Infinite Tucker Decomposition: Nonparametric Bayesian Models for Multiway Data Analysis

Zenglin Xu, Feng Yan {XU218,YAN12}@PURDUE.EDU
Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA

Yuan Qi ALANQI@CS.PURDUE.EDU
Departments of Computer Science and Statistics, Purdue University, West Lafayette, IN 47907 USA

Abstract

Tensor decomposition is a powerful computational tool for multiway data analysis. Many popular tensor decomposition approaches—such as the Tucker decomposition and CANDECOMP/PARAFAC (CP)—amount to multi-linear factorization. They are insufficient to model (i) complex interactions between data entities, (ii) various data types (*e.g.*missing data and binary data), and (iii) noisy observations and outliers. To address these issues, we propose tensor-variate latent nonparametric Bayesian models, coupled with efficient inference methods, for multiway data analysis. We name these models *InfTucker*. Using these *InfTucker* models, we conduct Tucker decomposition in an infinite feature space. Unlike classical tensor decomposition models, our new approaches handle both continuous and binary data in a probabilistic framework. Unlike previous Bayesian models on matrices and tensors, our models are based on latent Gaussian or t processes with nonlinear covariance functions. To efficiently learn the *InfTucker* models from data, we develop a variational inference technique on tensors. Compared with classical implementation, the new technique reduces both time and space complexities by several orders of magnitude. Our experimental results on chemometrics and social network datasets demonstrate that our new models achieved significantly higher prediction accuracy than the most state-of-art tensor decomposition approaches.

Appearing in *Proceedings of the 29^{th} International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

1 Introduction

Many real-world datasets with multiple aspects can be described by tensors (i.e., multiway arrays). For example, patient drug responses can be represented by a tensor with four modes (*person, medicine, biomarker, time*) and user customer ratings by a tensor with three modes (*user, item, time*). Given tensor-valued data, we want to model complex interactions embedded in data (e.g., drug interactions) and predict missing elements (e.g., unknown drug responses). Traditional multiway factor models—such as the Tucker decomposition (Tucker, 1966) and CANDECOMP/PARAFAC (CP) (Harshman, 1970)—have been applied for various applications (*e.g.*, computer vision (Shashua & Hazan, 2005) and chemometrics (Acar et al., 2011) etc). These models, however, face serious challenges for modeling complex multiway interactions. First, interactions between entities in each mode may be coupled together and highly nonlinear. The classical multi-linear models cannot capture these intricate relationships. Second, the data are often noisy, but the classical models are not designed to deal with noisy observations. Third, the data may contain many missing values. We need to first impute the missing values before we can apply the classical multiway factor models. Forth, the data may not be restricted to real values: they can be binary as in dynamic network data or have ordinal values for user-movie-ratings. But the classical models simply treat them as continuous data — this treatment would lead to degenerated predictive performance.

To address these challenges we propose a nonparametric Bayesian multiway analysis model, *InfTucker*. Based on latent Gaussian processes or t processes, it conducts the Tucker decomposition in an infinite dimensional feature space. It generalizes the elegant work of Chu & Ghahramani (2009) by capturing nonlinear interactions between different tensor modes. Grounded in a probabilistic framework, it naturally handles noisy observations and missing data. Furthermore, it handles various data types—binary

or continuous—by simply using suitable data likelihoods. Although *InfTucker* offers an elegant solution to multiway analysis, learning the model from data is computationally challenging. To overcome this challenge, we develop an efficient variational Bayesian approach that explores tensor structures to significantly reduce the computational cost. This efficient inference technique also enables the usage of nonlinear covariance functions for latent Gaussian and t processes on datasets with reasonably large sizes.

Our experimental results on chemometrics and social network datasets demonstrate that the *InfTucker* achieves significantly higher prediction accuracy than state-of-the-art tensor decomposition approaches—including High Order Singular Value Decomposition (HOSVD) (Lathauwer et al., 2000), Weighted CP (Acar et al., 2011) and nonnegative tensor decomposition (Shashua & Hazan, 2005).

2 Tensor Decomposition

A K-mode tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times \ldots \times n_K}$ has K modes, where the k-th mode has n_k dimensions. We use $y_{\mathbf{i}}$ to denote the $\mathbf{i} = (i_1, \ldots, i_K)$ entry of $\mathcal{Y}$.

In order to define tensor decomposition, two linear algebra operations on matrix are generalized to tensor—vectorization and tensor-matrix multiplication (Kolda & Bader, 2009). We define the vectorization operation, denoted by $\text{vec}(\mathcal{Y})$, to stack the tensor entries into a $n = \prod_{k=1}^{K} n_k$ by 1 vector. The entry $\mathbf{i} = (i_1, \ldots, i_K)$ of $\mathcal{Y}$ is mapped to the entry at position $j = i_K + \sum_{i=1}^{K-1}(i_k - 1)\prod_{k+1}^{K} n_k$ of $\text{vec}(\mathcal{Y})$[1].

The mode-k tensor-matrix multiplication of a tensor $\mathcal{W} \in \mathbb{R}^{r_1 \times \ldots \times r_K}$ and a matrix $\mathbf{U} \in \mathbb{R}^{s \times r_k}$ is denoted as $\mathcal{W} \times_k \mathbf{U}$, which is of size $r_1 \times \ldots \times r_{k-1} \times s \times r_{k+1} \times \ldots \times r_K$. It takes the products $\mathcal{W}$'s elements in the k-th dimension by the associated elements in the rows of $\mathbf{U}$ and sums the products. The corresponding entry-wise definition is

$$(\mathcal{W} \times_k \mathbf{U})_{i_1 \ldots i_{k-1} j i_{k+1} \ldots i_K} = \sum_{i_k=1}^{r_k} w_{i_1 \ldots i_K} u_{j i_k}. \quad (1)$$

There are two families of tensor decomposition, the Tucker family and the CP family. The Tucker family extends bilinear factorization models on matrix to handle tensor. Formally, Tucker decomposition defines a multi-linear form on tensor.

Definition 1 (Tucker decomposition) *The Tucker decomposition for a K-mode tensor $\mathcal{M} \in \mathbb{R}^{n_1 \times \ldots \times n_K}$ is*

$$\mathcal{M} = \mathcal{W} \times_1 \mathbf{U}^{(1)} \times_2 \ldots \times_K \mathbf{U}^{(K)} = [\![\mathcal{W}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(K)}]\!] \quad (2)$$

where $\mathcal{W} \in \mathbb{R}^{r_1 \times \ldots \times r_K}$ is the core *tensor, and $\mathbf{U}^{(k)} \in \mathbb{R}^{n_k \times r_k}$ are K latent factor matrices.*

The last identity in (2) is introduced by Kolda & Bader (2009) to compactly represent Tucker decomposition. We can relate Tucker decomposition with matrix-vector multiplication using the vec operation on tensors.

Proposition 2 *The Tucker decomposition* (2) *can be represented in a vectorized form*

$$\text{vec}([\![\mathcal{W}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(K)}]\!]) = \mathbf{U}^{(1)} \otimes \ldots \otimes \mathbf{U}^{(K)} \cdot \text{vec}(\mathcal{W}) \quad (3)$$

where $\otimes$ is the Kronecker product.

The alternating least square (ALS) method has been used to solve Tucker decomposition (Kolda & Bader, 2009). A graphical illustration of the Tucker decomposition on a 3-mode tensor is shown in Figure 1.

The CP family is a restricted form of the Tucker family. The entry-wise definition of CP for a tensor $\mathcal{M}$ is $m_{i_1 \ldots i_K} = \sum_{l=1}^{r} \lambda_l u_{i_1 l} \ldots u_{i_K l}$. We focus on generalizing Tucker decomposition to an infinite feature space using tensor-variate Gaussian or t processes in this paper.

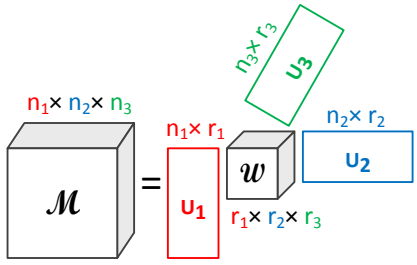


Figure 1. Illustration of Tucker decomposition on a 3-mode tensor $\mathcal{Y}$. The core tensor $\mathcal{W}$ is multiplied by matrices $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$ on different dimensions to obtain $\mathcal{Y}$.

3 Infinite Tucker Decomposition

In this section we present the infinite Tucker decomposition based on latent Gaussian processes and t processes. The following discussion is primarily for latent Gaussian processes. The model derivation for latent t processes is similar to that of latent Gaussian processes.

We extend classical Tucker decomposition in three aspects: i) flexible noise models for both continuous and binary observations; ii) an infinite core tensor to model complex interactions; and iii) latent Gaussian process or t prior process. More specifically, we assume the observed tensor $\mathcal{Y}$

[1] Unlike the usual column-wise vec-operation, applying our definition of vec() on matrices (2-mode tensors) gives the row-wise vectorization. This definition avoids the use of transpose in many equations throughout this paper.

is sampled from a latent real-valued tensor $\mathcal{M}$ via a probabilistic noise model $p(\mathcal{Y}|\mathcal{M}) = \prod_{\mathbf{i}} p(y_{\mathbf{i}}|m_{\mathbf{i}})$.

3.1 Tensor-variate Gaussian processes

We start by conducting Tucker decomposition for $\mathcal{M}$ with a core tensor $\mathcal{W}$ of uniform size, $r_1 = \ldots = r_K = r$. We assign independent standard normal distribution on $\mathcal{W}$, which is equivalent to

$$\mathrm{vec}(\mathcal{W}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4}$$

where $\mathbf{I}$ is the identity matrix. In light of the vectorized representation of Tucker decomposition (3), it is easy to see that $\mathrm{vec}(\mathcal{M})$ follows a normal distribution by marginalizing out $\mathrm{vec}(\mathcal{W})$.

$$\begin{aligned} \mathrm{vec}(\mathcal{M}) &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^{(1)} \otimes \ldots \otimes \mathbf{\Sigma}^{(K)}), \\ \mathbf{\Sigma}^{(k)} &= \mathbf{U}^{(k)}\mathbf{U}^{(k)\top} \end{aligned} \tag{5}$$

The covariance matrix of $\mathrm{vec}(\mathcal{M})$ has a Kronecker product structure. Because the covariance of any two elements $m_{\mathbf{i}}$ and $m_{\mathbf{j}}$ in the tensor $\mathcal{M}$ is

$$\mathrm{Cov}(m_{\mathbf{i}}, m_{\mathbf{j}}) = \prod_{k=1}^{K} \mathbf{\Sigma}^{(k)}_{i_k j_k} \tag{6}$$

we can interpret $\mathbf{\Sigma}^{(k)}$ as the covariance on the k-th mode. To model nonlinear relationships, we replace each row $\mathbf{u}^k_i$ of the latent feature matrix $\mathbf{U}^{(k)}$ by a nonlinear feature mapping $\phi(\mathbf{u}^k_i)$. We obtain an equivalent nonlinear covariance matrix $\mathbf{\Sigma}^{(k)}$ from nonlinear covariance function $k(\mathbf{u}^{(k)}_i, \mathbf{u}^{(k)}_j) = \langle \phi(\mathbf{u}^{(k)}_i), \phi(\mathbf{u}^{(k)}_j) \rangle$. The corresponding core tensor $\mathcal{W}$ after the feature mapping has the size of the mapped feature vector $\phi(\mathbf{u}^{(k)}_i)$ on mode k, which could be infinite. A rigorous exposition on the relation between core tensors and the feature vectors is given in Appendix B.

Note that by marginalizing out the core $\mathcal{W}$ in our model means, we effectively use all possible values of the latent core $\mathcal{W}$ for our model estimation. Using all possible values of $\mathcal{W}$ — instead of a particular single value of $\mathcal{W}$ — can greatly reduce the chance of overfitting and boost predictive performance (as demonstrated in our experiments).

We can also rewrite the normal probability density function (p.d.f.) in (5) using the vectorization relation (3) as

$$\begin{aligned} &\mathcal{N}(\mathrm{vec}(\mathcal{M}); \mathbf{0}, \mathbf{\Sigma}^{(1)} \otimes \ldots \otimes \mathbf{\Sigma}^{(K)}) \\ &= \mathcal{TN}(\mathcal{M}; \mathbf{0}, \mathbf{\Sigma}^{(1)}, \ldots, \mathbf{\Sigma}^{(K)}) \\ &= \frac{\exp\left\{-\frac{1}{2}\|[\![\mathcal{M}; (\mathbf{\Sigma}^{(1)})^{-\frac{1}{2}}, \ldots, (\mathbf{\Sigma}^{(K)})^{-\frac{1}{2}}]\!]\|^2\right\}}{(2\pi)^{n/2}\prod_{k=1}^{K}|\mathbf{\Sigma}^{(k)}|^{-\frac{n}{2n_k}}} \end{aligned} \tag{7}$$

where $n = \prod_k n_k$ and $\|\mathcal{X}\| = \sqrt{\sum_{\mathbf{i}} x^2_{\mathbf{i}}}$. The above equation (7) essentially defines a tensor-variate normal distribution $\mathcal{TN}(\mathbf{0}, \mathbf{\Sigma}^{(1)}, \ldots, \mathbf{\Sigma}^{(K)})$ that naturally generalizes the definition of matrix-variate normal distributions (Gupta & Nagar, 2000).

Combining nonlinear covariance functions and tensor-variate normal distributions, We can define tensor-variate Gaussian processes on $\mathcal{M}$.

Definition 3 (Tensor-variate Gaussian process) *A tensor-variate Gaussian process is a collection of random variables $\{m(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(K)})\}$, $\mathbf{u}^{(k)} \in \mathbb{R}^r$, whose finite joint probability over $\{(\mathbf{u}^{(1)}_{i_1}, \ldots, \mathbf{u}^{(K)}_{i_K})\}$ has the tensor-variate normal distribution density function. Specifically, given $\mathbf{U}^{(k)}$, the zero mean tensor-variate Gaussian process on $\mathcal{M}$ is denoted by*

$$\mathcal{M} \sim \mathcal{TGP}(\mathbf{0}, k(\cdot,\cdot)). \tag{8}$$

The finite probability density function is

$$p(\mathcal{M}|\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(K)}) = \mathcal{TN}(\mathcal{M}; \mathbf{0}, \mathbf{\Sigma}^{(1)}, \ldots, \mathbf{\Sigma}^{(K)}) \tag{9}$$

where $\mathbf{\Sigma}^{(k)} = k(\mathbf{U}^k, \mathbf{U}^k)$ is the covariance matrix.

Intuitively, a tensor-variate Gaussian process is equivalent to defining *infinite* Tucker decomposition with infinite feature mapping $\phi(\mathbf{u})$ and an infinite core tensor $\mathcal{W}_\infty$ whose elements are independent standard normal random variables

$$\mathcal{M} = [\![\mathcal{W}_\infty; \phi(\mathbf{U}^{(1)}), \ldots, \phi(\mathbf{U}^{(K)})]\!] \tag{10}$$

Definition 3 shows that probabilistic infinite Tucker decomposition of $\mathcal{M}$ can be realized by modeling $\mathcal{M}$ as a draw from a tensor-variate Gaussian process. Our definition of tensor-variate Gaussian processes generalizes matrix-variate Gaussian processes defined in (Yu et al., 2007) and (Yan et al., 2011).

Finally, to encourage sparsity in estimated $\mathbf{u}^{(k)}_i$—for easy model interpretation—we use a Laplace prior $\mathbf{u}^{(k)}_i \sim \mathcal{L}(\lambda) \propto \exp(-\lambda\|\mathbf{u}^{(k)}_i\|_1)$.

3.2 Tensor-variate t processes

Because of the strong relation between t-distributions and normal distributions—t distributions can be regarded as mixtures of Gaussian distributions weighted by Gamma distributions, we can easily define tensor-variate t processes as an alternative way to define infinite Tucker decomposition:

Definition 4 (Tensor-variate t Processes) *$\mathcal{M}$ follows a tensor-variate t process $\mathcal{TTP}(\nu, \mathbf{0}, k(\cdot,\cdot))$ with a degree of freedom $\nu > 2$, if $\mathcal{M}$ follows* tensor t *distribution with*

the following density

$$\mathcal{TT}(\mathcal{M};\nu,\mathbf{0},\{\mathbf{\Sigma}^{(k)}\}_{k=1}^K) = \frac{\Gamma(\frac{n+\nu}{2})\prod_{k=1}^K|\mathbf{\Sigma}^{(k)}|^{-\frac{n}{2n_k}}}{\Gamma(\frac{\nu}{2})(\nu\pi)^{\frac{n}{2}}} \left(1+\frac{1}{\nu}\|[\![\mathcal{M};(\mathbf{\Sigma}^{(1)})^{-\frac{1}{2}},\ldots,(\mathbf{\Sigma}^{(K)})^{-\frac{1}{2}}]\!]\|^2\right)^{-\frac{1}{2}(n+\nu)} \quad (11)$$

where $\Gamma(x)$ is the Gamma function.

3.3 Noise models

We use a noise model $p(\mathcal{Y}|\mathcal{M})$ to link the infinite Tucker decomposition and the tensor observation $\mathcal{Y}$.

Probit model: In this case, each entry of the observation is binary; that is, $y_{\mathbf{i}} \in \{0,1\}$. A probit function $p(y_{\mathbf{i}}|m_{\mathbf{i}}) = \Phi(m_{\mathbf{i}})^{y_{\mathbf{i}}}(1-\Phi(m_{\mathbf{i}}))^{1-y_{\mathbf{i}}}$ models the binary observation. Note that $\Phi(\cdot)$ is the standard normal cumulative distribution function. The probit model can be extended to handle multi-class and ordinal data as well.

Gaussian model: We use a Gaussian likelihood $p(y_{\mathbf{i}}|m_{\mathbf{i}}) = \mathcal{N}(y_{\mathbf{i}}|m_{\mathbf{i}},\sigma^2)$ to model the real-valued observation $y_{\mathbf{i}}$.

Missing values: We allow missing values in the observation. Let $\mathbb{O}$ denote the indices of the observed entries in $\mathcal{Y}$. Then we have $p(\mathcal{Y}_{\mathbb{O}}|\mathcal{M}_{\mathbb{O}})$ as the likelihood.

4 Algorithm

Given the observed tensor $\mathcal{Y}$, we aim to obtain the MAP estimate of component matrices $\mathbf{U}^{(k)}$ by maximizing the marginal likelihood $p(\mathcal{Y}|\{\mathbf{U}^{(k)}\}_{k=1}^K)p(\{\mathbf{U}^{(k)}\}_{k=1}^K)$. Integrating out $\mathcal{M}$ in the above equation is intractable for the probit noise. Therefore, we develop a variational expectation maximization (EM) algorithm. The inference for Gaussian noise is exact, but it can be derived in the variational EM framework as a special case. In the following paragraphs, we first present the inference and prediction algorithms for both of the noise models, and then describe an efficient algebraic approach to significantly reduce the computation complexity. Due to the space limitation, we only describe the inference algorithm for tensor-variate t processes. Tensor-variate Gaussian process inference can be derived similarly.

4.1 Inference

Probit noise: We follow the data augmentation scheme by Albert & Chib (1993) to decompose the probit model into $p(y_{\mathbf{i}}|m_{\mathbf{i}}) = \int p(y_{\mathbf{i}}|z_{\mathbf{i}})p(z_{\mathbf{i}}|m_{\mathbf{i}})dz_{\mathbf{i}}$. Let $\delta(\cdot)$ be the indicator function, we have

$$p(y_{\mathbf{i}}|z_{\mathbf{i}}) = \delta(y_{\mathbf{i}}=1)\delta(z_{\mathbf{i}}>0) + \delta(y_{\mathbf{i}}=0)\delta(z_{\mathbf{i}}\le 0),$$
$$p(z_{\mathbf{i}}|m_{\mathbf{i}}) = \mathcal{N}(z_{\mathbf{i}}|m_{\mathbf{i}},1)$$

It is well known that a t distribution can be factorized into a normal distribution convolved with a Gamma distribution, such that

$$\mathcal{TT}(\mathcal{M};\nu,\mathbf{0},\{\mathbf{\Sigma}^{(k)}\}_{k=1}^K) = \int \mathrm{Gam}(\eta;\nu/2,\nu/2)\cdot \mathcal{TN}(\mathcal{M};\mathbf{0},\{\eta^{-1/K}\mathbf{\Sigma}^{(k)}\}_{k=1}^K)d\eta, \quad (12)$$

The joint probability likelihood with data augmentation is

$$p(\mathcal{Y},\mathcal{Z},\mathcal{M},\eta,\mathcal{U}) = p(\mathcal{Y}|\mathcal{Z})p(\mathcal{Z}|\mathcal{M})p(\mathcal{M}|\eta,\mathcal{U})p(\eta)p(\mathcal{U}).$$

where $\mathcal{U} = [\mathbf{U}^{(1)},\ldots,\mathbf{U}^{(K)}]$, $p(\mathcal{M}|\eta,\mathcal{U})$ and $p(\eta)$ are the tensor-variate normal distribution and the Gamma distribution in (12). $p(\mathcal{U})$ is the Laplace prior.

Our variational EM algorithm consists of a variational E-step and a gradient-based M-step. In the E-step, we approximate the posterior distribution $p(\mathcal{Z},\mathcal{M},\eta|\mathcal{Y},\mathcal{U})$ by a fully factorized distribution $q(\mathcal{Z},\mathcal{M},\eta) = q(\mathcal{Z})q(\mathcal{M})q(\eta)$. Variational inference minimizes the Kullback-Leibler (KL) divergence between the approximate posterior and the true posterior.

$$\min_q \mathrm{KL}\left(q(\mathcal{Z})q(\mathcal{M})q(\eta)\|p(\mathcal{Z},\mathcal{M},\eta|\mathcal{Y},\mathcal{U})\right). \quad (13)$$

The variational approach optimizes one approximate distribution, *e.g.*, $q(\mathcal{Z})$, in (13) at a time, while having all the other approximate distributions fixed. We loop over $q(\mathcal{Z})$, $q(\mathcal{M})$ and $q(\eta)$ to iteratively optimize the KL divergence until convergence.

Given $q(\mathcal{M})$ and $q(\eta)$, the $q(z_{\mathbf{i}})$ is a truncated normal distribution

$$q(z_{\mathbf{i}}) \propto \mathcal{N}(\mathbb{E}_q[m_{\mathbf{i}}],1)\delta(z_{\mathbf{i}}>1), \quad (14)$$

$$\mathbb{E}_q[z_{\mathbf{i}}] = \mathbb{E}_q[m_{\mathbf{i}}] + \frac{(2y_{\mathbf{i}}-1)\mathcal{N}(\mathbb{E}_q[m_{\mathbf{i}}]|0,1)}{\Phi((2y_{\mathbf{i}}-1)\mathbb{E}_q[m_{\mathbf{i}}])}. \quad (15)$$

Given $q(\mathcal{Z})$ and $q(\eta)$, it is more convenient to write the optimized approximate distribution for $\mathcal{M}$ in its vectorized form. Let

$$\mathbf{\Sigma}_p = \mathbf{\Sigma}^{(1)} \otimes \ldots \otimes \mathbf{\Sigma}^{(K)} \quad (16)$$

be the covariance matrix of the tensor TP prior, we have

$$q(\mathrm{vec}(\mathcal{M})) = \mathcal{N}(\mathrm{vec}(\mathcal{M})|\boldsymbol{\mu},\mathbf{\Upsilon}), \quad (17)$$

$$\boldsymbol{\mu} = \mathrm{vec}(\mathbb{E}_q[\mathcal{M}]) = \mathbf{\Upsilon}\ \mathrm{vec}(\mathbb{E}_q[\mathcal{Z}]) \quad (18)$$

$$\mathbf{\Upsilon} = \mathbb{E}_q[\eta]^{-1}\mathbf{\Sigma}_p\left(\mathbf{I}+\mathbb{E}_q[\eta]^{-1}\mathbf{\Sigma}_p\right)^{-1}. \quad (19)$$

The optimized $q(\eta)$ is also a Gamma distribution:

$$q(\eta) = \mathrm{Gam}(\eta;\beta_1,\beta_2),\ \ \mathbb{E}_q[\eta] = \frac{\beta_1}{\beta_2},\ \ \beta_1 = \frac{\nu+n}{2},$$
$$\beta_2 = \frac{\nu+\boldsymbol{\mu}^\top\mathbf{\Sigma}_p^{-1}\boldsymbol{\mu}+\mathrm{tr}(\mathbf{\Sigma}_p^{-1}\mathbf{\Upsilon})}{2}.$$

Based on the variational approximate distribution obtained in the E-step, we maximize the expected log likelihood over $\mathcal{U}$ in the M-step.

$$\max_{\mathcal{U}} \mathbb{E}_q \left[\log p(\mathcal{Y}, \mathcal{Z}, \mathcal{M}, \eta | \mathcal{U}) p(\mathcal{U})\right]. \tag{20}$$

After eliminating constant terms, we need to solve the following optimization problem with regularization on $\mathbf{U}^{(k)}$

$$\begin{aligned}\min_{\mathcal{U}} f(\mathcal{U}) = \tau \|[\![\mathbb{E}_q[\mathcal{M}]; (\boldsymbol{\Sigma}^{(1)})^{-\frac{1}{2}}, \ldots, (\boldsymbol{\Sigma}^{(K)})^{-\frac{1}{2}}]\!]\|^2 \\ + \sum_{k=1}^{K} \frac{n}{n_k} \log |\boldsymbol{\Sigma}^{(k)}| + \tau \operatorname{tr}\left(\boldsymbol{\Sigma}_p^{-1} \boldsymbol{\Upsilon}\right), \end{aligned} \tag{21}$$

where $\tau = \mathbb{E}_q[\eta]$. In the above equation (21), $\boldsymbol{\Sigma}^{(k)} = k(\mathbf{U}^{(k)}, \mathbf{U}^{(k)})$ is considered as a function of $\mathbf{U}^{(k)}$. $\boldsymbol{\Upsilon}$ and τ are the statistics computed in the E-step, and they have fixed values. The gradient of $f(\mathcal{U})$ *w.r.t.* to a scalar $u_{ij}^{(k)}$ is

$$\begin{aligned}\frac{\partial f}{\partial u_{ij}^{(k)}} &= \frac{n}{n_k} \operatorname{tr}\left((\boldsymbol{\Sigma}^{(k)})^{-1} \frac{\partial \boldsymbol{\Sigma}^{(k)}}{\partial u_{ij}^{(k)}}\right) \\ &\quad + \tau \boldsymbol{\mu}^\top \boldsymbol{\Delta}^{(k)} \boldsymbol{\mu} + \tau \operatorname{tr}\left(\boldsymbol{\Delta}^{(k)} \boldsymbol{\Upsilon}\right) \qquad (22) \\ \boldsymbol{\Delta}^{(k)} &= (\boldsymbol{\Sigma}^{(1)})^{-1} \otimes \ldots \otimes (\boldsymbol{\Sigma}^{(k-1)})^{-1} \\ &\quad \otimes (\boldsymbol{\Sigma}^{(k)})^{-1} \frac{\partial \boldsymbol{\Sigma}^{(k)}}{\partial u_{ij}^{(k)}} (\boldsymbol{\Sigma}^{(k)})^{-1} \\ &\quad \otimes (\boldsymbol{\Sigma}^{(k+1)})^{-1} \otimes \ldots \otimes (\boldsymbol{\Sigma}^{(K)})^{-1} \qquad (23)\end{aligned}$$

With an ℓ_1 penalty on $f(\mathcal{U})$, we choose a projected scaled subgradient L-BFGS algorithm for optimization—due to its excellent performance (Schmidt, 2010).

Gaussian noise: The inference for the regression case follows the same format as the probit noise case. $\mathbb{E}_q[\mathcal{Z}]$ is replaced by $\mathcal{Y}_{\mathbb{O}}$, and we do not need to update $q(\mathcal{Z})$. The variational EM algorithm is only applied to the observed entries of $\mathcal{M}_{\mathbb{O}}$ and the covariance $[\Sigma_p]_{\mathbb{O},\mathbb{O}}$.

4.2 Prediction

Probit noise: Given a missing value index $\mathbf{i} = (i_1, \ldots, i_K)$, the predictive distribution is

$$p(y_{\mathbf{i}}|\mathcal{Y}) \approx \int p(y_{\mathbf{i}}|m_{\mathbf{i}}) p(m_{\mathbf{i}}|\mathcal{M}, \eta) q(\mathcal{M}) q(\eta) dm_{\mathbf{i}} d\mathcal{M} d\eta$$

The above integral is intractable, so we replace η integral $q(\eta)d\eta$ by the mode of its approximate posterior distribution $\tau^* = (\beta_1 - 1)/\beta_2$, thus the predictive distribution is approximated by

$$\begin{aligned}&\int p(y_{\mathbf{i}} = 1|z_{\mathbf{i}}) p(z_{\mathbf{i}}|m_{\mathbf{i}}) p(m_{\mathbf{i}}|\mathcal{M}, \tau^*) q(\mathcal{M}) dz_{\mathbf{i}} dm_{\mathbf{i}} d\mathcal{M} \\ &= \int \delta(z_{\mathbf{i}} > 0) \mathcal{N}(z_{\mathbf{i}}|\mu_{\mathbf{i}}(1), \nu_{\mathbf{i}}^2(1)) dz_{\mathbf{i}} = \Phi(\frac{\mu_{\mathbf{i}}(1)}{\nu_{\mathbf{i}}(1)}) \end{aligned} \tag{24}$$

where

$$\begin{aligned}k(\mathbf{i}, \mathbf{j}) &= \prod_{k=1}^{K} \Sigma^{(k)}(\mathbf{u}_{i_k}^{(k)}, \mathbf{u}_{j_k}^{(k)}),\ \mathbf{k} = [k(\mathbf{i}, \mathbf{j})]_{\mathbf{j} \in \mathbb{O}}^\top \\ \mu_{\mathbf{i}}(\rho) &= \mathbf{k}^\top (\boldsymbol{\Sigma}_p + \rho^2 \tau^* \mathbf{I})^{-1} \operatorname{vec}(\mathcal{Y}) \\ \nu_{\mathbf{i}}^2(\rho) &= 1 + \frac{1}{\tau^*}[k(\mathbf{i}, \mathbf{i}) - \mathbf{k}^\top (\boldsymbol{\Sigma}_p + \rho^2 \tau^* \mathbf{I})^{-1} \mathbf{k}]\end{aligned}$$

Gaussian noise: The predictive distribution for the regression case can be evaluate by a similar integral, which gives

$$p(y_{\mathbf{i}}|\mathcal{Y}_{\mathbb{O}}) = \mathcal{N}(z_{\mathbf{i}}|\mu_{\mathbf{i}}(\sigma), \nu_{\mathbf{i}}^2(\sigma)). \tag{25}$$

4.3 Efficient Algorithm

If $n = \prod_{k=1}^{K} n_k$ is the number of all elements in $\mathcal{Y}$, a naïve implementation of the above algorithm requires prohibitive $O(n^3)$ time complexity and $O(n^2)$ space complexity for each EM iteration. The key computation bottlenecks are the computations involving $\boldsymbol{\Upsilon}$, which are $\operatorname{tr}(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Upsilon})$ in (21), $\operatorname{tr}\left(\boldsymbol{\Delta}^{(k)}\boldsymbol{\Upsilon}\right)$ in (22) and $\boldsymbol{\Upsilon} \operatorname{vec}(\mathbb{E}_q[\mathcal{Z}])$ in (18).

To avoid this high complexity, we can make use of the Kronecker product structure and generalize the strategy used by Yan et al. (2011) on matrix-variate Gaussian processes. We assume $\mathbb{E}_q[\eta] = 1$ to simplify the computation. It is easy to adapt our computation strategies to $\mathbb{E}_q[\eta] \neq 1$. Let $\boldsymbol{\Sigma}^{(k)} = \mathbf{V}^{(k)} \boldsymbol{\Lambda}^{(k)} \mathbf{V}^{(k)\top}$ be the singular value decomposition (SVD) of the covariance matrix $\boldsymbol{\Sigma}^{(k)}$, $\mathbf{V}^{(k)}$ is an orthogonal matrix and $\boldsymbol{\Lambda}^{(k)}$ is a diagonal matrix. $\boldsymbol{\Upsilon}$ can be represented as

$$\begin{aligned}&\boldsymbol{\Upsilon} = \boldsymbol{\Sigma}_p (\mathbf{I} + \boldsymbol{\Sigma}_p)^{-1} = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^\top, \text{ where} \\ &\mathbf{V} = \mathbf{V}^{(1)} \otimes \ldots \otimes \mathbf{V}^{(K)} \\ &\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^{(1)} \otimes \ldots \otimes \boldsymbol{\Lambda}^{(K)} (\mathbf{I} + \boldsymbol{\Lambda}^{(1)} \otimes \ldots \otimes \boldsymbol{\Lambda}^{(K)})^{-1}\end{aligned}$$

It is obvious that $\mathbf{V}$ is an orthogonal matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix. The above relation implies that we can actually compute the singular value decomposition of $\boldsymbol{\Upsilon} = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^\top$ from covariance matrices $\boldsymbol{\Sigma}^{(k)}$.

Computing $\operatorname{tr}(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Upsilon})$ and $\operatorname{tr}\left(\boldsymbol{\Delta}^{(k)}\boldsymbol{\Upsilon}\right)$:
We define $\mathbf{d}_k = \operatorname{diag}(\mathbf{V}^{(k)} (\boldsymbol{\Sigma}^{(k)})^{-1} \mathbf{V}^{(k)^\top})^\top$ with $\boldsymbol{\Sigma}^{(k)}$ being a computed statistics in the E-step, $\operatorname{diag}(\boldsymbol{\Lambda})$ denotes the diagonal elements of $\boldsymbol{\Lambda}$, and $\mathcal{D}$ is a tensor of size $n_1 \times \ldots \times n_K$, such that $\operatorname{vec}(\mathcal{D}) = \operatorname{diag}(\boldsymbol{\Lambda})$. In order to efficiently compute $\operatorname{tr}(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Upsilon})$ appearing in equation (21), we use the following relations

$$\begin{aligned}&\operatorname{tr}(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\Upsilon}) = \operatorname{tr}(\boldsymbol{\Sigma}_p^{-1} \mathbf{V}^\top \boldsymbol{\Lambda} \mathbf{V}) = \operatorname{tr}(\boldsymbol{\Lambda} \mathbf{V} \boldsymbol{\Sigma}_p^{-1} \mathbf{V}^\top) \\ &= \operatorname{diag}(\mathbf{V} \boldsymbol{\Sigma}_p^{-1} \mathbf{V}^\top)^\top \operatorname{diag}(\boldsymbol{\Lambda}) \\ &= \mathbf{d}_1 \otimes \ldots \otimes \mathbf{d}_K \operatorname{diag}(\boldsymbol{\Lambda}) \\ &= \mathbf{d}_1 \otimes \ldots \otimes \mathbf{d}_K \operatorname{vec}(\mathcal{D}) \\ &= \mathcal{D} \times_1 \mathbf{d}_1 \ldots \times_K \mathbf{d}_K, \end{aligned} \tag{26}$$

Both time and space complexities of the last formula (26) is $O(n)$. The technique of (26) can be used to efficient calculate the last term $\mathrm{tr}\left(\mathbf{\Delta}^{(k)}\mathbf{\Upsilon}\right)$ in the gradient (22) as well. Furthermore, if we use the incremental EM algorithm, which only takes one gradient descent step for each M step, equation (26) can be simplified such that $\mathbf{d}_k$ is the inverse of the eigenvalues of $\mathbf{\Sigma}^{(k)}$.

$$\mathbf{d}_k = \mathrm{diag}\left((\mathbf{\Lambda}^{(k)})^{-1}\right)^\top$$

In practice, this incremental EM algorithm performs as good as EM with full optimization.

Computing $\mathbf{\Upsilon}\,\mathrm{vec}(\mathbb{E}_q[\mathcal{Z}])$:
For any tensor $\mathcal{A}$ of the same size as $\mathcal{D}$, $\mathbf{\Lambda}\,\mathrm{vec}(\mathcal{A})$ means multiplying the j-th element of $\mathrm{vec}(\mathcal{A})$ by the $\mathbf{\Lambda}_{jj}$, which is the j-th element of $\mathrm{vec}(\mathcal{D})$. So we have $\mathbf{\Lambda}\,\mathrm{vec}(\mathcal{A}) = \mathrm{vec}(\mathcal{D}\odot\mathcal{A})$, where $\odot$ denotes the entry-wise product.

In light of this relation and (3), we can efficiently compute (18) by

$$\begin{aligned}
&\mathbf{\Upsilon}\,\mathrm{vec}(\mathbb{E}_q[\mathcal{Z}]) = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\,\mathrm{vec}(\mathbb{E}_q[\mathcal{Z}])\\
&= \mathbf{V}\mathbf{\Lambda}\,\mathrm{vec}\left(\left[\!\left[\mathbb{E}_q[\mathcal{Z}]; \mathbf{V}^{(1)\top},\ldots,\mathbf{V}^{(K)\top}\right]\!\right]\right)\\
&= \mathbf{V}\,\mathrm{vec}\left(\mathcal{G}\odot\mathcal{D}\right),
\end{aligned}$$

where $\mathcal{G} = \left[\!\left[\mathbb{E}_q[\mathcal{Z}]; \mathbf{V}^{(1)\top},\ldots,\mathbf{V}^{(K)\top}\right]\!\right]$. It can be further simplified as:

$$\mathbf{\Upsilon}\,\mathrm{vec}(\mathbb{E}_q[\mathcal{Z}]) = \mathrm{vec}(\left[\!\left[\mathcal{G}\odot\mathcal{D}; \mathbf{V}^{(1)},\ldots,\mathbf{V}^{(K)}\right]\!\right]). \quad (27)$$

Equation (27) greatly reduces the time and space complexities of (18). The time complexity of each mode-k multiplication is $O(n_k n)$, and SVD of the covariance matrix on the k-th mode costs $O(n_k^3)$. So the total time complexity including other computation steps is $O(\sum_{k=1}^K n_k^3 + n_k n)$. If we assume the length of each mode are the same, the time complexity of our efficient algorithm is $O(n^{1+\frac{1}{K}})$ which is a huge reduction from $O(n^3)$ of the naïve algorithm. The space complexity is reduced to $O(n+\sum_{k=1}^K n_k^2)$ and $O(n)$ for tensors with uniform size for each mode, because we only need to store the covariance matrix for each mode rather than the $n^2\times n^2$ covariance matrix $\mathbf{\Sigma}_p$. We can further reduce the complexities by approximating the covariance matrices via truncated SVD.

5 Related Works

The *InfTucker* model extends Probabilistic PCA (PPCA) (Tipping & Bishop, 1999) and Gaussian process latent variable models (GPLVMs) (Lawrence, 2006). Using a Dirac covariance and a linear covariance for the two modes of a matrix, our model reduces to the PPCA model $\mathcal{TN}(0,\mathbf{I},\mathbf{U}\mathbf{U}^\top)$; similarly, using a Dirac covariance and a nonlinear covariance $\mathbf{K}$, our model reduces to the GPLVM model $\mathcal{TN}(0,\mathbf{K},\mathbf{I})$. While PPCA and GPLVM model interactions of one mode of a matrix and ignore the joint interactions of two modes, *InfTucker* does. Our model is also related to previous matrix-variate GPs (Yu et al., 2007). The main difference lies in the fact they used linear covariance functions to reduce the computational complexities and dealt with matrix-variate data for online recommendation and link prediction. Hoff (2011) proposed a hierarchical Bayesian extension to CANDECOMP/PARAFAC that captures the interactions of component matrices. Porteous et al. (2008) generalized hierarchical Dirichlet processes to handle tensor data. Unlike these approaches, ours can handle non-Gaussian noise and uses nonlinear covariance functions to model complex interactions. Both Hoff (2011) and Porteous et al. (2008) used Gibbs samplers for inference—requiring high computational cost and making their approach infeasible for tensors with moderate and large sizes.

The most closely related work is the probabilistic Tucker decomposition (pTucker) model (Chu & Ghahramani, 2009); actually the GP-based *InfTucker* with Gaussian noise function reduces to pTucker as a special case when using a linear covariance function. Our TP-based *InfTucker* further differs from pTucker by marginalizing out a scaling hyperparameter of the covariance function and handles non-Gaussian noise functions. A kernelized version of pTucker was also discussed by the authors without conducting any experiment on it. The kernelized pTucker approach captures nonlinear relationships between observations in the same way as kernel PCA. This is fundamentally different from our approach, which captures nonlinear relationships between the latent factors. In addition, the kernelized pTucker approach needs to explicitly calculate the Kronecker product of the latent factors, so it is difficult for pTucker to scale up to large datasets. In contrast, our inference method does not conduct any expensive Kronecker product—we have exploited properties of Kronecker products to greatly simplify the computations.

To handle missing data, enhance model interpretability, and avoid overfitting, several extensions (e.g., using nonnegativity constraints) to tensor decomposition have been proposed, including nonnegative tensor decomposition (NTD) (Shashua & Hazan, 2005) and Weighted tensor decomposition (WTD) (Acar et al., 2011). Unlike ours, these models either solve the core tensors explicitly, or do not handle nonlinear multiway interactions.

6 Experiments

We use *InfTucker*^{gp} and *InfTucker*^{tp} to denote the two new infinite Tucker decomposition models based on tensor-variate Gaussian and t processes, respectively. To evaluate

Data	amino	flow injection	bread
CP	0.053±0.002	0.051±0.005	0.238±0.001
TD	0.054±0.002	0.051±0.003	0.248±0.001
HOSVD	0.053±0.002	0.052±0.004	0.259±0.001
NCP	0.057±0.005	0.110±0.023	0.233±0.001
PTD	0.054±0.002	0.048±0.002	0.240±0.001
WCP	0.049±0.004	0.079±0.011	0.246±0.003
InfTucker^{gp}	**0.047**±0.003	0.049±0.002	0.232±0.001
InfTucker^{tp}	**0.047**±0.003	**0.046**±0.002	**0.225**±0.001

Table 1. The mean square errors (MSE) with standard errors. The results suggested that our new approaches–*InfTucker*^{gp} and *InfTucker*^{tp} —achieved higher prediction accuracy than all the competing approaches. In particular, the improvements of *InfTucker*^{tp} over all the other methods on all datasets (except *InfTucker*^{gp} on the amino dataset) are statistically significant ($p < 0.05$).

them, we conducted two sets of experiments, one on continuous tensor data and the other on binary tensor data, to evaluate the prediction accuracy on hold-out data. The hyperparameters are determined using cross-validation in our MAP inference framework. For both experiments, we compared *InfTucker* with the following tensor decomposition methods: CANDECOMP/PARAFAC (CP), Tucker decomposition (TD), Nonnegative CP (NCP), High Order SVD (HOSVD), Weighted CP (WCP) and Probabilistic Tucker Decomposition (PTD). We implemented PTD as described in the paper by Chu & Ghahramani (2009) and applied to a small continuous tensor data (*bread* as described in the 6.1). To handle larger and binary datasets, we used probit models and the efficient computation techniques described in Section 4.3 for PTD. For the other methods, we used the implementation of the tensor data analysis toolbox[2] developed by T. G. Kolda.

6.1 Experiment on continuous tensor data

Experimental setting. We used three continuous chemometrics datasets[3], *amino*, *bread*, and *flow injection*. The dimensions of the tensors are $5 \times 51 \times 201$, $10 \times 11 \times 8$, $12 \times 100 \times 89$, respectively.

All the above tensor data were normalized such that each element of the tensor has zero mean and unit variance (based on the vectorized representations). For each tensor, we randomly split it via 5-fold cross validation: each time four folds are used for training and one fold for testing. This procedure was repeated 10 times, each time with a different partition for the 5-fold cross validation. In *InfTucker*^{tp} , the degree of freedom ν in the tensor-variate t process is fixed to 10. We chose the Gaussian/exponential covariance functions $\Sigma^{(k)}(\mathbf{u}_i, \mathbf{u}_j) = e^{-\gamma\|\mathbf{u}_i - \mathbf{u}_j\|^t}$, where $t = 1, 2$ and γ is selected from $[0.01 : 0.05 : 1]$ by 5-fold cross validation. The regularization parameter λ for *InfTucker*^{gp} and *InfTucker*^{tp} is chosen from $\{1, 10, 100\}$.

Results. We compared the the prediction accuracy values of all the approaches on hold-out elements of the tensor data. 80% of the data are held out for training and the remaining 20% are used for testing in each run. For each comparison, we used the same number of latent factors, denoted as r, for all the approaches. We varied r from 3 to 5 and computed the averaged mean square errors (MSEs) and the standard errors of the MSEs. Based on cross-validation, we set $r = 3$. The MSEs on the three datasets are summarized in Table 1. Based on the prediction accuracies, PTD and WCP tie on the third best, while HOSVD is the worst (perhaps due to the strong nonnegativity constraint on the latent factors). Clearly, *InfTucker*^{gp} achieved higher prediction accuracies than all the previous approaches on all the datasets, and *InfTucker*^{tp} further outperformed *InfTucker*^{gp} for most cases.

6.2 Experiment on binary tensor data

Experimental setting. We extracted three binary social network datasets, *Enron*, *Digg1*, and *Digg2*, for our experimental evaluation. *Enron* is a relational dataset describing the three-way relationship: sender-receiver-time. This dataset, extracted from the Enron email dataset[4], has the dimensionality of $203 \times 203 \times 200$ with 0.01% non-zero elements. The *Digg1* and *Digg2* datasets were all extracted from a social news website `digg.com`[5]. *Digg1* describes a three-way interaction: news-keyword-topic, and *Digg2* describes a four-way interaction: user-news-keyword-topic. *Digg1* has the dimensionality of $581 \times 124 \times 48$ with 0.024% non-zero elements, and *Digg2* has the dimensionality of $22 \times 109 \times 330 \times 30$ with only 0.002% non-zero elements. Apparently these datasets are very sparse.

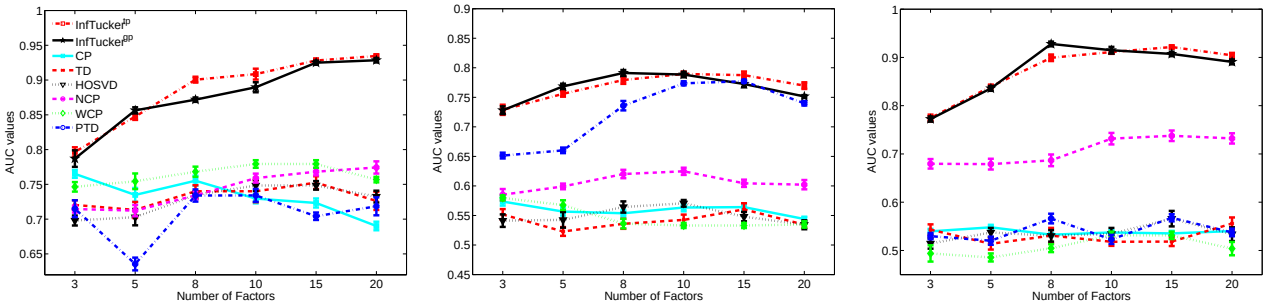
Results. We chose r from the range {3,5,8,10,15,20} based on cross-validation. Since the data are binary, we evaluated all these approaches by area-under-curve (AUC) values by averaged over 50 runs. 80% of the data are used for training and 20% are used for testing in each run. For *InfTucker*, the values to compute the AUCs are the predicted probability obtained from equation (24). For other methods, the values to compute the AUCs are the reconstructed values. The larger the averaged AUC value an approach achieves, the better it is. We reported the averaged AUC values for all algorithms in Figure 2. Again, the proposed *InfTucker*^{gp} and *InfTucker*^{tp} approaches signif-

[2] `http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/`

[3] Available from `http://www.models.kvl.dk/datasets`

[4] Available at `http://www.cs.cmu.edu/~enron/`

[5] Available at `http://www.public.asu.edu/~ylin56/kdd09sup.html`



(a) *Enron* (b) *digg1* (c) *digg2*

Figure 2. The AUC values of six algorithms on three multi-way networks. The dimensions of the latent factors are $r = 3, 5, 8, 10, 15, 20$ respectively.

icantly outperform all the others. Note that the nonprobabilistic approaches—such as CP and TD—- suffer severely from the least square minimization; given the sparse and binary training data, the least-square-minimization leads to too many predictions with zero values, a result of both overfitting and mis-model fitting. This experimental comparison fully demonstrates the advantages of *InfTucker* (stemming from the right noise models and the nonparametric Bayesian treatment).

7 Conclusions

To conduct multiway data analysis, we have proposed a new nonparametric Bayesian tensor decomposition framework, *InfTucker*, where the observed tensor is modeled as a sample from stochastic processes on tensors. In particular, we have employed tensor-variate Gaussian and t processes. This new framework can model nonlinear interactions between multi-aspects of the tensor data, handle missing data and noise, and quantify prediction confidence (based on predictive posterior distributions). We have also presented an efficient variational method to estimate *InfTucker* from data. Experimental results on chemometrics and social network datasets demonstrate that the superior predictive performance of *InfTucker* over the alternative tensor decomposition approaches.

Acknowledgment This work was supported by NSF IIS-0916443, NSF CAREER award IIS-1054903, and the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

References

Acar, E., Dunlavy, D. M., Kolda, T. G., and Mørup, M. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

Albert, James H and Chib, Siddhartha. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88:669–679, 1993.

Chu, Wei and Ghahramani, Zoubin. Probabilistic models for incomplete multi-dimensional arrays. *AISTATS*, 2009.

Gupta, Arjun K. and Nagar, D. K. *Matrix variate distributions*. CRC Press, 2000.

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

Hoff, Peter D. Hierarchical multilinear models for multiway data. *Computational Statistics & Data Analysis*, 55:530–543, 2011.

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

Lathauwer, Lieven De, Moor, Bart De, and Vandewalle, Joos. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl*, 21:1253–1278, 2000.

Lawrence, Neil. The Gaussian process latent variable model. Technical Report CS-06-03, The University of Sheffield, 2006.

Porteous, Ian, Bart, Evgeniy, and Welling, Max. Multi-HDP: A non-parametric Bayesian model for tensor factorization. In *AAAI*, 2008.

Schmidt, Mark. *Graphical Model Structure Learning with L1-Regularization*. PhD thesis, UBC, 2010.

Shashua, Amnon and Hazan, Tamir. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd ICML*, 2005.

Tipping, Michael E. and Bishop, Christopher M. Probabilistic principal component analysis. *Journal of The Royal Statistical Society Series B-statistical Methodology*, 61:611–622, 1999.

Tucker, Ledyard. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

Yan, Feng, Xu, Zenglin, and Qi, Yuan. Sparse matrix-variate gaussian process blockmodels for network modeling. In *UAI*, pp. 745–752. 2011.

Yu, Kai, Chu, Wei, Yu, Shipeng, Tresp, Volker, and Xu, Zhao. Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems*, 2007.