Bayesian Continuous-Time Tucker Decomposition

Shikai Fang [1 2] Akil Narayan [2 3] Robert M. Kirby [1 2] Shandian Zhe [1]

Abstract

Tensor decomposition is a dominant framework for multiway data analysis and prediction. Although practical data often contains timestamps for the observed entries, existing tensor decomposition approaches overlook or under-use this valuable temporal information. They either drop the timestamps or bin them into crude steps and hence ignore the temporal dynamics within each step or use simple parametric time coefficients. To overcome these limitations, we propose Bayesian Continuous-Time Tucker Decomposition (BCTT). We model the tensor-core of the classical Tucker decomposition as a time-varying function, and place a Gaussian process prior to flexibly estimate all kinds of temporal dynamics. In this way, our model maintains the interpretability while is flexible enough to capture various complex temporal relationships between the tensor nodes. For efficient and high-quality posterior inference, we use the stochastic differential equation (SDE) representation of temporal GPs to build an equivalent state-space prior, which avoids huge kernel matrix computation and sparse/low-rank approximations. We then use Kalman filtering, RTS smoothing, and conditional moment matching to develop a scalable message-passing inference algorithm. We show the advantage of our method in simulation and several real-world applications.

1. Introduction

Multiway interaction data is omnipresent in real-world applications, such as in online advertising, e-commerce and social networking. A popular and powerful framework for multiway interaction analysis and prediction is tensor decomposition, which aims to estimate a set of latent factors to represent the interaction nodes, and use the factors to reconstruct the observed tensor elements. The factors can reflect unknown patterns in data, such as communities across the nodes, and provide effective features to build downstream predictive tools, such as product rating for recommendation and clicks for advertisement display.

While many successful tensor decomposition methods have been developed (Tucker, 1966; Harshman, 1970; Chu and Ghahramani, 2009; Kang et al., 2012; Choi and Vishwanathan, 2014), these methods ignore or under-exploit the valuable time information, which often comes along with the tensor data, *e.g.,* at which time point a *user* purchased an *item* at a specific *Amazon store*. Current methods often throw out the timestamps or bin the timestamps into crude steps, *e.g.,* weeks or months, and augment the tensor with a time step mode (Xiong et al., 2010; Xu et al., 2012; Rogers et al., 2013; Zhe et al., 2015; 2016a; Du et al., 2018). While between the steps we can use conditional priors and/or nonlinear dynamics to model their transition, the temporal dependencies within each step are overlooked. The most recent work (Zhang et al., 2021) although introduces continuous-time coefficients into the CANDECOMP/PARAFAC (CP) decomposition (Harshman, 1970), its parametric modeling of the coefficients, *i.e.,* polynomial splines, might not be flexible enough to capture a variety of different temporal dynamics in data (*e.g.,* from simple linear to highly nonlinear).

To overcome these limitations, we propose BCTT, a novel continuous-time Bayesian dynamic decomposition model. We extend the classical Tucker decomposition, which accounts for every multiplicative interaction between the factors across different tensor modes and is highly interpretable and quite expressive. We model the tensor-core — weights of the factor interactions — as a time-varying function. We place a Gaussian process (GP) prior, a nonparametric function prior that can flexibly estimate all kinds of functions, not restricted to any specific parametric form. In this way, our model not only maintains the interpretability, but also can automatically capture different, complex temporal dynamics from data. For efficient and high-quality posterior inference, we construct a linear time-invariant (LTI) stochastic differential equation (SDE) (Hartikainen and Särkkä, 2010) as an equivalent representation of the temporal GP. Based on the LTI-SDE, we build a state-space prior, which

[1]School of Computing, University of Utah [2] Scientific Computing and Imaging (SCI) Institute, University of Utah [3]Department of Mathematics, University of Utah. Correspondence to: Shandian Zhe <zhe@cs.utah.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

is essentially a Gaussian Markov chain but is equivalent to the GP prior. In this way, we circumvent the expensive kernel matrix computation in the original GP, and do not need any low-rank or sparse approximations. Next, we develop a message-passing posterior inference algorithm in the expectation propagation framework. We use Kalman filtering and Rauch–Tung–Striebel (RTS) smoothing (Särkkä, 2013) to efficiently compute the posterior of the SDE states, and use conditional moment matching (Wang and Zhe, 2019) and multi-variate delta method (Bickel and Doksum, 2015) to overcome the intractability in moment matching. Both the time and space complexity of our inference algorithm is linear in the number of observed data points.

For evaluation, we examined our approach in both ablation study and real-world applications. On synthetic datasets, BCTT successful learned different temporal dynamics and recovered the clustering structures of the tensor nodes from their factor estimation. On three real-world temporal tensor datasets, BCTT significantly outperforms the competing dynamic decomposition methods, including discrete time factors and continuous time coefficients, often by a large margin. The structure of the learned tensor-core also shows interesting temporal evolution.

2. Background

Tensor Decomposition. Consider a K-mode tensor $\mathcal{Y} \in \mathbb{R}^{d_1 \times \cdots \times d_K}$, where d_k is the number of nodes in mode k. We use a K-elements tuple $\mathbf{i} = (i_1, \ldots, i_K)$ to index each entry of the tensor, and denote the entry value by $y_{\mathbf{i}}$. To factorize $\mathcal{Y}$ into a concise structure, we introduce a set of latent factors for the tensor nodes, $\mathcal{U} = \{\mathbf{U}^1, \ldots, \mathbf{U}^K\}$, where each $\mathbf{U}^k = [\mathbf{u}_1^k, \ldots, \mathbf{u}_{d_k}^k]^\top$ is a factor matrix, in which each row consists of the factors for a node j in mode k, namely $\mathbf{u}_j^k$ $(1 \le j \le d_k)$. Given the factorization form, we estimate the optimal factors $\mathcal{U}$ to reconstruct the tensor $\mathcal{Y}$, by minimizing a loss on the observed entries. The (arguably) most popular tensor factorization model is CANDECOMP/PARAFAC (CP) (Harshman, 1970), whose entry-wise form is given by

$$y_{\mathrm{i}} \approx \boldsymbol{\lambda}^\top(\mathbf{u}_{i_1}^1 \circ \ldots \circ \mathbf{u}_{i_K}^K) = \sum\nolimits_{r=1}^R \lambda_r \prod\nolimits_{k=1}^K u_{i_k,r}^k, \quad (1)$$

where $\circ$ is the Hadamard (element-wise) product, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_R)^\top$ and each $\mathbf{u}_{i_k}^k = (u_{i_k,1}^k, \ldots, u_{i_k,R}^k)^\top$. While simple and convenient, CP only accounts for the interaction between every r-th factor in different modes, *i.e.,* $\prod_{k=1}^K u_{i_k,r}^k$ (weighted by λ_r accordingly), and overlook all the other possible interactions.

Tucker decomposition (Tucker, 1966) is more interpretable and expressive than CP in that it considers all the possible interactions between the factors across the tensor modes. Specifically, Tucker decomposition assumes $\mathcal{Y} \approx \mathcal{W} \times_1 \mathbf{U}^1 \times_2 \ldots \times_K \mathbf{U}^K$ where $\mathcal{W} \in \mathbb{R}^{R_1 \times \ldots \times R_K}$ is parametric tensor-core and $\times_k$ is mode k tensor-matrix product (Kolda, 2006). The entry-wise form is therefore given by

$$\begin{aligned} y_{\mathbf{i}} &\approx \mathrm{vec}(\mathcal{W})^\top \left(\mathbf{u}_{i_1}^1 \otimes \ldots \otimes \mathbf{u}_{i_K}^K\right) \\ &= \sum_{r_1=1}^{R_1} \cdots \sum_{r_K=1}^{R_K} \left[w_{(r_1,\ldots,r_K)} \cdot \prod_{k=1}^K u_{i_k,r_k}^k \right] \end{aligned} \quad (2)$$

where $\mathrm{vec}(\cdot)$ is the vectorization and $\otimes$ is the Kronecker product. As we can see from (2), every interaction between the factors across the K modes is accounted for, $\{\prod_{k=1}^K u_{i_k,r_k}^k | 1 \le r_1 \le R_1, \ldots, 1 \le r_K \le R_K\}$. Each interaction is weighted by an element of the tensor-core. It is easy to see that CP is a special case of Tucker decomposition when we set all $R_k = R$ and $\mathcal{W}$ to be diagonal.

Gaussian Processes (GPs) are powerful Bayesian function estimators. Due to the nonparametric nature, GPs can automatically grasp the complexity of the target function underlying the data (*e.g.,* from linear to highly linear), not restricted to any parametric form. Specifically, suppose given N training examples, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$ and $\mathbf{y} = (y_1, \ldots, y_N)^\top$, we want to learn a function $f : \mathbb{R}^d \to \mathbb{R}$. We place a GP prior over the target function, and then any finite set of the function values follow a multivariate Gaussian distribution. Consider $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\top$ and we have $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{K})$, where $\mathbf{m}$ is the mean function value at the inputs and usually set to $\mathbf{0}$, and $\mathbf{K}$ is an $N \times N$ kernel matrix — each element $[\mathbf{K}]_{n,n'} = \kappa(\mathbf{x}_n, \mathbf{x}_{n'})$ and $\kappa(\cdot,\cdot)$ is a kernel function. A commonly used, powerful kernel is Matérn kernel,

$$k\left(\mathbf{x}_n, \mathbf{x}_{n'}\right) = \sigma^2 \frac{\left(\frac{\sqrt{2\nu}}{l}\alpha(\mathbf{x}_n, \mathbf{x}_{n'})\right)^\nu}{\Gamma(\nu)2^{\nu-1}} K_\nu \left(\frac{\sqrt{2\nu}}{l}\alpha(\mathbf{x}_n, \mathbf{x}_{n'})\right)$$

where $\alpha(\cdot,\cdot)$ is the distance function (usually the Euclidean distance), $\Gamma(\cdot)$ is the gamma function, $K_\nu(\cdot)$ is the modified Bessel function of the second kind, ν is the degree of freedom, l is length-scale and σ^2 magnitude. Given $\mathbf{f}$, we use a noise model $p(\mathbf{y}|\mathbf{f})$ to fit the observed function outputs, *e.g.,* $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \tau^{-1}\mathbf{I})$. We can then conduct Bayesian inference. The predictive distribution of the function value at a new input $\mathbf{x}^*$ is straightforward to obtain: since $[\mathbf{f}; f(\mathbf{x}^*)]$ follows a joint Gaussian distribution as well and $p(f(\mathbf{x}^*)|\mathbf{f})$ is a conditional Gaussian distribution.

SDE Representation of Temporal GPs. In the literature of stochastic differential equations (SDEs) (Särkkä et al., 2006; Oksendal, 2013), it is known that the solution of linear SDEs are Gaussian processes on time, namely, temporal GPs. From the other side, for temporal GPs with certain stationary kernels, we can construct an equivalent Linear Time-Invariant (LTI) SDE through spectral analysis (Hartikainen and Särkkä, 2010). Take the Matérn kernel with $\nu = m + \frac{1}{2}$ (where $m \in \mathbb{N}$) as an example. We can obtain its power spectral density as $S(\omega) = P(\mathrm{i}\omega)q_c P(-\mathrm{i}\omega)$, where $P(\mathrm{i}\omega) = \frac{1}{(\beta+\mathrm{i}\omega)^{m+1}}$, i indicates the imaginary part, $\beta = \sqrt{2\nu}/l$, and $q_c = \frac{2\sigma^2\pi^{1/2}\beta^{2m+1}\Gamma(m+1)}{\Gamma(m+1/2)}$. This is equivalent to feeding a white noise process with diffusion q_c into a system, who transfers the signal with $P(\mathrm{i}\omega)$ to generate the output. Via inverse Fourier transform, we know the output process is the solution of the SDE

$$\frac{\mathrm{d}^{m+1}f(t)}{\mathrm{d}t^{m+1}} + a_m\frac{\mathrm{d}^m f(t)}{\mathrm{d}t^m} + \ldots + a_0 f(t) = \xi(t), \quad (3)$$

where $\xi(t)$ is the white noise process with diffusion q_c, and $a_0, \ldots, a_m$ are the coefficients of the zeroth, first, till m-th term in the polynomial of $P(\mathrm{i}\omega)$'s denominator. This can be further written as an LTI-SDE, in which we define the state as $\mathbf{y}(t) = \left(f(t), \frac{\mathrm{d}f(t)}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}f^m(t)}{\mathrm{d}t}\right)^\top$, and

$$\frac{\mathrm{d}\mathbf{y}(t)}{\mathrm{d}t} = \mathbf{F}\mathbf{x}(t) + \mathbf{L}\xi(t), \quad (4)$$

where

$$\mathbf{F} = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_0 & \ldots & -a_{m-1} & -a_m \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

In general, although we cannot guarantee the power spectrum $S(\omega)$ of the kernel has a polynomial form in the denominator, we can apply Taylor approximation on $1/S(\omega)$ to construct an approximately equivalent LTI-SDE.

3. Model

While useful, existing tensor decomposition methods use discrete time steps, and hence can miss the temporal variations within each step. Although the latest work (Zhang et al., 2021) employs continuous-time coefficients in the CP decomposition (see $\boldsymbol{\lambda}$ in (1)), it uses polynomial splines to model these coefficients and might not be sufficient to capture more complex dynamics. The CP form can further restrict its capability of capturing temporal interactions between the factors across different tensor modes. To overcome these limitations, we propose BCTT, a Bayesian continuous-time Tucker decomposition approach.

Specifically, we model each element of the tensor-core $\mathcal{W}$ in the Tucker decomposition (2) as a time-varying (or trend) function so as to capture the temporal interactions across all the factor combinations. In order to flexibly estimate a variety of complex temporal variations, we place a GP prior over each element, $w_{\mathbf{r}}(t) \sim \mathcal{GP}\left(0, \kappa(t, t')\right)$ where $\mathbf{r} = (r_1, \ldots, r_K)$. Given the observed tensor entry values and time points, $\mathcal{D} = \{(\mathbf{i}_1, t_1, y_1), \ldots, (\mathbf{i}_N, t_N, y_N)\}$, we have a multi-variate Gaussian prior over the values of $w_{\mathbf{r}}(\cdot)$ at the observed timestamps, $p(\mathbf{w}_{\mathbf{r}}) = \mathcal{N}(\mathbf{w}_{\mathbf{r}}|\mathbf{0}, \mathbf{K}_{\mathbf{r}})$, where $\mathbf{w}_{\mathbf{r}} = [w_{\mathbf{r}}(t_1), \ldots, w_{\mathbf{r}}(t_N)]^\top$, $\mathbf{K}_{\mathbf{r}}$ is the $N \times N$ kernel matrix on the time points and each $[\mathbf{K}_r]_{n,n'} = \kappa(t_n, t_{n'})$. Given $\mathcal{W}(t_n) = \{w_{\mathbf{r}}(t_n)\}_{\mathbf{r}}$, we sample the observed entry value from $p(y_n|\mathcal{W}(t_n), \mathcal{U}) = \mathcal{N}\left(y_n|\mathrm{vec}\left(\mathcal{W}(t_n)\right)^\top\left(\mathbf{u}^1_{i_{n_1}} \otimes \ldots \otimes \mathbf{u}^K_{i_{n_K}}\right), \tau^{-1}\right)$, where τ is the inverse variance, for which we place a Gamma prior, $p(\tau) = \mathrm{Gam}(\tau|b_0, c_0)$. Here we only consider continuous observations. However, it is straightforward to extend our model and inference to other types of entry values. We further place a standard Gaussian prior over the latent factors $p(\mathcal{U}) = \prod_{k=1}^K \prod_{j=1}^{d_k} \mathcal{N}(\mathbf{u}^k_j|\mathbf{0}, \mathbf{I})$. The joint probability is

$$p(\mathcal{U}, \{\mathbf{w}_{\mathbf{r}}\}_{\mathbf{r}}, \tau, \mathbf{y}) = p(\mathcal{U})p(\tau) \cdot \prod_{\mathbf{r}=(1,\ldots,1)}^{(R_1,\ldots,R_K)} \mathcal{N}(\mathbf{w}_{\mathbf{r}}|\mathbf{0}, \mathbf{K}_{\mathbf{r}}) \cdot \prod_{n=1}^N p(y_n|\mathcal{W}(t_n), \mathcal{U}). \quad (5)$$

However, a straightforward formulation as in (5) brings in severe computational challenges. The joint probability includes many multivariate Gaussian distributions, *i.e.,* $\mathcal{N}(\mathbf{w}_{\mathbf{r}}|\mathbf{0}, \mathbf{K}_{\mathbf{r}})$. When the number of time points N is large, the calculation of each kernel matrix $\mathbf{K}_{\mathbf{r}}$ and its inverse (in the distribution) is extremely expensive or even infeasible ($\mathcal{O}(N^3)$ time complexity). To overcome this hurdle, we have to seek for various sparse GP approximations (Quiñonero-Candela and Rasmussen, 2005), which essentially use aggressive low-rank structures to approximate the kernel matrices.

To prevent sparse/low-rank approximations (which can be of low quality), we use SDEs to formulate our model so as to perform full GP inference with a linear cost in N. Specifically, we observe that each $w_{\mathbf{r}}(t)$ is actually a temporal GP. Therefore, we can construct an equivalent LTI-SDE. For convenience, we use the Matérn kernel with $\nu = 3/2 = 1 + 1/2$ for illustration. According to (4), for each $w_{\mathbf{r}}(t)$, we define a state $\boldsymbol{\gamma}_{\mathbf{r}}(t) = (w_{\mathbf{r}}, \frac{\mathrm{d}w_{\mathbf{r}}}{\mathrm{d}t})^\top$, and the SDE is

$$\frac{\mathrm{d}\boldsymbol{\gamma}_{\mathbf{r}}(t)}{\mathrm{d}t} = \mathbf{F}\boldsymbol{\gamma}_{\mathbf{r}} + \mathbf{L}\xi(t), \quad (6)$$

where $\mathbf{F} = [0, 1; -\beta^2, -2\beta]$, $\mathbf{L} = [0; 1]$, and the diffusion of the white noise $\xi(t)$ is $q_c = 4\beta^3\sigma^2$. The benefit of the LTI-SDE representation is that its discrete form (on $t_1, \ldots, t_N$) is a Gaussian Markov chain,

$$\begin{aligned} &p(\boldsymbol{\gamma}_{\mathbf{r}}(t_1)) = \mathcal{N}(\boldsymbol{\gamma}_{\mathbf{r}}(t_1)|\mathbf{0}, \mathbf{P}_\infty), \\ &p(\boldsymbol{\gamma}_{\mathbf{r}}(t_{n+1})|\boldsymbol{\gamma}_{\mathbf{r}}(t_n)) = \mathcal{N}\left(\boldsymbol{\gamma}_{\mathbf{r}}(t_{n+1})|\mathbf{A}_n\boldsymbol{\gamma}_{\mathbf{r}}(t_n), \mathbf{Q}_n\right) \end{aligned} \quad (7)$$

where $\mathbf{P}_\infty = [\sigma^2, 0; 0, \beta^2\sigma^2]$ is the stationary covariance calculated by solving the matrix Riccati equation (Lancaster and Rodman, 1995), $\Delta_n = t_{n+1} - t_n$ is the time difference, $\mathbf{A}_n = \exp(\mathbf{F}\Delta_n)$, and $\mathbf{Q}_n = \mathbf{P}_\infty - \mathbf{A}_n\mathbf{P}_\infty\mathbf{A}_n^\top$.

To represent all the temporal GPs in our model, we define a joint state $\overline{\gamma}(t)$ as the concatenation of all $\{\gamma_{\mathbf{r}}(t)\}_{\mathbf{r}}$. Accordingly, the discrete form of the SDE for $\overline{\gamma}(t)$ follows

$$\begin{aligned} p(\overline{\gamma}_1) &= \mathcal{N}(\overline{\gamma}_1|\mathbf{0}, \mathbf{\Sigma}), \\ p(\overline{\gamma}_{n+1}|\overline{\gamma}_n) &= \mathcal{N}(\overline{\gamma}_{n+1}|\mathbf{B}_n\overline{\gamma}_n, \mathbf{C}_n), \end{aligned} \tag{8}$$

where $\overline{\gamma}_n \triangleq \overline{\gamma}(t_n)$, $\mathbf{\Sigma} = \text{diag}(\mathbf{P}_\infty, \ldots, \mathbf{P}_\infty)$, $\mathbf{B}_n = \text{diag}(\mathbf{A}_n, \ldots, \mathbf{A}_n)$, and $\mathbf{C}_n = \text{diag}(\mathbf{Q}_n, \ldots, \mathbf{Q}_n)$. As we can see, this is essentially a state-space prior over the collection of states $\{\overline{\gamma}_n\}$. To extract the tensor-core $\mathcal{W}(t)$, we can use a sparse $\overline{R} \times 2\overline{R}$ matrix,

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & & & & & \\ & & 1 & 0 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 0 \end{pmatrix},$$

to obtain $\text{vec}(\mathcal{W}(t)) = \mathbf{H} \cdot \overline{\gamma}(t)$, where $\overline{R}$ is the size of the tensor-core, $\overline{R} = \prod_{k=1}^K R_k$.

Now, we replace the multivariate Gaussians in (5) by the state space prior in (8), and write the joint probability as

$$p(\mathcal{U}, \{\overline{\gamma}_n\}, \tau, \mathbf{y}) = p(\mathcal{U})p(\tau) \cdot p(\overline{\gamma}_1) \prod_{n=1}^{N-1} p(\overline{\gamma}_{n+1}|\overline{\gamma}_n)$$
$$\prod_{n=1}^{N} \mathcal{N}\left(y_n | (\mathbf{H}\overline{\gamma}_n)^\top \left(\mathbf{u}^1_{i_{n_1}} \otimes \ldots \otimes \mathbf{u}^K_{i_{n_K}}\right), \tau^{-1}\right). \tag{9}$$

Since each state $\overline{\gamma}_n$ is only dependent on its previous state $\overline{\gamma}_{n-1}$ (Markov property), we no longer need to compute a giant $N \times N$ covariance matrix nor need low-rank approximations. The state space prior enables us to develop an efficient, linear GP inference algorithm, as presented in the next section.

4. Algorithm

The exact posterior of our model is infeasible to calculate, because the likelihood of each data point n (arising from the entry-wise Tucker decomposition (2)) couples the relevant latent factors $\{\mathbf{u}^1_{i_{n_1}}, \ldots \mathbf{u}^K_{i_{n_K}}\}$ and state $\overline{\gamma}(t_n)$. To address this issue, we introduce Gaussian-Gamma likelihood approximations, and based on Kalman filtering (KF) (Kalman, 1960) and Rauch-Tung-Striebel (RTS) smoothing (Rauch et al., 1965) we develop an efficient message-passing algorithm in the expectation propagation (EP) framework (Minka, 2001a). See Fig.1.

4.1. Gaussian-Gamma Approximations for Efficient Filtering and Smoothing

Specifically, we approximate each data likelihood with

$$\begin{aligned} &\mathcal{N}\left(y_n | (\mathbf{H}\overline{\gamma}_n)^\top \left(\mathbf{u}^1_{i_{n_1}} \otimes \ldots \otimes \mathbf{u}^K_{i_{n_K}}\right), \tau^{-1}\right) \approx \ell_n \\ &\triangleq Z_n \cdot \prod_{k=1}^K \mathcal{N}\left(\mathbf{u}^k_{i_{n_k}} | \mathbf{m}^{k,n}_{i_{n_k}}, \mathbf{V}^{k,n}_{i_{n_k}}\right) \cdot \text{Gam}\left(\tau | b_n, c_n\right) \\ &\cdot \mathcal{N}\left(\mathbf{H}\overline{\gamma}_n \mid \boldsymbol{\beta}_n, \mathbf{S}_n\right), \end{aligned} \tag{10}$$

where Z_n is a normalization term (it will be canceled during inference). Hence we obtain the approximate posterior by

$$q(\mathcal{U}, \{\overline{\gamma}_n\}, \tau) \propto \prod_{k=1}^K \prod_{j=1}^{d_k} \mathcal{N}(\mathbf{u}^k_j|\mathbf{0}, \mathbf{I})\text{Gam}(\tau|b_0, c_0)$$
$$\prod_{n=1}^{N}\prod_{k=1}^{K} \mathcal{N}\left(\mathbf{u}^k_{i_{n_k}} | \mathbf{m}^{k,n}_{i_{n_k}}, \mathbf{V}^{k,n}_{i_{n_k}}\right) \text{Gam}\left(\tau | b_n, c_n\right) \tag{11}$$
$$p(\overline{\gamma}_1)\mathcal{N}\left(\mathbf{H}\overline{\gamma}_1 \mid \boldsymbol{\beta}_1, \mathbf{S}_1\right) \prod_{n=1}^{N-1} p(\overline{\gamma}_{n+1}|\overline{\gamma}_n)\mathcal{N}\left(\mathbf{H}\overline{\gamma}_n \mid \boldsymbol{\beta}_n, \mathbf{S}_n\right).$$

The parameters of the approximation terms, including $\{\mathbf{m}^{k,n}_{i_{n_k}}, \mathbf{V}^{k,n}_{i_{n_k}}, b_n, c_n, \boldsymbol{\beta}_n, \mathbf{S}_n\}$, will be updated and estimated during the message-passing inference. After that, we can obtain the (approximate) posterior of latent factors and noise inverse variance τ by merging relevant terms, $q(\mathbf{u}^k_j) \propto \mathcal{N}(\mathbf{u}^k_j|\mathbf{0}, \mathbf{I}) \prod_{i_{n_k}=j} \mathcal{N}(\mathbf{u}^k_{i_{n_k}} | \mathbf{m}^{k,n}_{i_{n_k}}, \mathbf{V}^{k,n}_{i_{n_k}})$, and $q(\tau) \propto \text{Gam}(\tau|b_0, c_0) \prod_{n=1}^N \text{Gam}(\tau|b_n, c_n)$, which have closed forms, *i.e.,* Gaussian or Gamma.

However, the posterior of the states $\overline{\gamma}_n$ is not easy to obtain, because $\overline{\gamma}_n$ are chained in the state space prior. Thanks to the Gaussian term $\mathcal{N}\left(\mathbf{H}\overline{\gamma}_n \mid \boldsymbol{\beta}_n, \mathbf{S}_n\right)$ introduced in (10) — by symmetry, we can view it as $\mathcal{N}\left(\boldsymbol{\beta}_n | \mathbf{H}\overline{\gamma}_n, \mathbf{S}_n\right)$ — a Gaussian likelihood (emission) of the virtual observation $\boldsymbol{\beta}_n$ following the state space prior of each $\overline{\gamma}_n$ (see the third line of (11)). Therefore, we can apply the standard KF in a forward pass and RTS smoothing in a backward pass to efficiently compute all the marginal posteriors $q(\overline{\gamma}_n)$ and $q(\overline{\gamma}_n, \overline{\gamma}_{n+1})$, with a linear cost in N (*i.e.,* $\mathcal{O}(N)$ complexity). Note that the standard KF and RTS can only be used for Gaussian emissions but they give exact results. For non-Gaussian likelihoods, we have to combine with extra approximations, such as extended KF and unscented KF (Särkkä, 2013), which can be unstable and more costly.

4.2. Message Passing with Conditional Moment Matching

To optimize the approximation terms in each ℓ_n (see (10)), we develop a message- passing algorithm in the EP framework. Specifically, at each step, given all ℓ_n, we first run KF and RST smoothing to calculate the posterior of each sate $q(\overline{\gamma}_n)$. The calculation is actually the standard message passing in chain graphical models (Bishop, 2006). Each

Gaussian term $\mathcal{N}\left(\mathbf{H}\overline{\gamma}_n \mid \boldsymbol{\beta}_n, \mathbf{S}_n\right)$ is the initial message sent from data point n to the state $\overline{\gamma}_n$, then we conduct KF to compute the message from each $\overline{\gamma}_n$ to $\overline{\gamma}_{n+1}$ (forward pass), and then RTS smoothing the messages from $\overline{\gamma}_{n+1}$ to $\overline{\gamma}_n$ (backward pass). The posterior $q(\overline{\gamma}_n)$ is obtained by aggregating all the messages sent to $\overline{\gamma}_n$ (*i.e.,* those from $\overline{\gamma}_{n-1}$, $\overline{\gamma}_{n+1}$ and data point n), which ends up with a Gaussian distribution.

Next, we use the state posteriors $\{q(\overline{\gamma}_n)\}$ to update the likelihood approximation terms in $\{\ell_n\}$ via EP. Specifically, for each data point n, we obtain a calibrated distribution by dividing the global posterior by the current approximation,

$$q^{\backslash n}(\boldsymbol{\Theta}_n) \propto \frac{q(\overline{\gamma}_n)q(\tau)\prod_{k=1}^K q(\mathbf{u}^k_{i_{n_k}})}{\ell_n} = \mathcal{N}(\boldsymbol{\eta}_n|\boldsymbol{\beta}^{\backslash n}, \mathbf{S}^{\backslash n})$$
$$\cdot \prod_{k=1}^K \mathcal{N}\left(\mathbf{u}^k_{i_{n_k}} | \mathbf{m}^{k,\backslash n}_{i_{n_k}}, \mathbf{V}^{k,\backslash n}_{i_{n_k}}\right) \mathrm{Gam}\left(\tau | b^{\backslash n}, c^{\backslash n}\right),$$

where $\boldsymbol{\eta}_n = \mathbf{H}\overline{\gamma}_n = \mathrm{vec}\left(\mathcal{W}(t_n)\right)$, and $\boldsymbol{\Theta}_n = \{\boldsymbol{\eta}_n, \{\mathbf{u}^k_{i_{n_k}}\}_k, \tau\}$ are all the random variables present in the n-th likelihood. The calibrated distribution integrates the information from all the other data points, *i.e.,* the context. To update the terms in ℓ_n, we construct a tilted distribution,

$$\widetilde{p}(\boldsymbol{\Theta}_n) \propto q^{\backslash n}(\boldsymbol{\Theta}_n)$$
$$\cdot \mathcal{N}\left(y_n | \boldsymbol{\eta}_n^\top \left(\mathbf{u}^1_{i_{n_1}} \otimes \ldots \otimes \mathbf{u}^K_{i_{n_K}}\right), \tau^{-1}\right). \tag{12}$$

We aim to project the tilted distribution back to our approximation family (exponential family), to obtain

$$q^*(\boldsymbol{\Theta}_n) = \mathcal{N}(\boldsymbol{\eta}_n | \boldsymbol{\beta}^*_n, \mathbf{S}^*_n) \tag{13}$$
$$\cdot \prod_{k=1}^K \mathcal{N}\left(\mathbf{u}^k_{i_{n_k}} | \mathbf{m}^{k,*}_{i_{n_k}}, \mathbf{V}^{k,*}_{i_{n_k}}\right) \mathrm{Gam}\left(\tau | b^*_n, c^*_n\right),$$

from which we update ℓ_n terms via dividing the calibrated distribution back,

$$\ell_n \leftarrow \frac{q^*(\boldsymbol{\Theta}_n)}{q^{\backslash n}(\boldsymbol{\Theta}_n)}. \tag{14}$$

The projection essentially is to minimize the Kullback-Leibler divergence from $\widetilde{p}(\boldsymbol{\Theta}_n)$ to $q^*(\boldsymbol{\Theta}_n)$, which can be done by moment matching. For example, the Gaussian posterior of $\boldsymbol{\eta}_n$ in (13) needs two moments — the expectation of $\boldsymbol{\eta}_n$ and $\boldsymbol{\eta}_n\boldsymbol{\eta}_n^\top$. So we need to compute them under the tilted distribution so as to match the parameters of $q^*(\boldsymbol{\eta}_n)$, namely $\boldsymbol{\beta}^*_n = \mathbb{E}_{\widetilde{p}}\left[\boldsymbol{\eta}_n\right]$ and $\mathbf{S}^*_n = \mathbb{E}_{\widetilde{p}}\left[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^\top\right] - \mathbb{E}_{\widetilde{p}}\left[\boldsymbol{\eta}_n\right]\mathbb{E}_{\widetilde{p}}\left[\boldsymbol{\eta}\right]^\top$.

The standard EP assumes the moment matching is computationally tractable. However, this is not the case in our model. Since $\boldsymbol{\eta}_n$ and the latent factors are coupled in the product and Kronecker product in the tilted distribution (12), we do not have a closed form of the moments. To address this problem, we use the idea of the conditional moment matching (Wang and Zhe, 2019). Take $\boldsymbol{\eta}_n$ as an example. Denote the required moments by $\boldsymbol{\phi}(\boldsymbol{\eta}_n) = \left(\boldsymbol{\eta}_n, \boldsymbol{\eta}_n\boldsymbol{\eta}_n^\top\right)$. The key observation is that we can decompose the expectation into a nested structure,

$$\mathbb{E}_{\widetilde{p}}\left[\boldsymbol{\phi}(\boldsymbol{\eta}_n)\right] = \mathbb{E}_{\widetilde{p}(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})}\left[\mathbb{E}_{\widetilde{p}(\boldsymbol{\eta}_n|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})}\left[\boldsymbol{\phi}(\boldsymbol{\eta})|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]\right]$$

where $\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n} \triangleq \boldsymbol{\Theta}_n \backslash \{\boldsymbol{\eta}_n\}$. Therefore, we can compute the conditional moment first, *i.e.,* the inner expectation, and then take expectation over the conditional moments, *i.e.,* the outer-expectation. Given all the other variables $\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}$ fixed, the conditioned tilted distribution $\widetilde{p}(\boldsymbol{\eta}_n|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})$ is simply a Gaussian. Hence, the conditional moment is easy to obtain,

$$\mathbb{E}\left[\boldsymbol{\eta}_n|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right] = \boldsymbol{\Sigma}_n\left(\left(\mathbf{S}^{\backslash n}\right)^{-1}\boldsymbol{\beta}^{\backslash n} + \tau y_n \mathbf{v}_n\right), \tag{15}$$
$$\mathbb{E}\left[\boldsymbol{\eta}_n\boldsymbol{\eta}_n^\top|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right] = \boldsymbol{\Sigma}_n + \mathbb{E}\left[\boldsymbol{\eta}_n|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]\mathbb{E}\left[\boldsymbol{\eta}_n|\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]^\top$$

where $\mathbf{v}_n = \mathbf{u}^1_{i_{n_1}} \otimes \ldots \otimes \mathbf{u}^K_{i_{n_K}}$ and $\boldsymbol{\Sigma}_n = \left(\left(\mathbf{S}^{\backslash n}\right)^{-1} + \tau\mathbf{v}_n\mathbf{v}_n^\top\right)^{-1}$.

Next, we need to take the outer-level expectation to obtain the moments, namely, computing the mean of the conditional moment under the marginal tilted distribution $\widetilde{p}(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})$. However, since $\widetilde{p}(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})$ is analytically intractable, the outer expectation does not have a closed form. To tackle this issue, we observe that the moment matching is also performed between $q(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})$ and $\widetilde{p}(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})$, and hence we can assume they are close, especially in high density regions. We then use the current posterior as the surrogate to compute the expected conditional moment,

$$\mathbb{E}_{\widetilde{p}}[\boldsymbol{\phi}(\boldsymbol{\eta}_n)] \approx \mathbb{E}_{q(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})}\left[\boldsymbol{\rho}_n\right] \tag{16}$$

where $\boldsymbol{\rho}_n$ is the conditional moment.

Nonetheless, since $\boldsymbol{\rho}_n$ is a nonlinear function of the conditioned variables $\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}$ (see (15)), we do not have a close form to compute (16) either. But we have already known the form of $q(\Theta_{\backslash \boldsymbol{\eta}_n})$, so we can use the multivariate delta method (Oehlert, 1992; Bickel and Doksum, 2015) to compute the expectation easily. Specifically, we use a first-order Taylor approximation to represent the conditional moments,

$$\boldsymbol{\rho}_n(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}) \approx \boldsymbol{\rho}_n\left(\mathbb{E}_q\left[\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]\right)$$
$$+ \mathbf{J} \cdot \left(\mathrm{vec}\left(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right) - \mathrm{vec}\left(\mathbb{E}_q\left[\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]\right)\right) \tag{17}$$

where $\mathbf{J}$ is the Jacobian at $\mathbb{E}_q\left[\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]$. Then taking the expectation over the Taylor approximation gives

$$\mathbb{E}_{q(\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n})}\left[\boldsymbol{\rho}_n\right] \approx \boldsymbol{\rho}_n\left(\mathbb{E}_q\left[\boldsymbol{\Theta}_{\backslash \boldsymbol{\eta}_n}\right]\right). \tag{18}$$

We refer to (Oehlert, 1992; Wolter, 2007) for the theoretical justifications and guarantees of the delta method. With the same approach, we can compute the moments for other random variables in $\boldsymbol{\Theta}$, including $\{\mathbf{u}^k_{i_{n_k}}\}$ and τ, and obtain

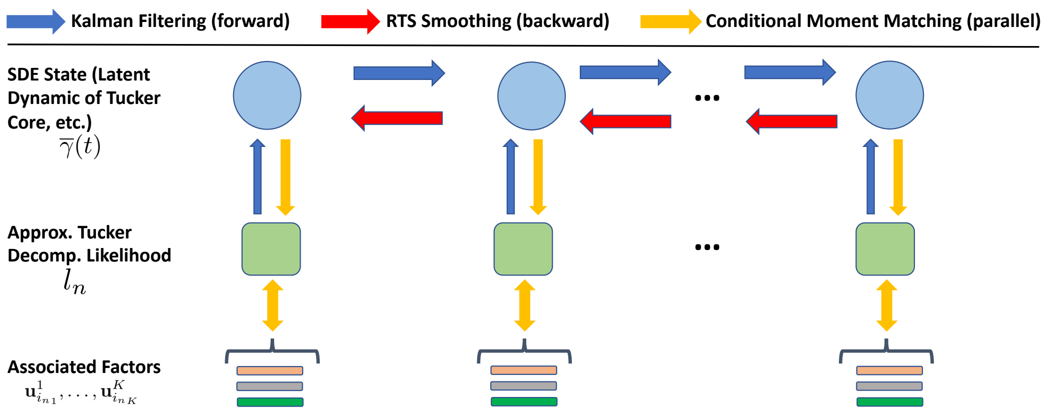

Kalman Filtering (forward)
RTS Smoothing (backward)
Conditional Moment Matching (parallel)
SDE State (Latent Dynamic of Tucker Core, etc.)
$\overline{\gamma}(t)$
Approx. Tucker Decomp. Likelihood
l_n
Associated Factors
$\mathbf{u}^1_{i_{n1}}, \ldots, \mathbf{u}^K_{i_{nK}}$
...

Figure 1. Graphical illustration of the message-passing inference algorithm.

Algorithm 1 BCTT

```
**Input:** $\mathcal{D} = \{(\mathbf{i}_1, t_1, y_1), \ldots, (\mathbf{i}_N, t_N, y_N)\}$, kernel hyper-parameters $l, \sigma^2$
Initialize approximation terms in (10) for each likelihood.
**repeat**
  Run KF and RTS smoothing to compute each $q(\overline{\gamma}_n)$
  **for** $n = 1$ **to** $N$ in parallel **do**
    Simultaneously update $\mathcal{N}(\mathbf{H}\overline{\gamma}_n|\boldsymbol{\beta}_n, \mathbf{S}_n)$, $\text{Gam}(\tau|b_n, c_n)$ and $\left\{\mathcal{N}\left(\mathbf{u}^k_{i_{n_k}}|\mathbf{m}^{k,n}_{i_{n_k}}, \mathbf{V}^{k,n}_{i_{n_k}}\right)\right\}_k$ in (10) with conditional moment matching and multi-variate delta method.
  **end for**
**until** Convergence
**Return:** $\{q(\mathcal{W}(t_n))\}_{n=1}^N$, $\{q(\mathbf{u}^k_j)\}_{1 \le k \le K, 1 \le j \le d_k}$, $q(\tau)$
```

their posterior in (13). Finally, we apply (14) to update the approximation terms in the likelihood.

While the derivation of the conditional moment matching is a bit lengthy, the implementation is straightforward. From (18) and (16), we just need to derive the form of the conditional moments (in our case, it is either Gaussian or Gamma), and then plug in the expectation of the conditioned variables under the current poster. For efficiency, we update the approximation factors of all the likelihoods in parallel, and then perform damping to be stable (Minka, 2001b). We repeatedly do message passing and conditional moment matching until convergence. The model inference is summarized in Algorithm 1.

4.3. Algorithm Complexity

In each iteration, our algorithm runs KF and RTS smoothing to go through data twice, so as to calculate the posterior of each state $\overline{\gamma}_n$, and then conduct conditional moment matching in parallel to update the likelihood approximation for each data point. The overall time complexity is $\mathcal{O}(N\overline{R})$, where $\overline{R}$ is the size of the tensor-core. The space complexity is $\mathcal{O}\left(N(\overline{R}^2 + \sum_{k=1}^K R_k^2)\right)$ which is to store the posterior of each state and the likelihood approximation terms at each data point. Hence, our algorithm enjoys a linear scalability with the growth of data. Note that our algorithm fulfills the full GP inference, without the need for any sparse or low-rank approximations. As a comparison, the naive GP model demands $\mathcal{O}(\overline{R}N^3)$ time and $\mathcal{O}(\overline{R}N^2)$ space complexity, and hence can be extremely expensive or infeasible for large N. In practice, it is possible that multiple entries were observed at the same time point. Adjusting our method for such cases is trivial. Since the number of states is smaller than N accordingly, the complexity is even lower.

5. Related Work

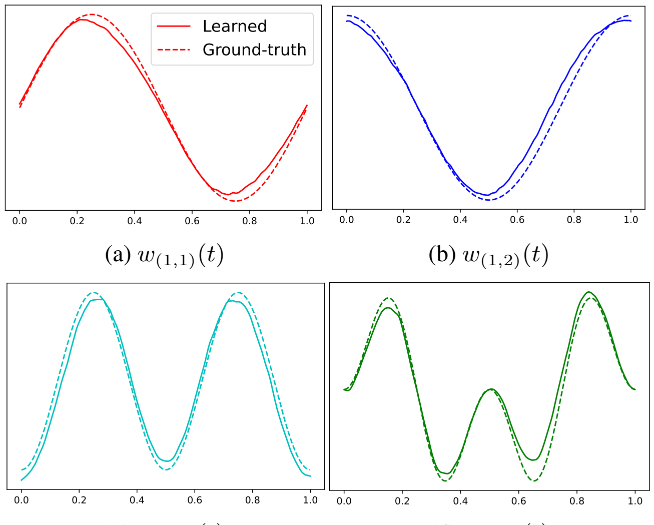
There are many tensor decomposition methods, *e.g.,* (Chu and Ghahramani, 2009; Kang et al., 2012; Choi and Vishwanathan, 2014; Zhe et al., 2016a; Liu et al., 2018; Pan et al., 2020b; Tillinghast and Zhe, 2021; Fang et al., 2021b; Tillinghast et al., 2022). To utilize time information, current methods expand the tensor with a time mode (Xiong et al., 2010; Rogers et al., 2013; Du et al., 2018; Zhe et al., 2016b; 2015; Ahn et al., 2021; Wu et al., 2019), which comprises a set of discrete time steps, *e.g.,* by hours or days. The observed entry values are then arranged into different time slices of the tensor. The factors of the time steps and tensor nodes are jointly learned during the decomposition. To better estimate the temporal relationships, a few more refined approaches model the transition between the time steps, *e.g.,* a conditional linear Gaussian prior in (Xiong et al., 2010), RNN (Wu et al., 2019), and kernel smoothing and regularization in (Ahn et al., 2021). To conduct continuous-time decomposition, the latest work (Zhang et al., 2021) uses polynomial splines to estimate the factor coefficients ($\boldsymbol{\lambda}$ in (1)) in the CP model as a time function. Another set of works (Schein et al., 2015; 2016; Zhe and Du, 2018; Pan et al., 2020a; Wang et al., 2020) decompose the events between the tensor nodes. The entry values are event counts or event sequences. These methods either use Poisson processes or more complex temporal point processes, such as Hawkes processes (Hawkes, 1971). However, these methods do not consider the result of events, *e.g.,* payment or product ratings.

Message passing is a general inference framework in probabilistic graphical models (Wainwright and Jordan, 2008). When the model has a chain or tree structure and the factors in the graph (*i.e.,* terms in probability) are tractable, message passing can perform exact inference in a highly efficient way. Kalman filter and RTS smoothing are examples. When the factors are complex (*e.g.,* not in the exponential family), the computation of the messages can be intractable. Minka (2001a) proposed a more general framework, Expectation propagation (EP), to handle the message computation via moment matching. However, it can still fail when moment matching is intractable. To address this problem, Wang and Zhe (2019) proposed conditional EP (CEP) that uses conditional moment matching, Taylor approximations and numerical quadrature to compute the intractable moments for fully factorized posteriors. CEP has been used in Bayesian CP decomposition (Wang and Zhe, 2019) and shown great performance. (Fang et al., 2021a) has used CEP in the streaming inference of a sparse Tucker decomposition model where a spike-and-slab prior is placed on the tensor-core and approximated on the fly to obtain the running posterior. Our work uses GPs to estimate a time-varying tensor-core to handle continuous time information in the Tucker decomposition framework. To avoid huge kernel matrix computations and/or low-rank approximations, we use LTI-SDEs to build an equivalent state-space prior, which is essentially a Gaussian Markov chain. Under the chain structure, we combine the message passing and moment matching for efficient inference. We use similar ideas as in (Wang and Zhe, 2019; Fang et al., 2021a) to compute the Gaussian messages to the SDE states. Given these messages, we then perform KF and RTS smoothing to calculate the posterior of the SDE states in an exact way, which are in turn used to update the approximation terms in each likelihood. In this way, we achieve the linear time complexity for our Tucker-GP model.

6. Experiment

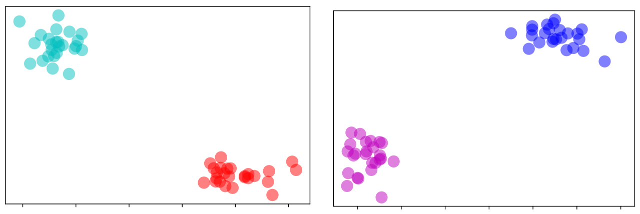
6.1. Ablation Study

We first evaluated BCTT on a synthetic task. We simulated a two-mode tensor, where each mode includes 50 nodes. For each node, we generated two latent factors that reflect a clustering structure in each mode. Specifically, for the nodes in mode 1, we sampled the latent factors $\mathbf{u}_j^1$ from $\mathcal{N}([-1;1], 0.1\mathbf{I})$ for $1 \le j \le 25$, and from $\mathcal{N}([1;-1], 0.1\mathbf{I})$ for $26 < j \le 50$. Similarly, for the nodes in mode 2, we sampled $\mathbf{u}_j^2 \sim \mathcal{N}([1;1], 0.1\mathbf{I})$ when $1 \le j \le 25$, and $\mathcal{N}([-1;-1], 0.1\mathbf{I})$ for $26 < j \le 50$.



(a) $w_{(1,1)}(t)$ (b) $w_{(1,2)}(t)$ (c) $w_{(2,1)}(t)$ (d) $w_{(2,2)}(t)$

Figure 2. Recovered temporal dynamics within factor interactions.



(a) $\mathbf{U}^1$ (b) $\mathbf{U}^2$

Figure 3. The estimated latent factors by BCTT

Given the latent factors, we generate the tensor entry values at any time t from

$$y_{\mathbf{i}}(t) = u^1_{i_1,1}u^2_{i_2,1}w_{(1,1)}(t) + u^1_{i_1,1}u^2_{i_2,2}w_{(1,2)}(t) + u^1_{i_1,2}u^2_{i_2,1}w_{(2,1)}(t) + u^1_{i_1,2}u^2_{i_2,2}w_{(2,2)}(t), \qquad (19)$$

where $w_{(1,1)}(t) = \sin(2\pi t)$, $w_{(1,2)}(t) = \cos(2\pi t)$, $w_{(2,1)}(t) = \sin(2\pi t)\sin(2\pi t)$, and $w_{(2,2)}(t) = \cos(2\pi t)\sin^2(2\pi t)$. These weight functions represent the four temporal interaction patterns between factors across the two modes, corresponding to the tensor-core $\mathcal{W}(t)$ in our model. We generated 2K observed entries from $t \in [0,1]$. We implemented our method BCTT with PyTorch (Paszke et al., 2019). We use the Matérn kernel with $\nu = 3/2$, and set $l = \sigma^2 = 0.1$. We ran our message-passing inference until convergence. The tolerance level was set to 10^{-3}. Then we compared the learned tensor-core $\mathcal{W}(t)$ with the ground-truth interaction functions between every pair of the factors across the two modes[1]. As we can see from Fig.

[1]We normalized each learned interaction function by the maximum posterior mean of the corresponding state. This is to address the identifiablility issue, since scaling $\mathcal{W}$ arbitrarily then re-scaling $\mathcal{U}$ accordingly do not change the Tucker decomposition loss (or likelihood).

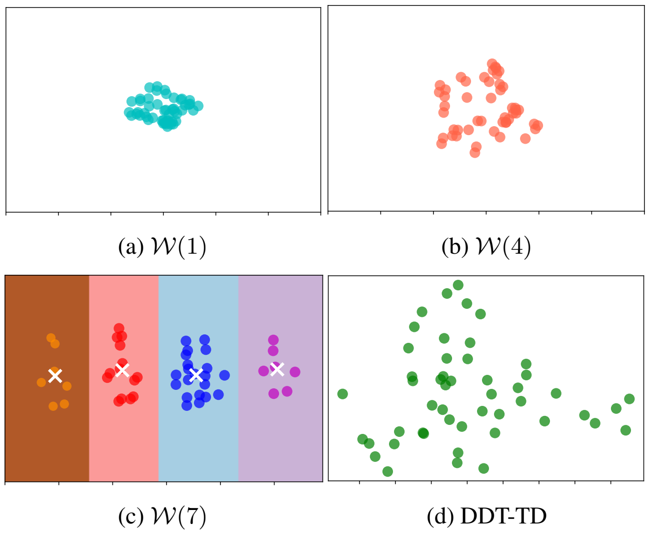

(a) $\mathcal{W}(1)$ (b) $\mathcal{W}(4)$ (c) $\mathcal{W}(7)$ (d) DDT-TD

Figure 4. The structures of learned tensor-core at different time points by BCTT (a-c) and the static tensor-score learned by dynamic discrete-time Tucker decomposition (DDT-TD).

2, our approach recovered each function pretty accurately, showing that BCTT has successful captured all the temporal dynamics within the factor interactions. Next, we show the learned factors in each mode in Fig. 3. The colors indicate the ground-truth cluster membership of the nodes. As we can see, our learned factors clearly revealed the hidden structures of the tensor nodes.

6.2. Real-World Applications

Next, we examined BCTT on three real-world benchmark datasets. (1) *MovieLen100K* (`https://grouplens.org/datasets/movielens/`), a two-mode (user, movie) tensor, of size 610×9729. The entry values are movie ratings at different time points. We have $100,208$ observed entries and their timestamps. (2) *AdsClick* (`https://www.kaggle.com/c/avazu-ctr-prediction`), a three-mode mobile ads click tensor, *(banner-position, site domain, app)*, of size $7 \times 2842 \times 4127$. We collected 50K observed entry values (number of clicks) at different time points (in ten days). *DBLP* (`https://dblp.uni-trier.de/xml/`), a three-mode tensor about bibliographic records in computer science from 2011 to 2021, (author, conference, keyword), of size $3731 \times 1935 \times 169$. The entry values are the numbers of publications. There are 50k entry values and their timestamps.

Methods. We compared with following state-of-the-art multilinear and nonparametric tensor decomposition algorithms with time information integrated. (1) CT-CP (Zhang et al., 2021), continuous-time CP decomposition, which uses polynomial splines to estimates $\boldsymbol{\lambda}$ in (1) as a trend function. (2) CT-GP, continuous-time GP decomposition, which extends (Zhe et al., 2016a) to use GPs to learn tensor element as a function of the latent factors and time $y_{\mathbf{i}}(t) = g(\mathbf{u}^1_{i_1}, \ldots, \mathbf{u}^K_{i_K}, t) \sim \mathcal{GP}(0, \kappa(\cdot,\cdot))$. (3) DT-GP, discrete-time GP decomposition, which expands the tensor with a discrete time mode and then applies GP decomposition. (4) DDT-CP, dynamic discrete-time CP decomposition, which on top of DT-CP, places an RNN-like dynamic prior over the time factors, $p(\mathbf{t}_j|\mathbf{t}_{j-1}) = \mathcal{N}(\mathbf{t}_j|\sigma(\mathbf{A}\mathbf{t}_{j-1}) + \mathbf{b}, v\mathbf{I})$ where $\sigma(\cdot)$ is a nonlinear activation, (5) DDT-TD and (6) DDT-GP, dynamic discrete-time Tucker and GP decomposition, which place the same dynamic prior as in DDT-CP.

Settings. All the methods were implemented by PyTorch. For {CT, DT, DDT}-GP, we used the square exponential kernel and sparse variational GP inference as in (Zhe et al., 2016b) for scalable model estimation. The number of pseudo inputs was 100. For CT-CP, we used 100 knots for the polynomial splines. Except BCTT, all the methods were trained with stochastic mini-batch optimization, with mini-batch size 100. We used ADAM optimization (Kingma and Ba, 2014). The learning rate was chosen from $\{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$. We re-scaled all the timestamps to $[0, 10]$ to ensure numerical stability. We examined all the methods with the number of factors $R \in \{3, 5, 7, 9\}$. Following (Xu et al., 2012; Kang et al., 2012; Zhe et al., 2016b), we randomly sampled 80% observed entry values and their time points for training, and then tested on the remaining entries. For discrete-time decomposition methods, we set the number of time steps to 50 (we tested with more steps but did not obtain improvement). We repeated the experiments for five times, and examined the average root mean-square-error (RMSE), average mean-absolute-error (MAE), and their standard deviations.

Results. As shown in Table 1, our approach BCTT outperforms the competing methods in all the cases except that in Table 1d, on *AdsClicks*, BCTT was the second best, and its MAE is slightly worse than CP-CT. In most cases, the improvement obtained by BCTT is large and significant ($p < 0.05$). It shows that our semi-parametric model BCTT not only maintains the interpretable structure as in Tucker decomposition, but also achieves a superior performance, even to full nonparametric models, *e.g.,* CT-GP and DDT-GP. This might because BCTT uses the state-space representation to enable full GP inference, without any low-rank/sparse approximation as needed in those GP baselines.

Furthermore, we investigated if our learned tensor-core $\mathcal{W}(t)$ can reflect temporal structural variations. To do so, we set $R = 7$ and ran BCTT on *DBLP* dataset. We looked at the tensor-core at three time points $t = 1, 4, 7$. The size of the tensor-core is $7 \times 7 \times 7$. We followed (Fang et al., 2021a) to fold the tensor-core to a 49×7 interaction matrix for each mode. Thus, each row expresses how strongly the combi-

RMSE	*MovieLens*	*AdsClicks*	*DBLP*
CT-CP	1.113 ± 0.004	1.337 ± 0.013	0.240 ± 0.007
CT-GP	0.949 ± 0.008	1.422 ± 0.008	0.227 ± 0.009
DT-GP	0.963 ± 0.008	1.436 ± 0.015	0.227 ± 0.007
DDT-GP	0.957 ± 0.008	1.437 ± 0.010	0.225 ± 0.006
DDT-CP	1.022 ± 0.003	1.420 ± 0.020	0.245 ± 0.004
DDT-TD	1.059 ± 0.006	1.401 ± 0.022	0.232 ± 0.09
BCTT	$\mathbf{0.922 \pm 0.002}$	$\mathbf{1.322 \pm 0.012}$	$\mathbf{0.214 \pm 0.009}$
MAE			
CT-CP	0.788 ± 0.004	0.787 ± 0.006	0.105 ± 0.001
CT-GP	0.714 ± 0.004	0.891 ± 0.011	0.092 ± 0.004
DT-GP	0.722 ± 0.008	0.893 ± 0.008	0.084 ± 0.003
DDT-GP	0.720 ± 0.003	0.894 ± 0.009	0.083 ± 0.001
DDT-CP	0.755 ± 0.002	0.901 ± 0.011	0.114 ± 0.002
DDT-TD	0.742 ± 0.006	0.866 ± 0.012	0.101 ± 0.001
BCTT	$\mathbf{0.698 \pm 0.002}$	$\mathbf{0.777 \pm 0.016}$	$\mathbf{0.084 \pm 0.001}$

(a) $R = 3$

RMSE	*MovieLens*	*AdsClicks*	*DBLP*
CT-CP	1.165 ± 0.008	1.324 ± 0.013	0.263 ± 0.006
CT-GP	0.965 ± 0.019	1.410 ± 0.015	0.227 ± 0.007
DT-GP	0.949 ± 0.007	1.425 ± 0.015	0.225 ± 0.008
DDT-GP	0.948 ± 0.005	1.421 ± 0.012	0.220 ± 0.006
DDT-CP	1.141 ± 0.007	1.623 ± 0.013	0.282 ± 0.011
DDT-TD	0.944 ± 0.003	1.453 ± 0.035	0.312 ± 0.072
BCTT	$\mathbf{0.895 \pm 0.007}$	$\mathbf{1.304 \pm 0.018}$	$\mathbf{0.202 \pm 0.009}$
MAE			
CT-CP	0.835 ± 0.006	0.792 ± 0.007	0.128 ± 0.001
CT-GP	0.717 ± 0.012	0.883 ± 0.016	0.092 ± 0.002
DT-GP	0.714 ± 0.005	0.886 ± 0.012	0.084 ± 0.001
DDT-GP	0.707 ± 0.004	0.882 ± 0.015	0.082 ± 0.003
DDT-CP	0.843 ± 0.003	1.082 ± 0.013	0.141 ± 0.004
DDT-TD	0.712 ± 0.002	0.903 ± 0.024	0.221 ± 0.047
BCTT	$\mathbf{0.679 \pm 0.001}$	$\mathbf{0.785 \pm 0.010}$	$\mathbf{0.080 \pm 0.001}$

(b) $R = 7$

RMSE	*MovieLens*	*AdsClicks*	*DBLP*
CT-CP	1.026 ± 0.002	1.335 ± 0.012	0.244 ± 0.005
CT-GP	0.970 ± 0.011	1.425 ± 0.011	0.229 ± 0.009
DT-GP	0.952 ± 0.012	1.428 ± 0.015	0.226 ± 0.007
DDT-GP	0.949 ± 0.007	1.417 ± 0.013	0.226 ± 0.007
DDT-CP	1.087 ± 0.012	1.515 ± 0.023	0.257 ± 0.006
DDT-TD	1.050 ± 0.005	1.403 ± 0.053	0.277 ± 0.026
BCTT	$\mathbf{0.901 \pm 0.002}$	$\mathbf{1.317 \pm 0.046}$	$\mathbf{0.204 \pm 0.009}$
MAE			
CT-CP	0.813 ± 0.003	0.796 ± 0.006	0.112 ± 0.001
CT-GP	0.731 ± 0.007	0.890 ± 0.012	0.093 ± 0.002
DT-GP	0.720 ± 0.016	0.888 ± 0.011	0.085 ± 0.001
DDT-GP	0.715 ± 0.003	0.879 ± 0.016	0.085 ± 0.001
DDT-CP	0.807 ± 0.003	0.958 ± 0.012	0.120 ± 0.002
DDT-TD	0.784 ± 0.015	0.831 ± 0.038	0.171 ± 0.043
BCTT	$\mathbf{0.684 \pm 0.001}$	$\mathbf{0.776 \pm 0.013}$	$\mathbf{0.082 \pm 0.001}$

(c) $R = 5$

RMSE	*MovieLens*	*AdsClicks*	*DBLP*
CT-CP	1.188 ± 0.002	1.335 ± 0.015	0.265 ± 0.004
CT-GP	0.935 ± 0.009	1.406 ± 0.008	0.227 ± 0.008
DT-GP	0.945 ± 0.005	1.410 ± 0.003	0.222 ± 0.008
DDT-GP	0.939 ± 0.003	1.411 ± 0.004	0.217 ± 0.003
DDT-CP	1.117 ± 0.011	1.580 ± 0.022	0.292 ± 0.007
DDT-TD	0.956 ± 0.005	1.473 ± 0.045	0.345 ± 0.096
BCTT	$\mathbf{0.891 \pm 0.003}$	$\mathbf{1.308 \pm 0.026}$	$\mathbf{0.198 \pm 0.006}$
MAE			
CT-CP	0.856 ± 0.003	$\mathbf{0.786 \pm 0.007}$	0.131 ± 0.001
CT-GP	0.703 ± 0.006	0.889 ± 0.009	0.094 ± 0.004
DT-GP	0.713 ± 0.003	0.880 ± 0.003	0.082 ± 0.002
DDT-GP	0.706 ± 0.005	0.874 ± 0.004	0.080 ± 0.001
DDT-CP	0.872 ± 0.006	1.024 ± 0.013	0.155 ± 0.005
DDT-TD	0.718 ± 0.004	0.923 ± 0.034	0.201 ± 0.053
BCTT	$\mathbf{0.678 \pm 0.002}$	0.787 ± 0.008	$\mathbf{0.079 \pm 0.002}$

(d) $R = 9$

Table 1. Prediction error and standard deviation. The results were averaged over five runs.

nation of factors in other modes interact with the factors in the current mode. To reflect the structure, we ran Principled Component Analysis (PCA), and show the first and second principled components in a plane. We also tested DDT-TD which learns a static tensor-core but using time factors and nonlinear dynamics. We looked at the results at mode 1. As shown in Fig. 4 a-c, we can see a clear structural variation. At $t = 1$, the tensor-core elements are quite concentrated, showing somewhat homogeneous interactions. The case is similar at $t = 4$ but the interaction strengths are more scattered. However, at $t = 7$, the strengths clearly formed four groups, exhibiting heterogeneous interaction patterns — a major shift. Together these imply the interaction between factors evolve with time. As a comparison, the tensor-core learned by DDT-TD do not reflect apparent structures or temporal patterns. It is inconvenient to examine how the interactions between the factors of the tensor nodes evolve.

7. Conclusion

We proposed BCTT, a continuous-time dynamic Tucker decomposition method. Our model maintains the interpretable structure while is flexible enough to capture various temporal dynamics within the factor interactions. Our LTI-SDE based message-passing inference avoids sparse GP approximations and enjoys a linear scalability with the data growth.

Acknowledgments

This work has been supported by MURI AFOSR grant FA9550-20-1-0358, NSF IIS-1910983, NSF CAREER Award IIS-2046295 and NSF DMS-1848508. We thank Shibo Li for implementing several baseline methods.

References

Ahn, D., Jang, J.-G., and Kang, U. (2021). Time-aware tensor decomposition for sparse tensors. Machine Learning, pages 1–22.

Bickel, P. J. and Doksum, K. A. (2015). Mathematical statistics: basic ideas and selected topics, volume I, volume 117. CRC Press.

Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

Choi, J. H. and Vishwanathan, S. (2014). Dfacto: Distributed factorization of tensors. In Advances in Neural Information Processing Systems, pages 1296–1304.

Chu, W. and Ghahramani, Z. (2009). Probabilistic models for incomplete multi-dimensional arrays. AISTATS.

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

Fang, S., Kirby, R. M., and Zhe, S. (2021a). Bayesian streaming sparse tucker decomposition. In Uncertainty in Artificial Intelligence, pages 558–567. PMLR.

Fang, S., Wang, Z., Pan, Z., Liu, J., and Zhe, S. (2021b). Streaming Bayesian deep tensor factorization. In International Conference on Machine Learning, pages 3133–3142. PMLR.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Model and conditions for an"explanatory"multimode factor analysis. UCLA Working Papers in Phonetics, 16:1–84.

Hartikainen, J. and Särkkä, S. (2010). Kalman filtering and smoothing solutions to temporal gaussian process regression models. In 2010 IEEE international workshop on machine learning for signal processing, pages 379–384. IEEE.

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. Biometrika, 58(1):83–90.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kang, U., Papalexakis, E., Harpale, A., and Faloutsos, C. (2012). Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 316–324. ACM.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kolda, T. G. (2006). Multilinear operators for higher-order decompositions, volume 2. United States. Department of Energy.

Lancaster, P. and Rodman, L. (1995). Algebraic riccati equations. Clarendon press.

Liu, B., He, L., Li, Y., Zhe, S., and Xu, Z. (2018). Neuralcp: Bayesian multiway data analysis with neural tensor decomposition. Cognitive Computation, 10(6):1051–1061.

Minka, T. P. (2001a). Expectation propagation for approximate Bayesian inference. In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, pages 362–369.

Minka, T. P. (2001b). A family of algorithms for approximate Bayesian inference. PhD thesis, Massachusetts Institute of Technology.

Oehlert, G. W. (1992). A note on the delta method. The American Statistician, 46(1):27–29.

Oksendal, B. (2013). Stochastic differential equations: an introduction with applications. Springer Science & Business Media.

Pan, Z., Wang, Z., and Zhe, S. (2020a). Scalable nonparametric factorization for high-order interaction events. In International Conference on Artificial Intelligence and Statistics, pages 4325–4335. PMLR.

Pan, Z., Wang, Z., and Zhe, S. (2020b). Streaming nonlinear bayesian tensor decomposition. In Conference on Uncertainty in Artificial Intelligence, pages 490–499. PMLR.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32:8026–8037.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. The Journal of Machine Learning Research, 6:1939–1959.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. AIAA journal, 3(8):1445–1450.

Rogers, M., Li, L., and Russell, S. J. (2013). Multilinear dynamical systems for tensor time series. Advances in Neural Information Processing Systems, 26:2634–2642.

Särkkä, S. (2013). Bayesian filtering and smoothing. Number 3. Cambridge University Press.

Särkkä, S. et al. (2006). Recursive Bayesian inference on stochastic differential equations. Helsinki University of Technology.

Schein, A., Paisley, J., Blei, D. M., and Wallach, H. (2015). Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1045–1054. ACM.

Schein, A., Zhou, M., Blei, D. M., and Wallach, H. (2016). Bayesian poisson tucker decomposition for learning the structure of international relations. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pages 2810–2819. JMLR.org.

Tillinghast, C., Wang, Z., and Zhe, S. (2022). Nonparametric sparse tensor factorization with hierarchical Gamma processes. In International Conference on Machine Learning. PMLR.

Tillinghast, C. and Zhe, S. (2021). Nonparametric decomposition of sparse tensors. In International Conference on Machine Learning, pages 10301–10311. PMLR.

Tucker, L. (1966). Some mathematical notes on three-mode factor analysis. Psychometrika, 31:279–311.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. Now Publishers Inc.

Wang, Z., Chu, X., and Zhe, S. (2020). Self-modulating nonparametric event-tensor factorization. In International Conference on Machine Learning, pages 9857–9867. PMLR.

Wang, Z. and Zhe, S. (2019). Conditional expectation propagation. In UAI, page 6.

Wolter, K. (2007). Introduction to variance estimation. Springer Science & Business Media.

Wu, X., Shi, B., Dong, Y., Huang, C., and Chawla, N. V. (2019). Neural tensor factorization for temporal interaction learning. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pages 537–545.

Xiong, L., Chen, X., Huang, T.-K., Schneider, J., and Carbonell, J. G. (2010). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In Proceedings of the 2010 SIAM International Conference on Data Mining, pages 211–222. SIAM.

Xu, Z., Yan, F., and Qi, Y. A. (2012). Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. In ICML.

Zhang, Y., Bi, X., Tang, N., and Qu, A. (2021). Dynamic tensor recommender systems. Journal of Machine Learning Research, 22(65):1–35.

Zhe, S. and Du, Y. (2018). Stochastic nonparametric event-tensor decomposition. In Advances in Neural Information Processing Systems, pages 6856–6866.

Zhe, S., Qi, Y., Park, Y., Xu, Z., Molloy, I., and Chari, S. (2016a). Dintucker: Scaling up Gaussian process models on large multidimensional arrays. In Thirtieth AAAI conference on artificial intelligence.

Zhe, S., Xu, Z., Chu, X., Qi, Y., and Park, Y. (2015). Scalable nonparametric multiway data analysis. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, pages 1125–1134.

Zhe, S., Zhang, K., Wang, P., Lee, K.-c., Xu, Z., Qi, Y., and Ghahramani, Z. (2016b). Distributed flexible nonlinear tensor factorization. In Advances in Neural Information Processing Systems, pages 928–936.