Nonparametric Factor Trajectory Learning for Dynamic Tensor Decomposition

Zheng Wang [1] **Shandian Zhe** [1]

Abstract

Tensor decomposition is a fundamental framework to analyze data that can be represented by multi-dimensional arrays. In practice, tensor data is often accompanied with temporal information, namely the time points when the entry values were generated. This information implies abundant, complex temporal variation patterns. However, current methods always assume the factor representations of the entities in each tensor mode are static, and never consider their temporal evolution. To fill this gap, we propose NONparametric FActor Trajectory learning for dynamic tensor decomposition (NONFAT). We place Gaussian process (GP) priors in the frequency domain and conduct inverse Fourier transform via Gauss-Laguerre quadrature to sample the trajectory functions. In this way, we can overcome data sparsity and obtain robust trajectory estimates across long time horizons. Given the trajectory values at specific time points, we use a second-level GP to sample the entry values and to capture the temporal relationship between the entities. For efficient and scalable inference, we leverage the matrix Gaussian structure in the model, introduce a matrix Gaussian posterior, and develop a nested sparse variational learning algorithm. We have shown the advantage of our method in several real-world applications.

1. Introduction

Data involving interactions between multiple entities can often be represented by multidimensional arrays, *i.e.,* tensors, and are ubiquitous in practical applications. For example, a four-mode tensor *(user, advertisement, web-page, device)* can be extracted from logs of an online advertising system, and three-mode *(patient, doctor, drug*) tensor from medical databases. As a powerful approach for tensor data analysis, tensor decomposition estimates a set of latent factors to represent the entities in each mode, and use these factors to reconstruct the observed entry values and to predict missing values. These factors can be further used to explore hidden structures from the data, *e.g.,* via clustering analysis, and provide useful features for important applications, such as personalized recommendation, click-through-rate prediction, and disease diagnosis and treatment.

In practice, tensor data is often accompanied with valuable temporal information, namely the time points at which each interaction took place to generate the entry value. These time points signify that underlying the data can be rich, complex temporal variation patterns. To leverage the temporal information, existing tensor decomposition methods usually introduce a time mode (Xiong et al., 2010; Rogers et al., 2013; Zhe et al., 2016a; 2015; Du et al., 2018) and arrange the entries into different time steps, *e.g.,* hours or days. They estimate latent factors for time steps, and may further model the dynamics between the time factors to better capture the temporal dependencies (Xiong et al., 2010). Recently, Zhang et al. (2021) introduced time-varying coefficients in the CANDECOMP/PARAFAC (CP) framework (Harshman, 1970) to conduct continuous-time decomposition. While successful, current methods always assume the factors of entities are static and invariant. However, along with the time, these factors, which reflect the entities' hidden properties, can evolve as well, such as user preferences, commodity popularity and patient health status. Existing approaches are not able to capture such variations and therefore can miss important temporal patterns.

To overcome this limitation, we propose NONFAT, a novel nonparametric dynamic tensor decomposition model to estimate time-varying factors. Our model is robust, flexible enough to learn various complex trajectories from sparse, noisy data, and capture nonlinear temporal relationships of the entities to predict the entry values. Specifically, we use Gaussian processes (GPs) to sample frequency functions in the frequency domain, and then generate the factor trajectories via inverse Fourier transform. Due to the nice properties of Fourier bases, we can robustly estimate the factor trajectories across long-term time horizons, even under sparse and noisy data. We use Gauss-Laguerre quadrature to efficiently compute the inverse Fourier transform.

[1]School of Computing, University of Utah. Correspondence to: Shandian Zhe <zhe@cs.utah.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Next, we use a second-level GP to sample entry values at different time points as a function of the corresponding factors values. In this way, we can estimate the complex temporal relationships between the entities. For efficient and scalable inference, we use the sparse variational GP framework (Hensman et al., 2013a) and introduce pseudo inputs and outputs for both levels of GPs. We observe a matrix Gaussian structure in the prior, based on which we can avoid introducing pseudo frequencies, reduce the dimension of pseudo inputs, and hence improve the inference quality and efficiency. We then employ matrix Gaussian posteriors to obtain a tractable variational evidence bound. Finally, we use a nested reparameterization procedure to implement a stochastic mini-batch variational learning algorithm.

We evaluated our method in three real-world applications. We compared with the state-of-the-art tensor decomposition methods that incorporate both continuous and discretized time information. In most cases, NONFAT outperforms the competing methods, often by a large margin. NONFAT also achieves much better test log-likelihood, showing superior posterior inference results. We showcase the learned factor trajectories, which exhibit interesting temporal patterns and extrapolate well to the non-training region. The entry value prediction by NONFAT also shows a more reasonable uncertainty estimate in both interpolation and extrapolation.

2. Preliminaries

2.1. Tensor Decomposition

In general, we denote a K-mode tensor or multidimensional array by $\mathcal{M} \in \mathbb{R}^{d_1 \times \ldots \times d_K}$. Each mode consists of d_k entities indexed by $1, \ldots, d_k$. We then denote each tensor entry by $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_K)$, where each element is the entity index in the corresponding mode. The value of the tensor entry, denoted by $m_{\mathbf{i}}$, is the result of interaction between the corresponding entities. For example, given a three-mode tensor *(customer, product, online-store)*, the entry values might be the purchase amount or payment. Given a set of observed entries, tensor decomposition aims to estimate a set of latent factors to represent the entities in each mode. Denote by $\mathbf{u}_j^k$ the factors of entity j in mode k. These factors can reflect hidden properties of the entities, such as customer interest and preference. We denote the collection of the factors in mode k by $\mathbf{U}^k = [\mathbf{u}_1^k, \ldots, \mathbf{u}_{d_k}^k]^\top$, and by $\mathcal{U} = \{\mathbf{U}^1, \ldots, \mathbf{U}^K\}$ all the factors for the tensor.

To learn these factors, a tensor decomposition model is used to fit the observed data. For example, Tucker decomposition (Tucker, 1966) assumes that $\mathcal{M} = \mathcal{W} \times_1 \mathbf{U}^1 \times_2 \ldots \times_K \mathbf{U}^K$, where $\mathcal{W} \in \mathbb{R}^{r_1 \times \ldots \times r_K}$ is called tensor core, and $\times_k$ is the tensor-matrix product at mode k (Kolda, 2006). If we restrict $\mathcal{W}$ to be diagonal, it becomes the popular CANDECOMP/PARAFAC (CP) decomposition (Harshman, 1970), where each entry value is decomposed as

$$m_{\boldsymbol{\ell}} = (\mathbf{u}_{\ell_1}^1 \circ \ldots \circ \mathbf{u}_{\ell_K}^K)^\top \boldsymbol{\lambda}, \tag{1}$$

where $\circ$ is the element-wise multiplication and $\boldsymbol{\lambda}$ corresponds to the diagonal elements of $\mathcal{W}$. While CP and Tucker are popular and elegant, they assume a multilinear interaction between the entities. In order to flexibly estimate various interactive relationships (*e.g.,* from simple linear to highly nonlinear), (Xu et al., 2012; Zhe et al., 2015; 2016a) view the entry value $m_{\boldsymbol{\ell}}$ as an unknown function of the latent factors and assign a Gaussian process (GP) prior to jointly learn the function and factors from data,

$$m_{\boldsymbol{\ell}} = f(\mathbf{u}_{\ell_1}^1, \ldots, \mathbf{u}_{\ell_K}^K), \quad f \sim \mathcal{GP}\left(0, \kappa(\mathbf{v}_{\boldsymbol{\ell}}, \mathbf{v}_{\boldsymbol{\ell}'})\right), \tag{2}$$

where $\mathbf{v}_{\boldsymbol{\ell}} = [\mathbf{u}_{\ell_1}^1; \ldots; \mathbf{u}_{\ell_K}^K]$, $\mathbf{v}_{\boldsymbol{\ell}'} = [\mathbf{u}_{\ell_1'}^1; \ldots; \mathbf{u}_{\ell_K'}^K]$ are the factors associated with entry $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$, respectively, *i.e.,* inputs to the function f, and $\kappa(\cdot, \cdot)$ is the covariance (kernel) function that characterizes the correlation between function values. For example, a commonly used one is the square exponential (SE) kernel, $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\eta})$, where η is the kernel parameter. Thus, any finite set of entry values $\mathbf{m} = [m_{\mathbf{i}_1}, \ldots, m_{\mathbf{i}_N}]$, which is a finite projection of the GP, follows a multivariate Gaussian prior distribution, $p(\mathbf{m}) = \mathcal{N}(\mathbf{m}|\mathbf{0}, \mathbf{K})$ where $\mathbf{K}$ is an $N \times N$ kernel matrix, $[\mathbf{K}]_{n,n'} = \kappa(\mathbf{v}_{\boldsymbol{\ell}_n}, \mathbf{v}_{\boldsymbol{\ell}_{n'}})$. To fit the observations $\mathbf{y}$, we can use a noise model $p(\mathbf{y}|\mathbf{m})$, *e.g.,* a Gaussian noisy model for continuous observations. Given the learned $\mathbf{m}$, to predict the value of a new entry $\boldsymbol{\ell}^*$, we can use conditional Gaussian,

$$p(m_{\boldsymbol{\ell}^*}|\mathbf{m}) = \mathcal{N}(m_{\boldsymbol{\ell}^*}|\mu^*, \nu^*) \tag{3}$$

where $\mu^* = \kappa(\mathbf{v}_{\ell^*}, \mathbf{V})\kappa(\mathbf{V}, \mathbf{V})^{-1}\mathbf{m}$, $\nu^* = \kappa(\mathbf{v}_{\ell^*}, \mathbf{v}_{\ell^*}) - \kappa(\mathbf{v}_{\ell^*}, \mathbf{V})\kappa(\mathbf{V}, \mathbf{V})^{-1}\kappa(\mathbf{V}, \mathbf{v}_{\ell^*})$, and $\mathbf{V} = [\mathbf{v}_{\boldsymbol{\ell}_1}, \ldots, \mathbf{v}_{\boldsymbol{\ell}_N}]^\top$, because $[\mathbf{m}; m_{\boldsymbol{\ell}^*}]$ also follows a multi-variate Gaussian distribution.

In practice, tensor data is often along with time information, *i.e.,* the time point at which each interaction occurred to generate the observed entry value. To exploit this information, current methods partition the time domain into steps $1, 2, \ldots, T$ according to a given interval, *e.g.,* one week or month. The observed entries are then binned into the T steps. In this way, a time mode is appended to the original tensor (Xiong et al., 2010; Rogers et al., 2013; Zhe et al., 2016a; 2015; Du et al., 2018), $\widehat{\mathcal{M}} \in \mathbb{R}^{d_1 \times \ldots \times d_K \times T}$. Then any tensor decomposition method can therefore be applied to estimate the latent factors for both the entities and time steps. To better grasp temporal dependencies, a dynamic model can be used to model the transition between the time factors. For example, (Xiong et al., 2010) placed a conditional prior over successive steps, $p(\mathbf{t}_{j+1}|\mathbf{t}_j) = N(\mathbf{t}_{j+1}|\mathbf{t}_j, \sigma^2\mathbf{I})$ where $\mathbf{t}_j$ are the latent factors for time step j. To leverage continuous time information, the most recent work (Zhang

et al., 2021) uses polynomial splines to model the coefficients $\boldsymbol{\lambda}$ in CP decomposition (see (1)) as a time-varying (trend) function.

2.2. Fourier Transform

Fourier transform (FT) is a mathematical transform that reveals the connection between functions in the time and frequency domains. In general, for any (complex) integrable function $f(t)$ in the time domain, we can find a corresponding function $\widehat{f}(\omega)$ in the frequency domain such that

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \widehat{f}(\omega) e^{i\omega t} \mathrm{d}\omega, \tag{4}$$

where $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$, and i indicates the imaginary part. The frequency function can be obtained via a convolution in the time domain,

$$\widehat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \mathrm{d}t. \tag{5}$$

The two functions $(f(t), \widehat{f}(\omega))$ is called a Fourier pair. Using the time function $f(t)$ to compute the frequency function $\widehat{f}(\omega)$, *i.e.,* (5), is called forward transform, while using $\widehat{f}(\omega)$ to recover $f(t)$, *i.e.,* (4), is called inverse transform.

3. Model

Although the existing tensor decomposition methods are useful and successful, they always assume the factors of the entities are static and invariant, even when the time information is incorporated into the decomposition. However, due to the complexity and diversity of real-world applications, those factors, which reflect the underlying properties of the entities, can evolve over time as well, *e.g.,* customer interest and preference, product quality, and song popularity. Hence, assuming fixed factors can miss important temporal variation patterns and hinder knowledge discovery and/or downstream predictive tasks. To overcome this issue, we propose NONFAT, a novel Bayesian nonparametric factor trajectory model for dynamic tensor decomposition.

Specifically, given the observed tensor entries and their timestamps, $\mathcal{D} = \{(\boldsymbol{\ell}_1, y_1, t_1), \ldots, (\boldsymbol{\ell}_N, y_N, t_N)\}$, we want to learn R factor trajectories (*i.e.,* time functions) for each entity j in mode k,

$$\mathbf{u}_j^k(t) = [u_{j,1}^k(t), \ldots, u_{j,R}^k(t)] \in \mathbb{R}^R.$$

To flexibly estimate these trajectories, an intuitive idea is to use GPs to model each $u_{j,r}^k(t)$, similar to that Xu et al. (2012); Zhe et al. (2016a) used GPs to estimate the decomposition function (see (2)). However, this modeling can be restrictive. When the time point is distant from the training time points, the corresponding covariance (kernel) function decreases very fast (consider the SE kernel for an example, $\kappa(t,t') = \exp(-\frac{1}{\eta}(t-t')^2)$). As a result, the GP estimate of the trajectory value — which is essentially an interpolation based on the training points (see (3)) — tends to be close to zero (the prior mean) and the predictive variance becomes large. That means, the GP estimate is neither accurate nor reliable. However, real-world tensor data are often very sparse and noisy — most entries only have observations at a few time points. Therefore, learning a GP trajectory outright on the time domain might not be robust and reliable enough, especially for many places that are far from the scarce training timestamps.

To address this issue, we find that from the Fourier transform view, any time function can be represented (or decomposed) by a series of Fourier bases $\{e^{i\omega t} | \omega \in \mathbb{R}\}$; see (4) and (5). These bases are trigonometric and have nice properties. We never need to worry that their values will fade to zero (or other constant) when t is large or getting away from the training timestamps. If we can obtain a reasonable estimate of their coefficients, *i.e.,* $\widehat{f}(\omega)$, we can use these bases to recover the time function in a much more reliable way.

Therefore, we turn to learning the factor trajectories from the frequency domain. Specifically, for each entity j of mode k, we learn a frequency function $\widehat{u}_{j,r}^k(\omega)$ $(1 \le r \le R)$, so that we can obtain the time trajectory via inverse Fourier transform (IFT),

$$u_{j,r}^k(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \widehat{u}_{j,r}^k(\omega) e^{i\omega t} \mathrm{d}\omega. \tag{6}$$

To get rid of the imaginary part, we require that $\widehat{u}_{j,r}^k(\omega)$ is symmetric, *i.e.,* $\widehat{u}_{j,r}^k(\omega) = \widehat{u}_{j,r}^k(-\omega)$, and therefore we have

$$u_{j,r}^k(t) = \frac{1}{\pi}\int_{0}^{\infty} \widehat{u}_{j,r}^k(\omega) \cos(\omega t) \mathrm{d}\omega. \tag{7}$$

However, even if the frequency function is given, the integral in (7) is in general analytically intractable. To overcome this issue, we use Gauss-Laguerre (GL) quadrature, which can solve the integral of the kind $\int_0^\infty e^{-x} g(x) \mathrm{d}x$ quite accurately. We write down (7) as

$$u_{j,r}^k(t) = \frac{1}{\pi}\int_{0}^{\infty} e^{-\omega} \left[\widehat{u}_{j,r}^k(\omega) e^{\omega}\right] \cos(\omega t) \mathrm{d}\omega, \tag{8}$$

and then use GPs to learn $\alpha_{j,r}^k(\omega) = \widehat{u}_{j,r}^k(\omega) e^{\omega}$. In doing so, not only do we still enjoy the flexibility of nonparametric estimation, we can also conveniently apply GL quadrature, without the need for any additional integral transform,

$$u_{j,r}^k(t) \approx \frac{1}{\pi}\sum_{c=1}^{C} \alpha_{j,r}^k(\widehat{\omega}_c) \cos(\widehat{\omega}_c t) \cdot \gamma_c \tag{9}$$

where $\{\widehat{\omega}_c\}$ and $\{\gamma_c\}$ are C quadrature nodes and weights.

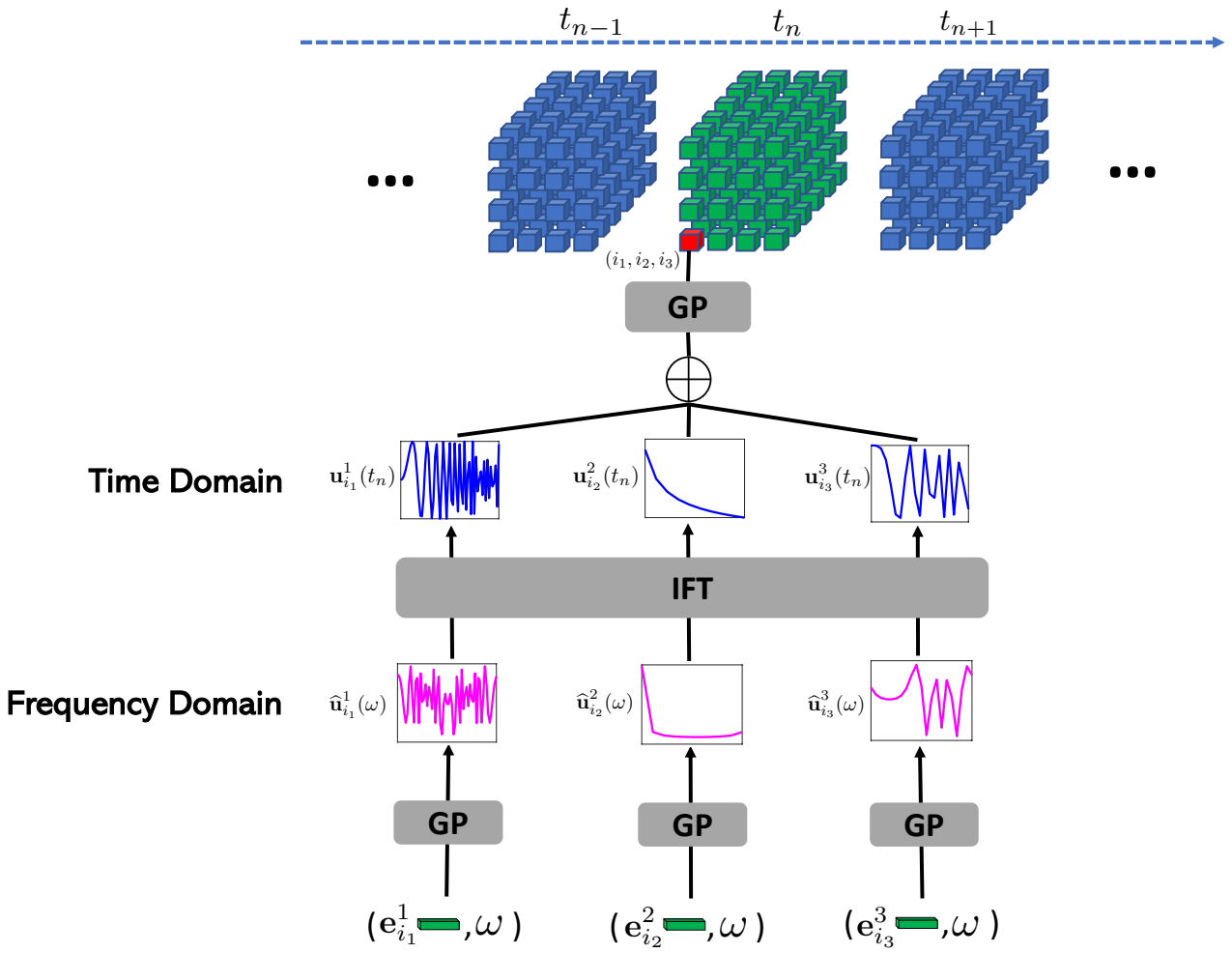


Figure 1. A graphical representation of our non-parametric factor trajectory learning model for dynamic tensor decomposition.

Next, we introduce a frequency embedding $\mathbf{e}^k_j \in \mathbb{R}^s$ for each entity j in mode k, and model $\alpha^k_{j,r}(\cdot)$ as a function of both the embedding and frequency,

$$\alpha^k_{j,r}(\omega) = f^k_r(\mathbf{e}^k_j, \omega). \tag{10}$$

The advantage of doing so is that we only need to estimate one function $f^k_r(\cdot,\cdot)$ to obtain the r-th frequency functions for all the entities in mode k. The frequency embeddings can also encode structural information within these entities (*e.g.,* groups and outliers). Otherwise, we have to estimate $d_k R$ functions in mode k, which can be quite costly and challenging, especially when d_k is large. We then apply a GP prior over f^k_r,

$$f^k_r(\mathbf{e}, \omega) \sim \mathcal{GP}\left(0, \kappa_r([\mathbf{e}; \omega], [\mathbf{e}'; \omega'])\right). \tag{11}$$

Given the factor trajectories, to obtain the value of each entry $m_{\boldsymbol{\ell}}$ at any time t, we use a second-level GP,

$$\begin{aligned} m_{\boldsymbol{\ell}}(t) &= g(\mathbf{u}^1_{\ell_1}(t), \ldots, \mathbf{u}^K_{\ell_K}(t)) \\ &\sim \mathcal{GP}\left(0, \kappa_g(\mathbf{v}_{\boldsymbol{\ell}}(t), \mathbf{v}_{\boldsymbol{\ell}'}(t))\right), \end{aligned} \tag{12}$$

where $\mathbf{v}_{\boldsymbol{\ell}}(t) = [\mathbf{u}^1_{\ell_1}(t); \ldots; \mathbf{u}^K_{\ell_K}(t)]$ and $\mathbf{v}_{\boldsymbol{\ell}'}(t) = [\mathbf{u}^1_{\ell'_1}(t); \ldots; \mathbf{u}^K_{\ell'_K}(t)]$. This is similar to (2). However, since the input consist of the values at (time-varying) trajectories, our second-level GP can flexibly estimate various temporal relationships between the entities. Finally, we sample the observed entry values from a Gaussian noisy model,

$$p(\mathbf{y}|\mathbf{m}) = \mathcal{N}(\mathbf{y}|\mathbf{m}, \sigma^2 \mathbf{I}), \tag{13}$$

where σ^2 is the noise variance, $\mathbf{y} = \{y_1, \ldots, y_N\}$ and $\mathbf{m} = \{m_{\boldsymbol{\ell}_1}(t_1), \ldots, m_{\boldsymbol{\ell}_N}(t_N)\}$. In this paper, we focus on continuous observations. However, our method can be easily adjusted to other types of observations. A graphical illustration of our model is given in Fig. 1.

4. Algorithm

The inference of our model is challenging. The GP prior over each $f_r^k(\cdot)$ (see (11)) demands we compute a multivariate Gaussian distribution of $\{f_r^k(\mathbf{e}_j^k, \widehat{\omega}_c)|1 \le j \le d_k, 1 \le c \le C\}$, and the GP over g (see (12) and (13)) a multivariate Gaussian distribution of $\{m_{\boldsymbol{\ell}_n}(t_n)|1 \le n \le N\}$. Hence, when the mode dimensions d_k and/or the number of observations N is large, the computation is very costly or even infeasible, not to mention the two-level GPs are tightly coupled (see (12)). To overcome the computational challenges, based on the variational sparse GP framework (Hensman et al., 2013b), we leverage our model structure to develop a nested stochastic mini-batch variational learning algorithm, presented as follows.

4.1. Sparse Variational ELBO Based on Matrix Gaussian Prior and Posterior

Specifically, given $\mathcal{D} = \{(\boldsymbol{\ell}_1, y_1, t_1), \ldots, (\boldsymbol{\ell}_N, y_N, t_N)\}$, the joint probability of our model is

$$p(\text{Joint}) = \prod_{k=1}^{K}\prod_{r=1}^{R} \mathcal{N}\left(\mathbf{f}_r^k|\mathbf{0}, \kappa_r(\mathbf{X}_f^k, \mathbf{X}_f^k)\right) \cdot \mathcal{N}\left(\mathbf{m}|\mathbf{0}, \kappa_g(\mathbf{X}_g, \mathbf{X}_g)\right)\mathcal{N}(\mathbf{y}|\mathbf{m}, \sigma^2\mathbf{I}) \quad (14)$$

where $\mathbf{f}_r^k$ is the concatenation of $\{f_r^k(\mathbf{e}_j^k, \widehat{\omega}_c)|1 \le j \le d_k, 1 \le c \le C\}$, $\mathbf{m} = [m_{\boldsymbol{\ell}_1}(t_1), \ldots, m_{\boldsymbol{\ell}_N}(t_N)]$, κ_r and κ_g are kernel functions, $\mathbf{X}_f^k$ are $d_kC \times (s+1)$ input matrix for $\mathbf{f}_r^k$, each row of $\mathbf{X}_f^k$ is an $(\mathbf{e}_j^k, \widehat{\omega}_c)$ pair, $\mathbf{X}_g = [\mathbf{v}_{\boldsymbol{\ell}_1}(t_1), \ldots, \mathbf{v}_{\boldsymbol{\ell}_N}(t_N)]^\top$ is the input matrix for $\mathbf{m}$, of size $N \times KR$. In our work, both κ_r and κ_r are chosen as SE kernels. Note that $\{\widehat{\omega}_c\}$ are the quadrature nodes.

First, we observe that for the first-level GP, the input matrix $\mathbf{X}_r^k$ is the cross combination of the d_k frequency embeddings $\mathbf{E}^k = [\mathbf{e}_1^k, \ldots, \mathbf{e}_{d_k}^k]^\top$ and C quadrature nodes $\widehat{\boldsymbol{\omega}} = [\widehat{\omega}_1; \ldots; \widehat{\omega}_C]$. Hence, we can rearrange $\mathbf{f}_r^k$ into a $d_k \times C$ matrix $\mathbf{F}_r^k$. Due to the multiplicative property of the kernel, *i.e.,* $\kappa_r([\mathbf{e}; \omega], [\mathbf{e}'; \omega']) = \kappa_r(\mathbf{e}, \mathbf{e}') \cdot \kappa_r(\omega, \omega')$, we observe $\mathbf{F}_r^k$ follows a matrix Gaussian prior distribution,

$$\begin{aligned} p(\mathbf{F}_r^k) &= N(\mathbf{f}_r^k|\mathbf{0}, \kappa_r(\mathbf{E}^k, \mathbf{E}^k) \otimes \kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})) \\ &= \mathcal{MN}(\mathbf{F}_r^k|\mathbf{0}, \kappa_r(\mathbf{E}^k, \mathbf{E}^k), \kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})) \quad (15) \\ &= \frac{\exp\left(-\frac{1}{2}\text{tr}\left(\kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})^{-1}\left(\mathbf{F}_r^k\right)^\top \kappa_r(\mathbf{E}^k, \mathbf{E}^k)^{-1}\mathbf{F}_r^k\right)\right)}{(2\pi)^{d_kC/2}|\kappa_r(\mathbf{E}^k, \mathbf{E}^k)|^{C/2}|\kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})|^{d_k/2}}. \end{aligned}$$

Therefore, we can compute the $d_k \times d_k$ row covariance matrix $\kappa_r(\mathbf{E}^k, \mathbf{E}^k)$ and $C \times C$ column covariance matrix $\kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})$ separately, rather than a giant full covariance matrix, *i.e.,* $\kappa_r(\mathbf{X}_f^k, \mathbf{X}_f^k)$ in (14) ($d_kC \times d_kC$). This can reduce the cost of the required covariance inverse operation from $\mathcal{O}\left((d_kC)^3\right)$ to $\mathcal{O}(C^3 + (d_k)^3)$ and hence save much cost. Nonetheless, when d_k is large, the row covariance matrix can still be infeasible to compute. Note that we do not need to worry about the column covariance, because the number of quadrature nodes C is small, *e.g.,* $C = 10$ is usually sufficient to give a high accuracy in numerical integration.

To address the challenge of computing the row covariance matrix, we introduce a_k pseudo inputs $\mathbf{Z}^k \in \mathbb{R}^{a_k \times s}$, where $a_k \ll d_k$. We consider the value of $f_r^k(\cdot)$ at the cross combination of $\mathbf{Z}_k$ and $\widehat{\boldsymbol{\omega}}$, which we refer to as the pseudo output matrix $\mathbf{G}_r^k$. Then $\{\mathbf{G}_r^k, \mathbf{F}_r^k\}$ follow another matrix Gaussian prior and we decompose the prior via

$$p(\mathbf{G}_r^k, \mathbf{F}_r^k) = p(\mathbf{G}_r^k)p(\mathbf{F}_r^k|\mathbf{G}_r^k), \quad (16)$$

where

$$\begin{aligned} p(\mathbf{G}_r^k) &= \mathcal{MN}\left(\mathbf{G}_r^k|\mathbf{0}, \kappa_r(\mathbf{Z}^k, \mathbf{Z}^k), \kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})\right), \\ p(\mathbf{F}_r^k|\mathbf{G}_r^k) &= \mathcal{MN}\left(\mathbf{F}_r^k|\boldsymbol{\Gamma}_r^k, \boldsymbol{\Omega}_r^k, \kappa_r(\widehat{\boldsymbol{\omega}}, \widehat{\boldsymbol{\omega}})\right), \quad (17) \end{aligned}$$

in which $\boldsymbol{\Gamma}_r^k = \kappa_r(\mathbf{E}^k, \mathbf{Z}^k)\kappa_r(\mathbf{Z}^k, \mathbf{Z}^k)^{-1}\mathbf{G}_r^k$, and $\boldsymbol{\Omega}_r^k = \kappa_r(\mathbf{E}^k, \mathbf{E}^k) - \kappa_r(\mathbf{E}^k, \mathbf{Z}^k)\kappa_r(\mathbf{Z}^k, \mathbf{Z}^k)^{-1}\kappa_r(\mathbf{Z}^k, \mathbf{E}^k)$.

Similarly, to address the computational challenge for the second-level GP, namely $\kappa_g(\mathbf{X}_g, \mathbf{X}_g)$ in (14), we introduce another set of a_g pseudo inputs $\mathbf{Z}_g \in \mathbb{R}^{a_g \times KR}$, where $a_g \ll N$. Denote by $\mathbf{h}$ the corresponding pseudo outputs — the outputs of $g(\cdot)$ function at $\mathbf{Z}_g$. We know $\{\mathbf{m}, \mathbf{g}\}$ follow a joint Gaussian prior, and we decompose their prior as well,

$$p(\mathbf{h}, \mathbf{m}) = p(\mathbf{h})p(\mathbf{m}|\mathbf{h}), \quad (18)$$

where $p(\mathbf{h}) = \mathcal{N}\left(\mathbf{h}|\mathbf{0}, \kappa_g(\mathbf{Z}_g, \mathbf{Z}_g)\right)$ and $p(\mathbf{m}|\mathbf{h}) = \mathcal{N}\big(\mathbf{m}|\kappa_g(\mathbf{X}_g, \mathbf{Z}_g)\kappa_g(\mathbf{Z}_g, \mathbf{Z}_g)^{-1}\mathbf{h}, \kappa_g(\mathbf{X}_g, \mathbf{X}_g) - \kappa_g(\mathbf{X}_g, \mathbf{Z}_g)\kappa_g(\mathbf{Z}_g, \mathbf{Z}_g)^{-1}\kappa_g(\mathbf{Z}_g, \mathbf{X}_g)\big)$.

Now, we augment our model with the pseudo outputs $\{\mathbf{G}_r^k\}$ and $\mathbf{h}$. According to (16) and (18), the joint probability becomes

$$p(\{\mathbf{G}_r^k, \mathbf{F}_r^k\}, \mathbf{h}, \mathbf{m}, \mathbf{y}) = \prod_{k=1}^{K}\prod_{r=1}^{R} p(\mathbf{G}_r^k)p(\mathbf{F}_r^k|\mathbf{G}_r^k) \cdot p(\mathbf{h})p(\mathbf{m}|\mathbf{h})\mathcal{N}(\mathbf{y}|\mathbf{m}, \sigma^2\mathbf{I}). \quad (19)$$

Note that if we marginalize out all the pseudo outputs, we recover the original distribution (14). To conduct tractable and scalable inference, we follow (Hensman et al., 2013b) to introduce a variational posterior of the following form,

$$\begin{aligned} &q(\{\mathbf{G}_r^k, \mathbf{F}_r^k\}, \mathbf{h}, \mathbf{m}) \\ &= \textstyle\prod_{k=1}^{K}\prod_{r=1}^{R} q(\mathbf{G}_r^k)p(\mathbf{F}_r^k|\mathbf{G}_r^k)q(\mathbf{h})p(\mathbf{m}|\mathbf{h}). \quad (20) \end{aligned}$$

We then construct a variational evidence lower bound (ELBO) (Wainwright and Jordan, 2008), $\mathcal{L} = \mathbb{E}_q\left[\log \frac{p(\{\mathbf{G}_r^k, \mathbf{F}_r^k\}, \mathbf{h}, \mathbf{m}, \mathbf{y})}{q(\{\mathbf{G}_r^k, \mathbf{F}_r^k\}, \mathbf{h}, \mathbf{m})}\right]$. Contrasting (19) and (20), we can see that all the conditional priors,

$p(\mathbf{F}_r^k|\mathbf{G}_r^k)$ and $p(\mathbf{m}|\mathbf{h})$, which are all giant Gaussian, have been canceled. Then we can obtain a tractable ELBO, which is additive over the observed entry values,

$$\mathcal{L} = -\sum_{k=1}^{K}\sum_{r=1}^{R}\mathrm{KL}\left(q(\mathbf{G}_r^k)\|p(\mathbf{G}_r^k)\right) - \mathrm{KL}\left(q(\mathbf{h})\|p(\mathbf{h})\right) + \sum_{n=1}^{N}\mathbb{E}_q\left[\log p(y_n|m_{\boldsymbol{\ell}_n}(t_n))\right], \quad (21)$$

where $\mathrm{KL}(\cdot\|\cdot)$ is the Kullback-Leibler (KL) divergence. We then introduce Gaussian posteriors for $q(\mathbf{h})$ and all $q(\mathbf{G}_r^k)$ so that the KL terms have closed forms. Since the number of pseudo inputs a_g and a_k are small (*e.g.,* 100), the cost is cheap. To further improve the efficiency, we use a matrix Gaussian posterior for each $\mathbf{G}_r^k$, namely

$$q(\mathbf{G}_r^k) = \mathcal{MN}(\mathbf{G}_r^k|\mathbf{A}_r^k, \mathbf{L}_r^k\left(\mathbf{L}_r^k\right)^\top, \mathbf{R}_r^k\left(\mathbf{R}_r^k\right)^\top), \quad (22)$$

where $\mathbf{L}_r^k$ and $\mathbf{R}_r^k$ are lower triangular matrices to ensure the row and column covariance matrices are positive semi-definite. Thereby, we do not need to compute the $a_kC \times a_kC$ full posterior covariance matrix.

Note that the matrix Gaussian (MG) view (15) not only can help us reduce the computation expense, but also improves the approximation. The standard sparse GP approximation requires us to place pseudo inputs in the entire input space. That is, the embeddings plus frequencies. Our MG view allows us to only introduce pseudo inputs in the embedding space (no need for pseudo frequencies). Hence it decreases approximation dimension and improves inference quality.

4.2. Nested Stochastic Mini-Batch Optimization

We maximize the ELBO (21) to estimate the variational posterior $q(\cdot)$, the frequency embeddings, kernel parameters, pseudo inputs $\mathbf{Z}^k$ and $\mathbf{Z}_g$ and other parameters. To scale to a large number of observations, each step we use a random mini-batch of data points to compute a stochastic gradient, which is based on an unbiased estimate of the ELBO,

$$\widehat{\mathcal{L}} = \text{KL-terms} + \frac{N}{B}\sum_{n\in\mathcal{B}}\mathbb{E}_q\left[\log p(y_n|m_{\boldsymbol{\ell}_n}(t_n))\right], \quad (23)$$

where $\mathcal{B}$ is the mini-batch, of size B. However, we cannot directly compute $\nabla\widehat{\mathcal{L}}$ because each expectation $\mathbb{E}_q\left[\log p(y_n|m_{\ell_n}(t_n))\right]$ is analytically intractable. To address this problem, we use a nested reparameterization procedure to compute an unbiased estimate of $\nabla\widehat{\mathcal{L}}$, which is also an unbiased estimate of $\nabla\mathcal{L}$, to update the model.

Specifically, we aim to obtain a parameterized sample of each $m_{\boldsymbol{\ell}_n}(t_n)$, plug it in the corresponding log likelihood and get rid of the expectation. Then we calculate the gradient. It guarantees to be an unbiased estimate of $\nabla\widehat{\mathcal{L}}$. To do so, following (18), we first draw a parameterized sample of the pseudo output $\mathbf{h}$ via its Gaussian posterior $q(\mathbf{h})$ (Kingma and Welling, 2013), and then apply the conditional Gaussian $p(m_{\boldsymbol{\ell}_n}(t_n)|\mathbf{h})$. We denote by $\widehat{\mathbf{h}}$ the sample of $\mathbf{h}$. However, the input to $\mathbf{m}_{\ell_n}(t_n)$ are values of the factor trajectories (see (12)), which are modeled by the GPs in the first level. Hence, we need to use the reparameterization trick again to generate a parameterized sample of the input $\mathbf{v}_{\boldsymbol{\ell}_n}(t_n) = [\mathbf{u}^1_{\ell_{n1}}(t_n); \ldots; \mathbf{u}^K_{\ell_{nK}}(t_n)]$. To do so, we generate a posterior sample of the pseudo outputs $\{\mathbf{G}_r^k\}$ at the first level. According to (22), we can draw a matrix Gaussian noise $\mathbf{S}_r^k \sim \mathcal{MN}(\mathbf{0}, \mathbf{I}, \mathbf{I})$ and obtain the sample by

$$\widehat{\mathbf{G}}_r^k = \mathbf{A}_r^k + \mathbf{L}_r^k\mathbf{S}_r^k(\mathbf{R}_r^k)^\top.$$

Then we use the conditional matrix Gaussian in (17) to sample $\boldsymbol{\alpha}_r^k \triangleq [f_r^k(\mathbf{e}^k_{\ell_{n_k}}, \widehat{\omega}_1), \ldots, f_r^k(\mathbf{e}^k_{\ell_{n_k}}, \widehat{\omega}_C)](1 \le k \le K)$. Since $\boldsymbol{\alpha}_r^k$ is actually a row of $\mathbf{F}_r^k$, the conditional matrix Gaussian is degenerated to an ordinary multivariate Gaussian and we generate the parameterized sample $\widehat{\boldsymbol{\alpha}}_r^k$ accordingly given $\mathbf{G}_r^k = \widehat{\mathbf{G}}_r^k$. Then we apply the GL quadrature (9) to obtain the sample of each trajectory value $\widehat{u}^k_{\ell_{n_k},r}(t_n)$, and the input to the second-level GP, $\widehat{\mathbf{v}}_{\boldsymbol{\ell}_n}(t_n) = [\widehat{\mathbf{u}}^1_{\ell_{n1}}(t_n); \ldots; \widehat{\mathbf{u}}^K_{\ell_{nK}}(t_n)]$. Now, with the sample of the pseudo output $\widehat{\mathbf{h}}$, we can apply $p(m_{\boldsymbol{\ell}_n}(t_n)|\mathbf{h} = \widehat{\mathbf{h}})$ to generate the sample of $\mathbf{m}_{\ell_n}(t_n)$. We can use any automatic differentiation library to track the computation of these samples, build computational graphs, and calculate the stochastic gradient.

4.3. Algorithm Complexity

The time complexity of our inference algorithm is $\mathcal{O}\left(RC^3 + KRa_k^3 + a_g^3 + B(KR(a_k^2 + C^2) + a_g^2)\right)$, in which B is the size of the mini-batch and the cubic terms arise from computing the KL divergence and inverse covariance (kernel) matrix on the pseudo inputs and quadrature nodes (across K modes and R trajectories). Hence, the cost is proportional to the mini-batch size. The space complexity is $\mathcal{O}(\sum_{k=1}^K\sum_{r=1}^R(a_k^2 + C^2 + a_kC) + a_g^2 + a_g + \sum_{k=1}^K d_ks)$, including the storage of the prior and variational posterior of the pseudo outputs and frequency embeddings.

5. Related Work

To exploit time information, existent tensor decomposition methods usually expand the tensor with an additional time mode, *e.g.,* (Xiong et al., 2010; Rogers et al., 2013; Zhe et al., 2016b; Ahn et al., 2021; Zhe et al., 2015; Du et al., 2018). This mode consists of a series of time steps, say by hours or weeks. A dynamic model is often used to model the transition between the time step factors. For example, Xiong et al. (2010) used a conditional Gaussian prior, Wu et al. (2019) used recurrent neural networks to model the time factor transition, and Ahn et al. (2021) proposed a kernel smoothing and regularization term. To deal with

continuous time information, the most recent work (Zhang et al., 2021) uses polynomial splines to model the CP coefficients ($\boldsymbol{\lambda}$ in (1)) as a time function. Another line of research focuses on the events of interactions between the entities, *e.g.,* Poisson tensor factorization (Schein et al., 2015; 2016), and those based on more expressive point processes (Zhe and Du, 2018; Pan et al., 2020a; Wang et al., 2020). While successful, these methods only model the event count or sequences. They do not consider the actual interaction results, like payment or purchase quantity in online shopping, and clicks/nonclicks in online advertising. Hence their problem setting is different. Despite the success of current tensor decomposition methods (Chu and Ghahramani, 2009; Choi and Vishwanathan, 2014; Zhe et al., 2016a; 2015; 2016b; Liu et al., 2018; Pan et al., 2020b; Tillinghast et al., 2020; Tillinghast and Zhe, 2021; Fang et al., 2021; Tillinghast et al., 2022), they all assume the factors of the entities are static and fixed, even when the time information is incorporated. To our knowledge, our method is the first to learn these factors as trajectory functions.

Our bi-level GP decomposition model can be viewed as an instance of deep Gaussian processes (DGPs) (Damianou and Lawrence, 2013). However, different from the standard DGP formulation, we conduct inverse Fourier transform over the output of the first-level GPs, before we feed them to the next-level GP. In so doing, we expect to better learn the factor trajectory functions in our problem. Our nested sparse variational inference is similar to (Salimbeni and Deisenroth, 2017), where in each GP level, we also introduce a set of pseudo inputs and outputs to create sparse approximations. However, we further take advantage of our model structure to convert the (finite) GP prior in the first level as a matrix Gaussian. The benefit is that we do not need to introduce pseudo inputs in the frequency domain and so we can reduce the approximation. We use a matrix Gaussian posterior to further accelerate the computation.

6. Experiment

6.1. Predictive Performance

We evaluated NONFAT in three real-world benchmark datasets. (1) *Beijing Air Quality*[1], hourly concentration measurements of 6 pollutants (*e.g.,* PM2.5, PM10 and SO2) in 12 air-quality monitoring sites across Beijing from year 2013 to 2017. We thereby extracted a 2-mode (pollutant, site) tensor, including 10K measurements and the time points for different tensor entries. (2) *Mobile Ads*[2], a 10-day click-through-rate dataset for mobile advertisements. We extracted a three-mode tensor *(banner-position, site domain, mobile app)*, of size $7 \times 2842 \times 4127$. The tensor includes 50K observed entry values (click number) at different time points. (3) *DBLP* [3], the bibliographic records in the domain of computer science. We downloaded the XML database, filtered the records from year 2011 to 2021, and extracted a three-mode tensor (author, conference, keyword) of size $3731 \times 1935 \times 169$ from the most prolific authors, most popular conferences and keywords. The observed entry values are the numbers of papers published at different years. We collected 50K entry values and their time points.

We compared with the following popular and/or state-of-the-art methods for dynamic tensor decomposition. (1) CPCT (Zhang et al., 2021), the most recent continuous-time decomposition algorithm, which uses polynomial splines to estimate the coefficients in the CP framework as a temporal function (see $\boldsymbol{\lambda}$ in (1)). (2) GPCT, continuous-time GP decomposition, which extends (Xu et al., 2012; Zhe et al., 2016b) by placing the time t in the GP kernel to learn the entry value as a function of both the latent factors and time t, *i.e.,* $m_{\boldsymbol{\ell}} = f(\mathbf{u}^1_{\ell_1}, \ldots, \mathbf{u}^K_{\ell_K}, t)$. (3) NNCT, continuous-time neural network decomposition, which is similar to (Liu et al., 2019) but uses time t as an additional input to the NN decomposition model. (4) GPDTL, discrete-time GP decomposition with linear dynamics, which expands the tensor with a time mode and jointly estimates the time factors and other factors with GP decomposition (Zhe et al., 2016b). In addition, we follow (Xiong et al., 2010) to introduce a conditional prior over consecutive steps, $p(\mathbf{t}_{j+1}|\mathbf{t}_j) = \mathcal{N}(\mathbf{t}_{j+1}|\mathbf{C}\mathbf{t}_j + \mathbf{b}, v\mathbf{I})$. Note this linear dynamic model is more general than the one in (Xiong et al., 2010) since the latter corresponds to $\mathbf{C} = \mathbf{I}$ and $\mathbf{b} = \mathbf{0}$. (5) GPDTN, discrete-time GP decomposition with nonlinear dynamics. It is similar to GPDTL except that the prior over the time factors becomes $p(\mathbf{t}_{j+1}|\mathbf{t}_j) = \mathcal{N}(\mathbf{t}_{j+1}|\sigma(\mathbf{C}\mathbf{t}_j)+\mathbf{b}, v\mathbf{I})$ where $\sigma(\cdot)$ is the nonlinear activation function. Hence, this can be viewed as an RNN-type transition. (6) NNDTL, discrete-time NN decomposition with linear dynamics, similar to GPDTL but using NN decomposition model. (7) NNDTN, discrete-time NN decomposition with nonlinear dynamics, which is similar to GPDTN that employs RNN dynamics over the time steps.

Experiment Setting. We implemented all the methods with PyTorch (Paszke et al., 2019). For all the GP baselines, we used SE kernel and followed (Zhe et al., 2016a) to use sparse variational GP framework (Hensman et al., 2013b) for scalable posterior inference, with 100 pseudo inputs. For NONFAT, the number of pseudo inputs for both levels of GPs was set to 100. For NN baselines, we used three-layer neural networks, with 50 neurons per layer and `tanh` as the activation. For nonlinear dynamic methods, including GPDTN and NNDTN, we used `tanh` as the acti-

[1] https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data

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

[3] https://dblp.uni-trier.de/xml/

Beijing Air Quality	$R = 2$	$R = 3$	$R = 5$	$R = 7$
NONFAT	$\mathbf{0.340 \pm 0.006}$	$\mathbf{0.315 \pm 0.001}$	$\mathbf{0.314 \pm 0.001}$	0.326 ± 0.006
CPCT	0.997 ± 0.001	0.997 ± 0.001	1.002 ± 0.002	1.002 ± 0.002
GPCT	0.372 ± 0.001	0.366 ± 0.001	0.363 ± 0.001	0.364 ± 0.001
NNCT	0.986 ± 0.002	0.988 ± 0.002	0.977 ± 0.012	0.987 ± 0.003
GPDTL	0.884 ± 0.001	0.884 ± 0.001	0.885 ± 0.001	0.884 ± 0.001
NNDTL	0.356 ± 0.003	0.358 ± 0.005	0.333 ± 0.003	$\mathbf{0.315 \pm 0.002}$
GPDTN	0.884 ± 0.001	0.884 ± 0.001	0.884 ± 0.001	0.884 ± 0.001
NNDTN	0.365 ± 0.005	0.337 ± 0.006	0.336 ± 0.003	0.319 ± 0.005
Mobile Ads				
NONFAT	0.652 ± 0.002	$\mathbf{0.635 \pm 0.003}$	$\mathbf{0.638 \pm 0.006}$	$\mathbf{0.637 \pm 0.005}$
CPCT	1.001 ± 0.004	0.986 ± 0.018	1.009 ± 0.009	0.971 ± 0.010
GPCT	0.660 ± 0.003	0.661 ± 0.003	0.662 ± 0.001	0.659 ± 0.003
NNCT	0.822 ± 0.001	0.822 ± 0.001	0.822 ± 0.001	0.822 ± 0.001
GPDTL	0.714 ± 0.006	0.695 ± 0.004	0.695 ± 0.004	0.695 ± 0.003
NNDTL	$\mathbf{0.646 \pm 0.003}$	0.646 ± 0.002	0.642 ± 0.003	0.640 ± 0.003
GPDTN	0.667 ± 0.003	0.661 ± 0.003	0.668 ± 0.003	0.669 ± 0.003
NNDTN	0.646 ± 0.004	0.645 ± 0.002	0.640 ± 0.003	0.638 ± 0.003
DBLP				
NONFAT	$\mathbf{0.188 \pm 0.003}$	$\mathbf{0.188 \pm 0.003}$	0.189 ± 0.003	$\mathbf{0.189 \pm 0.003}$
CPCT	1.004 ± 0.003	1.004 ± 0.002	1.005 ± 0.002	1.001 ± 0.004
GPCT	0.189 ± 0.003	0.191 ± 0.003	0.192 ± 0.003	0.196 ± 0.003
NNCT	0.188 ± 0.003	0.188 ± 0.003	$\mathbf{0.188 \pm 0.003}$	0.189 ± 0.003
GPDTL	0.208 ± 0.004	0.223 ± 0.003	0.221 ± 0.003	0.224 ± 0.003
NNDTL	0.188 ± 0.003	0.188 ± 0.003	0.189 ± 0.003	0.189 ± 0.003
GPDTN	0.206 ± 0.002	0.218 ± 0.003	0.224 ± 0.003	0.225 ± 0.002
NNDTN	0.188 ± 0.003	0.188 ± 0.003	0.188 ± 0.003	0.189 ± 0.003

Table 1. Root Mean-Square Error (RMSE). The results were averaged over five runs.

vation. For CPCT, we used 100 knots for polynomial splines. For discrete-time methods, we used 50 steps. We did not find improvement with more steps. All the models were trained with stochastic mini-batch optimization. We used ADAM (Kingma and Ba, 2014) for all the methods, and the mini-batch size is 100. The learning rate was chosen from $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$. To ensure convergence, we ran each methods for 10K epochs. To get rid of the fluctuation of the error due to the stochastic model updates, we computed the test error after each epoch and used the smallest one as the result. We varied the number of latent factors R from {2, 3, 5, 7}. For NONFAT, R is the number of factor trajectories. We followed the standard testing procedure as in (Xu et al., 2012; Kang et al., 2012; Zhe et al., 2016b) to randomly sample 80% observed tensor entries (and their timestamps) for training and tested the prediction accuracy on the remaining entries. We repeated the experiments for five times and calculated the average root mean-square-error (RMSE) and its standard deviation.

Results. As we can see from Table 1, in most cases, our methods NONFAT outperforms all the competing approaches, often by a large margin. In addition, NONFAT always achieves better prediction accuracy than GP methods, including GPCT, GPDTL and GPDTN. In most cases, the improvement is significant ($p < 0.05$). It shows that our bi-level GP decomposition model can indeed improves upon the single level GP models. Only in a few cases, the prediction error of NONFAT is slightly worse an NN approach. Note that, NONFAT learns a trajectory for each factor, and it is much more challenging than learning fixed-value factors, as done by the competing approaches.

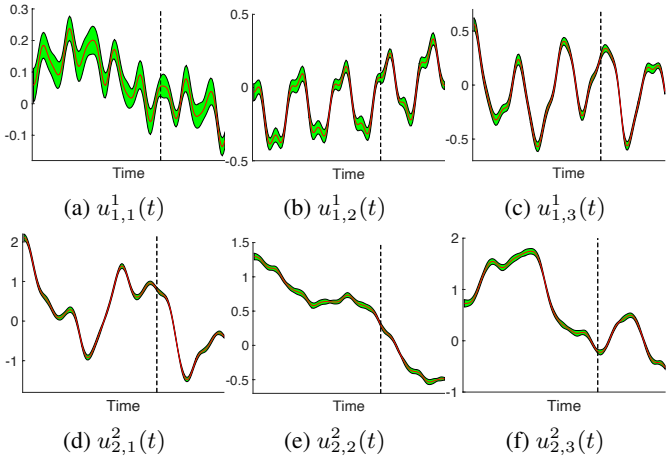
Furthermore, we examined our method in a probabilistic sense. We reported the test log-likelihood of NONFAT and the other probabilistic approaches, including GPCT, GPDTL and GPDTN, in Table 2. We can see that NONFAT largely improved upon the competing methods in all the cases, showing that NONFAT is advantageous not only in prediction accuracy but also in uncertainty quantification, which can be important in many decision tasks, especially with sparse and noisy data.

6.2. Investigation of Learning Result

Next, we investigated if our learned factor trajectories exhibit patterns and how they influence the prediction. To this end, we set $R = 3$ and ran NONFAT on *Beijing Air Quality* dataset. We show the learned factor trajectories for the first monitoring site in mode 1 in Fig. 2 a-c, and second pollutant (SO2) in mode 2 in Fig. 2 d-f. As we can see, they show different patterns. First, it is interesting to see that all the trajectories for the site exhibit periodicity, but with different

Beijing Air Quality	$R = 2$	$R = 3$	$R = 5$	$R = 7$
NONFAT	$\mathbf{-0.343 \pm 0.018}$	$\mathbf{-0.264 \pm 0.003}$	$\mathbf{-0.260 \pm 0.004}$	$\mathbf{-0.297 \pm 0.017}$
GPCT	-0.420 ± 0.001	-0.406 ± 0.001	-0.401 ± 0.001	-0.401 ± 0.001
GPDTL	-1.299 ± 0.001	-1.299 ± 0.001	-1.299 ± 0.001	-1.299 ± 0.001
GPDTN	-1.299 ± 0.001	-1.299 ± 0.001	-1.299 ± 0.001	-1.299 ± 0.001
Mobile Ads				
NONFAT	$\mathbf{-0.726 \pm 0.004}$	$\mathbf{-0.705 \pm 0.003}$	$\mathbf{-0.709 \pm 0.007}$	$\mathbf{-0.706 \pm 0.008}$
GPCT	-0.733 ± 0.002	-0.737 ± 0.005	-0.734 ± 0.004	-0.735 ± 0.004
GPDTL	-1.843 ± 0.009	-1.807 ± 0.006	-1.822 ± 0.008	-1.830 ± 0.003
GPDTN	-0.774 ± 0.003	-0.762 ± 0.004	-0.804 ± 0.006	-0.806 ± 0.003
DBLP				
NONFAT	$\mathbf{0.201 \pm 0.019}$	$\mathbf{0.201 \pm 0.019}$	$\mathbf{0.199 \pm 0.017}$	$\mathbf{0.199 \pm 0.017}$
GPCT	0.129 ± 0.009	0.105 ± 0.009	0.104 ± 0.011	0.087 ± 0.013
GPDTL	0.102 ± 0.023	0.004 ± 0.025	0.035 ± 0.019	0.022 ± 0.019
GPDTN	0.114 ± 0.012	0.041 ± 0.019	0.019 ± 0.020	0.013 ± 0.015

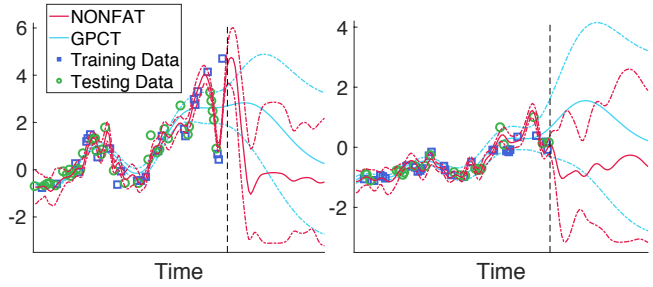
Table 2. Test log-likelihood. The results were averaged from five runs.



(a) $u^1_{1,1}(t)$ (b) $u^1_{1,2}(t)$ (c) $u^1_{1,3}(t)$

(d) $u^2_{2,1}(t)$ (e) $u^2_{2,2}(t)$ (f) $u^2_{2,3}(t)$

Figure 2. The learned factor trajectories.



(a) (6, 6) (b) (7,4)

Figure 3. Entry value prediction.

perturbation, amplitude, period, *etc.* This might relate to the working cycles of the sensors in the site. The trajectories for the pollutant is much less periodic and varies quite differently. For example, $u^2_{2,1}(t)$ decreases first and then increases, while $u^2_{2,3}(t)$ increases first and then decreases, and $u^2_{2,2}(t)$ keeps the decreasing trend. They represent different time-varying components of the pollutant concentration. Second, the vertical dashed line is the boundary of the training region. We can see that all the trajectories extrapolate well. Their posterior mean and standard deviation are stable outside of the training region (as stable as inside the training region). It demonstrates our learning from the frequency domain can yield robust trajectory estimates.

Finally, we showcase the prediction curves of two entries in Fig. 3. The prediction made by our factor trajectories can better predict the test points, as compared with GPCT. More important, outside the training region, our predictive uncertainty is much smaller than GPCT (right to the dashed vertical line), while inside the training region, our predictive uncertainty is not so small as GPCT that is close to zero. This shows NONFAT gives more reasonable uncertainty quantification in both interpolation and extrapolation.

7. Conclusion

We have presented NONFAT, a novel nonparametric Bayesian method to learn factor trajectories for dynamic tensor decomposition. The predictive accuracy of NONFAT in real-world applications is encouraging and the learned trajectories show interesting temporal patterns. In the future, we will investigate the meaning of these patterns more in-depth and apply our approach in more applications.

Acknowledgments

This work has been supported by NSF IIS-1910983 and NSF CAREER Award IIS-2046295.

References

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

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.

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In Artificial Intelligence and Statistics, pages 207–215.

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., Wang, Z., Pan, Z., Liu, J., and Zhe, S. (2021). 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.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013a). Gaussian processes for big data. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI).

Hensman, J., Fusi, N., and Lawrence, N. D. (2013b). Gaussian processes for big data. In Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, pages 282–290. AUAI Press.

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. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

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

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.

Liu, H., Li, Y., Tsang, M., and Liu, Y. (2019). CoSTCo: A Neural Tensor Completion Model for Sparse Tensors, page 324–334. Association for Computing Machinery, New York, NY, USA.

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.

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.

Salimbeni, H. and Deisenroth, M. P. (2017). Doubly stochastic variational inference for deep gaussian processes. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 4591–4602.

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., Fang, S., Zhang, K., and Zhe, S. (2020). Probabilistic neural-kernel tensor decomposition. In 2020 IEEE International Conference on Data Mining (ICDM), pages 531–540. IEEE.

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.

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.