be also helpful to optimize large-scale tensor factorization.

3.7. Experiments

In this section, first we investigate the validity of the proposed approximation scheme and the learning algorithm by using synthetic data. Then we evaluate the applicability of our methods for missing-values prediction and anomaly detection in multiple sensor measurements.

As the accuracy measure for missing-values prediction, we used the root mean square error (RMSE), the mean absolute error (MAE), and the area under the ROC curve (AUC) for Gaussian-, Poisson-, Bernoulli-attributes, respectively. Lower RMSE/MAE and higher AUC imply better results. We also used the AUC for the evaluation of anomaly detection.

For comparison, we prepared PARAFAC, the Tucker decomposition, and ETF whose attributes were assumed all Gaussian, which corresponds to pTucker (Chu and Ghahramani, 2009) with the EM algorithm. We used the N-way Toolbox for Matlab (Andersson and Bro, 2000) for implementation of PARAFAC and the Tucker decomposition with orthogonal constraints to factor matrices. All experiments were done with Xeon 2.93 GHz 8 core machines.

3.7.1 Synthetic Data

To confirm validity of our learning algorithm, we generated synthetic data sets by following the generative model of ETF. Firstly we compare approximation schemes in estimation accuracy and computational time. Secondly we show that how the size of $\underline{\tilde{\mathbf{X}}}$ affects the accuracy of parameter estimation in the online algorithm.

Comparison of approximations

We considered $9 \times 9 \times 9$ observation tensor $\underline{\mathbf{X}}$. The true values of parameters $\underline{\mathbf{Z}}$, $\mathbf{U}_1$, $\mathbf{U}_2$, and $\mathbf{U}_3$ were randomly drawn from their prior distribution with $\gamma_l = 3$. The

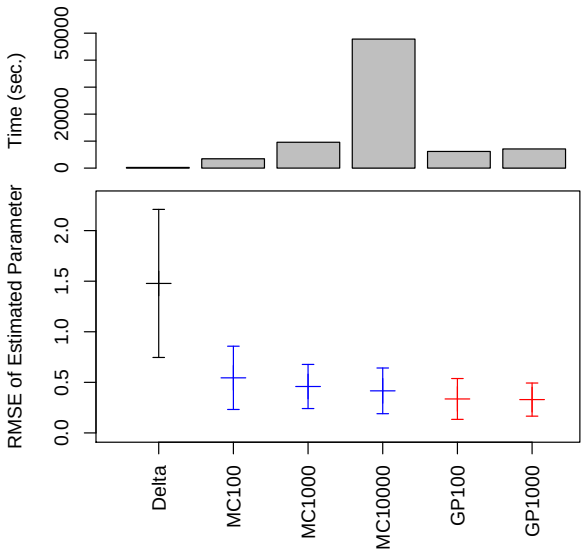

Figure 3.3: The results of the parameter estimation for the synthetic data set by using ETF with the zeroth-order delta method (Delta), MC, and GP approximation. Each error bar shows the mean and standard deviation of RMSEs over 50 trials. The bar chart indicates the mean of the computational cost (CPU time (sec)).

ranks of the core tensor (K_1, K_2, K_3) were set to $(3, 3, 3)$. We assumed that data elements $\{x_{ijk} \mid j, k = 1, \ldots, 9\}$ were distributed by Gaussian for $i = 1, 2, 3$, Poisson for $i = 4, 5, 6$, and Bernoulli for $i = 7, 8, 9$, respectively. Then we randomly divided $\underline{\mathbf{X}}$ into a training set (50%) and a testing set (50%) for the missing-values prediction task. We generated 20 missing patterns with different random seeds and examined errors of the estimation over the 20 trials. We compared the proposed approximation by the GP with the zeroth delta method (Delta) discussed in Section 3.3 and a simple Monte Carlo (MC) approach for the approximation of the expectation. Simple Monte Carlo approximates an expectation of an arbitrary function f as $\mathbb{E}[f(\theta)] \approx \frac{1}{N}\sum_{n=1}^{N} f(\theta_n)$ where the samples $\{\theta_n\}$ are drawn from the Gaussian-approximated posterior distribution $q(\vec{\boldsymbol{\theta}})$ (3.14). We set the sample size N to $100, 1000, 10000$ for MC and $100, 1000$ for our method, respectively. Note that the approximation errors of both MC and GP converge to 0 when $N \to \infty$. The dimensions of the core tensor $\{K_l\}$ and the hyper-parameter $\{\gamma_l\}$ were set to the true value.

The RMSEs between the estimated and true natural parameters are shown in Figure 3.3. The result shows that the delta method saved the computational time, while the estimation error was the worst. Additionally, we observed that the GP approximation method with $N = 100$ samples was more accurate than MC with

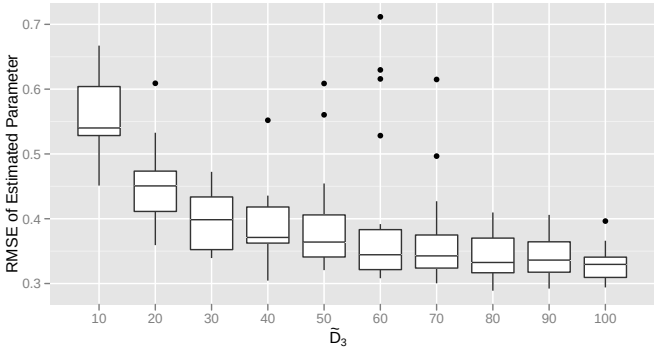

Figure 3.4: The results of the parameter estimation for the synthetic data set by using the online algorithm of ETF.

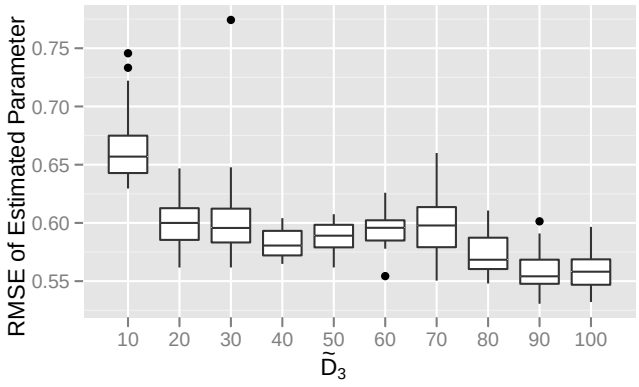

Figure 3.5: The result of a counter example.

10000 samples, and also the computational time (indicated by the bar chart) of GP was roughly 8 times faster than MC with 10000 samples.

Batch vs. online

We compared the batch algorithm (Algorithm 1) and the online algorithm (Algorithm 2) with the setting of the previous experiment except that the dimensions of the data tensor were given $9 \times 9 \times 100$. We used the GP approximation method with $N = 200$.

Figure 3.4 shows the RMSEs between the estimated and true natural parameters $\vec{\boldsymbol{\theta}}$. The horizontal axis indicates the value of $\tilde{D}_3$, the $\underline{\tilde{\mathbf{X}}}$'s dimension of the third mode, i.e., the values at $\tilde{D}_3 = 100$ shows the result of the batch algorithm. The result shows the online algorithm worked better when $\tilde{D}_3$ was sufficiently large.

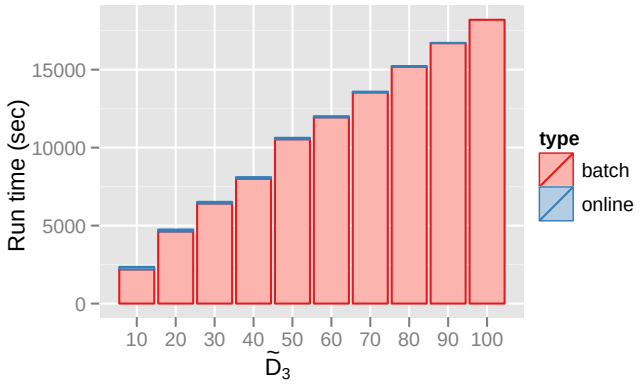

Figure 3.6: Comparison of computational times of the batch and the online part.

We also performed an experiment in an undesirable situation for the online algorithm. With the same parameter settings, we generated two $9 \times 9 \times 50$ tensors $\underline{\mathbf{X}}_1$ and $\underline{\mathbf{X}}_2$ with different random seeds, and we combined the two tensors into a $9 \times 9 \times 100$ tensor $\underline{\mathbf{X}}$. The result shown in Figure 3.5 indicates the errors were no longer monotonically decreasing as Figure 3.4, since the basis was not consistent in the entire data tensor.

We also compared the computational time which is shown in Figure 3.6. Each bar shows the total computational time; the red parts indicate the computational time for initial prepossessing with ETF batch and the blue parts indicate the remaining online procedure. The figure shows the computational time of the online procedure was mostly ignorable compared to the batch one.

3.7.2 Office-logging data

In this experiment, we used six temporal sequences of different types of sensor measurements. The sensors had measured various behaviors of researchers, such as Email sending/receiving, frequency of typing keyboard, and geometrical location, recorded in C&C Innovation Research Laboratories (CCIL), NEC Corporation. This data set has heterogeneity and is tensor structured. Each sequence contains the measurement for 20 members, recorded in daily working hours (9 a.m. – 5 p.m.) We aggregated the sequence of the measurements per hour and we got matrices $\mathbf{X}_1, \ldots, \mathbf{X}_6$ for each measurement; the (i, j)-th element of $\mathbf{X}_1$ contains the number of Emails that researcher i sent during a j-th time period. We summarize the data matrices in Table 3.2. We assumed that the data sets $\{\mathbf{X}_i\}$ were distributed by Poisson for $i = 1, 2, 3, 6$, and Gaussian for $i = 4, 5$, respectively.

Table 3.2: Description of each measurement of the office-logging data. *Min* and *Max* are minimum and maximum values of sensor measurements, respectively. The unit of $\mathbf{X}_4$, $\mathbf{X}_5$, and $\mathbf{X}_6$ is a centimeter.

	Name	Measurement	Type	Min	Max
$\mathbf{X}_1$	`email_send`	# of sent emails	Count	0.00	14.00
$\mathbf{X}_2$	`email_recv`	# of received emails	Count	0.00	15.00
$\mathbf{X}_3$	`type_freq`	# of typed keys on a PC	Count	0.00	50422.00
$\mathbf{X}_4$	`smloc_X`	X coordinate	Real	−550.17	3444.15
$\mathbf{X}_5$	`smloc_Y`	Y coordinate	Real	128.71	2353.55
$\mathbf{X}_6$	`mtv`	Movement distance	Non-negative	0.00	203136.96

Missing-values prediction

First we evaluated the performance of missing-value prediction. The aim here is to investigate the validity of our model assumption of the heterogeneity for the sensor measurements. Since the batch algorithm of ETF cannot handle the large data set due to the memory overflow, we used a one-month office-logging datarecorded from September 30 to December 28, 2009 (that is a $20 \times 235 \times 6$ tensor.) We used the GP approximation for ETF with $N = 200$ and $\gamma_1 = \gamma_2 = \gamma_3 = 1$. The rank, i.e., the dimensions of the core tensor, was prepared from $\{2 \times 2 \times 2, \ldots, 6 \times 6 \times 6, 6 \times 7 \times 7\}$. We randomly picked up 50% from observed elements of $\underline{\mathbf{X}}$ as a training and another 50% of that as a testing set. We generated 10 missing patterns with different random seeds and examined errors of the estimation over the 10 trials.

Test errors of missing-values prediction are shown in Figure 3.7. Overall, ETF and pTucker, the Bayesian methods, outperformed PARAFAC and Tucker for all sensor measurements. ETF outperformed or was comparable to pTucker in which the ranks were greater than $4 \times 4 \times 4$. We also performed a same experiment of missing-values prediction with different time periods (from June 30 to July 30, 2010), which yielded a similar consequence, shown in Figure 3.8, to the first data set.

Anomaly detection

Next we investigated an efficiency of anomaly detection. The aim here is to find unexpected events, which are distinguishable from routine works in CCIL, with the researchers' behavior measured by the sensors. The anomaly events were rare to happen, and we used long-term data set recorded from September 30, 2009 to July 31, 2010 ($20 \times 1927 \times 6$ tensor) to collect adequate amount of anomalies. NEC CCIL provided 34 irregular events in that period, which is listed in Table 3.3; we defined these 34 out of the total 1927 periods as anomalies in this experiment.

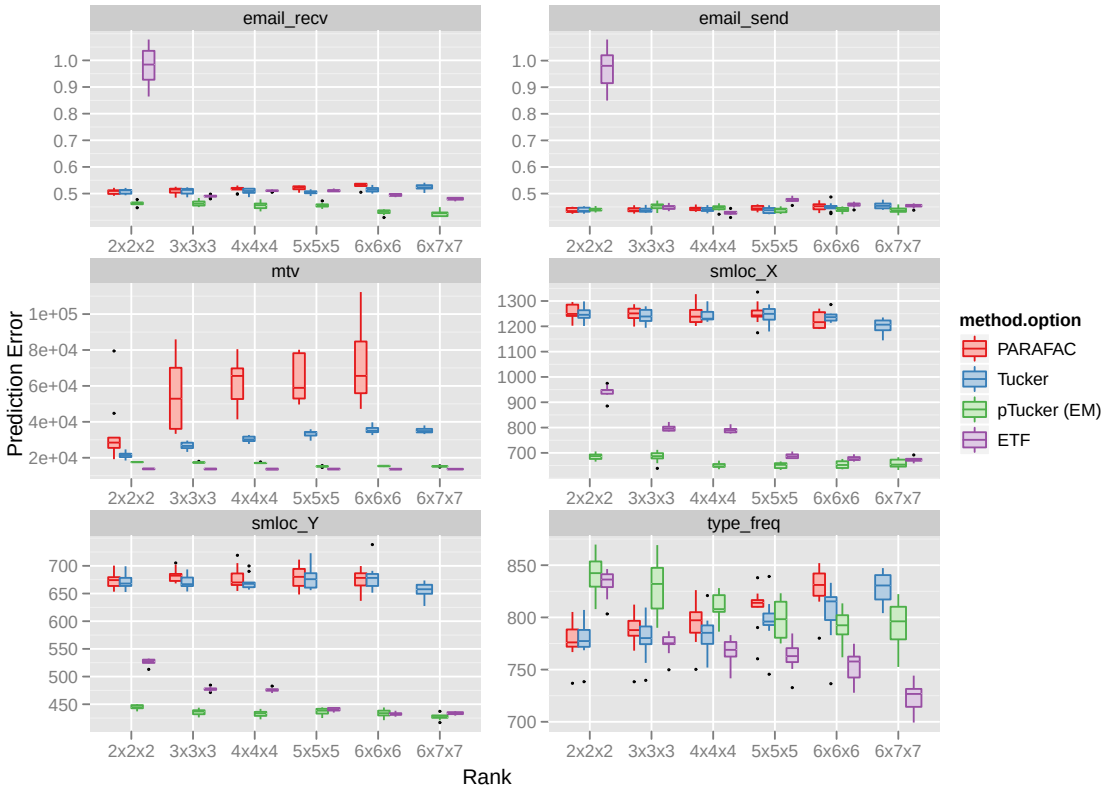

Figure 3.7: The results of the missing-values prediction for the office logging data (from September 30 to December 28, 2009).

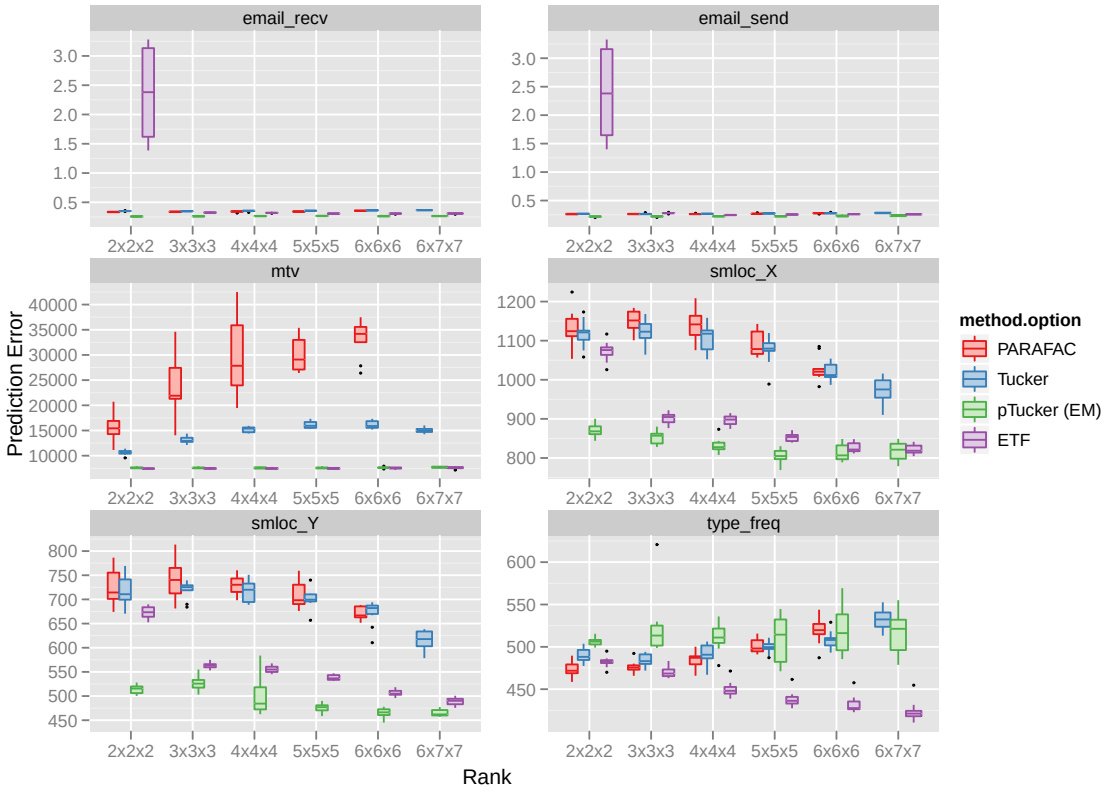

Figure 3.8: Another results of the missing-values prediction (from June 30 to July 30, 2010).

Table 3.3: List of irregular events of CCIL from September 30, 2009 to July 31, 2010.

Date	Time	Description
Dec 21	All day	Private incident
Dec 22	15:00 16:00	Monthly seminar
Jun 15	13:00 14:00	Visiting tour
Feb 23	15:00 16:00	Monthly seminar
Feb 26	14:00 15:00	Monthly meeting
Mar 12	15:00 16:00	Monthly seminar
Apr 2	10:00	Monthly meeting
Apr 16	16:00	Workshop
Apr 28	14:00	Monthly seminar
May 13	10:00	Visiting tour
May 28	10:00	Monthly meeting
May 31	10:00	Visiting tour
Jun 3	13:00 14:00	Mid-term meeting
Jun 11	14:00 15:00	Visiting tour
Jun 18	15:00	Visiting tour
Jun 25	10:00	Monthly meeting
Jul 09	16:00	Visiting tour
Jul 12	9:00	Stocktaking
Jul 22	10:00	Monthly meeting

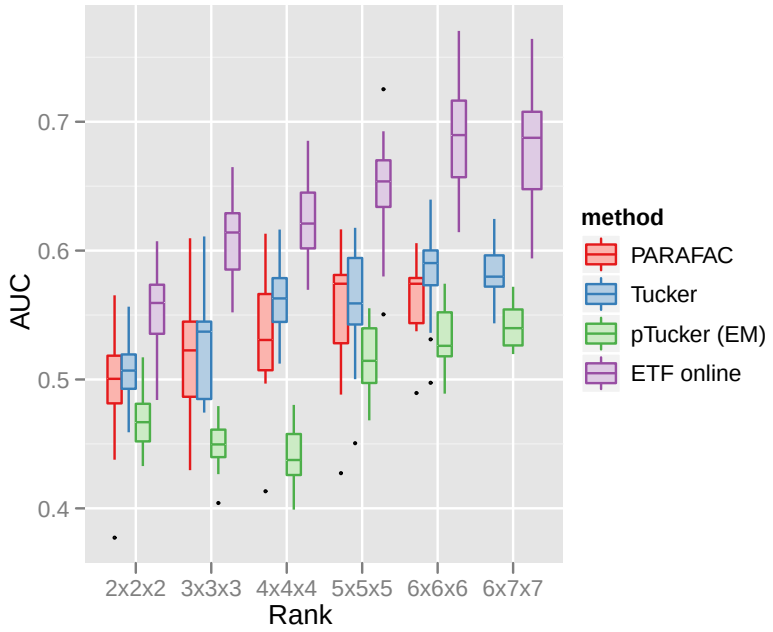

Figure 3.9: The results of the anomaly detection for the long-term office logging data.

First we evaluated area under the ROC curve (AUC) score. By using $DB(p, r)$ outlier detection scheme, we computed a maximum r under setting $p = 0.995$ for each row of a factor matrix. We used r as a degree of outlier; if the value r of $\mathbf{u}_{3,i}$ (i-th row of $\mathbf{U}_3$) was large then an irregular event would happen in i-th time period. We evaluated the performance of the anomaly detection by AUC score with respect to the r values for each event. AUC ideally takes its value in 0.5 to 1 and a higher AUC score means a better result.

The AUC score of each method is shown in Figure 3.9. We find that the accuracy of the anomaly detection by ETF was distinctively higher than the other methods in every rank. Figure 3.10 shows ROC curves with various settings of DB outlier's p values. For each method, we chose a rank which archives the best performance in AUC; $(4, 4, 4)$ for PARAFAC, $(5, 5, 5)$ for Tucker, $(6, 7, 7)$ for pTucker, and $(6, 6, 6)$ for ETF. The figure clearly shows ETF was totally accurate to detect anomalous events compared to PARAFAC, Tucker, or pTucker. This was because ETF naturally extracted the regular parts of the data under the appropriate assumption of exponential-family distributions; The other methods would fail to capture intrinsic features due to the assumption of Gaussian noise. The result also shows the choice of p was not sensitive for $p \geq 0.95$. We summarize computational times for each method in Figure 3.11. Although the run time of pTucker was exponentially growing, our online algorithm

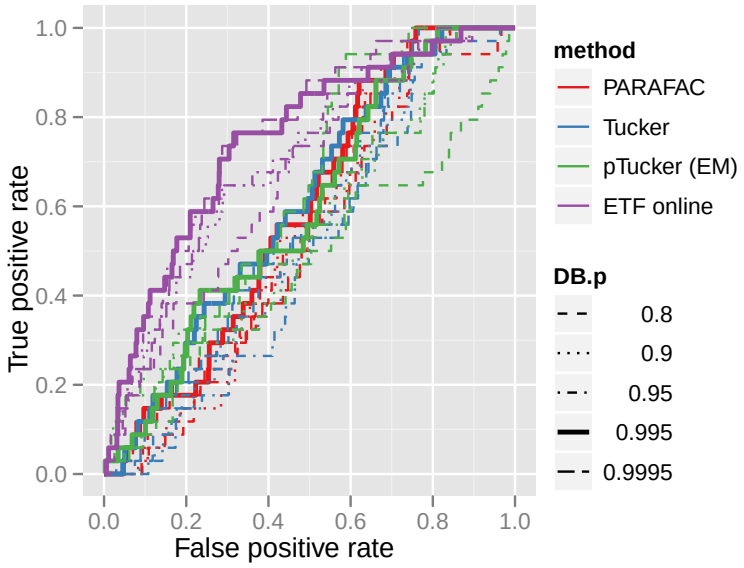

Figure 3.10: ROC curves for the anomaly detection.

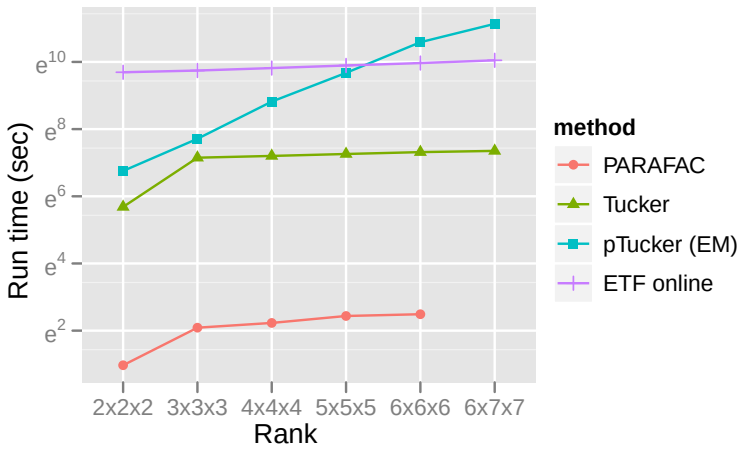

Figure 3.11: Computational times of the long-term office logging data. Note that the vertical axis takes logarithmic scale.

scaled well for larger rank. Actually, the computational time of the online algorithm was less than one minute per period; it was applicable for real-time computing.

3.8. Summary

In this chapter, we have proposed a procedure of exponential family tensor factorization for integrating heterogeneously attributed array data. We have employed the EM algorithm for Bayesian inference of the parameters with the GP approximation scheme. The online version of the proposed algorithm has allowed us to deal with real-time tracking of a large but slender data tensor. The experimental results have showed that the method has appropriately captured the heterogeneous data tensor and has been applicable to multiple sensor measurements for missing-values prediction and anomaly detection.

Chapter 4

Tensor Completion without Low-rank Assumption

4.1. Introduction

Data completion of a partially observed tensor has recently emerged with lots of applications. A particular example is recommender systems. In these situations, we have an $R \times C$ matrix $\mathbf{X}$[1] that represents relationships of *user* $\times$ *item* in which the (i, k)-th element x_{ik} contains i-user's preference such as a rating of k-th item. Normally we observe only $N \ll RC$ elements of $\mathbf{X}$. By predicting unobserved elements from the observations, we can effectively recommend new items for users. In some cases, side information such as users' age and items' price are simultaneously provided, that will improve prediction performance. If we observe time-dependent information, the problem takes a form of an array or *tensor* completion of a *user* $\times$ *item* $\times$ *time* data tensor.

Matrix factorization methods are generally used for matrix completion and popular techniques for recommender systems (Koren et al., 2009). Matrix factorization assumes the underlying (true) matrix of $\mathbf{X}$ is low-rank and estimate $\mathbf{X}$ by a low-rank approximation $\mathbf{X} \simeq \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are the low-dimensional latent features of users and items, respectively. For a data tensor $\underline{\mathbf{X}}$, tensor factorization such as the Tucker decomposition is used.

Kernel-based approaches such as Gaussian process (GP) models for collaborative filtering have recently been developed. In the kernel methods, relationships among preferences are represented as a Gram matrix (a covariance matrix in the GP models) determined by kernel functions in which the measurements are assumed to be externally given (e.g., side information). Yu et al. (2007) proposed a concept of the *tensor GP*, in which the covariances of user preferences are factorized into user-wise and item-wise covariances.

1. As mentioned, a matrix is a special case of a tensor.

Although both methods are based on different mechanisms, we clearly understand the common characteristic through the perspective of *similarities*. Tensor factorization can be seen as a framework that learns similarities of users and items as the parameters. While tensor factorization can flexibly determine the similarities from observed elements, the low-rank constraint for the similarity matrices is necessary to avoid overfitting and reduce computational complexity. In the GP-based models, the covariance matrix naturally represents similarities of users and items. Since the similarity measurements are assumed to be externally given, they potentially handle the full-rank similarity (covariance) matrix without any constraint. However, the exact inference of the GP for large-scale data sets is computationally infeasible; it requires $\mathcal{O}(N^3)$ computational cost. Low-rank approximations of the covariance matrix such as Nyström approximation are widely used in practice (Drineas and Mahoney, 2005).

In this chapter, we introduce a new framework for partially observed tensor completion problems based on the idea of the tensor GP. We employ *self-measuring* similarities in which the measurements are the elements of $\mathbf{X}$ themselves. The missing values are embedded in the Gram matrix as latent variables, which involves a form of kernel learning. Side information can be exploited but are not indispensable in our framework. We develop an efficient algorithm for the exact inference for prediction with $\mathcal{O}(\prod_{l=1}^{L} D_l \sum_{q=1}^{L} D_q)$ computational cost for a tensor $\mathbf{X} \in \mathbb{R}^{D_1,\ldots,D_L}$. Required memory space is $\mathcal{O}(N)$. Our algorithm enables us to deal with similarity (covariance) matrices as full-rank, and thus our method is applicable even if the underlying matrix or tensor is full-rank. We also show that our framework includes K-nearest neighbor, matrix and tensor factorization, and GP-based methods as special cases. Our method is evaluated in the standard collaborative filtering problem and show that it attains the lowest prediction error in the data set. The feasibility for multi-variate data analysis (i.e., data tensors) is also explored.

4.2. Model

In this section, first we introduce our model for a data matrix. Next, we show its dual representation with a kernel function related to GP models. Further generalizations including tensorization of the model are also discussed.

4.2.1 Pairwise linear regression

We have a partially observed $R \times C$ data matrix $\mathbf{X}$ with row-specific features $\boldsymbol{\phi}_i \in \mathbb{R}^A$ for $i = 1, \ldots, R$ and column-specific features $\boldsymbol{\psi}_k \in \mathbb{R}^B$ for $k = 1, \ldots, C$. We assume that all the elements of $\mathbf{X}$ is centered at 0, which can be achieved by subtracting the empirical mean $\frac{1}{N}\sum_{(i,k)\in\mathcal{I}} x_{ik}$ from $\mathbf{X}$, where $\mathcal{I}$ is an index set of the observed elements, and we have $N = |\mathcal{I}|$ observations $\{x_{ik}|(i,k) \in \mathcal{I}\}$. We also denote an N-dimensional vector which contains the observed elements of $\mathbf{X}$ in a certain order without overlapping by $\vec{\mathbf{x}}_{\mathcal{I}}$. For later convenience, we introduce an observation matrix $\mathbf{P} \in \{0,1\}^{N\times RC}$ that removes the unobserved elements, i.e., $\vec{\mathbf{x}}_{\mathcal{I}} = \mathbf{P}(\mathrm{vec}\ \mathbf{X})$.

We employ the idea of supervised learning approaches to missing-values prediction. Suppose that each element x_{ik} is a label and a feature pair $(\boldsymbol{\phi}_i, \boldsymbol{\psi}_k)$ is a corresponding input. Thus our goal is to predict the missing values $\{x_{jl}|(j,l) \notin \mathcal{I}\}$ via a supervised learning model with a training data $\mathcal{D} = \{(x_{ik}, \{\boldsymbol{\phi}_i, \boldsymbol{\psi}_k\})|(i,k) \in \mathcal{I}\}$. We model $\mathbf{X}$ as a pairwise regression form

$$x_{ik} = f(\boldsymbol{\phi}_i, \boldsymbol{\psi}_k) + \varepsilon_{ik}, \quad \varepsilon_{ik} \sim N(0, \sigma^2), \tag{4.1}$$

where ε_{ik} is an i.i.d. Gaussian observation noise and f is a bilinear mapping with a weight matrix $\mathbf{W}$[2]:

$$f(\boldsymbol{\phi}, \boldsymbol{\psi}) = \sum_{a=1}^{A} \sum_{b=1}^{B} w_{ab} \phi_a \psi_b = \boldsymbol{\phi}^\top \mathbf{W} \boldsymbol{\psi}. \tag{4.2}$$

In the bilinear form, $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are fully interacted through $\mathbf{W}$, i.e., there are individual free parameters $\{w_{ab}|a = 1, \ldots, A, b = 1, \ldots, B\}$ against to all the combination of feature pairs $\{\phi_1, \ldots, \phi_A\} \times \{\psi_1, \ldots, \psi_B\}$. The bilinear form can be rewritten as a standard linear model:

$$f(\boldsymbol{\phi}_i, \boldsymbol{\psi}_k) = \vec{\mathbf{w}}^\top (\boldsymbol{\psi}_k \otimes \boldsymbol{\phi}_i) = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik} \tag{4.3}$$

where $\vec{\mathbf{w}} = \text{vec } \mathbf{W} \in \mathbb{R}^{AB}$ is a vectorization of $\mathbf{W}$ and $\otimes$ denotes the Kronecker product. To fit the model to $\mathbf{X}$, we estimate a MAP solution of $\vec{\mathbf{w}}$ with a standard Gaussian prior $\vec{\mathbf{w}} \sim N(\mathbf{0}, \mathbf{I})$, which corresponds to a least square solution with a quadratic regularization

$$\begin{aligned} \hat{\vec{\mathbf{w}}} &= \operatorname*{argmin}_{\vec{\mathbf{w}}} J(\vec{\mathbf{w}}), \\ J(\vec{\mathbf{w}}) &\equiv \sum_{(i,k) \in \mathcal{I}} \left|\left| x_{ik} - \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik} \right|\right|^2 + \sigma^2 \left|\left| \vec{\mathbf{w}} \right|\right|^2. \end{aligned} \tag{4.4}$$

Note that the loss function (4.4) is the negative joint log-likelihood of our model.

4.2.2 Kernel representation and self-measuring similarity

Suppose that we have the labels $\mathbf{X}$ but not the inputs $\{\boldsymbol{\phi}_i\}$ and $\{\boldsymbol{\psi}_k\}$, that cannot handle by the supervised framework any more. Instead, we create the feature vectors from $\mathbf{X}$ itself. We construct $\boldsymbol{\xi}_{ik}$, the feature of x_{ik}, from the i-th row vector $\mathbf{x}_{i:}$ and the k-th column vector $\mathbf{x}_{:k}$. Moreover, we restrict the form of $\boldsymbol{\xi}_{ik}$ that can be factorized into the features of row and column as same as Equation (4.3). Then we have a pairwise feature function $\boldsymbol{\xi}$:

$$\boldsymbol{\xi}(\mathbf{x}_{i:}, \mathbf{x}_{:k}) = \boldsymbol{\psi}(\mathbf{x}_{:k}) \otimes \boldsymbol{\phi}(\mathbf{x}_{i:}) \tag{4.5}$$

2. Here we use $\mathbf{W}$ as a different variable from that used in Chapter 2 and 3

where $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are (nonlinear) feature functions of the row and the column, respectively.

The representer theorem guarantees that the solution to (4.4) can be represented by a linear sum of a kernel function. By substituting $\boldsymbol{\xi}_{ik} = \boldsymbol{\xi}(\mathbf{x}_{i:}, \mathbf{x}_{:k})$, the dual representation of the model (4.3) is given by

$$\sum_{(j,l)\in\mathcal{I}} \beta_{jl} k\Big((\mathbf{x}_{i:}, \mathbf{x}_{:k}), (\mathbf{x}_{j:}, \mathbf{x}_{:l})\Big) = \boldsymbol{\beta}^\top \mathbf{k}_{ik} \tag{4.6}$$

where $k(\cdot,\cdot)$ is a positive semi-definite (PSD) kernel function defined by

$$\begin{aligned} k\Big((\mathbf{x}_{i:}, \mathbf{x}_{:k}), (\mathbf{x}_{j:}, \mathbf{x}_{:l})\Big) &= \langle \boldsymbol{\xi}(\mathbf{x}_{i:}, \mathbf{x}_{:k}), \boldsymbol{\xi}(\mathbf{x}_{j:}, \mathbf{x}_{:l})\rangle \\ &= \langle \boldsymbol{\phi}(\mathbf{x}_{i:}), \boldsymbol{\phi}(\mathbf{x}_{j:})\rangle \langle \boldsymbol{\psi}(\mathbf{x}_{:k}), \boldsymbol{\psi}(\mathbf{x}_{:l})\rangle. \end{aligned}$$

Note that the pairwise kernel $k\left((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\right)$, called *Kronecker kernel* (Basilico and Hofmann, 2004; Kashima et al., 2009b), is factorized into a product of the row-specific kernel $\Sigma(\mathbf{x},\mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x})\rangle$ and the column-specific kernel $\Omega(\mathbf{y},\mathbf{y}') = \langle \boldsymbol{\psi}(\mathbf{y}), \boldsymbol{\psi}(\mathbf{y}')\rangle$. This property plays an important role in developing our efficient learning algorithm discussed in Section 4.3.1.

By substituting the MAP solution $\hat{\vec{\mathbf{w}}}$ into Equation (4.2), we obtain the solution $\hat{\boldsymbol{\beta}} = (\mathbf{K}_\mathcal{I} + \sigma^2\mathbf{I})^{-1}\vec{\mathbf{x}}_\mathcal{I}$ where $\mathbf{K}_\mathcal{I} = \mathbf{P}\mathbf{K}\mathbf{P}^\top$ and the Gram matrix $\mathbf{K} \in \mathbb{R}^{RC\times RC}$ takes the form $\mathbf{K} = \boldsymbol{\Omega}\otimes\boldsymbol{\Sigma}$. Note that $\mathbf{K}$ is a matrix of so called *self-measuring* similarity, because the similarities of $\mathbf{X}$ are measured via the kernel function $k(\cdot,\cdot)$ by using $\mathbf{X}$ itself. The idea of self-measuring similarity was originally used with K-nearest neighbor methods (see Section 4.4.) To compute the kernel function in which the input contains missing values $\{x_{ik}|(i,k)\notin\mathcal{I}\}$, we introduce latent variables $\mathbf{z}_{\setminus\mathcal{I}} \equiv \{z_{ik}|(i,k)\notin\mathcal{I}\}$. Instead of the partially observed data matrix $\mathbf{X}$, we use the completed matrix $\tilde{\mathbf{X}}$ as an input of the kernel function, where $\tilde{x}_{ik} = x_{ik}$ if $(i,k)\in\mathcal{I}$ otherwise $\tilde{x}_{ik} = z_{ik}$.

If the observed matrix represents reflective relationships, i.e., $\mathbf{X}$ is symmetric, our model can naturally handle it by just setting $\boldsymbol{\Sigma} = \boldsymbol{\Omega}$.

4.2.3 As a Bayesian probabilistic model

This model can be naturally extended to a Gaussian process. Recall that we have a standard Gaussian prior $N(\vec{\mathbf{w}}|\mathbf{0},\mathbf{I})$ for $\vec{\mathbf{w}}$. Since a likelihood of f can be seen as the Dirac measure $p(f|\vec{\mathbf{w}},\mathbf{x}_{i:},\mathbf{x}_{:k}) = \mathbb{I}(f(\mathbf{x}_{i:},\mathbf{x}_{:k}) = \vec{\mathbf{w}}^\top\boldsymbol{\xi}_{ik})$, by marginalizing out $\vec{\mathbf{w}}$ from Equation (4.2) a tensor Gaussian process prior (Yu et al., 2007) with self-measuring covariance function is obtained:

$$f(\mathbf{x}_{i:},\mathbf{x}_{:k}) \sim \mathcal{GP}\Big(0, k\Big((\mathbf{x}_{i:},\mathbf{x}_{:k}),(\mathbf{x}_{i':},\mathbf{x}_{:k'})\Big)\Big). \tag{4.7}$$

The fact can be verified by computing the mean and the covariance:

$$\begin{aligned}
\mathbb{E}[f_{ik}] &= \int f_{ik} \int \mathbb{I}(f_{ik} = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik}) N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}\mathbf{w} \mathrm{d}f_{ik} \\
&= \int \int f_{ik} \mathbb{I}(f_{ik} = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik}) N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}f_{ik} \mathrm{d}\mathbf{w} \\
&= \int \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik} N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}\mathbf{w} \\
&= 0, \\
\mathrm{cov}[f_{ik}, f_{jl}] &= \int \int f_{ik} f_{jl} \left\{ \int \mathbb{I}(f_{ik} = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik}) \mathbb{I}(f_{jl} = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{jl}) N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}\mathbf{w} \right\} \mathrm{d}f_{ik} \mathrm{d}f_{jl} \\
&= \int \int \int f_{ik} f_{jl} \mathbb{I}(f_{ik} = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik}) \mathbb{I}(f_{jl} = \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{jl}) N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}f_{ik} \mathrm{d}f_{jl} \mathrm{d}\mathbf{w} \\
&= \int \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{ik} \vec{\mathbf{w}}^\top \boldsymbol{\xi}_{jl} N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}\mathbf{w} \\
&= \boldsymbol{\xi}_{ik}^\top \left\{ \int \vec{\mathbf{w}} \vec{\mathbf{w}}^\top N(\vec{\mathbf{w}}|\mathbf{0}, \mathbf{I}) \mathrm{d}\mathbf{w} \right\} \boldsymbol{\xi}_{jl} \\
&= \boldsymbol{\xi}_{ik}^\top \boldsymbol{\xi}_{jl},
\end{aligned}$$

where we denote $f(\mathbf{x}_{i:}, \mathbf{x}_{:k})$ by f_{ik} for simplicity. The mean and the variance of the predictive distribution of (4.7) are then given by

$$\mathbb{E}[x_{ik} \mid \mathcal{D}, \mathbf{z}_{\setminus\mathcal{I}}] = \mathbf{k}_{ik}^\top (\mathbf{K}_\mathcal{I} + \sigma^2 \mathbf{I})^{-1} \mathbf{x}_\mathcal{I}, \tag{4.8}$$

$$\mathrm{var}[x_{ik} \mid \mathcal{D}, \mathbf{z}_{\setminus\mathcal{I}}] = c_{ik} - \mathbf{k}_{ik}^\top (\mathbf{K}_\mathcal{I} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{ik}, \tag{4.9}$$

where $c_{ik} = k_{ik,ik} + \sigma^2$. Note that the predictive mean (4.8) is the same as Equation (4.6) with the solution $\hat{\boldsymbol{\beta}}$; since the noise ε is Gaussian, and the predictive mean is located at the same point as the MAP.

In Equation (4.1), if the covariance matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{\Omega}$ are known, then $\mathbf{F} \equiv [f(\boldsymbol{\phi}_i, \boldsymbol{\psi}_k)]_{i,k}$ follow a zero-mean matrix Gaussian distribution $N(\mathbf{F} \mid \mathbf{0}, \boldsymbol{\Sigma}, \boldsymbol{\Omega})$, which is defined as

$$\begin{aligned}
N(\mathbf{F} \mid \mathbf{M}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) &= \frac{1}{(2\pi)^{RC/2} |\boldsymbol{\Omega}|^{R/2} |\boldsymbol{\Sigma}|^{C/2}} \\
&\times \exp\left(-\frac{1}{2} \mathrm{tr}\left[\boldsymbol{\Omega}^{-1} (\mathbf{F} - \mathbf{M})^T \boldsymbol{\Sigma}^{-1} (\mathbf{F} - \mathbf{M}) \right] \right).
\end{aligned} \tag{4.10}$$

A matrix Gaussian distribution is a special case of a Gaussian distribution which parametrizes a covariance matrix by the Kronecker product. This limitation reduces the number of parameters from $\mathcal{O}(R^2 C^2)$ to $\mathcal{O}(R^2 + C^2)$.

4.2.4 Further extensions

Tensor modeling

We can naturally extend our model to handle a higher order data tensor. Suppose that we have an L-th order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{D_1 \times \cdots \times D_L}$. Let $\underline{\mathbf{X}}_i^{(l)} \in \mathbb{R}^{D_1 \times \cdots \times D_{l-1} \times D_{l+1} \times \ldots D_L}$

denotes the i-th slice of the l-th mode of $\underline{\mathbf{X}}$. The model for $\underline{\mathbf{X}}$ is reasonably induced through Equation (4.6) where $\mathbf{K}$ is defined as

$$\mathbf{K} = \mathbf{\Sigma}_L \otimes \cdots \otimes \mathbf{\Sigma}_1, \quad [\mathbf{\Sigma}_l]_{mn} = \Sigma_l(\underline{\mathbf{X}}^{(l)}_m, \underline{\mathbf{X}}^{(l)}_n) \tag{4.11}$$

where $\Sigma_l(\cdot,\cdot)$ is a PSD kernel function for the l-th mode. In this case, the weight parameter in Equation (4.2) also becomes an L-th order tensor and the resulting primal representation is considered as the Tucker decomposition model (2.1) with an infinite-dimensional latent feature space.

Side information

If we have side information $\{\mathbf{s}_i | i = 1, \ldots, R\}$ (e.g., demographic data of users such as age) for each row, we exploit them by combining them with the self-measuring covariance function. For example, we extend the kernel function into a sum form (4.12a) or a product form (4.12b):

$$\Sigma(\mathcal{U}_i, \mathcal{U}_j) = \alpha_1 \Sigma'(\mathbf{x}_{i:}, \mathbf{x}_{j:}) + \alpha_2 \Sigma''(\mathbf{s}_i, \mathbf{s}_j), \tag{4.12a}$$

$$\Sigma(\mathcal{U}_i, \mathcal{U}_j) = \Sigma'(\mathbf{x}_{i:}, \mathbf{x}_{j:}) \Sigma''(\mathbf{s}_i, \mathbf{s}_j), \tag{4.12b}$$

for $\alpha_1, \alpha_2 > 0$. Note that in both sum- and product-form, if kernel functions Σ' and Σ'' are PSD, then the resulting kernel function Σ is still PSD (Rasmussen and Williams, 2006). The idea of the incorporation of side information with the sum form is previously discussed by Abernethy et al. (2009).

Additive kernel

We can extend the kernel k to an additive form, i.e.,

$$k\Big((\mathcal{U}_i, \mathcal{V}_k), (\mathcal{U}_j, \mathcal{V}_l)\Big) = \sum_{p=1}^{P} \alpha_p k_p\Big((\mathcal{U}_i, \mathcal{V}_k), (\mathcal{U}_j, \mathcal{V}_l)\Big)$$

where $k_p((\mathcal{U}_i, \mathcal{V}_k), (\mathcal{U}_j, \mathcal{V}_l)) = \Sigma_p(\mathcal{U}_i, \mathcal{U}_j)\Omega_p(\mathcal{V}_k, \mathcal{V}_l)$. If Σ_p and Ω_p for $p = 1, \ldots, P$ are PSD with $\alpha_1, \ldots, \alpha_P \geq 0$, then the resulting kernel k holds PSD. For example, if we set $P = 2$ and employ Kronecker delta function as Σ_1 and Ω_2, then we have the Gram matrix represented as $\mathbf{K} = \alpha_1(\mathbf{\Omega} \otimes \mathbf{I}) + \alpha_2(\mathbf{I} \otimes \mathbf{\Sigma}) = (\alpha_1 \mathbf{\Omega}) \oplus (\alpha_2 \mathbf{\Sigma})$ where $\oplus$ denotes the Kronecker sum: $\mathbf{A} \oplus \mathbf{B} = \mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{B}$. When $\alpha_1 = \alpha_2 = 1$, the kernel function is specially called *Cartesian kernel*, that has recently been applied to data mining (Kashima et al., 2009b) and geostatistics (Agovic et al., 2011).

4.3. Learning Algorithm

In this section, first we introduce an efficient algorithm to compute the mean of the predictive distribution with assuming that the latent variables are known. Then we discuss how to estimate the latent variables and the hyper-parameters.

4.3.1 Prediction of missing values

As mentioned, we need to compute the inverse of $(\mathbf{K}_{\mathcal{I}} + \sigma^2\mathbf{I})$ in Equation (4.8) to obtain the predictive means, in which the naïve computational cost is $\mathcal{O}(N^3)$. Instead, we solve the linear equation

$$\vec{\mathbf{x}}_{\mathcal{I}} = (\mathbf{K}_{\mathcal{I}} + \sigma^2\mathbf{I})\boldsymbol{\beta} \tag{4.13}$$

with respect to $\boldsymbol{\beta}$. Note that $(\mathbf{K}_{\mathcal{I}} + \sigma^2\mathbf{I})$ is positive definite when $\sigma^2 > 0$. We solve Equation (4.13) using the conjugate gradient method (Shewchuk, 1994), which is an iterative method to solve a linear system in which the matrix is positive definite. Each iteration of the conjugate gradient needs to perform a matrix-vector multiplication; in our case that corresponds to the multiplication of $\mathbf{K}_{\mathcal{I}}$ and an N-dimensional vector, which requires $\mathcal{O}(N^2)$ computation and $\mathcal{O}(N)$ memory space.

The computational cost of the multiplication can still be reduced by exploiting a structure in the covariance matrix. Because of the structure of the Kronecker product in $\mathbf{K}_{\mathcal{I}}$, a multiplication of $\mathbf{K}_{\mathcal{I}}$ and an N-dimensional vector $\mathbf{v}$ is rewritten as

$$\mathbf{K}_{\mathcal{I}}\mathbf{v} = \mathbf{P}(\boldsymbol{\Omega} \otimes \boldsymbol{\Sigma})\mathbf{P}^\top\mathbf{v} = \text{vec } \mathbf{P}(\boldsymbol{\Sigma}\mathbf{V}\boldsymbol{\Omega}) \tag{4.14}$$

where $\mathbf{V} \in \mathbb{R}^{R\times C}$ is a matrix form of $\mathbf{P}^\top\mathbf{v}$, i.e., vec $\mathbf{V} = \mathbf{P}^\top\mathbf{v}$. This technique, called *vec-trick* (Vishwanathan et al., 2007; Kashima et al., 2009a), reduces the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(RC(R + C))$. We apply the same technique when $\mathbf{K}$ is modeling for a data tensor as described in Section 4.2.4. When we have a $D_1 \times \cdots \times D_L$ data tensor, the computational complexity of the vec-trick is $\mathcal{O}(\prod_{l=1}^{L} D_l(\sum_{q=1}^{L} D_q))$.

Suppose we stop the iteration of the conjugate gradient when the ℓ_2 error of $\hat{\boldsymbol{\beta}}_l$ (i.e., between the solution at the l-th iteration and a solution of Equation (4.13) $\boldsymbol{\beta}_*$) is less than the error of the initial values $\hat{\boldsymbol{\beta}}_0$ with a tolerance ϵ, i.e., $||\boldsymbol{\beta}_* - \hat{\boldsymbol{\beta}}_l||_2 \leq \epsilon||\boldsymbol{\beta}_* - \hat{\boldsymbol{\beta}}_0||_2$. Then the maximum number of iterations is bounded $l \leq \frac{1}{2}\sqrt{\kappa}\log\left(\frac{2}{\epsilon}\right)$ where κ is the condition number of $(\mathbf{K}_{\mathcal{I}} + \sigma^2\mathbf{I})$ which is defined as the ratio of its maximum and the minimum eigenvalue. The total cost for obtaining the solution $\hat{\boldsymbol{\beta}}$ is $\mathcal{O}(\sqrt{\kappa}RC(R+C))$.

Let γ be the observation rate, i.e. $\gamma \equiv N/(RC)$, and the computational complexity of the naïve approach is rewritten as $\mathcal{O}(\gamma^3(RC)^3)$. If $\mathbf{X}$ is nearly square, i.e., $R \simeq C$, then our algorithm is much faster than the naïve one when the observation ratio γ is roughly greater than $\kappa^{\frac{1}{6}}/R^{\frac{2}{3}}$.

Note that the variances of the predictive distribution (4.9) can also be obtained by a similar way. Although it requires to solve multiple linear equations (4.13) where $\vec{\mathbf{x}}_{\mathcal{I}} = \mathbf{P}(\mathbf{k}_{ik} \otimes \mathbf{k}_{ik})$ for each (i, k)-th element, the parallel computation of them is possible.

4.3.2 Estimation of latent variables and model selection

Since we have a probabilistic model, the empirical Bayesian approach, i.e., maximizing the marginal likelihood, is one of the desirable methods to estimate the latent

Algorithm 3 Computation of predictive means.

Initialize $\hat{\mathbf{z}}_{\backslash\mathcal{I}}^{(0)}$ by row-wise or column-wise means of $\mathbf{X}$
for $l = 1$ to maximum number of iterations **do**
  Construct $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ with $\mathbf{X}$, $\hat{\mathbf{z}}_{\backslash\mathcal{I}}^{(l-1)}$, and side information
  Solve $\vec{\mathbf{x}}_{\mathcal{I}} = (\mathbf{K}_{\mathcal{I}} + \sigma^2\mathbf{I})\boldsymbol{\beta}$ by the conjugate gradient descent with a tolerance ϵ
  Compute predictive means (4.8) for unobserved elements
  Update $\hat{\mathbf{z}}_{\backslash\mathcal{I}}^{(l)}$ by the predictive means
end for
Return $\hat{\mathbf{z}}_{\backslash\mathcal{I}}^{(l)}$

variables. However, the optimization of the marginal log-likelihood is computationally infeasible especially for large-scale data. Instead, we use the predictive means as estimators of the latent variables, i.e., $\hat{\mathbf{z}}_{\backslash\mathcal{I}} = \mathbb{E}[\mathbf{x}_{\backslash\mathcal{I}}|\mathcal{D}]$. As discussed in the previous subsection, the computation of the predictive means only require $\mathcal{O}(\sqrt{\kappa}RC(R+C))$ time and $\mathcal{O}(N)$ space, which is generally better than the empirical Bayesian approach. Note that the values of $\hat{\mathbf{z}}_{\backslash\mathcal{I}}$ affect the predictive mean (4.8) through the kernel functions, thus we iteratively perform the estimation procedure. As an initial value of $\hat{z}_{ik}$ for $(i,k) \notin \mathcal{I}$, we use a row-wise mean $\frac{1}{|\mathcal{I}_{,k}|}\sum_{j\in\mathcal{I}_{,k}}\mathbf{x}_{jk}$ or a column-wise mean $\frac{1}{|\mathcal{I}_{i,}|}\sum_{l\in\mathcal{I}_{i,}}\mathbf{x}_{il}$, where $\mathcal{I}_{,k}$ ($\mathcal{I}_{i,}$) is a set of row (column) indices of observed element in the k-th column (i-th row) of $\mathbf{X}$. We summarize the entire algorithm as a pseudo code in Algorithm 3. Note that when both $\boldsymbol{\phi}(\cdot)$ and $\boldsymbol{\psi}(\cdot)$ are finite dimensional, this EM-like iterative method can be interpreted as an approximation of the EM algorithm (see Appendix B.1 for more details.)

We determine the hyper-parameter σ^2 and kernel parameters by cross-validation. Similarly, this model selection can be done with the same computational complexity of the prediction.

4.4. Connection to other methods

Although there are a lot of approaches for recommender systems, their characteristics can be specified by the following two intrinsic principles in general: (i) how to measure similarities between the observed elements, and (ii) how to use the similarity to construct the prediction model for the unobserved elements. In this point of view, we show the connection from our model to K-nearest neighbor, matrix factorization, and GP-based approaches.

4.4.1 K-nearest neighbor

The K-nearest neighbor (KNN) algorithm is one of the most popular approaches of collaborative filtering. The user-based KNN method (Resnick et al., 1994) measure the similarity between the i-th user (row) and the other j-th user for $j = 1, \ldots, R$ by

using $\mathbf{x}_{i:}$ and $\mathbf{x}_{j:}$. Then the predictive value of x_{ik} is given as

$$\hat{x}_{ik}^{\mathrm{KNN}} = \frac{\sum_{j \in \mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:}) x_{jk}}{\sum_{j \in \mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:})} \tag{4.15}$$

where $s(\cdot, \cdot)$ is a similarity function such as the Pearson correlation and $\mathcal{R}_i$ is an index set of top-K related users to the i-th user. When $\mathbf{x}_{i:}$ and $\mathbf{x}_{j:}$ contain missing values, the similarity $s(\mathbf{x}_{i:}, \mathbf{x}_{j:})$ is computed based only on observed pairs $\{(x_{il}, x_{jl}) | l \in \mathcal{I}_{i,} \cap \mathcal{I}_{j,}\}$. Note that we assume the reflectiveness for $\{\mathcal{R}_i | i = 1 \ldots, R\}$, i.e., $j \in \mathcal{R}_i$ if and only if $i \in \mathcal{R}_j$ for $i, j = 1, \ldots, R$.

Given $\{\mathcal{R}_i | i = 1, \ldots, R\}$, the loss function of the user-based KNN is explicitly written as the following weighted squared loss

$$\sum_{(i,k) \notin \mathcal{I}} \sum_{j \in \mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:}) (x_{jk} - y_{ik})^2, \tag{4.16}$$

where y_{ik} is a parameter of a predictive value of x_{ik}. The minimizer of y_{ik} in Equation (4.16) is equivalent to the predictive value of the user-based KNN (4.15) (see Appendix B.2.) Compared to our loss function (4.4), there is no parametric structure in the model $\{y_{ik} | (i, k) \notin \mathcal{I}\}$. Instead, the importance weight of each observed sample x_{jk} $((j, k) \in \mathcal{I})$ for a missing element x_{ik} is introduced by $s(\mathbf{x}_{i:}, \mathbf{x}_{j:})$. Note that when a shift-invariant similarity $s(\mathbf{x}, \mathbf{x}') = g(\mathbf{x} - \mathbf{x}')$ is given, the predictive value of the user-based KNN (4.15) is a conditional mean of the *Nardaraya-Watson* model (Bishop, 2007) in which the joint distribution for a pair of the i-th user and the k-th item is defined as

$$p_{ik}(\mathbf{x}, y) = p(\mathbf{x}, y \mid \mathcal{D}_{ik}) = \frac{1}{|\mathcal{R}_i|} \sum_{j \in \mathcal{R}_i} h(\mathbf{x} - \mathbf{x}_{j:}, y - x_{jk})$$

where $\mathcal{D}_{ik} \equiv \{(\mathbf{x}_{j:}, x_{jk}) | j \in \mathcal{R}_i\}$ is training data and $h(\mathbf{x}, y)$ is the component density function defined as $\int_{-\infty}^{\infty} h(\mathbf{x}, y) \mathrm{d}y = g(\mathbf{x})$. We see that the KNN independently models the distribution of a pair $(\mathbf{x}_{i:}, y_{ik})$ and there is no transfer of knowledge to other pairs.

By defining $\Sigma_{ij} = \bar{s}(\mathbf{x}_{i:}, \mathbf{x}_{j:}) \mathbb{I}(j \in \mathcal{R}_i)$ and $\Omega_{kl} = \mathbb{I}(k = l)$ with the normalized similarity $\bar{s}(\cdot, \cdot)$, the predictive value of the user-based KNN (4.15) is also rewritten as $\hat{x}_{ik}^{\mathrm{KNN}} = \mathbf{k}_{ik}^{\top} \vec{\mathbf{x}}_{\mathcal{I}}$. It can be seen as an approximation of the predictive mean (4.8) with taking $(\mathbf{K}_{\mathcal{I}} + \sigma^2 \mathbf{I})^{-1} = \mathbf{I}$. We obtain an analogous result for the item-based KNN (Sarwar et al., 2001). A unifying approach of the user- and the item-based KNN was also proposed in (Wang et al., 2006).

4.4.2 Matrix and tensor factorization

In the pairwise linear model (4.2), we treat $\boldsymbol{\phi}_i$ and $\boldsymbol{\psi}_k$ as the *given* feature vectors of the i-th row and the k-th column, respectively. Here we consider they are hidden variables that we have to estimate from observed data. Suppose that $\boldsymbol{\phi}_i$ and $\boldsymbol{\psi}_k$ are both Q-dimensional vectors satisfying $Q \leq \min(R, C)$ and we give a Dirac measure

$p(\mathbf{W}) = \mathbb{I}(\mathbf{W} = \mathbf{I})$ as a prior of $\mathbf{W} \in \mathbb{R}^{Q \times Q}$. If the all elements of $\mathbf{X}$ are observed, then the probability distribution of the model (4.3) is simply rewritten as

$$p(\mathbf{X} \mid \mathbf{\Phi}, \mathbf{\Psi}, \mathbf{W}) \propto \exp\left(-\frac{1}{2\sigma^2}\left|\left|\mathbf{X} - \mathbf{\Phi}\mathbf{W}\mathbf{\Psi}^\top\right|\right|_{\text{Fro}}^2\right) \tag{4.17}$$

where $\mathbf{\Phi}^\top = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_R)$ and $\mathbf{\Psi}^\top = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_C)$. $||\cdot||_{\text{Fro}}$ denotes the Frobenius norm defined by $||\mathbf{A}||_{\text{Fro}} = \sqrt{\text{tr}(\mathbf{A}^\top\mathbf{A})}$. We estimate maximum marginal likelihood solutions of $\{\boldsymbol{\phi}_i\}$ and $\{\boldsymbol{\psi}_k\}$ that are achieved by minimizing the unregularized squared loss function

$$\underset{\mathbf{\Phi}, \mathbf{\Psi}}{\text{argmin}} \left|\left|\mathbf{X} - \mathbf{\Phi}\mathbf{\Psi}^\top\right|\right|_{\text{Fro}}^2 \tag{4.18}$$

Here, matrix factorization can be interpreted as a learning framework of row-wise and column-wise similarities under the low-rank constraints $\mathbf{\Sigma} = \mathbf{\Phi}\mathbf{\Phi}^\top$ and $\mathbf{\Omega} = \mathbf{\Psi}\mathbf{\Psi}^\top$. This is in contrast to our model, which assumes $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ are both full-rank. Bayesian extensions of matrix factorization were discussed in (Salakhutdinov and Mnih, 2008a,b).

A more general case corresponds to a Bayesian extension of the Tucker decomposition called *pTucker* model (Chu and Ghahramani, 2009). In Equation (4.17), suppose that we have a latent factor $\mathbf{\Phi}_l \in \mathbb{R}^{D_l \times Q_l}$ for $l = 1, \ldots, L$. By marginalizing the weights $\underline{\mathbf{W}}$ by a standard Gaussian prior, we obtain a tensor Gaussian distribution (Dutilleul, 1999; Hoff, 2010). This is an extension of the matrix Gaussian distribution (4.10) with parametrizing an l-th mode-specific covariance matrix $\mathbf{\Sigma}_l$ as $\mathbf{\Sigma}_l = \mathbf{\Phi}_l\mathbf{\Phi}_l^\top$ and estimates $\{\mathbf{\Phi}_l\}$. Yu et al. (2009) proposed an nonparametric extension of probabilistic PCA, which can be seen as a special case of a two-mode pTucker model where $\mathbf{\Sigma}_1$ is set as an identity matrix. Its cost function contains 'logdet' regularization term, and it enforces an estimator of $\mathbf{\Sigma}_2$ to be low-rank.

Our model is also related to the CUR decomposition (Mahoney and Drineas, 2009). Here we employ the idea of self-measuring similarity to Equation (4.2), and suppose that the feature functions $\boldsymbol{\phi}_i = \boldsymbol{\phi}(\mathbf{x}_{i:})$ and $\boldsymbol{\psi}_i = \boldsymbol{\psi}(\mathbf{x}_{:k})$ are identity functions with the row- and column-wise sparsity, i.e. $\boldsymbol{\phi}(\mathbf{x}_{i:}) = \mathbb{I}(i \in \mathcal{R})\mathbf{x}_{i:}$ and $\boldsymbol{\psi}(\mathbf{x}_{:k}) = \mathbb{I}(k \in \mathcal{C})\mathbf{x}_{:k}$ where $\mathcal{R}$ and $\mathcal{C}$ are arbitrary index sets of rows and columns, respectively. Then the loss function (4.4) is transformed to the following form

$$||\mathbf{X} - \mathbf{X}_{\mathcal{C}}\mathbf{W}\mathbf{X}_{\mathcal{R}}||_{\text{Fro}}^2 + \sigma^2 ||\mathbf{W}||_{\text{Fro}}^2 \tag{4.19}$$

where $\mathbf{X}_{\mathcal{R}} \in \mathbb{R}^{R \times |\mathcal{R}|}$ ($\mathbf{X}_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}| \times C}$) is a submatrix of $\mathbf{X}$ that consists from $|\mathcal{R}|$ row vectors $\{\mathbf{x}_{i:} | i \in \mathcal{R}\}$ ($|\mathcal{C}|$ column vectors $\{\mathbf{x}_{:k} | k \in \mathcal{C}\}$). When $\sigma^2 = 0$, the loss function (4.19) is equivalent to the one of the CUR decomposition and the solution is given by $\hat{\mathbf{W}} = \mathbf{X}_{\mathcal{C}}^+\mathbf{X}\mathbf{X}_{\mathcal{R}}^+$ where $\mathbf{A}^+$ is the pseudo inverse of $\mathbf{A}$.

4.4.3 GP-based models

Yu et al. (2007) presented the idea of the tensor GP and proposed its approximation and an efficient learning algorithm for large-scale data sets. The approximated model is roughly equivalent to matrix factorization with the regularization incorporating with side information. Bonilla et al. (2008) apply the tensor GP framework to multi-task learning. After specifying the row covariance $\mathbf{\Sigma}$ with row-specific side information, they empirically estimate the column covariance $\mathbf{\Omega}$. That is in contrast to the tensor GP, which estimates both $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ by a nonparametric way. Probabilistic matrix addition (Agovic et al., 2011) is a GP model that employs the Cartesian kernel (see Section 4.2.4) as a covariance function instead of the Kronecker kernel. Abernethy et al. (2009) proposed more general framework which generalizes the loss functions $J(\mathbf{W})$ (4.4) with imposing a low-rank constraint for the weight parameter $\mathbf{W}$, which results the model as low-rank.

In many cases, exact computation of these GP-based methods is infeasible and several approximations such as a low-rank approximation of a Gram matrix are widely used. Note that our learning algorithm described in Section 4.3.1 can directly be used in these models.

4.5. Experimental results

In this section, first we investigate the behavior of the EM-like iterative update of the latent variables which we discussed in Section 4.3.2 by synthetic data sets. Next we evaluate the applicability for a real recommendation problem and data tensors. All experiments were done with a Xeon 2.93 GHz 8 core machine.

4.5.1 Toy data set

We randomly generate ten 100×100 data matrices by following the generative model (4.1) and (4.10) which we set $\sigma^2 = 0.3$. We employ the RBF kernel as the covariance functions.

Figure 4.1 shows the training and testing errors with growing the number of iterations of the EM-like iterative method, which includes the testing error when we use the mean, the row-wise means, and the column-wise means of the observed elements as predictive values for comparison. We see that the larger σ^2 works to prevent overfitting to the training (observed) elements. Our framework achieves same or better prediction accuracy than the naïve predictors even for the full-rank matrix completion problem. Figure 4.2 is the result when the ranks of both $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ are 2. In contrast to the result of the full rank covariance matrices (Figure 4.1), the iterative method improves the testing errors in the first few iterations.

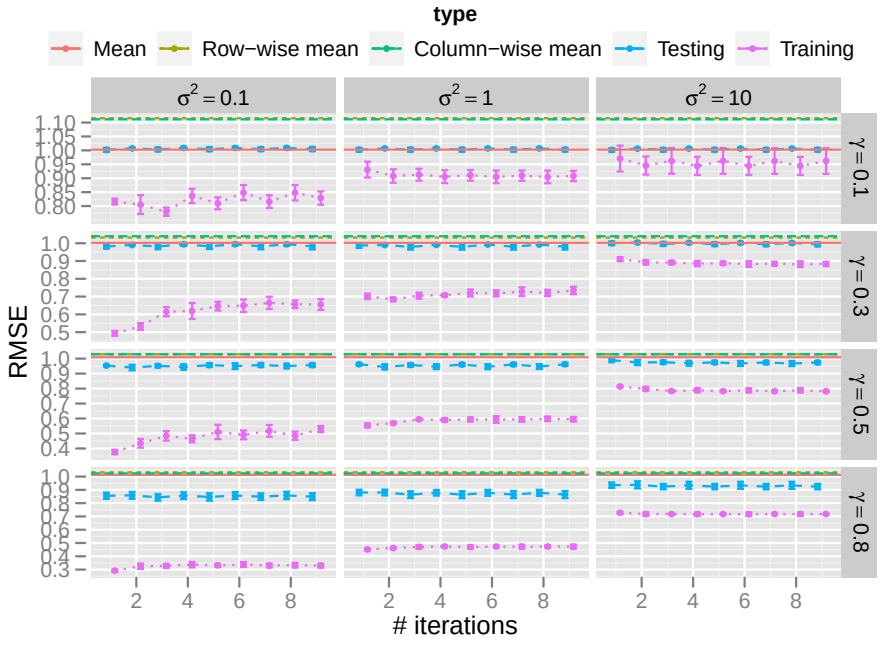

Figure 4.1: RMSE v.s. the number of iterations of the EM-like heuristic with various settings of σ^2 and the observation ratio γ. The errorbars represent the standard deviation.

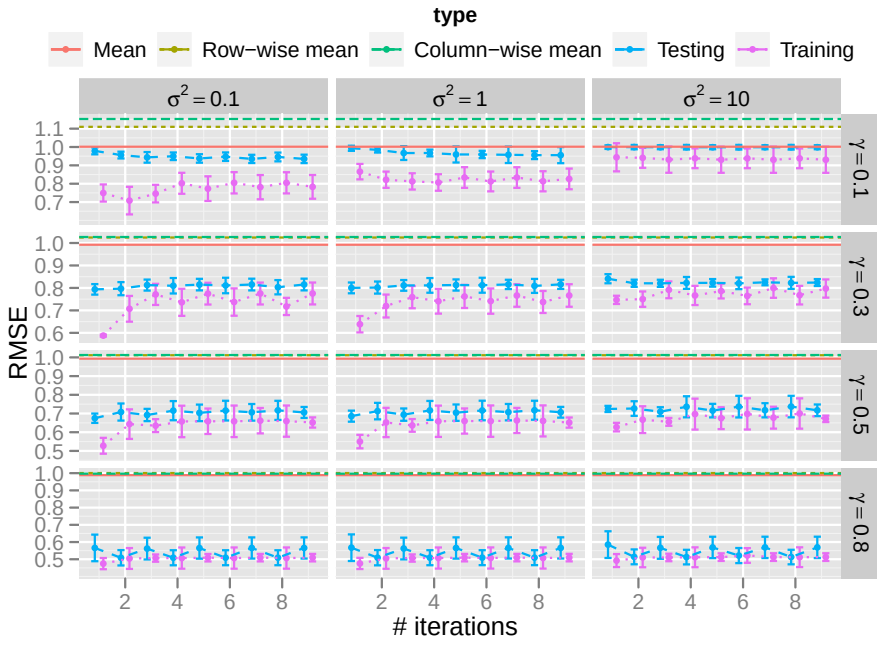

Figure 4.2: RMSE v.s. the number of iterations when $\mathbf{\Sigma}$ and $\mathbf{\Omega}$ are low-rank.

4.5.2 Collaborative filtering

We use the Movielens 100k data set[3], which contains $100,000$ ratings $x_{ik} \in \{1,2,3,4,5\}$ for $1,682$ movies labeled by 943 users. The observation ratio γ is 0.06. The data set contains side information: user-specific features (e.g., age, gender, ...) and movie-specific features (release date, genre, ...). The data set provides $90,570$ ratings for training and remaining $9,430$ ratings for testing. After learning with the training data set, we evaluate the RMSE for the testing data set.

We employ the RBF and the linear kernel as the covariance functions, and the hyper-parameters including σ^2 are determined by three-fold cross validation. We prepare two similarities: a self-measuring similarity ("Self-measuring"), and a combination of self-measuring and side information ("Combination"). In "Combination", we use the sum form (4.12a) for the linear kernel and the product form (4.12b) for the RBF kernel. We compare with standard methods for recommendation system which includes user- and item-based KNN with the Pearson correlation and matrix factorization. We also prepare the CUR decomposition (Mahoney and Drineas, 2009). To handle the missing values in the CUR decomposition, the missing values in $\mathbf{X}_{\mathcal{C}}$ and $\mathbf{X}_{\mathcal{R}}$ in Equation (4.19) are replaced by row- or column-wise means. The tensor GP model is also included in the comparison, which is equivalent to our method given the similarity measured by side information.

3. `http://www.grouplens.org/node/73`

Table 4.1: RMSEs on the Movielens 100k data set.

Method	RMSE	Time
Tensor GP (linear)	1.1225	26s
Tensor GP (RBF)	1.0517	7m01s
CUR (row)	1.0245	5s
CUR (col)	1.0199	5s
KNN (user)	0.9507	7s
KNN (item)	0.9354	42s
Matrix Factorization	0.9345	1m38s
Self-measuring (linear)	0.9340	45m41s
Self-measuring (RBF)	0.9308	16m22s
Combination (linear)	0.9294	45m30s
Combination (RBF)	**0.9256**	18m25s

We summarize the prediction errors in Table 4.1. The result shows that our method outperforms the comparative approaches. It suggests that the side information are not enough to measure similarities compared to the self-measuring approach. Nevertheless, the combination with the side information ("Combination") further improves the prediction performance compared to using the self-measuring alone ("Self-measuring"). The best score ("Combination" with the RBF kernel) in Table 4.1 is also the best over other 76 methods listed in mlcomp.org[4] as of Oct, 2011.

4.5.3 Three-way tensor data sets

We use five real data sets called "Amino acids", "Flow injection", "Bread sensory", "Sugar process", and "Fermentation process".[5][6] Note that both "Amino acids" and "Flow injection" are known as rank deficit problems (Bro, 1998): "Amino acids" is almost perfectly modeled by rank-3 *PARAFAC* (Harshman, 1970), a special case of the Tucker decomposition. "Flow injection" is also captured by a similar low-dimensional model.

For each data tensor, we randomly choose 50% of its elements as training data (i.e. observed elements) for 100 different random seeds. The rest elements are used for testing. We rescale each data tensor by its standard deviation to align the scales of all data sets. We compare with the Tucker decomposition, pTucker, and PARAFAC. We use the `N-way Toolbox` (Andersson and Bro, 2000) for the implementations of the Tucker and PARAFAC. Here we omit the tensor GP model, since some of the data sets do not contain side information. We use the RBF kernel that has individual scale parameters λ_l for each l-th mode. Throughout this experiment, we fix the hyper-parameters as $\sigma^2 = 1$ (same as pTucker) and $\lambda_l = D_l / D_1 D_2 D_3$ where D_l is

4. `http://mlcomp.org/datasets/341`
5. `http://www.models.kvl.dk/datasets/`
6. Here we follow the experimental settings in (Chu and Ghahramani, 2009).

Table 4.2: Ranks of PARAFAC, Tucker, and pTucker selected by `tucktest`.

Data Set	Amino acids	Flow injection	Bread sensory	Sugar process	Fermentation process
ranks	$4 \times 4 \times 4$	$4 \times 4 \times 4$	$5 \times 5 \times 5$	$5 \times 5 \times 5$	$4 \times 4 \times 4$

Table 4.3: RMSEs on the tensor data sets.

Data Set	Amino acids	Flow injection	Bread sensory	Sugar process	Fermentation process
Dimensions	$5 \times 201 \times 61$	$12 \times 100 \times 89$	$10 \times 11 \times 8$	$268 \times 571 \times 7$	$338 \times 15 \times 15$
PARAFAC	0.0295	0.0612	2.2831	0.3336	0.2444
Tucker	**0.0256**	0.0523	1.6026	0.3319	0.2486
pTucker	0.0273	0.1476	0.6053	0.2214	0.2251
Self-measuring	0.0350	**0.0428**	**0.5261**	**0.0421**	**0.1284**

the dimensionality of the l-mode. The ranks of the tensor factorization methods are preliminary specified by `tucktest` function of the `N-way toolbox` for each data set (see Table 4.2.)

We summarize the result in Table 4.3. The result clearly shows the low-rank tensor factorization methods performs well for the rank deficit data sets ("Amino acids" and "Flow injection"). In the other data sets, on the contrary, our method outperformed the low-rank models. To qualify the the characteristics of the proposed and the low-rank methods, we visualize slices of recovered data tensors at Figure 4.3. Note that here we set the observation ratio to 30% to clarify the difference. In "Amino acids" data set (the top-panel of Figure 4.3), the tensor factorization methods smoothly recovered the missing values in the low-dimensional feature spaces. In contrast, we see some jaggies in the reconstructed elements of our method. The situation is completely different in "Sugar process" data set (the bottom-panel of Figure 4.3). On one hand, the slices recovered by the low-rank tensor factorization are entirely flat and they failed to keep the wavy patterns, which are appeared in the original source. This is because the underlying data matrix would not be low-rank and the tensor factorization could not capture it in the low-dimensional space. Our method, on the other hand, successfully reconstructed the missing values with keeping the fine wavy patterns.

4.6. Summary

In this chapter, we have presented a new kernel-based framework for matrix and tensor completion problems. The proposed framework has separated row and column kernel matrices that can be readily calculated only from observed elements. We have also proposed an efficient conjugate gradient-based algorithm that exploits the structure of the Kronecker product in the Gram matrix. The algorithm allows the model to represent a full-rank matrix or tensor. On the Movielens 100k data set, we have shown that the proposed approach achieves the lowest error outperforming both

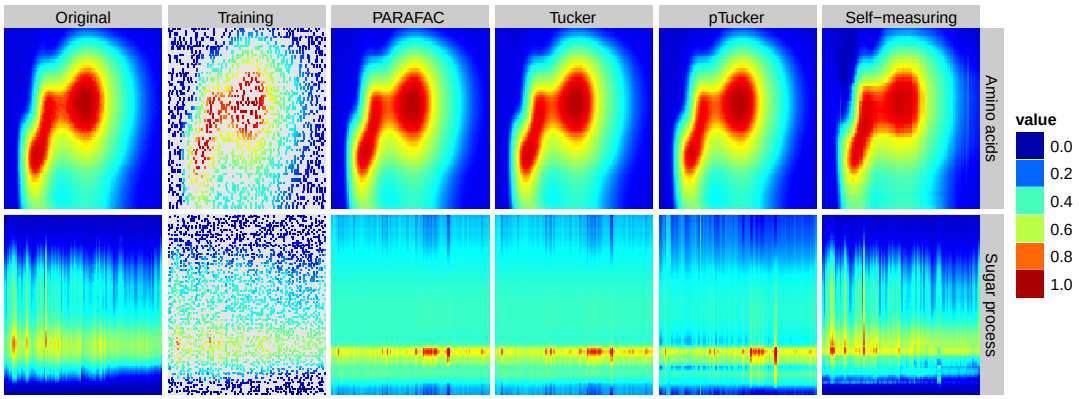

Figure 4.3: The results of reconstruction on "Amino acids" and "Sugar process" data sets, in which the fourth slice of the first mode and fifth slice of the third mode are illustrated in top and bottom panel, respectively. Original: original data sources. Training: training data sets with missing values indicated as blanks (70% of the elements are missing.) PARAFAC, Tucker, pTucker, and Self-measuring: reconstructed slice by each method.

KNN and matrix factorization. On four real-world tensor data, we have shown that we can achieve better error when the true underlying structure is full rank.

Chapter 5

Conclusion

This thesis has presented the two extensions of the Tucker decomposition. First, to handle heterogeneous tensors, we have generalized the likelihood distribution to the exponential family and have proposed the batch and the online procedure for the approximated Bayesian estimation of the parameters. Next, the kernelization of the Tucker decomposition has been introduced with the moderately fast iterative learning algorithm relaxing the low-rank assumption for the given tensor. The performance of the two models has been evaluated with some synthetic and real data sets and in the experimental results the proposed methods have outperformed the existing tensor factorization methods when the standard assumptions – the Gaussian observation model and the low-rank assumption – have been not holded.

5.1. Discussion and Future Works

The determination of the rank of ETF is an important issue for real applications. We can specify the rank by using standard techniques of model selection such as cross validation and empirical Bayes, while these approaches requires a large amount of computational cost. In last few years, several authors reformulated the Tucker decomposition as a convex problem with an explicit low-rank constraint (Tomioka et al., 2011; Signoretto et al., 2011; Gandy et al., 2011), which is analogous to the notion of compressive sensing (Fazel). These approaches will enable to determine the rank automatically from the theoretical aspects and such extension will be a promising future work for ETF.

As described in Section 3.3.1, the posterior of ETF is approximated by the Gaussian density. Due to the log-concavity of the likelihood and the prior, the posterior is also log-concave (uni-modal), and the Gaussian approximation is reasonable. The Laplace approximation allows us to implement a faster algorithm rather than a numerical approach based on the MC sampling, but we lose the information about the higher-order moments.

Another issue of ETF is that we must *a priori* specify the distribution for each attribute before learning the model. In some cases, it is difficult to choose the appropriate exponential family distribution. The model selection of the distribution is an open problem.

The conjugate-gradient-based algorithm discussed in Chapter 4 reduces the computational complexity compared to the conventional approach. However, the algorithm is still expensive to scale for large-scale data such as the Netflix data set. To handle such large-scale data, it is necessary to take account the sparsity of the data. Given, for example, a partially observed matrix $\mathbf{X}$ containing $N \ll RC$ elements. If we employ the linear kernel in which the missing elements are filled by 0, the complexity of the vec-trick (4.14) is reduced from $\mathcal{O}(RC(R+C))$ to $\mathcal{O}(N\max(R,C))$. It is also possible to employ other regularization terms instead of the ℓ_2^2 norm in Equation (4.4). For example, the regularization with the ℓ_1 norm introduces the sparsity in the parameter which improves the interpretation of the model and also saves the memory space. Recently several authors have developed efficient optimization methods for non-differentiable regularizations such as the ℓ_1 norm (Duchi and Singer, 2009; Langford et al., 2009).

Acknowledgement

まずはじめに博士課程の 3 年間をご指導して頂いた池田和司先生に深く感謝します．まだ修士のころに池田先生からテンソル分解というキーワードを教えてもらい，その当時取り組んでいた行列分解よりも一般的な手法があることを知ったのがこの博士課程での研究を始めた大きなきっかけでした．当時日本でテンソル分解を研究している人はまだ少なく，おかげでまだやりつくされていない分野で自由に研究を進めることが出来ました．

松本裕治先生にはお忙しいなか副指導教員を引き受けて頂きました．博士課程進学の際には松本研にも興味があり先生の教授室で相談に乗って頂いたことがありました．最終的には池田研に進学したため松本先生にはお手数をおかけしてしまいましたが，当時じっくり相談に乗って頂いたことで進学についてより深く考えるとができました．ありがとうございます．

同じく副指導教員を引き受けて頂いた渡邊一帆先生は，通常の輪講に加え学生主体の論文読み会や輪講などにも積極的に参加して頂きました．Chapter 3 で議論している指数型分布族やその近似方法についても色々と議論して頂き，とても参考になりました．また博士論文の原稿の細かい点までチェックして頂きました．ありがとうございます．

竹之内高志先生には博士課程 3 年間ほぼずっと直にご指導頂き，本博士論文の内容のほぼ全体に関して議論させて頂きました．打合せの際に自分が詰めの甘い式展開や実験結果を持っていくとそのいい加減さを必ず見破って駄目な点を丁寧にご指摘されました．そのおかげで，どんなささいな問題に対しても真摯に対応するという，研究に対する基本姿勢を身に付けることが出来たと思います．また統計学の基本的なセンスが竹之内先生の指導を通してある程度身に付けられたことも自分にとっては大きな幸運でした．

柴田智広先生には NEC との共同研究において大変お世話になりました．Chapter 3 のほぼ全ての内容はその際の共同研究に基づいた結果となっています．柴田先生がオーガナイズされた共同研究に参加したことでプロジェクトのスケジューリングやミーティングにおける段取りの決めかたなど，研究のメタな部分に関することを学ぶことができました．

NEC C&C イノベーション研究所の方々にはセンサデータ解析に関する共同研究の際にお世話になりました．特に山田敬嗣さん，國枝和雄さん，加藤大志さん，神谷祐樹さんには研究の方向性やデータの取得方法などについて様々な示唆を頂きました．

東京大学の鹿島久嗣先生，冨岡亮太先生には Chapter 4 の研究を始めとする様々な共同研究にて大変お世話になりました．特に鹿島先生には IBIS ワークショップ 2008 でお会いしてから，私の研究アイデアの相談に乗って頂いたり数々の新しい研究ネタを教えて頂きました．また冨岡先生には共同研究を通じてテンソル分解の最適化など関して学ばさせて頂きました．

池田研秘書の谷本史さんには学会出張や研究費関係でいつもお世話になりました．特に自分がドイツに居る時，書類のコピーといった様々な用事を引き受けてくださり，大変ありがたかったです．足立敏美さんには博士課程 3 年時に学振の書類関係や博士論文製本の際にお世話になりました．計算機クラスタ Maui に関して作村論一先生には何度もお世話になりました．為井智也さんからはコーヒーに対するこだわりを教え

て頂き，おかげで人生が少し豊かになった気がします．船谷浩之さんには一年上の先輩ということで，学業やプライベートに関して様々なことに相談に乗って頂きました．6 階のエレベーターホールで毎日料理を作っていたのはいい思い出です．同期の小林幹浩くんには主に食の面で色々とお世話になりました．いつかマンガ部屋でみんなに作ってくれたバナナケーキが絶品でした．Mauricio Alexandre Parente Burdelis さんとは席が 2 年間半ずっと隣同士だったこともあり研究の合間に色々な話をしました．論文の英語添削をいつも快く引き受けてくれ，論文の締切りで焦っていた私にとって仏のようにありがたい存在でした．後輩の中村政義くんとは自分が博士一年目のときにチームを組んで UCSD データマイニングコンテスト 2009 に出場し，中村くんが発見したデータの特徴抽出法のおかげでコンテストを優勝できました．このおかげで実データ解析には特徴抽出が一番大事な要素であることを身を持って学べました．

博士課程 3 年目には NEC 情報メディアプロセッシング研究所にて 3ヶ月弱インターンシップに受け入れて頂き，実際のビジネスの現場で研究を行うという貴重な経験をさせて頂きました．特にデータマイニングアンドビジネスアナリシス・テクノロジーグループの藤巻遼平さんには北米出張中の忙しいなかメンターを引き受けて頂き，スカイプやメールにてプログラミングから証明のフォローまで非常に丁寧な指導をして頂きました．また同グループの森永聡さん，小阪勇気さんにも大変お世話になりました．

研究室外の方にも大変お世話になりました．自然言語処理学講座の小町守さんにはインターンシップ応募書類を添削して頂いたり修士のときから様々なことでお世話になりました．大阪大学の植野剛さんには研究のことや将来のことなどを gmail のチャットでいつも相談に乗って頂きました．京都大学の前田新一先生や五十嵐康伸さんにはバスケットボールによく誘って頂きました．

I am grateful to Prof. Klaus-Robert Müller and Dr. Motoaki Kawanabe for accepting me as a visiting researcher in Technische Universität Berlin. Studying in Berlin is a valuable experience for me. Sometimes I miss a cold winter in Berlin.

最後に陰に陽に自分を支えてくれた親族に感謝します．

Appendix A

Appendix for Chapter 3

A.1. Gaussian Process

Here we consider an approximation of $\psi_h(\theta)$ by the GP. Because $\psi_h(\theta)$ is univariate, we only consider the case in which the input is one-dimensional for simplicity. First we randomly generate N samples $\mathbf{s} \equiv (\theta_1, \ldots, \theta_N)^\top$ from an arbitrary distribution. Note that, unlike the normal regression settings, we know the functional form of $\psi_h(\theta)$, e.g., $\psi_h(\theta) = e^\theta$ for Poisson distribution, and thus outputs $\mathbf{y}_h \equiv (\psi_h(\theta_1), \ldots, \psi_h(\theta_N))^\top$ is noise-free. For a given GP prior, the joint distribution over the outputs $\mathbf{y}_h$ is the Gaussian distribution with zero-mean and a $N \times N$ covariance matrix $\mathbf{K}$, where the (qr)-th element k_{qr} of the covariance $\mathbf{K}$ is represented by a pre-determined covariance function $k(\theta_q, \theta_r)$. For convenience, we employ a Gaussian kernel as the covariance function:

$$k(\theta_q, \theta_r; \alpha) = \exp\left(-\frac{\alpha}{2}\left|\left|\theta_q - \theta_r\right|\right|^2\right). \tag{A.1}$$

The hyper-parameter α controls the smoothness of outputs. Using the Bayes theorem, the predictive distribution of $\psi_{h*} \equiv \psi_h(\theta_*)$ at a new input θ_* with training data $\mathcal{D}_h \equiv \{\mathbf{s}, \mathbf{y}_h\}$ is given by a univariate Gaussian distribution

$$p(\psi_{h*} \mid \theta_*, \mathcal{D}_h) = N(\psi_{h*} \mid m_h(\theta_*), v_h^2(\theta_*)). \tag{A.2}$$

The mean $m_h(\theta_*)$ and the variance $v_h^2(\theta_*)$ are given by

$$m_h(\theta_*) = \mathbf{b}_h^\top \mathbf{k}_*, \quad v_h^2(\theta_*) = k_{**} - \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{k}_* \tag{A.3}$$

where $\mathbf{k}_* = (k_{*1}, \ldots, k_{*N})$, $k_{*n} = k(\theta_*, \theta_n)$, and $\mathbf{b}_h \equiv \mathbf{K}^{-1}\mathbf{y}_h$.

A.2. Derivative of GP

Here we consider to predict a derivative $\psi_h' \equiv \frac{\partial \psi}{\partial \theta}$ at a new input x_*. A straightforward approach is to prepare another GP and independently model ψ_h'. However, ψ_h and

ψ'_h are mutually correlated, and the joint modeling of them improves the accuracy of the function approximation. The covariance function between ψ_h and ψ'_h is given by the following lemma.

Lemma 3 (O'Hagan (1992)) *Let f be a Gaussian process over one-dimensional inputs $a, b \in \mathbb{R}$ where $\mathbb{E}[f(a)] = 0$ and $\mathrm{cov}[f(a), f(b)] = k(a, b)$ with an arbitrary covariance function $k(\cdot, \cdot)$. Then the covariance between (any higher order) derivatives of the outputs of f is represented by a (higher order) derivative of k.*

Proof Since $\mathrm{cov}[\cdot, \cdot]$ and the derivative operator are commutative, then

$$\mathrm{cov}\left[\left.\frac{\partial^n f}{\partial x^n}\right|_{x_i}, \left.\frac{\partial^m f}{\partial x^m}\right|_{x_j}\right] = \left.\frac{\partial^n}{\partial a^n}\frac{\partial^m}{\partial b^m}k\right|_{a=x_i, b=x_j}. \tag{A.4}$$

■

By applying the result of lemma 3 with the Gaussian covariance function, we observe the following corollary.

Corollary 4 *If the covariance function $k(\cdot, \cdot)$ is the Gaussian kernel function* (A.1)*, then the covariance function between an output and its derivative, and that between the derivatives are given by*

$$\begin{aligned}\mathrm{cov}\left[\left.\frac{\partial f}{\partial x}\right|_{x_i}, f(x_j)\right] &= -\mathrm{cov}\left[f(x_i), \left.\frac{\partial f}{\partial x}\right|_{x_j}\right] \\ &= -\alpha(x_i - x_j)k(x_i, x_j), \qquad \text{(A.5a)} \\ \mathrm{cov}\left[\left.\frac{\partial f}{\partial x}\right|_{x_i}, \left.\frac{\partial f}{\partial x}\right|_{x_j}\right] &= \alpha\{1 - \alpha(x_i - x_j)^2\}k(x_i, x_j). \qquad \text{(A.5b)}\end{aligned}$$

Lemma 3 allows us to predict derivatives of outputs and, in addition, to use derivative information as training inputs. To improve the approximation accuracy, we generate N' inputs $\mathbf{s}' \equiv \{\theta_{N+1}, \ldots, \theta_{N+N'}\}$ from an arbitrary distribution and calculate the corresponding derivative observations $\mathbf{y}'_h \equiv \{\psi'_h(\theta_{N+1}), \ldots, \psi'_h(\theta_{N+N'})\}$ as an additional training data set $\mathcal{D}'_h \equiv \{\mathbf{s}', \mathbf{y}'_h\}$.

Now we have a joint distribution of $\{\mathbf{y}_h, \mathbf{y}'_h\}$ with the covariance functions (A.5). The predictive mean and the variance of the output $\psi_h(\theta_*)$ are given by

$$m_h(\theta_*) = \boldsymbol{\beta}_h^\top \boldsymbol{\kappa}_*, \quad v_h^2(\theta_*) = k_{**} - \boldsymbol{\kappa}_*^\top \mathbf{C}^{-1}\boldsymbol{\kappa}_*$$

where $\boldsymbol{\kappa}_* = (\mathbf{k}_*^\top, -\mathbf{k}_*'^\top)^\top$, $\boldsymbol{\beta}_h = \mathbf{C}^{-1}(\mathbf{y}_h^\top, \mathbf{y}_h'^\top)^\top$, and

$$\mathbf{C} = \begin{pmatrix} \mathbf{K}(\mathbf{s}, \mathbf{s}) & \mathbf{K}'(\mathbf{s}, \mathbf{s}') \\ \mathbf{K}'(\mathbf{s}', \mathbf{s}) & \mathbf{K}''(\mathbf{s}', \mathbf{s}') \end{pmatrix}. \tag{A.6}$$

k'_{ij} and k''_{ij} are defined by equation (A.5a) and (A.5b), respectively and $\mathbf{k}'_* = (k'_{*1}, \ldots, k'_{*N'})$. The predictive mean and the variance of the derivative $\psi'_h(\theta_*)$ are also given by

$$m'_h(\theta_*) = \boldsymbol{\beta}_h^\top \boldsymbol{\kappa}'_*, \quad {v'_h}^2(\theta_*) = k''_{**} - {\boldsymbol{\kappa}'_*}^\top \mathbf{C}^{-1} \boldsymbol{\kappa}'_* \tag{A.7}$$

where $\boldsymbol{\kappa}'_* = ({\mathbf{k}'_*}^\top, {\mathbf{k}''_*}^\top)^\top$ and $\mathbf{k}''_* = (k''_{*1}, \ldots, k''_{*N'})$.

A.3. Marginalization of GP with Gaussian Density

In this section, we show that the marginalization of a GP has a closed form solution when the covariance function is a Gaussian kernel and the input follows a Gaussian distribution. For later convenience, first we introduce the following lemma:

Lemma 5 *Let $g_i(x) = \exp(-\frac{\alpha_i}{2} ||x - x_i||^2), i = 1, \ldots, M$ be Gaussian basis functions with scale parameters $\alpha_i \geq 0$ and location parameters x_i. Let y be a random variable following a Gaussian distribution $N(y|x_0, \alpha_0^{-1})$, then, for a finite positive integer n, we observe that*

$$\int y^n \left(\prod_{i=1}^{M} g_i(y) \right) N(y \mid x_0, \alpha_0^{-1}) \mathrm{d}y \tag{A.8}$$

$$= \mathbb{E}[y^n \mid \mathbb{E}_p[x], \beta^{-1}] \sqrt{\frac{\alpha_0}{\beta}} \prod_{i=0}^{M} g_i(\mathbb{E}_p[x]) \tag{A.9}$$

where $\beta = \sum_{i=0}^{M} \alpha_i$ and $p(x)$ is the mixture of the delta functions $p(x) = \sum_{i=0}^{M} \frac{\alpha_i}{\beta} \delta(x - x_i)$. $\mathbb{E}[y^n|\mu, \sigma^2]$ denotes the n-th order moment of a Gaussian distribution $N(y|\mu, \sigma^2)$.

Proof We observe that

$$\log N(y \mid x_0, \alpha_0^{-1}) \prod_{i=1}^{M} g_i(y) \tag{A.10}$$

$$= \log C_0 - \frac{\beta}{2} \sum_{i=0}^{M} \frac{\alpha_i}{\beta} (y - x_i)^2 \tag{A.11}$$

$$= \log C_0 - \frac{\beta}{2} \mathbb{E}_p[(y - x)^2] \tag{A.12}$$

$$= \log C_0 - \frac{\beta}{2} (y^2 - 2\mathbb{E}_p[x] y + \mathbb{E}_p[x^2]) \tag{A.13}$$

$$= \log C_0 - \frac{\beta}{2} \{(y - \mathbb{E}_p[x])^2 + \mathrm{var}_p[x]\} \tag{A.14}$$

where $C_0 = \sqrt{\frac{\alpha_0}{2\pi}}$ is the normalization term and $\mathrm{var}_p[x] = \mathbb{E}_p[x^2] - \mathbb{E}_p[x]^2$. Then we have

$$N(y \mid x_0, \alpha_0^{-1}) \prod_{i=1}^{M} g_i(y) \tag{A.15}$$

$$= C_0 \exp(-\frac{\beta}{2}\mathrm{var}_p[x]) \exp(-\frac{\beta}{2}(y - \mathbb{E}_p[x])^2) \tag{A.16}$$

$$= \frac{C_0}{C_\beta} \exp(-\frac{\beta}{2}\mathrm{var}_p[x]) N(y \mid \mathbb{E}_p[x], \beta^{-1}) \tag{A.17}$$

where $C_\beta = \sqrt{\frac{\beta}{2\pi}}$ is the normalization term. Then we obtain

$$\int y^n N(y \mid x_0, \alpha_0^{-1}) \prod_{i=1}^{M} g_i(y) \mathrm{d}y \tag{A.18}$$

$$= \frac{C_0}{C_\beta} \exp(-\frac{\beta}{2}\mathrm{var}_p[x]) \int y^n N(y \mid \mathbb{E}_p[x], \beta^{-1}) \mathrm{d}y \tag{A.19}$$

$$= \sqrt{\frac{\alpha_0}{\beta}} \exp(-\frac{\beta}{2}\mathrm{var}_p[x]) \mathbb{E}[y^n \mid \mathbb{E}_p[x], \beta^{-1}] \tag{A.20}$$

Since $\mathrm{var}_p[x] = \sum_{i=0}^{M} \frac{\alpha_i}{\beta}(x_i - \mathbb{E}_p[x])^2$,

$$\exp(-\frac{\beta}{2}\mathrm{var}_p[x]) = \prod_{i=0}^{M} \exp(-\frac{\alpha_i}{2}(x_i - \mathbb{E}_p[x])^2) \tag{A.21}$$

$$= \prod_{i=0}^{M} g_i(\mathbb{E}_p[x]) \tag{A.22}$$

■

Note that the result of lemma 5 with $n = 0$ is known as Bayes-Hermite Quadrature (O'Hagan, 1991; Rasmussen and Ghahramani, 2003).

A.4. Proof of Theorem 1

Now we prove theorem 1 by using corollary 4 and Lemma 5.

Proof Corollary 4 and equation (A.3) show that $m^{(p)}(x_*)$, the p-th order derivative of the GP's predictive mean at the new input x_*, represented as a product of a polynomial of x_* and a Gaussian basis function. Then, the marginalization of $m^{(p)}(x_*)$ by a Gaussian distribution $N(x_*|\mu_*, \sigma_*^2)$ can be written as a closed form solution by applying lemma 5. ■

A.5. Some examples of theorem 1

By using the result of theorem 1, we derive specific examples of the expectation approximation required in equation (3.9), (3.12), and (3.17). We assume that θ_* is a Gaussian-distributed random variable with a mean μ_* and a variance λ_*^{-1}. We would like to marginalize the predictive distribution: $\int p(\psi_{h*} \mid \theta_*, \mathcal{D}_h) N(\theta_* \mid \mu_*, \lambda^{-1}) \mathrm{d}\theta_*$. Although the marginal distribution is not a Gaussian[1], the mean and the variance can be obtained when we employ the Gaussian kernel as the covariance function. The predictive mean $\bar{m}_{h*}$ and variance $\bar{v}_{h*}^2$ for the uncertain input θ_* are written as

$$\begin{aligned}
\bar{m}_{h*} &= \iint \psi_{h*} p(\psi_{h*}|\mathcal{D}_h, \theta_*) p(\theta_*|\mu_*, \lambda_*^{-1}) \mathrm{d}\psi_{h*} \mathrm{d}\theta_* \\
&= \mathbb{E}_{\theta_*}[m_h(\theta_*)] = \mathbf{b}_h^\top \mathbb{E}_{\theta_*}[\mathbf{k}_*], && \text{(A.23a)} \\
\bar{v}_{h*} &= \iint (\psi_{h*} - m_{h*})^2 p(\psi_{h*}|\mathcal{D}_h, \theta_*) p(\theta_*|\mu_*, \lambda_*^{-1}) \mathrm{d}\psi_{h*} \mathrm{d}\theta_* \\
&= \mathbf{b}_h^\top \mathbb{E}_{\theta_*}[\mathbf{k}_* \mathbf{k}_*^\top] \mathbf{b}_h + 1 - \mathrm{tr}(\mathbf{K}^{-1} \mathbb{E}_{\theta_*}[\mathbf{k}_* \mathbf{k}_*^\top]) - \bar{m}_{h*}^2. && \text{(A.23b)}
\end{aligned}$$

where

$$\mathbb{E}_{\theta_*}[k_{*i}] = \sqrt{\frac{\lambda_*}{\lambda_* + \alpha}} k(e_i, \mu_*; \lambda_*) k(e_i, \theta_i; \alpha) \tag{A.24}$$

$$\begin{aligned}
\mathbb{E}_{\theta_*}[k_{*i} k_{*j}] = \sqrt{\frac{\lambda_*}{\lambda_* + 2\alpha}} & k(E_{ij}, \mu_*; \lambda_*) \\
& \times k(E_{ij}, \theta_i; \alpha) k(E_{ij}, \theta_j; \alpha), && \text{(A.25)}
\end{aligned}$$

$$e_i = \frac{\lambda_* \mu_* + \alpha \theta_i}{\lambda_* + \alpha}, \quad E_{ij} = \frac{\lambda_* \mu_* + \alpha(\theta_i + \theta_j)}{\lambda_* + 2\alpha}. \tag{A.26}$$

By combining the result of equation (A.7), we have

$$\begin{aligned}
\bar{m}'_{h*} &= \mathbf{b}_h^\top \mathbb{E}_{\theta_*}[\mathbf{k}'_*], && \text{(A.27a)} \\
\bar{v}'_{h*} &= \mathbf{b}_h^\top \mathbb{E}_{\theta_*}[\mathbf{k}'_* (\mathbf{k}'_*)^\top] \mathbf{b}_h + 1 \\
&\quad - \mathrm{tr}(\mathbf{K}^{-1} \mathbb{E}_{\theta_*}[\mathbf{k}'_* (\mathbf{k}'_*)^\top]) - (\bar{m}'_{h*})^2. && \text{(A.27b)}
\end{aligned}$$

where

$$\begin{aligned}
\mathbb{E}_{\theta_*}[k'_{*i}] &= \alpha \mathbb{E}_{\theta_*}[k_{*i}](\theta_i - e_i), && \text{(A.28)} \\
\mathbb{E}_{\theta_*}[k'_{*i} k'_{*j}] &= \alpha^2 \mathbb{E}_{\theta_*}[k_{*i} k_{*j}] \{\theta_i \theta_j - 2E_{ij}(\theta_i + \theta_j) && \text{(A.29)} \\
&\quad + E_{ij}^2 + (\lambda_* + 2\alpha)^{-1}\}. && \text{(A.30)}
\end{aligned}$$

1. We can easily verify this by calculating that the third order derivative of the joint distribution $p(\psi_{h*} \mid \theta_*, \mathcal{D}_h) N(\theta_* \mid \mu_*, \lambda^{-1})$ with respect to θ_* is not zero.

Algorithm 4 PREPARE_GP.

Input: $\vec{\boldsymbol{\theta}}_0$, N
  Sample $\mathbf{s}$ from $\vec{\boldsymbol{\theta}}_0$
  $\mathbf{C} = \mathbf{K}^{-1}(\mathbf{s}, \mathbf{s})$
  for $h = 1, \ldots, H$ **do**
    $\mathbf{y}_h = (\psi_h(s_1), \ldots, \psi_h(s_N))^\top$
    $\boldsymbol{\beta}_h = \mathbf{C}\mathbf{y}_h^\top$
  end for
  Return $\{\boldsymbol{\beta}_h \mid h = 1, \ldots, H\}$

By using the result of lemma 5 with $n = 1$, we also have

$$\begin{aligned}\mathbb{E}_{\theta_*}[\theta_* k'_{*i}] &= \alpha \mathbb{E}_{\theta_*}[k_{*i}]\{\theta_i e_i - e_i^2 - (\lambda_* + \alpha)^{-1}\} \\ &= e_i \mathbb{E}_{\theta_*}[k'_{*i}] - \mathbb{E}_{\theta_*}[k_{*i}](\lambda_* + \alpha)^{-1} \end{aligned} \tag{A.31}$$

Finally we obtain the solutions of the following expectations:

$$\mathbb{E}_q[m_h(\theta_d)] = \mathbf{b}_h^\top \mathbb{E}_{\theta_d}[\mathbf{k}_d] \tag{A.32}$$

$$\mathbb{E}_q[m'_h(\theta_d)] = \mathbf{b}_h^\top \mathbb{E}_{\theta_d}[\mathbf{k}'_d] \tag{A.33}$$

$$\mathbb{E}_q[\mathbf{M}'^{(l)}\{\boldsymbol{\Theta}^{(l)}\}^\top]_{ij} = \begin{cases} \sum_{a=1}^{D_{\setminus l}} \mathbb{E}_q[\theta_{ia}^{(l)} M'^{(l)}_{ia}(\theta_{ia})] & \text{if } i = j \\ \sum_{a=1}^{D_{\setminus l}} \mathbb{E}_q[\theta_{ia}^{(l)}]\mathbb{E}_q[M'^{(l)}_{ja}(\theta_{ja})] & \text{otherwise.} \end{cases} \tag{A.34}$$

The further expectations in equations (A.32)-(A.34) are given by equations (A.24), (A.28), and (A.31).

A.6. Algorithm PREPARE_GP

Here we summarize the subroutine PREPARE_GP, a prepossessing procedure for the GP approximation used in algorithm 1 and 2. Figure 4 shows the pseudo code of PREPARE_GP.

For the GP approximation in the M-step (section 3.3.2), we need to determine the training inputs $\mathbf{s}$. For accurate approximation, it is important to cover an area by the training inputs in which the posterior is dense and/or functions ψ and ψ' take a large value. Thus, we randomly choose N one-dimensional inputs θ_n $(n = 1, \ldots, N)$ from each dimension of the posterior mean $\vec{\boldsymbol{\theta}}_0 \equiv \mathbf{W}\vec{\mathbf{z}}_0$ according to the weights $\boldsymbol{\psi}(\vec{\boldsymbol{\theta}}_0)$. In an area in which the inputs are sparse, on the other hand, the mean of GP is close to zero because the mean function of the GP prior is set to zero. This property would be

problematic when we apply the gradient-based optimization, since the cost function $\bar{\mathcal{L}}(\theta)$ diverges when $\theta \to \infty$. To avoid this problem, we use a barrier function instead of the mean of GP when the input of GP is out of the area $[\min(\mathbf{s}), \max(\mathbf{s})]$. We use the zeroth-order delta method as the barrier function.

Appendix B

Appendix for Chapter 4

B.1. Interpretation of EM-like iterative method

In the linear model (4.4), the EM-like iterative algorithm can be interpreted as the EM algorithm with the following approximations. In the E-step, given the estimated mean of the latent variables $\hat{\mathbf{Z}}_{l-1}$, we use the predictive distribution $p(\mathbf{X}|\mathcal{D}, \hat{\mathbf{Z}}_{l-1})$ as a posterior of $\mathbf{Z}$ in $(l-1)$-th iteration, i.e., $p(\mathbf{Z}|\mathcal{D}) \approx p(\mathbf{X}|\mathcal{D}, \hat{\mathbf{Z}}_{l-1})$. In the M-step, the expected log-likelihood is replaced by its zeroth-order approximation, i.e.,

$$\mathbb{E}_{\mathbf{Z}}[J(\mathbf{w}, \mathbf{Z})] \approx J(\mathbf{w}, \hat{\mathbf{Z}}_l) \tag{B.1}$$

where $\hat{\mathbf{Z}}_l$ denotes the mean of the approximation of the posterior $p(\mathbf{X}|\mathcal{D}, \hat{\mathbf{Z}}_{l-1})$, i.e., the predictive values of $\mathbf{X}$ at $(l-1)$-th iteration. With the approximation, the solution of $\mathbf{w}$ in the linear Equation (4.13) is also the solution in the M-step. Note that the approximated expected log-likelihood (B.1) is no longer the lower bound of the marginal likelihood; it does not guarantee the convergence of the algorithm.

B.2. Minimizer of loss function of KNN

By taking differential of the loss function (4.16) with respect to y_{ik} and setting it to 0, we obtain the predictive value of a missing element x_{ik} for $(i,k) \notin \mathcal{I}$. The derivative is written as

$$\begin{aligned}
&\frac{\partial}{\partial y_{ik}} \sum_{(i,k)\notin\mathcal{I}} \sum_{j\in\mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:})(x_{jk} - y_{ik})^2 \\
&= 2 \sum_{j\in\mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:})(x_{jk} - y_{ik}) = 0.
\end{aligned}$$

Then we have

$$\begin{aligned}
y_{ik} \sum_{j\in\mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:}) &= \sum_{j\in\mathcal{R}_i} s(\mathbf{x}_{i:}, \mathbf{x}_{j:}) x_{jk}, \\
y_{ik} &= \hat{x}_{ik}^{\mathrm{KNN}}.
\end{aligned}$$

B.3. Details about experiment of collaborative filtering

In this section, we describe the experimental settings used in the experiment of collaborative filtering and its additional results.

B.3.1 Proposed methods

For the covariance functions, we prepared the RBF kernel $\Sigma(\mathbf{x}, \mathbf{x}') = \Omega(\mathbf{x}, \mathbf{x}') = \exp(-\lambda ||\mathbf{x} - \mathbf{x}'||^2)$ and a scaled linear kernel $\frac{1}{D}\sum_{d=1}^{D} x_d x'_d$ where D is the dimensionality of $\mathbf{x}$ and $\mathbf{x}'$.

We choose the hyper-parameters by three-fold cross validation from candidates $\sigma^2 \in \{1, 0.5, 0.1, 0.05\}$ and $\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ for the RBF kernel and $\sigma^2 \in \{10, 1, 0.1\}$ for the linear kernel. Table B.1 shows the chosen hyper-parameters.

Table B.1: Selected hyper-parameters.

		Tensor GP	Self-measuring	Combination
RBF	σ^2	0.1	0.5	0.5
	λ	0.1	0.001	0.001
Linear	σ^2	10	0.1	0.1

We set the tolerance of the conjugate gradient ϵ as 10^{-3}. As the initial values of $\{z_{ik}\}$, we use the row-mean for $\mathbf{\Sigma}$ and the column mean for $\mathbf{\Omega}$. For fair comparison, the number of the EM-like iteration is determined by early stopping with a validation set randomly drawn 5% of training data.

B.3.2 KNN and matrix factorization

We used `MyMediaLite`[1] as the implementations of KNN and matrix factorization. We set the hyper-parameters by following the examples specially recommended for the Movielens 100k dataset.[2]

B.3.3 CUR decomposition

We used the Matlab code[3] of CUR decomposition (Mahoney and Drineas, 2009), which was implemented by Christos Boutsidis.

We set the rank of the CUR decomposition as 64, selected by three-fold cross validation from rank $\in \{2^1, 2^2, 2^3, \ldots, 2^{10}\}$.

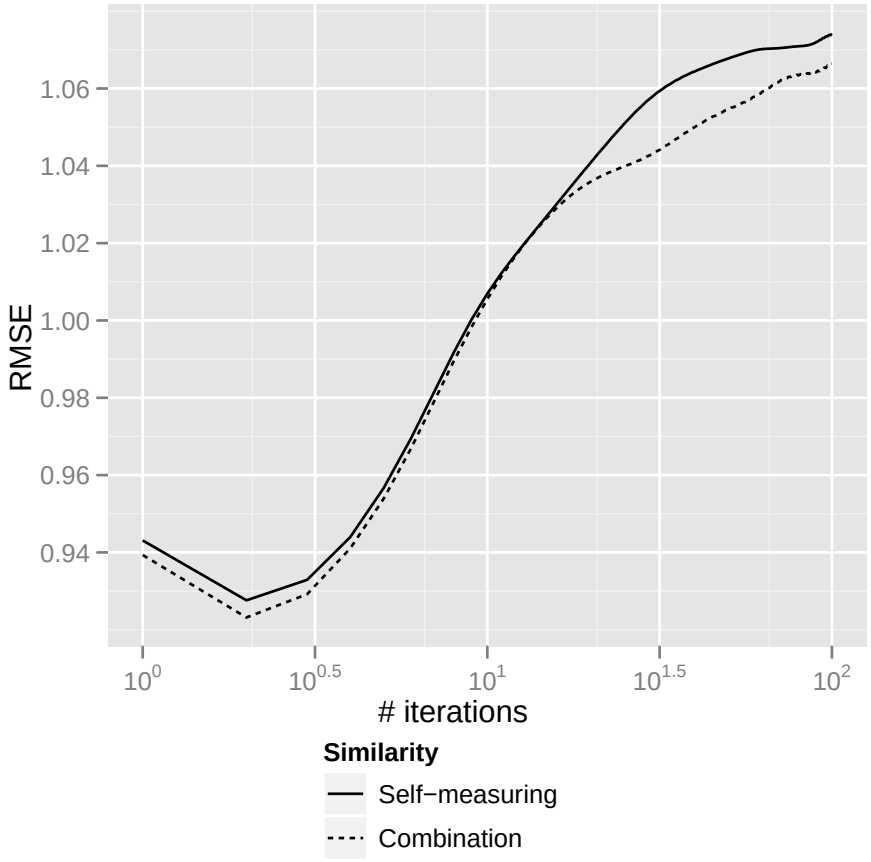

Figure B.1: Test errors v.s. the number of iterations of the EM-like heuristic with the RBF kernel.

B.3.4 Behaviour of EM-like iteration

Figure B.1 shows the EM-like heuristic drastically improves the prediction accuracy at the second iteration, while the third or further iterations make the prediction performance worse.

B.4. Details about experiment of tensor data sets

In this section, we describe the experimental settings used in the experiment of the tensor data sets and its additional results.

B.4.1 pTucker

We implemented the pTucker model by the EM algorithm. Note that missing values did not involve throughout the algorithm as the original implementation (Chu and Ghahramani, 2009).

B.4.2 Results with various settings of σ^2

Figure B.2 shows how the regularization parameter[4] effects to reconstruction of unobserved elements. In figure B.2, the result of the reconstruction is better for small σ^2. When the regularization is strong (i.e. σ^2 is large), the reconstruction result tend to be unshaped and blurred. Nevertheless, the regularization caused by σ^2 has a different effect compared to the low-rank constraint. The corresponding RMSEs are shown in figure B.3. Figure B.4 and B.5 are the results for "Sugar process" data set and show similar results.

1. `http://www.ismll.uni-hildesheim.de/mymedialite`
2. `http://www.ismll.uni-hildesheim.de/mymedialite/examples/datasets.html`
3. `http://www.cs.rpi.edu/~boutsc/files/AlgorithmCUR.m`
4. Originally σ^2 is a coefficient of L_2-norm regularization of $\mathbf{W}$ (see Equation (4.4).)

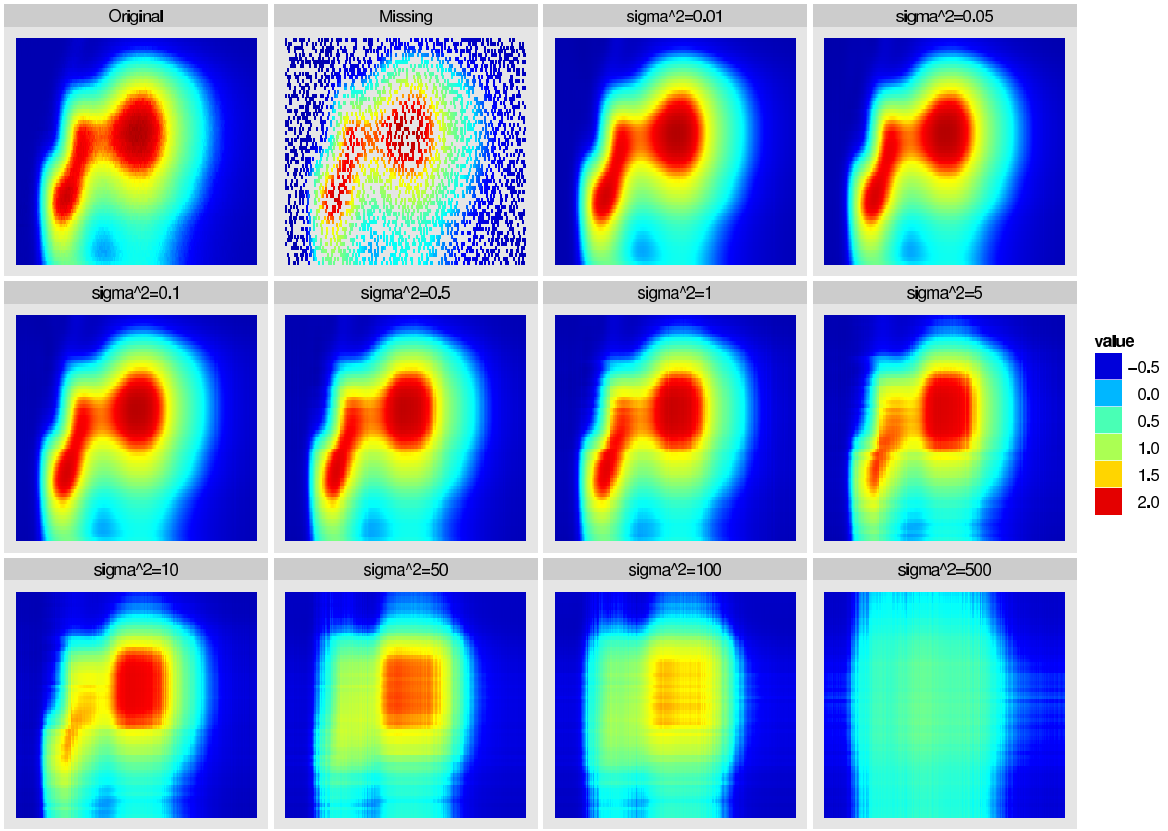

Figure B.2: Reconstruction results of the "Amino acids" data set with various σ^2 settings.

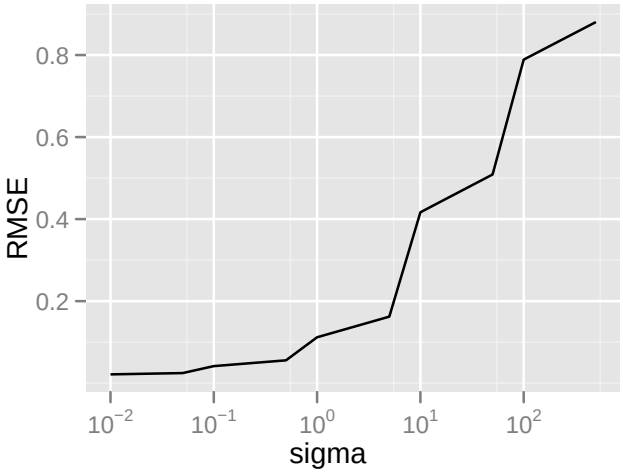

Figure B.3: Testing errors of the "Amino acids" data set with various σ^2 settings.

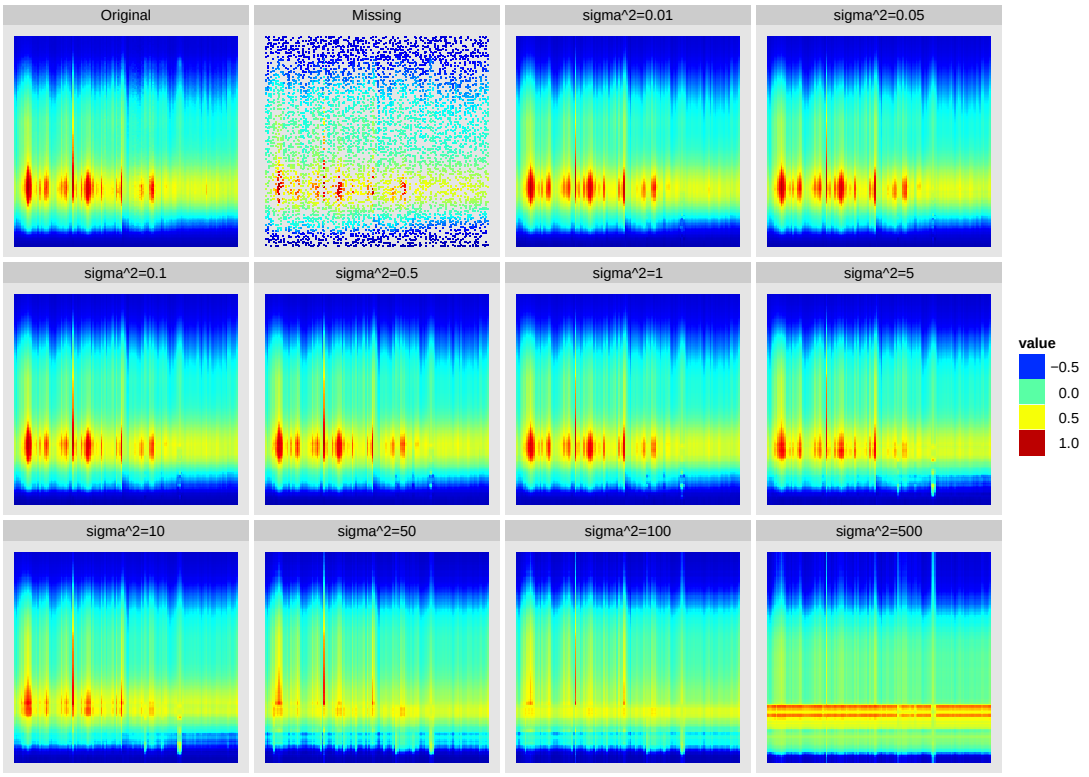

Figure B.4: Reconstruction results of the "Sugar process" data set with various σ^2 settings.

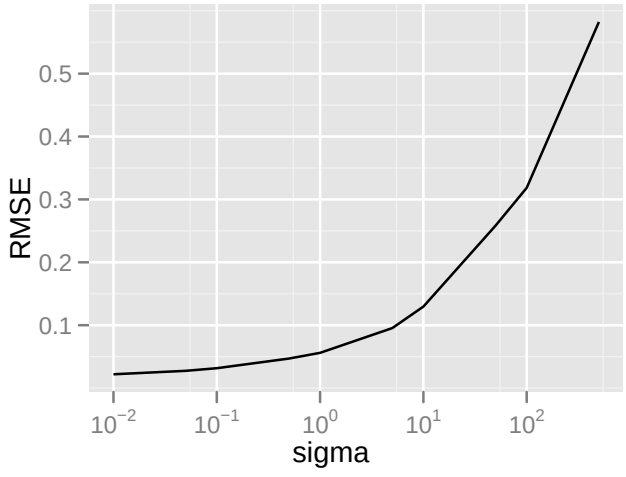

Figure B.5: Testing errors of the "Sugar process" data set with various σ^2 settings.

Bibliography

J. Abernethy, F. Bach, T. Evgeniou, and J. P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *J. Mach. Learn. Res.*, 10:803–826, June 2009. ISSN 1532-4435. URL http://portal.acm.org/citation.cfm?id=1577098.

E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1):6–20, 2009. ISSN 1041-4347. doi: 10.1109/TKDE.2008.112. URL http://dx.doi.org/10.1109/TKDE.2008.112.

A. Agovic, A. Banerjee, and S. Chatterjee. Probabilistic matrix addition. In *International Conference on Machine Learning (ICML)*, 2011.

B. Alexeev, M. Forbes, and J. Tsimerman. Tensor rank: some lower and upper bounds. In *IEEE Conference on Computational Complexity*, pages 283–291. IEEE Computer Society, Feb. 2011.

C. Andersson and R. Bro. The n-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, Aug. 2000. ISSN 01697439. doi: 10.1016/S0169-7439(00)00071-X. URL http://dx.doi.org/10.1016/S0169-7439(00)00071-X.

J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 9+, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015394. URL http://dx.doi.org/10.1145/1015330.1015394.

D. P. Bertsekas and D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, Sept. 1999. ISBN 1886529000. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/1886529000.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2007. ISBN 0387310738. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0387310738.

D. M. Blei and J. D. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems 21*, 2007.

E. V. Bonilla, K. M. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008. URL `nips08.pdf`.

L. Bottou and Y. LeCun. Large scale online learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press. URL `http://leon.bottou.org/papers/bottou-lecun-2004`.

R. Bro. *Multi-way analysis in the food industry. Models, algorithms and applications*. PhD thesis, Department of Analytical Chemistry, University of Amsterdam, Amsterdam, 1998.

W. Chu and Z. Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, 2009.

A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S. Amari. Non-Negative tensor factorization using alpha and beta divergences. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2007.*, volume 3, pages III–1393–III–1396, June 2007. doi: 10.1109/ICASSP.2007.367106. URL `http://dx.doi.org/10.1109/ICASSP.2007.367106`.

M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal components analysis to the exponential family. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.2812`.

L. De Lathauwer and B. De Moor. From matrix to tensor: Multilinear algebra and signal processing. In *INSTITUTE OF MATHEMATICS AND ITS APPLICATIONS CONFERENCE SERIES*, volume 67, pages 1–16. Citeseer, 1998.

L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-(R_1,R_2,. . .,R_N) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000a. ISSN 0895-4798. doi: 10.1137/S0895479898346995. URL `http://dx.doi.org/10.1137/S0895479898346995`.

L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, Mar. 2000b. ISSN 0895-4798. doi: 10.1137/S0895479896305696. URL `http://dx.doi.org/10.1137/S0895479896305696`.

P. Drineas and M. W. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, Dec. 2005. ISSN 1532-4435. URL http://portal.acm.org/citation.cfm?id=1194916.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, Dec. 2009. ISSN 1532-4435. URL http://portal.acm.org/citation.cfm?id=1577069.1755882.

D. M. Dunlavy, T. G. Kolda, and W. P. Kegelmeyer. Multilinear algebra for analyzing data with multiple linkages. Technical Report SAND2006-2079, Sandia National Laboratories, 2006. URL http://www.prod.sandia.gov/cgi-bin/techlib/access-control.pl/2006/062079.pdf.

P. Dutilleul. The mle algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123, 1999. doi: 10.1080/00949659908811970. URL http://dx.doi.org/10.1080/00949659908811970.

M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University. URL http://www.cds.caltech.edu/\~{}maryam.

S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010+, Feb. 2011. ISSN 0266-5611. doi: 10.1088/0266-5611/27/2/025010. URL http://dx.doi.org/10.1088/0266-5611/27/2/025010.

L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, Nov. 2007. ISBN 0262072882. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0262072882.

D. L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, Jan. 1997. ISSN 00189219. doi: 10.1109/5.554205. URL http://dx.doi.org/10.1109/5.554205.

R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84, 1970.

J. Hastad. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, Dec. 1990. ISSN 01966774. doi: 10.1016/0196-6774(90)90014-6. URL http://dx.doi.org/10.1016/0196-6774(90)90014-6.

P. D. Hoff. Separable covariance arrays via the tucker product, with applications to multivariate relational data. Aug. 2010. URL http://arxiv.org/abs/1008.2169.

H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, and K. Tsuda. Link propagation: A fast semi-supervised learning algorithm for link prediction. In *SDM*, pages 1099–1110. SIAM, 2009a.

H. Kashima, S. Oyama, Y. Yamanishi, and K. Tsuda. On pairwise kernels: An efficient alternative and generalization analysis. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, pages 1030–1037, Berlin, Heidelberg, 2009b. Springer-Verlag. ISBN 978-3-642-01306-5. doi: 10.1007/978-3-642-01307-2_110. URL http://dx.doi.org/10.1007/978-3-642-01307-2_110.

E. M. Knorr, R. T. Ng, V. Tucakov, E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, Feb. 2000. ISSN 1066-8888. doi: 10.1007/s007780050006. URL http://dx.doi.org/10.1007/s007780050006.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, Sept. 2009. doi: 10.1137/07070111X. URL http://dx.doi.org/10.1137/07070111X.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10:777–801, June 2009. ISSN 1532-4435. URL http://portal.acm.org/citation.cfm?id=1577097.

J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 497–504, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273559. URL http://dx.doi.org/10.1145/1273496.1273559.

J. Liu, P. Musialski, P. Wonka, and J. P. Ye. Tensor completion for estimating missing values in visual data. In *ICCV09*, pages 2114–2121, 2009.

W. Liu, X. Tang, and J. Liu. Bayesian tensor inference for sketch-based facial photo hallucination. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2141–2146, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL http://portal.acm.org/citation.cfm?id=1625621.

X. Liu, L. De Lathauwer, F. Janssens, and B. De Moor. Hybrid clustering on multiple information sources via HOSVD. In *The Seventh International Symposium on Neural Networks (ISNN 2010)*, 2010.

M. W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, Jan. 2009. doi: 10.1073/pnas.0803205106. URL http://dx.doi.org/10.1073/pnas.0803205106.

J. Ma**v**zgut, P. Ti**v**no, M. Bodén, and H. Yan. Multilinear decomposition and topographic mapping of binary tensors. In *Proceedings of the 20th international conference on Artificial neural networks: Part I*, ICANN'10, pages 317–326, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15818-8, 978-3-642-15818-6. URL http://portal.acm.org/citation.cfm?id=1886399.

P. McCullagh and J. A. Nelder. *Generalized Linear Models, Second Edition (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 2 edition, Aug. 1989. ISBN 0412317605. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0412317605.

S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian exponential family PCA. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, 2009.

M. Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):24–40, 2011. URL http://onlinelibrary.wiley.com/doi/10.1002/widm.1/full.

M. Mørup and L. K. Hansen. Automatic relevance determination for multiway models. *Journal of Chemometrics, Special Issue: In Honor of Professor Richard A. Harshman*, 23(7-8):352–363, 2009. URL http://www2.imm.dtu.dk/pubdb/p.php?5806.

S. Nakajima, M. Sugiyama, and R. Tomioka. Global analytic solution for variational bayesian matrix factorization. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1768–1776. 2010.

S. Nakajima, M. Sugiyama, and D. Babacan. On bayesian PCA: Automatic dimensionality selection and analytic solution. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 497–504, New York, NY, USA, June 2011. ACM.

A. O'Hagan. Some Bayesian numerical analysis. *Bayesian Statistics*, 4:345–363, 1992.

A. O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, Oct. 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191. URL http://dx.doi.org/10.1109/TKDE.2009.191.

P. Pudlák, V. Rödl, and J. Sgall. Boolean circuits, tensor ranks, and communication complexity. *SIAM J. Comput.*, 26:605–633, June 1997. ISSN 0097-5397. doi: 10.1137/S0097539794264809. URL http://dx.doi.org/10.1137/S0097539794264809.

C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. *Advances in Neural Information Processing Systems*, 15(15):505–512, 2003.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2006. ISBN 026218253X. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/026218253X.

S. Rendle and L. S. Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: 10.1145/1718487.1718498. URL http://dx.doi.org/10.1145/1718487.1718498.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9351.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008a.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 25, 2008b.

B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372071. URL http://dx.doi.org/10.1145/371920.372071.

A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning*, pages 792–799. ACM New York, NY, USA, 2005.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 1994. URL http://portal.acm.org/citation.cfm?id=865018.

M. Signoretto, R. Van de Plas, B. De Moor, and J. A. K. Suykens. Tensor versus matrix completion: A comparison with application to spectral data. *Signal Processing Letters, IEEE*, 18(7):403–406, July 2011. ISSN 1070-9908. doi: 10.1109/LSP.2011.2151856. URL http://dx.doi.org/10.1109/LSP.2011.2151856.

A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 650–658, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401969. URL http://dx.doi.org/10.1145/1401890.1401969.

J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, New York, NY, USA, 2006. ACM Press. ISBN 1595933395. doi: 10.1145/1150402.1150445. URL http://dx.doi.org/10.1145/1150402.1150445.

L. Tang, X. Wang, and H. Liu. Uncovering groups via heterogeneous interaction analysis. In *IEEE International Conference on Data Mining (ICDM'09)*, 2009.

D. Tao, M. Song, X. Li, J. Shen, J. Sun, X. Wu, C. Faloutsos, and S. J. Maybank. Bayesian Tensor Approach for 3-D Face Modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(10):1397–1410, 2008.

R. Tomioka, K. Hayashi, and H. Kashima. Estimation of low-rank tensors via convex optimization. Mar. 2011. URL http://arxiv.org/abs/1010.0789.

L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, Sept. 1966. doi: 10.1007/BF02289464. URL http://dx.doi.org/10.1007/BF02289464.

S. V. N. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1449–1456. MIT Press, Cambridge, MA, 2007.

J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 501–508, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148257. URL http://dx.doi.org/10.1145/1148170.1148257.

M. Wedel and W. Kamakura. Factor analysis with (mixed) observed and latent variables in the exponential family. *Psychometrika*, 66(4):515–530, Dec. 2001. ISSN 0033-3123. doi: 10.1007/BF02296193. URL http://dx.doi.org/10.1007/BF02296193.

L. Xiong, X. Chen, T. K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SIAM Data Mining 2010 (SDM 10)*, 2010.

K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu. Stochastic relational models for discriminative link prediction. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1553–1560. MIT Press, Cambridge, MA, 2007.

K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 211–218, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1571979. URL http://dx.doi.org/10.1145/1571941.1571979.

Publication List

Refereed Journal Papers

1. 自己計測類似度を用いたマルチタスクガウス過程．林浩平，竹之内高志，冨岡亮太，鹿島久嗣．採録決定．人工知能学会論文

International Conferences and workshops

1. Self-measuring Similarity for Multi-task Gaussian Process. Kohei Hayashi, Takashi Takenouchi, Ryota Tomioka, and Hisashi Kashima. In Proceedings of the ICML Workshop on Unsupervised and Transfer Learning, 2011.

2. Exponential family tensor factorization for missing-values prediction and anomaly detection. Kohei Hayashi, Takashi Takenouchi, Tomohiro Shibata, Yuki Kamiya, Daishi Kato, Kazuo Kunieda, Keiji Yamada, and Kazushi Ikeda. IEEE 10th International Conference on Data Mining (ICDM), pp. 216 225. 2010.

Domestic Conferences and workshops

1. カーネル法に基づく行列あるいはテンソル補完．林浩平，竹之内高志，冨岡亮太，鹿島久嗣．第 14 回情報論的学習理論ワークショップ (IBIS'11), 2011.

2. 大規模データのための指数族テンソル因子化法. 林浩平, 竹之内高志, 柴田智広, 池田和司. 第 13 回情報論的学習理論ワークショップ (IBIS'10), 2010.

3. 分布の違いを考慮したテンソル因子化法. 林浩平, 池田和司. 第 54 回システム制御情報学会研究発表講演会, 2010.

4. 指数族テンソル因子化法による欠損値予測と異常検知. 林浩平, 竹之内高志, 柴田智広, 神谷祐樹, 加藤大志, 國枝和雄, 山田敬嗣, 池田和司. 人工知能学会研究会資料, SIG-DMSM-A903-13, 2010.

5. Sparse exponential family PCA with heterogeneous attributes. 林浩平, 竹之内高志, 柴田智広, 池田和司. 第 12 回情報論的学習理論ワークショップ (IBIS'09), 2009.