Bayesian Latent Factor Model for Higher-order Data

Zerui Tao ZERUI.TAO@RIKEN.JP
Xuyang Zhao ZHAO@SIP.TUAT.AC.JP
Toshihisa Tanaka TANAKAT@CC.TUAT.AC.JP
Department of Electrical and Electronic Engineering
Tokyo University of Agriculture and Technology, Tokyo, Japan
RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan

Qibin Zhao[*] QIBIN.ZHAO@RIKEN.JP
RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Latent factor models are canonical tools to learn low-dimensional and linear embedding of original data. Traditional latent factor models are based on low-rank matrix factorization of covariance matrices. However, for higher-order data with multiple modes, i.e., tensors, this simple treatment fails to take into account the mode-specific relations. This ignorance leads to inefficiency in analysis of complex structures as well as poor data compression ability. In this paper, unlike covariance matrices, we investigate high-order covariance tensor directly by exploiting tensor ring (TR) format and propose the Bayesian TR latent factor model, which can represent complex multi-linear correlations and achieves efficient data compression. To overcome the difficulty of finding the optimal TR-ranks and simultaneously imposing sparsity on loading coefficients, a multiplicative Gamma process (MGP) prior is adopted to automatically infer the ranks and obtain sparsity. Then, we establish an efficient parameter-expanded EM algorithm to learn the maximum a posteriori (MAP) estimate of model parameters. Finally, we evaluate our model on covariance estimation, latent factor learning and image inpainting problems.

Keywords: Bayesian latent factor model, Tensor ring decomposition, MGP prior

1. Introduction

Latent factor models provide promising tools for inferring latent structures and dimension reduction. Traditional latent factor models aim to tackle with vector features and seek for low-dimensional linear embedding of original data. Specifically, supposing the data have $P \times P$ covariance matrix $\boldsymbol{V}$, latent factor models find a low-rank representation $\boldsymbol{V} = \boldsymbol{W}\boldsymbol{W}^\intercal + \boldsymbol{\Sigma}$, where $\boldsymbol{W} \in \mathbb{R}^{P\times K}$ is the loading matrix with $K \ll P$ and $\boldsymbol{\Sigma}$ is diagonal. Adopting this approximation, we can use K latent factors to represent the original data for downstreaming learning tasks, such as clustering and classification. Latent factor models have achieved great success in sparse factor learning (Carvalho et al., 2008), covariance estimation (Bhattacharya and Dunson, 2011), canonical correlation analysis (Zhao et al., 2016), covariance regression (Fox and Dunson, 2015) and so on.

[*] Corresponding author

Despite the achievements of traditional latent factor models, they are not designed to model higher-order data, i.e., tensors. When dealing with tensor data, one approach is to vectorize them and then apply traditional models. However, this naïve treatment may suffer from the curse of dimensionality and fail to take into account the mode-specific relations. First, recall that the size of the loading matrix is $P \times K$, where P grows exponentially with the order of tensors. Second, there are many mode-specific relations among higher-order data and the vectorization operation destroys original structures. Hence, this over-simplification leads to inefficiency in modeling complex latent structures and poor data compression ability. On the other hand, traditional tensor decomposition also has limitations for higher-order data. For example, CP decomposition is not expressive and the size of Tucker decomposition grows exponentially with the tensor order.

To overcome these drawbacks, we try to leverage the merits of tensor networks (TNs) (Cichocki et al., 2016) to factor models. TNs are shown to be powerful tools to represent higher-order tensors, since their sizes grow linearly with tensor orders. In this work, instead of finding low-rank approximation of *covariance matrices*, our motivation is to directly investigate tensor decomposition for *covariance tensors*. For higher-order data, we reckon that the covariances can be naturally represented by tensors. For example, for a matrix data $\boldsymbol{Y}^{(n)} \in \mathbb{R}^{I \times J}$, the covariance $\boldsymbol{\mathcal{V}}$ is an order-4 tensor of shape $I \times J \times I \times J$, where $\boldsymbol{\mathcal{V}}_{ijmn} = \mathrm{var}(\boldsymbol{Y}_{ij}, \boldsymbol{Y}_{mn})$. To model the covariance tensor, we suppose that it admits the tensor ring (TR) format (Zhao et al., 2016), which is a TN with linear structure. By exploiting the TR format on covariance tensors, we factorize the loading matrix to several small core tensors, which has much more compact formulation than the traditional latent factor models, due to the compression ability of TR format. Meanwhile, the core tensors are stacked in a hierarchical manner, which is more expressive and capable of modeling complex multi-linear relations.

Since the TR-ranks is a vector, it is hard to tune the TR-ranks as well as the factor numbers, which are shown to be essential to the performance. Moreover, it is desirable to obtain sparse loading core tensors for learning interpretable latent factors. To address these issues, we extend the multiplicative Gamma process (MGP) prior (Bhattacharya and Dunson, 2011) to TR format, which automatically infer the TR-ranks and obtain sparsity in loading coefficients simultaneously. Then, we establish efficient Parameter-eXpanded Expectation Maximization (PX-EM) algorithm to find the maximum a posteriori (MAP) estimate of model parameters. Finally, we validate our model on covariance estimation, latent factor learning and tensor completion. The results show that our model has better performance in modeling high dimensional data.

Notations Tensors are denoted by bold calligraphic letters, e.g., $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_D}$. Matrices are denoted by bold capital letters, e.g., $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$. Vectors are denoted by bold letters, e.g., $\boldsymbol{x} \in \mathbb{R}^{I}$. We denote a sequence of indexes by upright boldface letter, e.g., $\mathbf{i} = [i_1, \ldots, i_D]$ and $\mathbf{i}_{-d} = [i_1, \ldots, i_{d-1}, i_{d+1}, \ldots, i_D]$. We use subscripts to denote the elements, e.g., $\boldsymbol{\mathcal{X}}_{\mathbf{i}}$ is the $\mathbf{i}$-th element of $\boldsymbol{\mathcal{X}}$. We adopt the *little-endian* convention[1] for indexing, denoted by an overline on the index. For example, if $\boldsymbol{x} = \mathrm{vec}(\boldsymbol{\mathcal{X}})$, then the $\bar{\mathbf{i}}$-th element of $\boldsymbol{x}$ is $\boldsymbol{\mathcal{X}}_{\mathbf{i}}$. Notation "$\otimes$" denotes Kronecker product and "$*$" denotes Hadamard product. Normal, matrix normal and Gamma distributions are denoted as $\mathcal{N}$, $\mathcal{MN}$ and Ga respectively.

1. In little-endian convention, we have $\bar{\mathbf{i}} = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 I_2 + \cdots + (i_D - 1)I_1 \cdots I_{D-1}$.

2. Related Work

Traditional latent factor models have long history in statistics. Sparse Bayesian techniques are widely used to find sparse latent factors. For example, Carvalho et al. (2008) adopted the spike-and-slab prior to shrink the latent factors. Bhattacharya and Dunson (2011) proposed latent factor model with the MGP prior, which is able to induce infinite number of sparse factors. More recently, Zhao et al. (2016) designed a hierarchical prior for group sparsity. Besides these vector-based models, the latent factors models are also applied on matrix and tensor data recently. Wang et al. (2019) proposed the matrix factor model for time series analysis, by adopting a bi-linear construction. Chen et al. (2019); Han et al. (2020) extended the matrix factor model to higher-order cases by assuming the data admits Tucker format. Our approach is quite different from these tensor-based factor models. First, we use Bayesian techniques to automatically generate sparse factors, while their models are based on power iterations and highly rely on truncations. Second, instead of using the Tucker format, we adopt much expressive tensor network structures. Indeed, Tucker factor model is equivalent to the Kronecker structure of the covariance matrix, while the matrix-TR format is a more expressive extension (Cichocki et al., 2016).

Our model is also related to Bayesian tensor factorizations. Adopting the fully Bayesian framework, Zhao et al. (2015) proposed CP model with automatic relevance determination (ARD) prior to automatically infer tensor ranks. Similarly, Rai et al. (2014); Stevens et al. (2017) applied MGP prior on CP decomposition. Also, Xu et al. (2012); Zhe et al. (2015) proposed non-parametric Tucker decomposition. Very recently, Hawkins and Zhang (2021) proposed a Bayesian neural network with TT representation of the coefficients and Long et al. (2020) established a variational inference (VI) algorithm for TR completion with ARD prior. However, the ARD prior is not able to induce local sparsity and we focus on different problems and establish different algorithms in this work.

3. Preliminaries

3.1. Bayesian Latent Factor Model

Considering N observed data $\{\boldsymbol{y}^{(n)}\}_{n=1}^{N} \in \mathbb{R}^P$ the generic form of a latent factor model is

$$\boldsymbol{y}^{(n)} = \boldsymbol{W}\boldsymbol{\eta}^{(n)} + \boldsymbol{\epsilon}^{(n)}, \quad \forall n = 1, \dots, N, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{P\times K}$ is called the loading matrix, $\boldsymbol{\eta}^{(n)} \in \mathbb{R}^K$ are latent factors and $\boldsymbol{\epsilon}^{(n)} \in \mathbb{R}^P$ are noises. The latent factors are supposed to follow standard Gaussian distribution, i.e., $\boldsymbol{\eta}^{(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_K)$. Moreover, suppose $\boldsymbol{\epsilon}^{(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1^2, \dots, \sigma_P^2)$. Under such conditions, we have $\boldsymbol{y}^{(n)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{V})$, where $\boldsymbol{V} = \boldsymbol{W}\boldsymbol{W}^\intercal + \boldsymbol{\Sigma}$. Hence, the main task of latent factor models is to find a low-rank representation of the covariance matrix.

When applying latent factor models, low dimensional and sparse latent factors are desirable. Bayesian shrinkage techniques are developed to induce low-rank and sparse factors. Specifically, Bhattacharya and Dunson (2011) designed a Multiplicative Gamma Process (MGP) prior to encourage both global and local sparsity. Under the MGP framework, the prior distribution of the loading matrix is set as

$$\boldsymbol{W}_{ij} \mid \phi_{ij}, u_j \sim \mathcal{N}(0, \phi_{ij}^{-1} u_j^{-1}), \quad \phi_{ij} \sim Ga(\nu, \nu), \quad u_j = \prod_{l=1}^{j} \delta_l, \quad \delta_l \sim Ga(\alpha, 1),$$

for $i = 1, \ldots, P$, $j = 1, \ldots, K$, where u_j are global shrinkage parameters for low-rankness and ϕ_{ij} are local shrinkage parameters for sparsity.

3.2. Tensor Ring Decomposition

Now we introduce some basics of the Tensor Ring (TR) decomposition (Zhao et al., 2016), which is a linearly structured tensor network. In particular, when one of the TR-ranks becomes 1, the TR format reduces to the Tensor Train (TT) format (Oseledets, 2011). Hence, all the properties discussed here are applied for TT format. More about tensor algebra can be found in the survey articles Kolda and Bader (2009); Cichocki et al. (2016).

Tensor Unfolding For an order-3 tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, the lateral slices are $\boldsymbol{\mathcal{X}}_{:j:}, \forall j = 1, \ldots, J$. In this paper, we commonly denote the lateral slices as $\boldsymbol{X}[j]$. Tensors can be transformed to matrices by permutations and reshapings. We hereby introduce two types of matricization operations,

1. **Mode-d unfolding**: For $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$, the mode-$d$ unfolding is denoted as $\boldsymbol{X}_{[d]} \in \mathbb{R}^{I_d \times \prod_{j \neq d} I_j}$, and

$$\boldsymbol{\mathcal{X}}_{[d]}(i_d, \overline{i_{d+1} \cdots i_D i_1 \cdots i_{d-1}}) = \boldsymbol{\mathcal{X}}(i_1, \ldots, i_D),$$

   where $\boldsymbol{\mathcal{X}}(i_1, \ldots, i_D)$ is a Matlab-like notation representing the elements of $\boldsymbol{\mathcal{X}}$.

2. **Classical mode-d unfolding**: For $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$, the classical mode-$d$ unfolding is denoted as $\boldsymbol{X}_{(d)} \in \mathbb{R}^{I_d \times \prod_{j \neq d} I_j}$, and

$$\boldsymbol{\mathcal{X}}_{(d)}(i_d, \overline{i_1 \cdots i_{d-1} i_{d+1} \cdots i_D}) = \boldsymbol{\mathcal{X}}(i_1, \ldots, i_D).$$

Tensor Ring Format For an order-D tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$, the TR format denoted as

$$\boldsymbol{\mathcal{X}} = \ll \boldsymbol{\mathcal{Q}}^{(1)}, \ldots, \boldsymbol{\mathcal{Q}}^{(D)} \gg, \tag{2}$$

where $\boldsymbol{\mathcal{Q}}^{(d)} \in \mathbb{R}^{R_d \times I_d \times R_{d+1}}, \forall d = 1, \ldots, D$ are core tensors and $R_{D+1} = R_1$. The sequence $\{R_d\}_{d=1}^{D}$ is called TR-rank. Each element of the full tensor $\boldsymbol{\mathcal{X}}$ can be expressed as matrix product of the core tensors, namely, $\boldsymbol{\mathcal{X}}_{\mathbf{i}} = \operatorname{tr}\left(\boldsymbol{Q}^{(1)}[i_d] \cdots \boldsymbol{Q}^{(D)}[i_D]\right)$, where $\boldsymbol{Q}^{(d)}[i_d] \in \mathbb{R}^{R_d \times R_{d+1}}$ is the i_d-th lateral slice of the d-th core tensor.

Tensor Subchains The subchains of TR is defined as tensor contractions among a subsequence of core tensors. For example, the left subchain $\boldsymbol{\mathcal{Q}}^{<d} \in \mathbb{R}^{R_1 \times \prod_{j=1}^{d-1} I_j \times R_d}$, right subchain $\boldsymbol{\mathcal{Q}}^{>d} \in \mathbb{R}^{R_{d+1} \times \prod_{j=d+1}^{D} I_j \times R_1}$ are defined as

$$\boldsymbol{Q}^{<d}[\overline{i_1 \cdots i_{d-1}}] = \prod_{j=1}^{d-1} \boldsymbol{Q}^{(j)}[i_j], \quad \boldsymbol{Q}^{>d}[\overline{i_{d+1} \cdots i_D}] = \prod_{j=d+1}^{D} \boldsymbol{Q}^{(j)}[i_j].$$

Similarly, we can define $\boldsymbol{\mathcal{Q}}^{\neq d} \in \mathbb{R}^{R_{d+1}\times\prod_{j=1,j\neq d}^{D} I_j \times R_d}$ as

$$\boldsymbol{Q}^{\neq d}[\overline{i_{d+1}\cdots i_D i_1 \cdots i_{d-1}}] = \prod_{j=d+1}^{D} \boldsymbol{Q}^{(j)}[i_j] \prod_{j=1}^{d-1} \boldsymbol{Q}^{(j)}[i_j].$$

If $\boldsymbol{\mathcal{X}}$ admits the TR format (2), then

$$\boldsymbol{X}_{[d]} = \boldsymbol{Q}^{(d)}_{(2)} (\boldsymbol{Q}^{\neq d}_{[2]})^{\intercal}, \quad \forall d = 1, \ldots, D. \tag{3}$$

4. Bayesian Tensor Ring Latent Factor Model

In this section, we introduce the proposed Bayesian tensor ring latent factor (TRLF) model. By using the TR format, TRLF is suitable to model high dimensional data. Moreover, by using MGP prior, our model can obtain low-rank and sparse factors.

Model Formulation In stead of finding low-rank matrix factorization of the covariance matrix, i.e., Eq. (1), we generalize the latent factor model to higher-order data. Now suppose the observed data is an order-D tensor $\boldsymbol{\mathcal{Y}}^{(n)} \in \mathbb{R}^{P_1\times\cdots\times P_D}$. For N observations, we stack them into an order-$(D+1)$ tensor, denoted as $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{P_1\times\cdots\times P_D\times N}$. Then we extend Eq. (1) using the TR format, namely,

$$\boldsymbol{\mathcal{Y}} = \ll \boldsymbol{\mathcal{Q}}^{(1)}, \ldots, \boldsymbol{\mathcal{Q}}^{(D)}, \boldsymbol{\eta} \gg + \boldsymbol{\mathcal{E}}, \tag{4}$$

where $\boldsymbol{\mathcal{Q}}^{(d)} \in \mathbb{R}^{R_d\times P_d\times R_{d+1}}$ are loading core tensors, $\boldsymbol{\eta} \in \mathbb{R}^{R_D\times N\times R_1}$ is the latent factor and $\boldsymbol{\mathcal{E}}$ is the noise tensor with the same size of the data. Since the latent factors are matrices here, we assume that they follow the standard matrix Normal distribution, namely,

$$\boldsymbol{\eta}^{(n)} \sim \mathcal{MN}(\boldsymbol{0}, \boldsymbol{I}_{R_{D+1}}, \boldsymbol{I}_{R_1}), \quad \forall n = 1, \ldots, N. \tag{5}$$

Moreover, we suppose all the noises independently follow Gaussian distribution,

$$\boldsymbol{\mathcal{E}}^{(n)}_{p_1\cdots p_D} \sim \mathcal{N}(0, \tau^{-1}), \quad \forall n = 1, \ldots, N. \tag{6}$$

Note that instead of one single loading matrix $\boldsymbol{W}$ in Eq. (1), our model represents the loading matrices by D loading core tensors $\{\boldsymbol{\mathcal{Q}}^{(d)}\}_{d=1}^{D}$. Such construction has two main advantages. First, the core tensors have a much more compact form. If we vectorize the data $\boldsymbol{\mathcal{Y}}$, the size of the loading matrix grows exponentially with tensor order D. However, by directly tackle with the tensors, the parameter numbers grow linearly with D. Second, in our model, the core tensors are stacked in a deep and hierarchical manner, which can capture complex multi-linear relations.

Now we introduce more intuitions about our model and connections with model (1). Despite the enormous size of parameter, the naïve vectorization dismisses the mode-specific relations in the tensor data. To this end, we introduce the covariance tensor for higher-order data $\boldsymbol{\mathcal{Y}}^{(n)}$, i.e., $\boldsymbol{\mathcal{V}}_{p_1\cdots p_D p'_1\cdots p'_D} = \mathrm{var}(\boldsymbol{\mathcal{Y}}^{(n)}_{p_1\cdots p_D}, \boldsymbol{\mathcal{Y}}^{(n)}_{p'_1\cdots p'_D})$, where $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{P_1\cdots\times P_D\times P_1\cdots\times P_D}$ is an order-$(2D)$ tensor. By adopting model (4), we have

$$\boldsymbol{\mathcal{V}}_{p_1\cdots p_D p'_1\cdots p'_D} = \tau^{-1} + \mathrm{tr}\left(\boldsymbol{Q}^{(1)}[p_1]\cdots\boldsymbol{Q}^{(D)}[p_D]\cdot(\boldsymbol{Q}^{(D)}[p'_D])^{\intercal}\cdots(\boldsymbol{Q}^{(D)}[p'_D])^{\intercal}\right).$$

The low-rank covariance tensor follows a symmetric TR format and if we reshape $\boldsymbol{\mathcal{V}}$ to matrix form, this is a matrix-TR format, which is a much more expressive extension of the Kronecker structure (Oseledets, 2011). Indeed, given Eq. (3), (4), (5) and (6), by proper permutations and reshapings, the distribution of the observed data is

$$\mathrm{vec}(\boldsymbol{\mathcal{Y}}^{(n)}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{V}), \tag{7}$$

where $\boldsymbol{V} = \boldsymbol{Q}^{\leq D}_{[2]}(\boldsymbol{Q}^{\leq D}_{[2]})^\intercal + \mathrm{diag}(\tau^{-1})$. Note that we do not need to compute the above vector form distribution. All the computations can be conducted by the core tensors.

According to Eq. (4), (5) and (6), the conditional distribution of the data is

$$\log p(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\mathcal{Q}}, \boldsymbol{\eta}, \tau) = \frac{NP_1 \cdots P_D}{2} \log \tau - \frac{\tau}{2} \sum_{n=1}^{N} \sum_{p_1=1}^{P_1} \cdots \sum_{p_D=1}^{P_D} \left(\boldsymbol{\mathcal{Y}}^{(n)}_{p_1 \cdots p_D} - \hat{\boldsymbol{\mathcal{Y}}}^{(n)}_{p_1 \cdots p_D} \right)^2, \tag{8}$$

where $\hat{\boldsymbol{\mathcal{Y}}} = \ll \boldsymbol{\mathcal{Q}}^{(1)}, \ldots, \boldsymbol{\mathcal{Q}}^{(D)}, \boldsymbol{\eta} \gg$.

Prior Distributions To get sparse loading core tensors, we extend the Multiplicative Gamma Process (MGP) (Bhattacharya and Dunson, 2011) to multi-way scenario. For each elements of the core tensors, we assume

$$\boldsymbol{Q}^{(d)}_{jh}[i] \mid \phi^{(d)}_{jih}, u^{(d)}_j, u^{(d+1)}_h \sim \mathcal{N}\left(0, (\phi^{(d)}_{jih})^{-1} (u^{(d)}_j)^{-1} (u^{(d+1)}_h)^{-1}\right),$$

for $i = 1, \ldots, P_d$, $j = 1, \ldots, R_d$, $h = 1, \ldots, R_{d+1}$ and $d = 1, \ldots, D$, where $\{u^{(d)}\}_{d=1}^D$ are global shrinkage prior to induce sparse and low-rank estimators and $\{\phi^{(d)}\}_{d=1}^D$ are local shrinkage prior to prevent the model from over shrinkage. Then, we put the MGP on the global shrinkage parameters $\boldsymbol{u}$, namely,

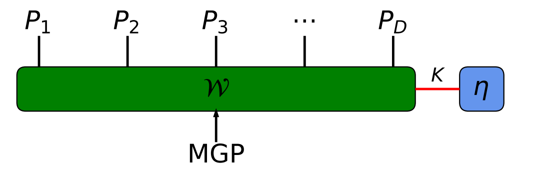
$$u^{(d)}_h = \prod_{l=1}^{h} \delta^{(d)}_l, \quad \delta^{(d)}_l \sim Ga(\alpha_\delta, 1),$$

where α_δ is set larger than 1 to encourage sparsity. Furthermore, the local shrinkage follows Gamma distribution $\phi^{(d)}_{jih} \sim Ga(\nu, \nu)$. Finally, we assume the noise precision follows Gamma distribution $\tau \sim Ga(\alpha_\tau, \beta_\tau)$. The graphical illustration is shown in Figure 1.

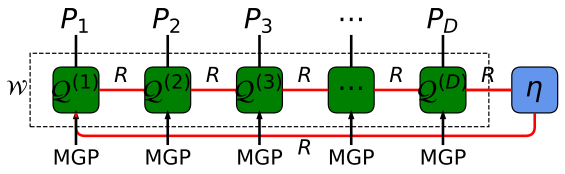
For simplicity, we denote all the parameters in our model as $\boldsymbol{\Theta} = \{\boldsymbol{\mathcal{Q}}, \boldsymbol{\eta}, \boldsymbol{\phi}, \boldsymbol{\delta}, \tau\}$. The joint distribution of our model is $\log p(\boldsymbol{\mathcal{Y}}, \boldsymbol{\Theta}) = \log p(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\Theta}) + \log p(\boldsymbol{\Theta})$, where $p(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\Theta})$ is defined in Eq. (8) and

$$\log p(\boldsymbol{\Theta}) = \log p(\boldsymbol{\mathcal{Q}} \mid \boldsymbol{\phi}, \boldsymbol{u}) + \log p(\boldsymbol{\phi}) + \log p(\boldsymbol{\delta}) + \log p(\tau) + \log p(\boldsymbol{\eta}).$$

Identifiability As most latent factor models, the proposed model is not identifiable. To be specific, if we apply some orthogonal transformations on the neighborhood core tensor, for instance, let $\tilde{\boldsymbol{Q}}^{(d)}[i] = \boldsymbol{Q}^{(d)}[i]\boldsymbol{P}^\intercal$ and $\tilde{\boldsymbol{Q}}^{(d+1)}[j] = \boldsymbol{P}\boldsymbol{Q}^{(d)}[j]$, where $\boldsymbol{P}^\intercal \boldsymbol{P} = \boldsymbol{I}$, we have $\tilde{\boldsymbol{Q}}^{(d)}[i] \cdot \tilde{\boldsymbol{Q}}^{(d+1)}[j] = \boldsymbol{Q}^{(d)}[i] \cdot \boldsymbol{Q}^{(d+1)}[j]$ and the covariance estimation does not change. This unidentifiability sometimes makes the posterior hard to be optimized. To this end, we adopt the parameter-expansion technique to optimize the transformation. This method is widely used in latent factor models and is shown to encourage sparse factors (Ročková and George, 2016; Zhao et al., 2016)



(a) InfLF(Bhattacharya and Dunson, 2011)



(b) TRLF

Figure 1: Graphical representation of matrix latent factor model and the proposed model. Each node represents a random variable and the legs are dimensions. Connected legs mean contractions over corresponding dimensions. MGP means the prior.

5. Algorithm

Since our model is fully conjugate, it is convenient to establish a Gibbs sampler to approximate the true posterior (See supplementary material). However, the Gibbs sampler usually converges slowly. Since we only concern about the point estimate of the model parameters for most of times, we establish an efficient EM algorithm to learn the MAP estimate. Moreover, we adopt the Parameter-eXpansion (PX) technique (Liu et al., 1998) to improve the EM algorithm. In this section, we firstly introduce the ordinary EM algorithm. Then we adjust it to the PX-EM by adopting a double-rotation step.

For our model, we treat $\boldsymbol{\eta}$ as latent variables and optimize them in the E-step. Other parameters are optimized in the M-step. In this section, we denote subscripts $\mathbf{p} = [p_1, \ldots, p_D]$ and $\mathbf{p}_{-d} = [p_1, \ldots, p_{d-1}, p_{d+1}, \ldots, p_D]$ for simplicity. We also adopted the little endian convention as $\overline{\mathbf{p}}$ and $\overline{\mathbf{p}_{-d}}$. Moreover, we denote the expectation w.r.t $\boldsymbol{\eta}$ as $\langle\cdot\rangle$.

E-step In the E-step we compute the expectation of the latent variables. For our model, we have

$$\langle \tilde{\boldsymbol{\eta}}^{(n)} \rangle = \boldsymbol{\Lambda}_{\boldsymbol{\eta}}^{(n)} \left(\sum_{\mathbf{p}} \tau \boldsymbol{\mathcal{Y}}_{\mathbf{p}}^{(n)} \mathrm{vec}(\boldsymbol{Q}^{\leq D, \intercal}[\overline{\mathbf{p}}]) \right)$$

$$\langle \tilde{\boldsymbol{\eta}}^{(n)} \cdot (\tilde{\boldsymbol{\eta}}^{(n)})^{\intercal} \rangle = \langle \tilde{\boldsymbol{\eta}}^{(n)} \rangle \cdot \langle \tilde{\boldsymbol{\eta}}^{(n)} \rangle^{\intercal} + \boldsymbol{\Lambda}_{\boldsymbol{\eta}}^{(n)},$$

where $\tilde{\boldsymbol{\eta}}^{(n)} = \mathrm{vec}(\boldsymbol{\eta}^{(n)})$ and

$$\left(\boldsymbol{\Lambda}_{\boldsymbol{\eta}}^{(n)}\right)^{-1} = \tau \sum_{\mathbf{p}} \mathrm{vec}((\boldsymbol{Q}^{\neq D, \intercal}[\overline{\mathbf{p}}])\mathrm{vec}(\boldsymbol{Q}^{\neq D, \intercal}[\overline{\mathbf{p}}])^{\intercal} + \boldsymbol{I}_K,$$

where $\boldsymbol{Q}^{\neq D, \intercal}[\overline{\mathbf{p}}]$ is the transpose of the subchain defined in Section 3.2.

M-step In the M-step, we maximize the expectation of the log-likelihood function. To make the notations consistent, we denote $\boldsymbol{\mathcal{Q}}^{(D+1)} = \boldsymbol{\eta}$ and $P_{D+1} = N$ in this section. To update the core tensors, for $d = 1, \ldots, D$, we have

$$\tilde{\boldsymbol{q}}^{(d)}[p_d] = \boldsymbol{\Lambda}_{\boldsymbol{Q}^{(d)}}^{(p_d)} \left(\sum_{n=1}^{N} \sum_{\mathbf{p}_{-d}} \tau \boldsymbol{\mathcal{Y}}_{\mathbf{p}}^{(n)} \langle \mathrm{vec}(\boldsymbol{Q}^{\neq d, \intercal}[\overline{\mathbf{p}_{-d}}]) \rangle \right),$$

where $\tilde{\boldsymbol{q}}^{(d)}[p_d] = \mathrm{vec}(\boldsymbol{Q}^{(d)}[p_d])$ and

$$\left(\boldsymbol{\Lambda}_{\boldsymbol{Q}^{(d)}}^{(p_d)}\right)^{-1} = \sum_{\mathbf{p}_{-d}} \tau \langle \mathrm{vec}(\boldsymbol{Q}^{\neq d,\intercal}[\overline{\mathbf{p}_{-d}}]) \mathrm{vec}(\boldsymbol{Q}^{\neq d,\intercal}[\overline{\mathbf{p}_{-d}}])^\intercal \rangle \\ + \mathrm{diag}(\mathrm{vec}(\boldsymbol{\phi}_{:p_d:}^{(d)})) * (\mathrm{diag}(\boldsymbol{u}^{(d+1)}) \otimes \mathrm{diag}(\boldsymbol{u}^{(d)})),$$

where $\mathrm{diag}(\cdot)$ is diagonal matrix and "$*$" denotes the Hadamard product. To update the local shrinkage parameter $\boldsymbol{\phi}$, for $d = 1, \ldots, D$, we have $\phi_{jih}^{(d)} = a_\phi / b_\phi$, where

$$a_\phi = 1/2 + \nu, \quad b_\phi = \nu + \langle (\boldsymbol{Q}_{jk}^{(d)}[i])^2 \rangle u_j^{(d)} u_h^{(d+1)}.$$

The update rule for the global shrinkage is $\delta_h^{(d)} = a_\delta / b_\delta$. For $d = 2, \ldots, D-1$, we have

$$a_\delta = \alpha_\delta + \frac{(P_d R_{d+1} + P_{d-1} R_d)(R_d - h + 1)}{2},$$

$$b_\delta = 1 + \frac{1}{2} \sum_{l=h}^{R_d} \left[\sum_{i=1}^{P_d} \boldsymbol{u}^{(d+1)} \left\langle \boldsymbol{Q}_{l\cdot}^{(d)}[i] * \boldsymbol{Q}_{l\cdot}^{(d)}[i] \right\rangle + \sum_{i=1}^{P_{d-1}} \boldsymbol{u}^{(d-1)} \left\langle \boldsymbol{Q}_{\cdot l}^{(d-1)}[i] * \boldsymbol{Q}_{\cdot l}^{(d-1)}[i] \right\rangle \right] u_{l,-h}^{(d)},$$

where $u_{l,-h}^{(d)} = \prod_{j=1, j\neq h}^{l} \delta_j^{(d)}$. For $d = 1$, the update rule is similar, except that the term involves $u^{(d-1)}$ is dropped. For $d = D$, we should drop the term with $u^{(d+1)}$. See supplementary material for details. Finally, we update the noise precision by $\tau = a_\tau / b_\tau$, where

$$a_\tau = N/2 + \alpha_\tau, \quad b_\tau = \beta_\tau + \left\langle \sum_{i=1}^{N} \left\| \boldsymbol{\mathcal{Y}}^{(i)} - \hat{\boldsymbol{\mathcal{Y}}}^{(i)} \right\|_F^2 \right\rangle.$$

PX-EM Since the latent factor model is not identifiable, some rotations on the latent factors may greatly benefit the learning process. In traditional latent factor models, the rotation step is designed for vectors. However, in our model, the latent factors are matrices. Hence, we develop a double-rotation step, which can be split into the left rotation and the right rotation. To begin with, we rewrite model (4) as

$$\boldsymbol{\mathcal{Y}} = \ll \tilde{\boldsymbol{\mathcal{Q}}}^{(1)}, \ldots, \tilde{\boldsymbol{\mathcal{Q}}}^{(D)}, \boldsymbol{\eta} \gg + \boldsymbol{\mathcal{E}}, \quad \boldsymbol{\eta}^{(n)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{A}, \boldsymbol{B}),$$

where $\forall i = 1, \ldots, P_1$ and $\forall j = 1, \ldots, P_D$,

$$\tilde{\boldsymbol{Q}}^{(1)}[i] = \boldsymbol{R}_1^{-1} \boldsymbol{Q}^{(1)}[i], \quad \tilde{\boldsymbol{Q}}^{(D)}[j] = \boldsymbol{Q}^{(D)}[j] \boldsymbol{R}_2^{-1}.$$

By setting $\boldsymbol{R}_2$ as the lower triangular part of the Cholesky decomposition of $\boldsymbol{B}$ and $\boldsymbol{R}_1$ as the upper triangular part of the Cholesky decomposition of $\boldsymbol{A}$, it is equivalent to model (4). In the PX-EM algorithm, we add MGP prior on the loading tensor $\tilde{\boldsymbol{\mathcal{Q}}}^{(1)}$ and $\tilde{\boldsymbol{\mathcal{Q}}}^{(D)}$, instead of $\boldsymbol{\mathcal{Q}}^{(1)}$ and $\boldsymbol{\mathcal{Q}}^{(D)}$. After each iteration of the EM algorithm, an additional double-rotation step is added, by $\boldsymbol{Q}^{(1)}[i] = \boldsymbol{R}_1 \tilde{\boldsymbol{Q}}^{(1)}[i]$ and $\boldsymbol{Q}^{(D)}[i] = \tilde{\boldsymbol{Q}}^{(D)}[i] \boldsymbol{R}_2$. The rotation matrices can be computed by maximizing the log-likelihood function (Ročková and George, 2016), as follows,

$$\boldsymbol{A} = \left\langle \sum_{n=1}^{N} \boldsymbol{\eta}^{(n),\intercal} \boldsymbol{B}^{-1} \boldsymbol{\eta}^{(n)} \right\rangle / (N R_1), \quad \boldsymbol{B} = \left\langle \sum_{n=1}^{N} \boldsymbol{\eta}^{(n)} \boldsymbol{A}^{-1} \boldsymbol{\eta}^{(n),\intercal}) \right\rangle / (N R_D).$$

Complexity Analysis The storage complexity of our model is $\mathcal{O}(\sum_{d=1}^{D} P_d R^2 + KN)$, where P_d is the dimension of each order, K is the factor number, N is the sample size and we suppose the TR-ranks equal to R. The storage complexity grows linearly with the data order D and the sample size N.

The computational complexity is $\mathcal{O}(\sum_{d=1}^{D} P_d R^6 + ND\sum_{d=1}^{D} P_d R^4 + NPD^2R^3)$ per iteration, where $P = P_1 \cdots P_D$ is the feature length. If $\boldsymbol{\mathcal{Y}}$ does not have any special structure, this complexity inevitably grows linearly with the feature length P, which can be extremely large as order D grows. However, if the data $\boldsymbol{\mathcal{Y}}$ is given by TR format of rank r, the computational complexity can be reduced to $\mathcal{O}(\sum_{d=1}^{D} P_d R^6 + ND\sum_{d=1}^{D} P_d R^4 + ND\sum_{d=1}^{D} P_d R^2 r^2)$, which grows quadratically with the data order D and can be generalized to higher-order scenarios. Even when the data is given by a matrix form, we can tensorize it and use the TR-SVD (Zhao et al., 2016) or TRA (Wang et al., 2017) algorithm to get an initial representation, which is reasonable in practice since we have assumed that $\boldsymbol{\mathcal{Y}}$ is low rank.

6. Experiments

To test our model, we conduct experiments on covariance estimation, latent factor learning and data imputation. We initialize the proposed model with both TR and TT format, denoted as TRLF and TTLF, respectively. All of our experiments are performed on a GNU/Linux workstation with Intel Xeon E5-2690 3.50GHz CPU and 64GB memory.

6.1. Synthetic Data Analysis

6.1.1. Scalability with Tensor Orders

Firstly, we show that our model scales linearly with the tensor orders, if the observed tensor admits low rank structures. Specifically, we suppose the observed tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_D \times N}$ is given in TR format of rank 10. We set $I_1 = \cdots = I_D = 4$, $N = 100$ and $D = 2, 3, \ldots, 10$. Then we generate TR cores of $\boldsymbol{\mathcal{Y}}$ from standard Gaussian distribution. For our model, we directly use the TR representation of $\boldsymbol{\mathcal{Y}}$. As a comparison, we apply the InfLF (Bhattacharya and Dunson, 2011) on vectorized $\boldsymbol{\mathcal{Y}}$. InfLF can be regarded as a vector form of our model. For our model, we set the TR-ranks as 20 and for InfLF, we set the factor number as 20. We report the time costs for 100 iterations and number of parameters in Figure 2. It shows that both the time cost and parameters of our model grows polynomially with the tensor order, which reveals the potential of our model in high-order applications.

6.1.2. Covariance Estimation

Covariance estimation is an important and difficult problem in high dimensional statistics. However, if the target covariance matrix is highly structured, e.g., low-rank and sparse, we show that our model will greatly boost the performance. In the synthetic data analysis, we artificially design some covariance matrices and test the performance of our model.

Data generation We consider data of feature length 1000 and different sample sizes. We pick 4 kind of loading matrices, 1). EXP, which is generated using Exponential functions, e.g., $\boldsymbol{W}_{exp}(i,j) = a\exp(-(i-j)^2/b)$, and then the covariance matrix is computed by $\boldsymbol{V}_{exp} = \boldsymbol{W}_{exp}\boldsymbol{W}'_{exp} + \tau\boldsymbol{I}$. 2). PED, which is a Periodic function $\boldsymbol{W}_{ped}(i,j) = a\exp(-\sin^2(\pi|i -$

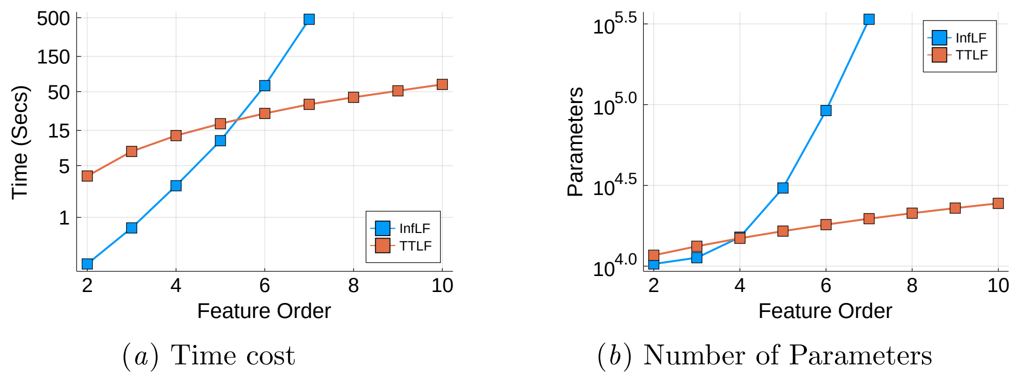


Figure 2: Figure (a) shows the times costs and figure (b) shows the parameters numbers. The x-axes are feature orders. The y-axes are time costs in seconds and parameter numbers, respectively. Both y-axes are in log scale.

$j|)/b)$ and $\boldsymbol{V}_{ped} = \boldsymbol{W}_{ped}\boldsymbol{W}'_{ped} + \tau\boldsymbol{I}$. 3). LIN $\otimes$ EXP, which is computed by $\boldsymbol{V}_{lin\otimes ped} = \boldsymbol{W}_{lin} \otimes \boldsymbol{W}_{ped}\boldsymbol{W}'_{ped} + \tau\boldsymbol{I}$, where $\boldsymbol{W}_{lin}(i,j) = ai \cdot j$. 4). LIN $\otimes$ EXP, which is computed by $\boldsymbol{V}_{lin\otimes ped} = \boldsymbol{W}_{lin} \otimes \boldsymbol{W}_{ped}\boldsymbol{W}'_{ped} + \tau\boldsymbol{I}$. See Figure 3 (a) for illustrations. From the top row to the bottom row are EXP, PED, LIN $\otimes$ EXP and LIN $\otimes$ PED, respectively. We set $\tau = 1\text{e}{-3}$ for all the cases. Finally we sample the data from Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{V})$, where $\boldsymbol{V}$ is the covariance matrices generated above. We choose sample size of $N = \{100, 500, 1000, 1500\}$ and repeat every experiment for 50 times.

Baseline models We compare our model with the following baselines: 1). **LW**, a James-Stein type model which shrinks the sample covariance matrix to the Ledoit-Wolf target matrix (Ledoit and Wolf, 2004). 2). **POET** (Fan et al., 2013), the Principal Orthogonal Complement Thresholding, which is a low rank matrix model. 3). **InfLF** (Bhattacharya and Dunson, 2011), the Infinite Latent Factor model, which is a Bayesian latent factor model using the MGP prior. Also, it can be regarded as a vector form of our model. 4). **HOLQ** (Gerard and Hoff, 2016), the Higher-Oder LQ decomposition, which is a generalization of Tucker decomposition. It assumes that the covariance matrix follows the Kronecker structure,

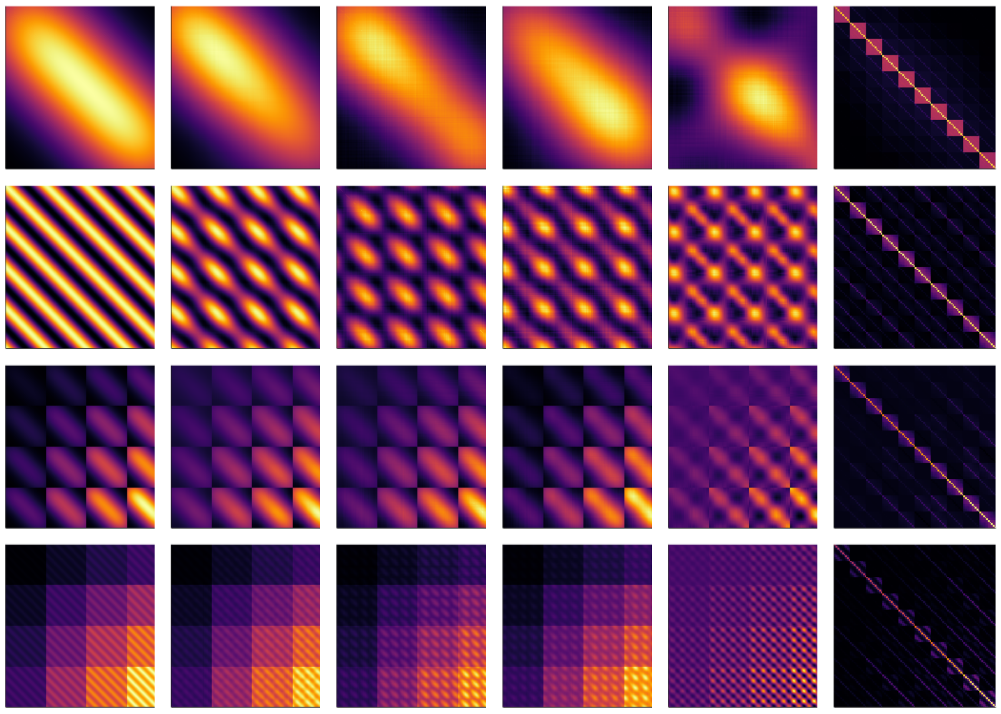
$$\boldsymbol{V} = \boldsymbol{V}^{(\boldsymbol{D})} \otimes \cdots \otimes \boldsymbol{V}^{(\boldsymbol{1})}, \tag{9}$$

where $\boldsymbol{V}^d, \forall d = 1, \ldots, D$ are small covariance matrices of each modes of the tensor data.

For LW, POET and InfLF, we directly use the data of shape $1000 \times N$. For HOLQ and our model, we reshape the data to shape $10 \times 10 \times 10 \times N$. For POET, we use factor number 50, because it is not sensitive to large factor numbers. For InfLF and our model, we set the factor number as 30 and adaptive tune it during training. Figure 3 shows the visualization results for data size $N = 100$. It results that our model achieves the best recovery performance. Also, we found that the HOLQ does not perform well on these data. The reason may be that our synthetic data does not match its assumption in Eq. (9). In specific, the HOLQ requires that the true covariance matrix can be decomposed into Kronecker product of small covariance matrices, which are positive semi-definite (PSD).

Although our data also has the Kronecker structure, the loading matrices, e.g., $\boldsymbol{W}_{exp}\boldsymbol{W}'_{exp}$, may be low-rank and hence are not PSD.

To evaluate the performance, we use the Log-Euclidean Distance (LED) (Vemulapalli and Jacobs, 2015), which is a measurement for symmetric positive definite matrices. The results are shown in Figure 4. We plot the median value, 1/4 and 3/4 quantiles of all the experiments. For the simulation data, our model outperforms the baseline models, especially when the sample size is small.



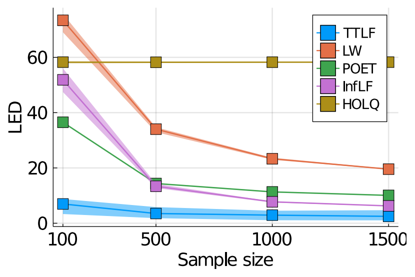
(a) Truth (b) TTLF (c) LW (d) POET (e) InfLF (f) HOLQ

Figure 3: Covariance estimation for synthetic data of sample size 100. For the top row to the bottom row is shape EXP, PED, LIN $\otimes$ EXP and LIN $\otimes$ PED.

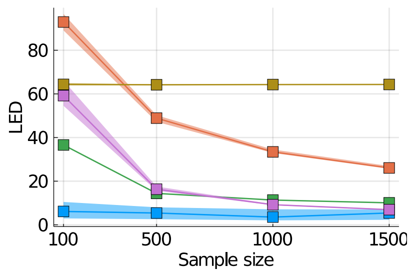
6.2. Real Data Covariance Estimation

In this subsection, we test our model on three real data sets, i.e., Arcene dataset[2], Madelon dataset[3] and EEG dataset[4]. The Arcene has feature length 10000 and consists of 100 training samples and 100 testing sample, where the training/testing set has 44 positive samples and 56 negative samples. The Madelon has feature of length 500 and two classes. Each class has 1000 samples for training and 300 samples for testing. The EEG has 5 classes, where each class has 100 samples of signal length 4096. For the EEG data, which

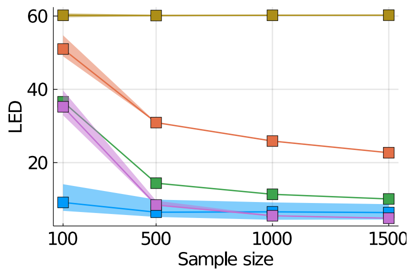
2. https://archive.ics.uci.edu/ml/datasets/Arcene
3. https://archive.ics.uci.edu/ml/datasets/Madelon
4. http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3



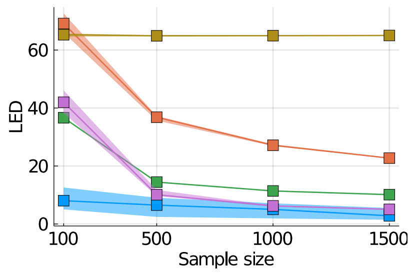
(a) EXP



(b) PED



(c) LIN $\otimes$ EXP



(d) LIN $\otimes$ PED

Figure 4: Results of covariance estimation in synthetic data analysis. Each subfigure shows results of different shapes. The x-axis is sample sizes and the y-axis is the LED.

is a time series, we use the first 50 samples as training and the last 50 samples as testing. Since it is hard to scale the LW and POET algorithm to the high dimensional cases, we compare our model with InfLF and HOLQ in this subsection. For preprocessing, we simply centralize the data, so that the mean of each feature is zero. Moreover, for HOLQ and our model, we reshape the Arcene to shape $10 \times 10 \times 10 \times 10 \times N$, the Madelon to shape $5 \times 10 \times 10 \times N$ and the EEG to shape $16 \times 16 \times 16 \times N$. We initialize the factor number as 40 and adaptively tune it during training.

Since the true covariance is unknown, we use the classification accuracy to evaluate the performance. In particular, we firstly estimate the covariance matrices of each class using the training data, e.g., $\hat{\boldsymbol{V}}_l$, where l is the label. Then for the test data x, the prediction of the label is $\hat{l} = \arg\max_l \text{LL}(x, \hat{\boldsymbol{V}}_l)$, where $\text{LL}(\cdot,\cdot)$ is the log-likelihood function. We report the accuracy (ACC) and AUC value in Table 1. The InfLF algorithm fails to converge for the Arcene data. This may reveal the advantage of tensor-based model in high-dimensional cases. Our model outperforms both InfLF and HOLQ.

Table 1: Results for the classification using the estimated covariance.

Data	Arcene		Madelon		EEG	
Metric	ACC	AUC	ACC	AUC	ACC	AUC
TTLF	**0.730**	**0.797**	**0.652**	**0.687**	**0.640**	**0.913**
InfLF	NA	NA	0.571	0.655	0.228	0.758
HOLQ	0.640	0.700	0.592	0.617	0.532	0.840

6.3. Supervised Learning with Extracted Factors

In this subsection, we show that the TR/TTLF model actually learn some meaningful latent factors of the original data. We use the U.S. Postal Service (USPS) data to illustrate the experiments[5]. This dataset totally consists of 9298 grayscale images of the handwritten digits from 0 to 9. For each of the images, the shape is 16×16 and the value ranges from -1.0 to 1.0. The whole dataset is split into a training set of size 7291 and a test set of size 2007. In this experiment, we compare our model with the InfLF. For our model, we tensorize each image to $4 \times 4 \times 4 \times 4$.

We stack the train and test set together and use TR/TTLF model to extract the latent factors without using the information about the labels. Then we feed supervised learning algorithms with the learned latent factors, e.g., support vector machine (SVM). For simplicity, we use the same parameter for the SVM in all the experiments without any tuning. The results are shown in Figure 5. We can see that the classification accuracy increases as the factor number growing. All the latent factor models improve the performance of the vanilla SVM and the TTLF has the best results. The classification experiment reveals that our model is potential as an unsupervised data preprocessing method. It reduces the feature dimension significantly while increasing the classification accuracy.

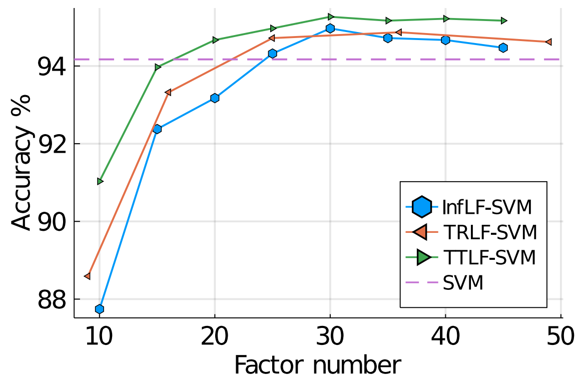


Figure 5: Classification using the latent factors.

6.4. Image Inpainting

Finally, the proposed model can be easily extended to tackle with missing data. When the data is partially observed, the only difference is the conditional distribution in Eq. (8). Supposing the set of observed index is Ω, the conditional distribution Eq. (8) becomes

$$\log p(\boldsymbol{\mathcal{Y}} \mid \boldsymbol{\mathcal{Q}}, \boldsymbol{\eta}, \tau) = \frac{M}{2} \log \tau - \frac{\tau}{2} \sum_{i=1}^{N} \sum_{p_1 \cdots p_D \in \Omega} \left(\boldsymbol{\mathcal{Y}}^{(i)}_{p_1 \cdots p_D} - \hat{\boldsymbol{\mathcal{Y}}}^{(i)}_{p_1 \cdots p_D} \right)^2 ,$$

where M is the number of observed entries and the prior distribution remains the same. Then we can use the same algorithm with fully observed case to maximize the posterior. We test the performance on the image inpainting problem. For one single image, we split it into patches and then stack them together. For example, for an image of shape $256 \times 256 \times 3$, we can split it into 256 patches of shape $16 \times 16 \times 3$.

We compare our model with several low-rank tensor completion algorithms, including Bayesian CP Factorization (BCPF) (Zhao et al., 2015), Tensor Ring Alternating Least Square (TRALS) (Wang et al., 2017), Tensor Train Weighted OPTimization (TTWOPT)

5. https://web.stanford.edu/~hastie/StatLearnSparsity_files/DATA/zipcode.html

(Yuan et al., 2019b), Tensor Ring Low-Rank Factorization (TRLRF) (Yuan et al., 2019a). The BCPF model also uses sparse Bayesian priors to automatically choose latent factors and requires no hyperparameters. However, for the rest of models, we have to carefully tune the hyperparameters, e.g., the TT/TR-ranks. Here we compute those models under several TT/TR-rank settings and select the best performance. However, it should be noted that this is not realistic in many applications where the true signals are unknown. For our model, we simply initialize with TR format of rank 15, and we truncate the redundant factors while training, in order to reduce computational cost.

We choose 8 standard pictures from USC- SIPI[6] database as benchmark and randomly generate masks of missing rate 0.5, 0.7 and 0.9. For numerical metrics, we use the Relative Standard Error (RSE) and the Peak Signal-to-Noise Ration (PSNR), which are two of the mostly used criteria to evaluate visual data. Smaller RSE and higher PSNR indicate better performance. We average the numerical results for the 8 pictures, as shown in Table 2. When the missing rate is 0.5, the TRLRF is slightly better than the proposed TRLF model, but the results are comparable. However, as the missing rate grows, our model outperforms others significantly.

Table 2: Results for image inpainting.

Missing	Metric	TRLF	TRALS	TRLRF	TTWOPT	BCPF
50%	RSE↓	0.0580	0.0738	**0.0555**	0.0783	0.0833
	PSNR↑	30.16	28.41	**31.03**	27.70	27.18
70%	RSE↓	**0.0759**	0.1133	0.1000	0.1272	0.1113
	PSNR↑	**27.63**	24.46	25.81	23.31	24.58
90%	RSE↓	**0.1268**	0.5340	0.2286	0.2321	0.1813
	PSNR↑	**23.01**	11.17	18.33	17.98	20.24

7. Conclusion

Latent factor modeling is a classical problem in statistical learning. This concept includes a wide range of algorithms for dimension reduction and sparse learning. However, traditional latent factor models are not designed for higher-order data. To address this issue, we try to combine the Bayesian latent factor model with TNs. By assuming the covariances are highly structured and can be approximated by TR format, we design the TRLF model to extract latent factors of the original data. We adopt the MGP prior to impose low-rank and sparse latent factors simultaneously and designed efficient PX-EM algorithm to find the MAP estimate of model parameters. Results show that our model outperforms in several high-dimensional modeling problems. For future research, we are interested in several directions: 1) Non-linear extensions; 2) Scalable inference algorithms for large datasets, such as amortized inference; 3) Exploring the TN formats of covariance matrices in other field.

6. http://sipi.usc.edu/database/

Acknowledgements

Zerui Tao was supported by the RIKEN Junior Research Associate (JRA) Program. Qibin Zhao was supported by JSPS KAKENHI (Grant No. 20H04249, 20H04208).

References

A. Bhattacharya and D. B. Dunson. Sparse Bayesian infinite factor models. *Biometrika*, 98 (2):291–306, 2011.

Carlos Marinho Carvalho, Jeffrey Chang, Joseph E. Lucas, Joseph R. Nevins, Quanli Wang, and Mike West. High-dimensional sparse factor modeling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008.

Rong Chen, Dan Yang, and Cun hui Zhang. Factor models for high-dimensional tensor time series. *arXiv preprint arXiv:1905.07530*, 2019.

Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9 (4-5):249–429, 2016.

Jianqing Fan, Yuan Liao, and Martina Mincheva. Large covariance estimation by thresholding principal orthogonal complements. *Journal of The Royal Statistical Society Series B-statistical Methodology*, 75(4):603–680, 2013.

Emily B Fox and David B Dunson. Bayesian nonparametric covariance regression. *The Journal of Machine Learning Research*, 16(1):2501–2542, 2015.

David Gerard and Peter Hoff. A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505:57–84, 2016.

Yuefeng Han, Rong Chen, Dan Yang, and Cun-Hui Zhang. Tensor factor model estimation by iterative projection. *arXiv preprint arXiv:2006.02611*, 2020.

Cole Hawkins and Zheng Zhang. Bayesian tensorized neural networks with automatic rank selection. *Neurocomputing*, 453:172–180, 2021.

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

Olivier Ledoit and Michael Wolf. Honey, I shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004.

Chuanhai Liu, Donald B Rubin, and Ying Nian Wu. Parameter expansion to accelerate em: the px-em algorithm. *Biometrika*, 85(4):755–770, 1998.

Zhen Long, Ce Zhu, Jiani Liu, and Yipeng Liu. Bayesian low rank tensor ring model for image completion. *arXiv preprint arXiv:2007.01055*, 2020.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33 (5):2295–2317, 2011.

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

Veronika Ročková and Edward I George. Fast Bayesian factor analysis via automatic rotations to sparsity. *Journal of the American Statistical Association*, 111(516):1608–1622, 2016.

Andrew Stevens, Yunchen Pu, Yannan Sun, Gregory Spell, and Lawrence Carin. Tensor-dictionary learning with deep kruskal-factor analysis. In *Artificial Intelligence and Statistics*, pages 121–129. PMLR, 2017.

Raviteja Vemulapalli and David W. Jacobs. Riemannian metric learning for symmetric positive definite matrices. *arXiv preprint arXiv:1501.02393*, 2015.

Dong Wang, Xialu Liu, and Rong Chen. Factor models for matrix-valued high-dimensional time series. *Journal of Econometrics*, 208(1):231–248, 2019.

Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Efficient low rank tensor ring completion. In *IEEE International Conference on Computer Vision*, pages 5697–5705, 2017.

Zenglin Xu, Feng Yan, and Yuan Qi. Infinite tucker decomposition: nonparametric Bayesian models for multiway data analysis. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1675–1682, 2012.

Longhao Yuan, Chao Li, Danilo Mandic, Jianting Cao, and Qibin Zhao. Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion. In *Proceedings of the AAAI*, volume 33, pages 9151–9158, 2019a.

Longhao Yuan, Qibin Zhao, Lihua Gui, and Jianting Cao. High-order tensor completion via gradient-based optimization under tensor train format. *Signal Processing: Image Communication*, 73:53–61, 2019b.

Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1751–1763, 2015.

Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

Shiwen Zhao, Chuan Gao, Sayan Mukherjee, and Barbara E. Engelhardt. Bayesian group factor analysis with structured sparsity. *Journal of Machine Learning Research*, 17(1): 6868–6914, 2016.

Shandian Zhe, Zenglin Xu, Xinqi Chu, Yuan Qi, and Youngja Park. Scalable nonparametric multiway data analysis. In *Artificial Intelligence and Statistics*, pages 1125–1134. PMLR, 2015.