InfTucker: t-Process based Infinite Tensor Decomposition

Zenglin Xu
Department of CS
Purdue University
xu218@purdue.edu

Feng Yan
Department of CS
Purdue University
yan12@purdue.edu

Yuan Qi
Departments of CS and Statistics
Purdue University
alanqi@cs.purdue.edu

September 1, 2011

Abstract

Tensor decomposition is a powerful tool for multiway data analysis. Many popular tensor decomposition approaches—such as the Tucker decomposition and CANDECOMP/PARAFAC (CP)—conduct 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 a tensor-variate latent t process model, InfTucker, for robust multiway data analysis: it conducts robust Tucker decomposition in an infinite feature space. Unlike classical tensor decomposition models, it handles both continuous and binary data in a probabilistic framework. Unlike previous nonparametric Bayesian models on matrices and tensors, our latent t-process model focuses on multiway analysis and uses nonlinear covariance functions. To efficiently learn InfTucker from data, we develop a novel variational inference technique on tensors. Compared with classical implementation, the new technique reduces both time and space complexities by several orders of magnitude. This technique can be easily adopted in other contexts (e.g., multitask learning) where we encounter tensor-variate t processes or Gaussian processes. Our experimental results on chemometrics and social network datasets demonstrate that the new InfTucker model achieves significantly higher prediction accuracy than several state-of-art tensor decomposition approaches including High Order Singular Value Decomposition (HOSVD), Weighted CP, and nonnegative tensor decomposition.
1 Introduction

Many real world datasets are naturally described by multiway arrays or tensors. For example, histories of email correspondences can be represented by a tensor with 4 modes (sender, receiver, date, content); and user customer ratings can be represented by a tenor with four modes (user, item, rating, time). Therefore there is a critical need for principled multiway data analysis tools.

Traditional multiway factor models—such as the Tucker decomposition [21] and CANDECOMP/PARAFAC (CP) [6]—have been widely applied in many areas (e.g., network traffic analysis [24], computer vision [17] and social network analysis [18, 14, 19], etc). These models, however, face serious challenges for modeling complex multiway interactions. First the 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 tensor data are often noisy and contain many outliers; classical models are not designed to handle outliers. Third, the data may not be restricted to real values; they can be binary as in dynamic network data or they may have ordinal values for user-movie-ratings. The classical models treat all these data as real-valued and this treatment would lead to degenerated predictive performance.

To address these challenges, we propose a robust nonparametric multiway analysis model, InfTucker. It conducts the Tucker decomposition in an infinite dimensional feature space, which captures nonlinear interactions between different tensor modes. Compared with the tensor-variate Gaussian process approaches [7, 23], the \( t \)-process-based InfTucker is more robust (not sensitive to data outliers). Furthermore, it handles various data types (e.g., incomplete or binary data) by adopting suitable probabilistic data likelihoods.

Although our nonparametric models offer elegant solutions to robust multiway analysis, computationally it is difficult to train them from multi-aspect data. To overcome this computational difficulty, we develop an efficient variational Bayesian approach that explores the tensor structures to reduce the computational cost.

In summary, the major contributions of this paper include:

i. We propose a tensor-variate latent \( t \) process model, InfTucker, for robust multiway data analysis. Recently several nonparametric Bayesian models on matrices and tensors [5, 23, 4] have been developed, but they are either designed for different tasks (e.g., multitask learning)—rather than multiway analysis—or use a linear kernel to make the computation feasible in practice. By contrast, based on latent \( t \)-processes, our model conducts robust nonlinear Tucker decomposition.

ii. We present a novel technique that reduces the computational complexity of variational inference on tensors. Specifically, for a \( K \)-mode tensor with \( n_k \) entities in each mode, our algorithm has \( O(\sum_{k=1}^{K} n_k^2 + (\sum_{k=1}^{K} n_k) \prod_{k=1}^{K} n_k) \) time and \( O(\sum_{k=1}^{K} n_k^2 + \prod_{k=1}^{K} n_k) \) space complexities, instead of \( O(\prod_{k=1}^{K} n_k^2) \) time and \( O(\prod_{k=1}^{K} n_k^2) \) space complexities of a naive inference technique. The efficient inference technique enables us to use tensor-variate \( t \)-processes with nonlinear covariance functions.

iii. Our experimental results on chemometrics and social network datasets demonstrate that the InfTucker achieves significantly higher prediction accuracy than...
several state-of-art tensor decomposition approaches including High Order Singular Value Decomposition (HOSVD) [10], Weighted CP [1] and nonnegative tensor decomposition [17].

2 Preliminary

Notations. Throughout this paper, we denote scalars by lower case letters (e.g. \( a \)), vectors by bold lower case letters (e.g. \( \mathbf{a} \)), matrices by bold upper case letters (e.g. \( A \)), and tensors by calligraphic upper case letters (e.g. \( \mathcal{A} \)). Calligraphic upper case letters are also used for probability distributions, e.g., \( \mathcal{N}(\mu, \Sigma) \). We use \( u_{ij} \) to represent the \((i,j)\) entry of a matrix \( U \), \( y_i \) to represent the \( i = (i_1, \ldots, i_K) \) entry of a tensor \( \mathcal{Y} \). \( U \otimes V \) denotes the Kronecker product of the two matrices there. We define the vectorization operation, denoted by \( \text{vec}(\mathcal{Y}) \), to stack the tensor entries into a \( \prod_{k=1}^K n_k \) by 1 vector. The entry \( i = (i_1, \ldots, i_K) \) of \( \mathcal{Y} \) is mapped to the entry at position \( j = i_K + \sum_{k=1}^{K-1} (i_k - 1) \prod_{k+1}^K n_k \) of \( \text{vec}(\mathcal{Y}) \). The mode-\( k \) product of a tensor \( \mathcal{W} \in \mathbb{R}^{r_1 \times \cdots \times r_K} \) with a matrix \( U \in \mathbb{R}^{n_k \times r_k} \) is denoted as \( \mathcal{W} \times_k U \) and it is of size \( r_1 \times \cdots \times r_{k-1} \times n \times r_{k+1} \times \cdots \times r_K \). The corresponding entry-wise definition is \( (\mathcal{W} \times_k U)_{i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_K} = \sum_{j=1}^{r_k} u_{i_1, \ldots, i_k, j, i_{k+1}, \ldots, i_K} \).

Tensor decomposition: There are two families of tensor decomposition, the Tucker family and the CP family. The Tucker family extends bilinear factorization models to handle tensor datasets. For an observed \( K \)-mode tensor \( \mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_K} \), the general form of Tucker decomposition is

\[
\mathcal{Y} = \mathcal{W} \times_1 U^{(1)} \times_2 \cdots \times_K U^{(K)}
\]

where \( \mathcal{W} \in \mathbb{R}^{r_1 \times \cdots \times r_K} \) is the core tensor, and \( U^{(k)} \in \mathbb{R}^{n_k \times r_k} \) are \( K \) latent factor matrices. As in Kolda and Bader [9], we collectively denote the group of \( K \) matrices as a Tucker tensor with a identity core \( \mathcal{U} = [U^{(1)}, \ldots, U^{(K)}] \)—this allows us to compactly represent the Tucker decomposition as \( \mathcal{Y} = \mathcal{W} \times \mathcal{U} \). The vector form of (1) is

\[
\text{vec}(\mathcal{W} \times \mathcal{U}) = U^{(1)} \otimes U^{(2)} \otimes \cdots \otimes U^{(K)} \cdot \text{vec}(\mathcal{W})
\]

The CP family is a restricted form of the Tucker family. The entry-wise definition of CP is \( y_{i_1, \ldots, i_K} = \sum_{l=1}^{r} \lambda_l u_{i_1, \ldots, i_k, l} \). The alternating least square (ALS) method has been used to solve both Tucker decomposition and CP [9].

3 Robust infinite Tucker decomposition by \( t \) processes

In this section we present the robust infinite Tucker decomposition based on latent \( t \) 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 \( t \) process prior, which makes the model robust to outliers.

\( ^1\)Unlike the usual column-wise vec-operation, our definition of \( \text{vec}(\cdot) \) on matrices is row-wise, which avoids the use of transpose in many equations throughout this paper.
More specifically, we assume the observed tensor $Y$ is sampled from a latent real-valued tensor $M$ via a probabilistic noise model $p(Y|M) = \prod_i p(y_i|m_i)$.

We conduct Tucker decomposition of $M$ with a core tensor $W$ of infinite size. To do so, we use a countably infinite feature mapping for the rows of the component matrices $U^{(k)} \in \mathbb{R}^{n_k \times r}$, $k = 1, \ldots, K$. Let $u^{(k)}_i$ denotes the $i$-th row of $U^{(k)}$. A feature mapping $\phi : \mathbb{R}^r \to \mathbb{R}^{n_k}$ maps each $u^{(k)}_i$ to the infinite feature space $\phi(u^{(k)}_i)$, where $\mathbb{N}_0$ denotes the countable infinity. The inner product of the feature mapping is denoted as $\Sigma_{ij} = \langle \phi(u^{(k)}_i), \phi(u^{(k)}_j) \rangle$. Let $\phi^r(u^{(k)}_i) = [\phi_1(u^{(k)}_i), \ldots, \phi_r(u^{(k)}_i)]$ denote the first $r$ coordinates of $\phi(u^{(k)}_i)$. $W \in \mathbb{R}^{n_k}$ denote an infinite $K$-mode core tensor, and $W^{(r)} = (w_k)_{k=1}^r \in \mathbb{R}^{r^K}$ denote the first $r$ dimensions in every mode of $W$. The infinite Tucker decomposition $\mathcal{M} = W \times \phi(U)$ for the latent tensor $M$ can be formally defined as the limit of a series of finite Tucker decompositions.

$$\mathcal{M} = \lim_{r \to \infty} W^{(r)} \times_1 \phi^{(r)}(U^{(1)}) \times_2 \ldots \times_K \phi^{(r)}(U^{(K)})$$

where $\phi^{(r)}(U^{(k)}) = [\phi^{(r)}(u^{(k)}_1) \ldots, \phi^{(r)}(u^{(k)}_{n_k})]^{T}$.

As shown in the next Section, we use a latent tensor-variate $t$ process prior on $W$ and then marginalize it out to obtain a $t$ process over $\mathcal{M}$—this makes the estimation of $\mathcal{M}$ to the presence of strong noise and outliers.

### 3.1 Tensor-variate $t$ processes

Before formally defining the tensor-variate $t$ process, we denote the domain of the mode $k$ by $U_k$, the $K$ covariance functions by $\Sigma^{(k)} : U_k \times U_k \to \mathbb{R}$, the covariance matrices by a Tucker tensor $S^{-\frac{1}{2}} = [(\Sigma^{(1)})^{\frac{1}{2}}, \ldots, (\Sigma^{(K)})^{\frac{1}{2}}]$ and $n = \prod_{k=1}^K n_k$.

The norm of the a tensor $\|A\|$ is defined as $\sqrt{\sum_i a_i^2}$. Then we define tensor-variate $t$ processes as follows.

**Definition 1 (Tensor-variate $t$ Distributions and Tensor-variate $t$ Processes)** Given $K$ location sets $U_k$, $k = 1, \ldots, K$, let $b : U_1 \times \ldots \times U_K \to \mathbb{R}$ be the mean function. $M = \{f(u^{(1)}_1, \ldots, u^{(K)}_k) | u^{(k)} \in U_k\}$ is a set of random tensor variables where $f : U_1 \times \ldots \times U_K \to \mathbb{R}$ is a random function. For any finite sets $\{u^{(1)}_1, \ldots, u^{(k)}_k\}_{k=1}^K$, let $M = \{f(u^{(1)}_1, \ldots, u^{(K)}_k) | j \in \mathbb{R}^{n_1 \times \ldots \times n_K}\}$, where $j_k = 1, \ldots, n_k$, be a random tensor and $B = [b(u^{(1)}_1, \ldots, u^{(K)}_k)]_{j \in \mathbb{R}^{n_1 \times \ldots \times n_K}}$ be the mean tensor. Let $\Gamma(x)$ be the Gamma function. The set $M$ follows a tensor-variate $t$ process $\mathcal{T}(\nu, \mu, \Sigma)$ with degree of freedom $\nu > 2$, if $M$ follows tensor $t$ distribution with the following density

$$\mathcal{T}(\mathcal{M} | \nu, \mu, \Sigma) = \Gamma(\frac{n+\nu}{2}) \prod_{k=1}^K \Sigma^{(k)}_k \frac{1}{\Gamma(\frac{\nu}{2}) \frac{1}{2}} \left(1 + \frac{1}{\nu} ||( \mathcal{M} - B) \times S^{-\frac{1}{2}} ||^2 \right)^{-\frac{n+\nu}{2}}$$

In this paper, we set the mean function to be zero, i.e. $B = 0$. Let $T(\nu, \mu, \Sigma)$ denotes $t$ distribution with mean $\mu$, covariance $\Sigma$ and degree of freedom $\nu$. If the latent tensor $\mathcal{M}$ is drawn from a tensor-variate $t$ process, then $vec(\mathcal{M}) \sim T(0, \Psi)$, where $\Psi =$. 

4
Theorem 2 Let $U_k \subset \mathbb{R}^r$, and $\Sigma_r^{(k)}(u_i^{(k)}, u_j^{(k)}) = \langle \phi(r)(u_i^{(k)}), \phi(r)(u_j^{(k)}) \rangle$ be a series of covariance functions. Define a multi-linear function by

$$t^{(r)}(u^{(1)}, \ldots, u^{(K)}) = \mathcal{W}^{(r)} \times_1 \phi^{(r)}(u^{(1)}) \times_K \phi^{(r)}(u^{(K)}), \ u^{(k)} \in U_k$$

(5)

If $\mathcal{W}^{(r)} \sim \mathcal{T}(\nu, 0, \{I_r\}_{k=1}^K)$, then $t^{(r)}(u^{(1)}, \ldots, u^{(K)})$ follows a tensor-variate $t$ distribution $\mathcal{T}_T \mathcal{P}(\nu, 0, \{\Sigma_r^{(k)}\}_{k=1}^K)$, and it converges to $\mathcal{T}_T \mathcal{P}(\nu, 0, \{\Sigma^{(k)}\}_{k=1}^K)$ in distribution as $r \to \infty$.

The proof can be found in the appendix. The above theorem shows that probabilistic infinite Tucker decomposition of $\mathcal{M}$ can be realized by modeling $\mathcal{M}$ as a draw from a tensor-variate $t$ process on the location vectors induced from the unknown component matrices $U^{(k)}$. Our definition of tensor-variate $t$ processes generalizes matrix-variate $t$ process defined in [25]. Theorem 2 also suggests a constructive definition of tensor-variate processes for general covariance functions.

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

### 3.2 Noise models

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

**Probit model:** In this case, each entry of the observation is binary; that is, $y_i \in \{0, 1\}$. A probit function $p(y_i|m_i) = \Phi(m_i) = \Phi(m_i)^{y_i}(1 - \Phi(m_i))^{1-y_i}$ models the binary observation. Note that $\Phi(\cdot)$ is the standard normal cumulative distribution function.

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

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

Other noise models include modified probit models for ordinal regression and multi-class classification [2], null category noise models for semi-supervised classification [12]. In this paper we focus on probit and Gaussian models.

### 4 Inference

Given the observed tensor $Y$, we aim to estimate the component matrices $U^{(k)}$ by maximizing the marginal likelihood $p(Y|\{U^{(k)}\}_{k=1}^K)p(\{U^{(k)}\}_{k=1}^K)$. Integrating out $\mathcal{M}$ in the above equation is intractable however. Therefore, we resort to approximate inference; more specifically, we develop a variational expectation maximization (EM) algorithm. In the following paragraphs, we first present the inference algorithms for both of the noise models, and then describe an efficient algebraic approach to significantly reduce the computation complexity.
4.1 Variational EM for binary classification

We follow the data augmentation scheme by Albert and Chib [2] to decompose the probit model into \( p(y_i|m_i) = \int p(y_i|z_i)p(z_i|m_i)dz_i \). Let \( \delta(\cdot) \) be the indicator function, we have

\[
p(y_i|z_i) = \delta(y_i = 1)\delta(z_i > 0) + \delta(y_i = 0)\delta(z_i \leq 0), \ p(z_i|m_i) = \mathcal{N}(z_i|m_i, 1) \tag{6}
\]

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

\[
\mathcal{T}_T(M|\nu, 0, \{\Sigma^{(k)}\}_{k=1}^K) = \int \mathcal{T}_N(M|0, \{\eta^{-1/\nu}\Sigma^{(k)}\}_{k=1}^K) \text{Gam}(\eta|\nu/2, \nu/2)d\eta \tag{7}
\]

\[
\mathcal{T}_N(M|B, \{\Sigma^{(k)}\}_{k=1}^K) = (2\pi)^{-\nu/2} \prod_{k=1}^K |\Sigma^{(k)}|^{-\nu/2} \exp \left(-\frac{1}{2} \| (M - B) \times \mathcal{S}^{-\frac{1}{2}} \|^2 \right) \tag{8}
\]

where \( \mathcal{T}_N \) denotes the tensor-variate normal distribution. The joint probability likelihood with data augmentation is

\[
p(Y, Z, M, \eta, U) = p(Y|Z)p(Z|M)p(M|\eta, U)p(\eta)p(U). \tag{9}
\]

where \( p(M|\eta, U) \) and \( p(\eta) \) is the tensor-variate normal distribution and the Gamma distribution in (7). \( p(U) \) 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 \( p(Z, M, \eta|Y, U) \) by a fully factorized distribution \( q(Z, M, \eta) = q(Z)q(M)q(\eta) \). Variational inference minimizes the Kullback-Leibler (KL) divergence between the approximate posterior and the true posterior.

\[
\min_{q} \text{KL}(q(Z)q(M)q(\eta)||p(Z, M, \eta|Y, U)). \tag{10}
\]

The variational approach optimizes one approximate distribution, e.g., \( q(Z) \), in (10) at a time, while having all the other approximate distributions fixed [3]. We loop over \( q(Z) \), \( q(M) \) and \( q(\eta) \) to iteratively optimize the KL divergence until convergence.

Given \( q(M) \) and \( q(\eta) \), the \( q(z_i) \) is a truncated normal distribution

\[
q(z_i) \propto \mathcal{N}(\mathbb{E}_q[m_i], 1)\delta(z_i > 1), \ \mathbb{E}_q[z_i] = \mathbb{E}_q[m_i] + \frac{(2y_i - 1)\mathbb{E}_q[m_i]|0, 1)}{\Phi((2y_i - 1)\mathbb{E}_q[m_i])}. \tag{11}
\]

Given \( q(Z) \) and \( q(\eta) \), it is more convenient to write the optimized approximate distribution for \( M \) in its vectorized form. Let \( \Sigma_p = \Sigma^{(1)} \otimes \ldots \otimes \Sigma^{(K)} \), we have

\[
q(\text{vec}(M)) = \mathcal{N}(\text{vec}(M)|\mu, \Psi), \ \mu = \text{vec}(\mathbb{E}_q[M]) = \Psi \text{ vec}(\mathbb{E}_q[Z]) \tag{12}
\]

\[
\Psi = \mathbb{E}_q[\eta]^{-1} \Sigma_p \left( I + \mathbb{E}_q[\eta]^{-1} \Sigma_p \right)^{-1}. \tag{13}
\]
The optimized $q(\eta)$ is also a Gamma distribution:

$$q(\eta) = \text{Gam}(\eta | \beta_1, \beta_2), \quad \mathbb{E}_q[\eta] = \frac{\beta_1}{\beta_2}, \quad \beta_1 = \frac{\nu + n}{2}, \quad \beta_2 = \frac{\nu + \mu^T \Sigma_p^{-1} \mu + \text{tr}(\Sigma_p^{-1} \Upsilon)}{2}.$$  

(14)

Based on the variational approximate distribution obtained in the E-step, we maximize the expected log likelihood over $\mathcal{U} = [\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(K)}]$ in the M-step.

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

(15)

After eliminating constant terms, we need to solve the following optimization problem

$$\min_{\mathcal{U}} f(\mathcal{U}) = \sum_{k=1}^{K} \frac{n}{n_k} \log |\Sigma^{(k)}| + \tau \|\mathbb{E}_q[\mathcal{M}] \times S^{-1/2}\|^2 + \tau \text{tr}(\Sigma_p^{-1} \Upsilon) + \lambda \sum_{k=1}^{K} \|\mathbf{U}_k\|_1,$$

(16)

where $\tau = \mathbb{E}_q[\eta]$. In the above equation (16), $\Sigma^{(k)} = \Sigma^{(k)}(\mathbf{U}^{(k)}, \mathbf{U}^{(k)})$ is considered as a function of $\mathbf{U}_k$, and $S^{-1/2}$ is a function of $\mathcal{U}$. $\Upsilon$ and $\tau$ are the statistics computed in the E-step, and they have fixed values. Omitting the last $\ell_1$ term in (16), the gradient of $f(\mathcal{U})$ w.r.t. to a scalar $u^{(k)}_{ij}$ is given in the appendix.

With an $\ell_1$ penalty on $f(\mathcal{U})$, we choose a projected scaled subgradient L-BFGS algorithm for optimization—due to its excellent performance [16].

### 4.2 Regression

**Inference:** The inference for the regression case follows the same format as the binary classification case. The only changes are: 1) replacing $\mathbb{E}_q[\mathbf{Z}]$ by $\mathbf{Y}$ and skipping updating $q(\mathbf{Z})$. 2) The variational EM algorithm are only applied to the observed entries.

**Prediction:** Given a missing value index $i = (i_1, \ldots, i_K)$, the predictive distribution is

$$p(y_i | \mathbf{Y}_O) \approx \int p(y_i | m_i) p(m_i | \mathcal{M}, \eta) q(\mathcal{M})q(\eta) d\mathcal{M} d\eta$$

The above integral is intractable, so we replace $\eta$ 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

$$\int p(y_i | m_i) p(m_i | \mathcal{M}, \tau^*) q(\mathcal{M}) d\mathcal{M}$$

$$= \mathcal{N}(y_i | \mathbf{k}^T (\Sigma_p + \sigma^2 \tau^* \mathbf{I})^{-1} \mathbf{vec}(\Upsilon), \frac{1}{\tau^*} [k(i, i) - \mathbf{k}^T (\Sigma_p + \sigma^2 \tau^* \mathbf{I})^{-1} \mathbf{k}])$$

(17)

where $k(i, j) = \prod_{k=1}^{K} \Sigma^{(k)}(u^{(k)}_{ik}, u^{(k)}_{jk})$ and $\mathbf{k} = [k(i, j)]_{i,j \in \mathcal{O}}$. 


4.3 Efficient Algorithms

A naïve implementation of the above algorithm requires prohibitive $O(\prod_{k=1}^{K} n_k^2)$ time complexity and $O(\prod_{k=1}^{K} n_k^2)$ space complexity for each EM iteration. The key computation bottlenecks are the operations involving $\Sigma$ defined in equation (13). To avoid this high complexity, we can make use of the Kronecker product structure. We assume $E_q[y] = 1$ to simplify the computation, it is easy to adapt our computation strategies to $E_q[y] \neq 1$. Let $\Sigma^{(k)} = V^{(k)} \Lambda^{(k)} V^{(k)\top}$ be the singular value decomposition of the covariance matrix $\Sigma^{(k)}$, $V^{(k)}$ is an orthogonal matrix and $\Lambda^{(k)}$ is a diagonal matrix. $\Sigma$ can be represented as

$$
\Sigma = \sum_{k=1}^{K} \Lambda^{(k)} V^{(k)\top} \otimes \ldots \otimes V^{(k)} \Lambda^{(k)} V^{(k)\top} + \sum_{k=1}^{K} \Lambda^{(k)} V^{(k)\top} \otimes \ldots \otimes V^{(k)} \Lambda^{(k)} V^{(k)\top}^{-1} = \sum_{k=1}^{K} \Lambda^{(k)} (I + \Lambda^{(k)})^{-1} V^{(k)\top} \otimes \ldots \otimes V^{(k)} \Lambda^{(k)} (I + \Lambda^{(k)})^{-1} V^{(k)\top}.
$$

Let $V = [V^{(1)}, \ldots, V^{(K)}]$, $A = \Lambda^{(1)} (I + \Lambda^{(1)})^{-1} \otimes \ldots \otimes \Lambda^{(K)} (I + \Lambda^{(K)})^{-1}$. It is obvious that $V$ is an orthogonal matrix and $A$ is a diagonal matrix. The above relation implies that we can actually compute the singular value decomposition of $\Sigma = AVA$ from covariance matrices $\Sigma^{(k)}$.

In order to efficiently compute $\text{tr}(\Sigma^{-1}_p \Sigma^{(k)})$ appearing in equation (16), we use the following relations

$$
\text{tr}(\Sigma^{-1}_p \Sigma^{(k)}) = \text{tr}(\Sigma^{-1}_p V^{\top} \Lambda V) = \text{tr}(\Lambda V \Sigma^{-1}_p V^{\top}) = \text{diag}(V \Sigma^{-1}_p V^{\top}) \text{ diag}(\Lambda) = d_1 \otimes \ldots \otimes d_K \text{ diag}(\Lambda) = d_1 \otimes \ldots \otimes d_K \text{ vec}(D) = D \times_1 d_1 \times_2 \ldots \times_K d_K,
$$

where $d_k = \text{diag}(V^{(k)} (\Sigma^{-1}_p)^{(k)} V^{(k)\top})$ with $\Sigma^{(k)}$ being a computed statistics in the E-step, $\text{ diag}(\Lambda)$ denotes the diagonal elements of $\Lambda$, and $D$ is a tensor of size $n_1 \times \ldots \times n_K$, such that $\text{ vec}(D) = \text{ diag}(\Lambda)$. Both time and space complexities of the last formula (18) is $O(\prod_{k=1}^{K} n_k)$.

We denote $V = [V^{(1)}, \ldots, V^{(K)}]$ and $V^{\top} = [V^{(1)\top}, \ldots, V^{(K)\top}]$. For any tensor $A$ of the same size as $D$, $\text{ vec}(A)$ means multiplying the $j$-th element of $\text{ vec}(A)$ by the $j$-th element of $\text{ vec}(D)$. So we have $\Lambda \text{ vec}(\Lambda) = \text{ vec}(D \otimes \Lambda)$, where $\otimes$ denotes the Hadamard product, i.e. entry-wise product. In light of this relation, we can efficiently compute (12) by

$$
\text{ vec}(E_q[Z]) = \text{ vec} \left[ \left( (E_q[Z] \times V^{\top}) \otimes D \right) \times V \right].
$$

The right-hand side of Equation (19) effectively reduce the time and space complexities of the left-hand side operations to $O(\sum_{k=1}^{K} n_k^2 + (\sum_{k=1}^{K} n_k) \prod_{k=1}^{K} n_k)$ and $O(\sum_{k=1}^{K} n_k^2 + \prod_{k=1}^{K} n_k)$, respectively.

We can further reduce the complexities by approximating the covariance matrices via truncated SVD. If the first $t_k$ leading eigenvalues and the corresponding eigenvectors are preserved, the time complexity of our algorithm is $O(\sum_{k=1}^{K} n_k t_k + (\sum_{k=1}^{K} n_k) \prod_{k=1}^{K} t_k)$, and the space complexity is $O(\sum_{k=1}^{K} n_k t_k + \prod_{k=1}^{K} t_k)$. 
Data | CP | TD | HOSVD | NCP | WCP | InfTucker
---|---|---|---|---|---|---

amm | 0.053±0.002 | 0.051±0.002 | 0.053±0.002 | 0.057±0.005 | 0.049±0.012 | **0.047±0.003**
flow | 0.051±0.005 | 0.051±0.003 | 0.052±0.004 | 0.110±0.073 | 0.079±0.035 | **0.046±0.002**
bread | 0.226±0.032 | 0.234±0.043 | 0.234±0.034 | 0.235±0.035 | **0.218±0.029** | **0.217±0.032**
suger | 0.173±0.004 | 0.169±0.004 | 0.167±0.004 | 0.177±0.004 | 0.138±0.003 | **0.071±0.003**

Table 1: The MSE values averaged over 50 runs. The lowest MSE value and those not significantly worse than it (achieved by t-test with 95% confidence level), are highlighted in the bold font.

5 Related Works

The InfTucker model extends Probabilistic PCA (PPCA) [20] and Gaussian process latent variable models (GPLVMs) [11]: 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 related to previous tensor-variate GPs Yu et al. [23], Yu and Chu [22]. 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.

Our model is closely related to the probabilistic Tucker-3 (pTucker) model [5]; actually InfTucker reduces to pTucker as a special case as $\nu \to \infty$. Also, Hoff [7] proposed a hierarchical Bayesian extension to CANDECOMP/PARAFAC that captures the interaction of component matrices. Unlike both [5]'s and Hoff [7]'s approach, ours handles non-Gaussian noise and uses nonlinear covariance functions to model complex interactions. In addition, [7] used a Gibbs sampler for inference—its convergence incurs high computational cost and makes this approach infeasible for tensors with moderate and large sizes.

To handle missing data, enhance model interpretability, and avoid overfitting, several extensions to non-probabilistic tensor decomposition have been proposed, including nonnegative tensor decomposition (NTD) [17, 8, 13, 15] and Weighted tensor decomposition (WTD) [1]. These models solve the core tensors explicitly and do not handle nonlinear multiway interactions.

Finally, note that the inference technique described in Section 4 can be adopted for Gaussian process or $t$-process multi-task learning [4, 25]. Let $M$ be the number of tasks and $N$ be the number of data points in each task. Our inference technique can be used to reduce their time complexity from $O(M^3 N^3)$ to $O(M^3 + N^3)$ and the space complexity from $O(M^2 N^2)$ to $O(N^2 + M^2)$.

6 Experiments

To evaluate the proposed InfTucker decomposition method, we conduct two batches of experiments: one for the setting with the Gaussian noise and the other for the probit noise. For both settings, we compare InfTucker with the following conventional and state-of-the-arts tensor decomposition methods: CP, Tucker decomposition (TD), Nonnegative CP (NCP), High Order SVD (HOSVD), and Weighted CP (WCP). All of
these comparison methods are implemented in the tensor toolbox\(^2\).

**Experiment on Gaussian Noise Model**

**Experimental setting:** We employ four continuous chemometrics datasets\(^3\), namely amino, bread, flow injection, and sugar. Their dimensions are \(5 \times 201 \times 61\), \(10 \times 11 \times 8\), \(12 \times 100 \times 59\), and \(268 \times 571 \times 7\), respectively. All the above datasets are normalized to have zero mean and unit variance. For each dataset, we randomly split all the tensor coordinates to 5 folds via cross-validation: each time one fold used for testing and the other four folds for training. This procedure is repeated for 10 times. Thus in total, for each dataset, all algorithms are repeated 50 times. In InfTucker, the degree of freedom \(\nu\) in the tensor-variate \(t\) process is fixed to 10. We employ the Gaussian/exponential covariance functions \(\Sigma^{(k)}(u_i, u_j) = e^{-\gamma\|u_i - u_j\|^t}\), where \(t = 1, 2\) and \(\gamma\) is selected from \([0.01 : 0.05 : 1]\) by 5-fold cross validation. The regularization parameter \(\lambda\) for InfTucker is chosen from \([1, 10, 100]\).

**Results:** Since for all these datasets we do not know the true latent factors, we use the prediction accuracy on hold out edges to compare all these models. For easy comparison of all algorithms, we set the same number of latent factors for all modes. We varied the number of the latent factors, \(r\), from 3 to 5 and computed the averaged MSE values and standard deviations for all algorithms. Due to the relatively low dimensionality of certain modes in the above mentioned tensors, we find that when \(r = 3\), the MSEs are all near optimal. Thus we only reported the MSEs for \(r = 3\) in Table 1, where the lowest MSE value and those not significantly worse than it (achieved by t-test with 95% confidence level) are highlighted. It is not hard to find that InfTucker achieves the lowest MSEs on all data. This may be due to the so-called “kernel trick” that the non-linear covariance functions provide more flexibility in modeling the interaction among all modes. Among the comparison methods, WCP ranks the second best, while HOSVD is the worst due to the strong constraint on the non-negativity of latent factors.

Figure 1: 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.

**Experiment on Probit Noise Model**

**Experimental setting:** We extracted three binary multi-way social network datasets, noted as Enron, Digg1, and Digg2, as our evaluation data. Enron is a relational dataset

\(^2\)http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/

\(^3\)Available from http://www.models.kvl.dk/datasets
describing the three-way relation: sender-receiver-email. This dataset, extracted from
the Enron email\textsuperscript{4}, has the dimensionality of 203 × 203 × 200 with 0.01% non-zero
elements. The Digg1 and Digg2 datasets are all extracted from a social news website
digg.com\textsuperscript{5}. Digg1 describes a three-way relation: news-keyword-topic, and Digg2
describes a four-way relation: user-news-keyword-topic. Digg1 has the dimensionality
of 581 × 124 × 48 with 0.024% non-zero elements, and Digg2 has the dimensionality
of 22 × 109 × 330 × 30 with 0.002% non-zero elements. All these datasets are very
sparse.

\textbf{Results:} We adopt similar settings to the Gaussian noise case except that \( r \) is varied
from \{3,5,8,10,15,20\} due to the larger dimension of each mode. Since the data are
binary, MSE cannot well reveal the existence of a missing link in social networks.
Therefore, we evaluate all the algorithms by Area Under Curve (AUC) values averaged
over 50 runs. The larger the AUC values, the better. In InfTucker, we use \( \mathbb{E}_q [\mathcal{M}] \) as
predictions to calculate the AUC values. We report the averaged AUC values for all
algorithms in Figure 1. It is observed that the proposed InfTucker model sufficiently
outperforms all the other models. The failure of the traditional models like CP and TD
is that the least square minimization used in these models easily leads to almost all zero
predictions since the tensor datasets are highly sparse. This experiment indicates the
necessity of probit noise models for binary data.

\section{Conclusion}

We proposed a new nonparametric Bayesian tensor decomposition framework, Inf-
Tucker, where the observation tensor is modeled as a draw from a tensor-variate \( t \)
process. This framework is able to flexibly model the interdependence among multi-
modes by conducting tensor decomposition in the infinite feature space. It can also deal
with several types of data, e.g. incomplete data and binary data, via various noise mod-
eling approaches, while most traditional methods rely on least square estimations. As a
significant contribution, we also proposed an efficient variational approach to estimate
the component matrices. Experimental results on chemometrics and social network
datasets demonstrate that the new InfTucker model achieves significantly higher pre-
diction accuracy than several state-of-art tensor decomposition approaches.

\textsuperscript{4}Available from http://www.cs.cmu.edu/~enron/
\textsuperscript{5}Available from http://www.public.asu.edu/~ylin56/kdd09sup.html
References


A Proof of Theorem 2

Proof Sketch If $\mathcal{W}(r) \sim TT(\nu, 0, \{I_r\}_{k=1}^K)$, then $\text{vec}(\mathcal{W}(r)) \sim TT(\nu, 0, I_rK)$. Let $U_k, k = 1, \ldots, K$ be $K$ location sets (matrices) as used in (3), we have

$$\text{vec} \left( \mathcal{W}(r) \times \phi(r)(U) \right) = \phi(r)(U(1)) \otimes \cdots \otimes \phi(r)(U(K)) \text{vec}(\mathcal{W}(r)) \quad (20)$$

Thus, $\text{vec} \left( \mathcal{W}(r) \times \phi(r)(U) \right) \sim TT(\nu, 0, \Sigma_1 \otimes \cdots \otimes \Sigma_K)$, where $\Sigma_k(i, j) = \Sigma_r(i, j)$ are the covariance matrix. Inverting (20) gives $\mathcal{W}(r) \times \phi(r)(U) \sim TT(0, \{\Sigma_k\}_{k=1}^K)$, which proves $f(r)$ follows the tensor Gaussian process.

From the definition of inner product in $\ell_2$, we have the following identity on the convergence of covariance function.

$$\Sigma_k(u_i^{(k)}, u_j^{(k)}) = \lim_{r \to \infty} \Sigma_r(u_i^{(k)}, u_j^{(k)}), \forall u_i^{(k)}, u_j^{(k)} \in U_k$$

Convergence in distribution follows from this convergence result.

B Gradient of $f(U)$

$$\frac{\partial f}{\partial u_{ij}^{(k)}} = \frac{n}{n_k} \text{tr} \left( (\Sigma^{(k)})^{-1} \frac{\partial \Sigma^{(k)}}{\partial u_{ij}^{(k)}} \right) + \tau \mu^T \Delta^{(k)} \mu + \tau \text{tr} \left( \Delta^{(k)} \Upsilon \right) \quad (21)$$

$$\Delta^{(k)} = (\Sigma^{(1)})^{-1} \otimes \cdots \otimes (\Sigma^{(k-1)})^{-1} \otimes (\Sigma^{(k)})^{-1} \frac{\partial \Sigma^{(k)}}{\partial u_{ij}^{(k)}} (\Sigma^{(k)})^{-1}$$

$$\otimes (\Sigma^{(k+1)})^{-1} \otimes \cdots \otimes (\Sigma^{(K)})^{-1}$$
C Original figures

Figure 2: The AUC values of six algorithms on Enron. The dimensions of the latent factors are $d = 3, 5, 8, 10, 15, 20$ respectively. InfTucker outperforms the other algorithms significantly on all datasets.
Figure 3: The AUC values of six algorithms on digg1. The dimensions of the latent factors are $d = 3, 5, 8, 10, 15, 20$ respectively. *InfTucker* outperforms the other algorithms significantly on all datasets.
Figure 4: The AUC values of six algorithms on digg2. The dimensions of the latent factors are $d = 3, 5, 8, 10, 15, 20$ respectively. \textit{InfTucker} outperforms the other algorithms significantly on all datasets.