Infinite Tucker Decomposition: Nonparametric Bayesian Models for Multiway Data Analysis

Zenglin Xu, Feng Yan, and Yuan (Alan) Qi

Presented by Eunsu Ryu
July 6, 2012
Introduction

• Many real-world datasets can be described by tensors
• Traditional multi-way factor models (Tucker, 1966; Harshman 1970) are unable to capture:
  • Coupled, nonlinear interactions between the entities
  • Noisy observations
  • Data containing missing values
  • Data with values other than real numbers (e.g. binary)
• Contributions
  • Nonparametric Bayesian multiway analysis model
  • Efficient inference algorithm
Tensor Algebra

- **Vectorization:** given $\mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_K}$, stack the tensor entries into a vector $\text{vec}(\mathbf{y}) \in \mathbb{R}^n$, where $n = \prod n_k$.

  $$y(i_1, \ldots, i_K) = [\text{vec}(\mathbf{y})]_j, \quad j = i_K + \sum_{k=1}^{K-1} (i_k - 1) \prod_{l=k+1}^{K} n_l.$$ 

- **Mode-\(k\) tensor-matrix multiplication:** given $\mathbf{W} \in \mathbb{R}^{r_1 \times \cdots \times r_K}$, and $\mathbf{U} \in \mathbb{R}^{s \times r_k}$, define $\mathbf{W} \times_k \mathbf{U} \in \mathbb{R}^{r_1 \times \cdots \times r_{k-1} \times s \times r_{k+1} \times \cdots \times r_K}$ s.t.

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

- Diagrams illustrating the operations.
Tucker Decomposition

**Definition:** The *Tucker decomposition* for $\mathcal{M} \in \mathbb{R}^{n_1 \times \cdots \times n_K}$ is

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

where $\mathcal{W} \in \mathbb{R}^{r_1 \times \cdots \times r_K}$ is the *core tensor*, and $U^{(k)} \in \mathbb{R}^{n_k \times r_k}$ is the mode-$k$ latent *factor matrix*

- **Remarks**
  - $\text{vec}(\mathcal{W}; U^{(1)}, \ldots, U^{(K)}) = U^{(1)} \otimes \cdots \otimes U^{(K)} \cdot \text{vec}(\mathcal{W})$

- **Infinite Tucker Decomposition**
  - Infinite-dimensional generalization of the Tucker decomposition with an infinite core tensor $\mathcal{W}$. 
Proposed Model

• The data $\mathbf{y}$ is sampled from a latent tensor $\mathcal{M}$ via
  \[
p(\mathbf{y}|\mathcal{M}) = \prod_i p(y_i|m_i)
  \]
  where $i = (i_1, ..., i_K)$.
• $\mathcal{M}$ has the Tucker decomposition
  \[
  \mathcal{M} = \mathcal{W} \times_1 \mathbf{U}^{(1)} \times_2 \ldots \times_K \mathbf{U}^{(K)}
  \]
  where the core tensor $\mathcal{W}$ can have infinite size.
• Tensor-variate Gaussian or t process prior on $\mathcal{M}$ (next slide).
• Laplace prior on the $\mathbf{U}^{(k)}$’s: $u_{i_k}^{(k)} \sim \mathcal{L}(\lambda) \propto \exp(-\lambda \|u_{i_k}^{(k)}\|_1)$.
• Flexible noise models:
  • Probit model: $p(y_i|m_i) = \Phi(m_i)^{y_i}(1 - \Phi(m_i))^{1-y_i}$, $y_i \in \{0, 1\}$.
  • Gaussian model: $p(y_i|m_i) = \mathcal{N}(y_i|m_i, \sigma^2)$
  • Missing values: $p(\mathbf{y}\_\emptyset|\mathcal{M}\_\emptyset)$, where $\emptyset$ denotes the indices of the observed entries in $\mathbf{y}$. 

Tensor-variate Gaussian Process

- **Prior on \( \mathcal{W} \):** let \( r_1 = \cdots = r_K = r \) and assign \( \text{vec}(\mathcal{W}) \sim \mathcal{N}(0, I) \). Then

\[
\text{vec}(\mathcal{M}) \sim \mathcal{N}(0, \Sigma^{(1)} \otimes \cdots \otimes \Sigma^{(K)}),
\]

where \( \Sigma^{(k)} = U^{(k)} U^{(k)T} \) is the covariance along the \( k \)-th mode, and

\[
\text{cov}(m_{i_1, \ldots, i_K}, m_{j_1, \ldots, j_K}) = \prod_{k=1}^{K} \Sigma_{i_k j_k}^{(k)}.
\]

- **Tensor-variate Gaussian distribution:** tensor \( \mathcal{M} \) follows a tensor-variate normal distribution defined by:

\[
\mathcal{TN}(\mathcal{M}; 0, \Sigma^{(1)}, \ldots, \Sigma^{(K)}) = \frac{\exp \left\{ -\frac{1}{2} \| [\mathcal{M}; (\Sigma^{(1)})^{-\frac{1}{2}}, \ldots, (\Sigma^{(K)})^{-\frac{1}{2}}] \|^2 \right\}}{(2\pi)^{n/2} \prod_{k=1}^{K} |\Sigma^{(k)}|^{-\frac{n}{2n_k}}}.
\]

where \( n = \prod n_k \), and \( \| \mathcal{X} \| = \sqrt{\sum_i x_i^2} \).
Tensor-variate Gaussian Process

- Modeling nonlinear relationships: replace each row $u_i^{(k)}$ of $U^{(k)}$ by a nonlinear feature mapping $\phi(u_i^{(k)})$ to obtain

$$
\Sigma_{ij}^{(k)} = k(u_i^{(k)}, u_j^{(k)}) = \langle \phi(u_i^{(k)}), \phi(u_j^{(k)}) \rangle
$$

where $\phi(u_i^{(k)})$ can have infinite dimensions.

**Definition:** A tensor-variate Gaussian process is a collection of random variables $\{m(u_1^{(1)}, \ldots, u_K^{(K)})\}$, $u_i^{(k)} \in \mathbb{R}^r$ whose finite joint probability over $\{m(u_1^{(1)}, \ldots, u_i^{(K)})\}$ follows the tensor-variate Gaussian distribution. Specifically, given $U^{(k)}$,

$$
\mathcal{M} \sim TGP(0, k(\cdot, \cdot)) \Rightarrow p(\mathcal{M} | U^{(1)}, \ldots, U^{(K)}) = \mathcal{N}(\mathcal{M}; 0, \Sigma^{(1)}, \ldots, \Sigma^{(K)})
$$

with the covariance matrix $\Sigma^{(k)} = k(U^k, U^k)$. 
Tensor-variate Gaussian Process

• Remarks:
  • TGP is equivalent to defining Tucker decomposition with infinite feature mapping $\phi(u)$, and an infinite core tensor $\mathcal{W}_\infty$.
    $$\mathcal{M} = [\mathcal{W}_\infty; \phi(U^{(1)}), \ldots, \phi(U^{(K)})]$$
  • Each element of $\mathcal{W}_\infty$ is a standard normal random variable.
  • Infinite tucker decomposition of the tensor $\mathcal{M}$ is a draw from a tensor-variate Gaussian process.
  • TGP is a generalization of the matrix-variate Gaussian process defined in (Yu et al., 2007) and (Yan et al., 2011)
Tensor-variate T Process

**Definition:** $\mathcal{M}$ follows a *tensor-variate t process* $\mathcal{T}\mathcal{T}\mathcal{P}(\nu, k(\cdot, \cdot))$ with a degree of freedom $\nu > 2$ if $\mathcal{M}$ follows the tensor-variate t distribution with the density:

$$
\mathcal{T}\mathcal{T}(\mathcal{M}; \nu, 0, \{\Sigma^{(k)}\}_{k=1}^K) = \frac{\Gamma\left(\frac{n+\nu}{2}\right) \prod_{k=1}^K |\Sigma^{(k)}|^{-\frac{n}{2\nu}}}{\Gamma\left(\frac{\nu}{2}\right)(\nu\pi)^{\frac{n}{2}} \left(1 + \frac{1}{\nu}\|\mathcal{M}; (\Sigma^{(1)})^{-\frac{1}{2}}, \ldots, (\Sigma^{(K)})^{-\frac{1}{2}}\|^2\right)^{-\frac{1}{2}(n+\nu)}}
$$

- **Remark**
  - $\mathcal{T}\mathcal{T}(\mathcal{M}; \nu, 0, \{\Sigma^{(k)}\}_{k=1}^K) = \int \text{Gam}(\eta; \nu/2, \nu/2) \mathcal{T}\mathcal{N}(\mathcal{M}, 0, \{\eta^{-1/K}\Sigma^{(k)}\}_{k=1}^K) d\eta$
Inference

• Given the observation \( \mathbf{y} \in \mathbb{R}^{n_1 \times \cdots \times n_K} \), want the MAP estimate of the \( U^{(k)} \)'s by maximizing \( p(\mathbf{y}|\{U^{(k)}\}_{k=1}^K)p(\{U^{(k)}\}_{k=1}^K) \).

• The paper only describes the inference for \( \mathcal{TTP} \) prior on \( \mathcal{M} \).

• Probit noise
  
  • Augmentation: \( p(y_i|m_i) = \int p(y_i|z_i)p(z_i|m_i)dz_i \), where \( p(z_i|m_i) = \mathcal{N}(z_i|m_i, 1) \), and \( p(y_i|z_i) = \delta(y_i = 1)\delta(z_i > 0) + \delta(y_i = 0)\delta(z_i \leq 0) \).

  • Joint likelihood: \( p(\mathbf{y}, \mathbf{z}, \mathcal{M}, \eta, \mathcal{U}) = p(\mathbf{y}|\mathbf{z})p(\mathbf{z}|\mathcal{M})p(\mathcal{M}|\eta, \mathcal{U})p(\eta)p(\mathcal{U}) \), where \( \mathcal{U} = [\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(K)}] \), \( \mathcal{M} \sim \mathcal{TNN}(0, \{\eta^{-1/K}\Sigma^{(k)}\}_{k=1}^K) \), and \( \eta \sim \text{Gam}(\nu/2, \nu/2) \).

  • E-step: compute \( q(\mathbf{z}, \mathcal{M}, \eta) = q(\mathbf{z})q(\mathcal{M})q(\eta) \) that minimizes \( \text{KL}(q(\mathbf{z}, \mathcal{M}, \eta)||p(\mathbf{z}, \mathcal{M}, \eta|\mathbf{y}, \mathcal{U})) \).
Inference (Probit noise)

- **E-step (continued)**
  - $q(z_i) \propto \mathcal{N}(E_q[m_i], 1)\delta(z_i > 1)$, where $E_q[z_i] = E_q[m_i] + \frac{(2y_i - 1)\mathcal{N}(E_q[m_i]|0, 1)}{\Phi((2y_i - 1)E_q[m_i])}$.
  - $q(\text{vec}(\mathcal{M})) = \mathcal{N}(\text{vec}(\mathcal{M})|\mu, \Psi)$, where $\mu = \text{vec}(E_q[\mathcal{M}]) = \Psi \text{vec}(E_q[Z])$
    $$\Psi = E_q[\eta]^{-1} \Sigma_p \left( I + E_q[\eta]^{-1} \Sigma_p \right)^{-1},$$
    and $\Sigma_p = \Sigma^{(1)} \otimes \ldots \otimes \Sigma^{(K)}$.
  - $q(\eta) = \text{Gam}(\eta; \beta_1, \beta_2)$, $E_q[\eta] = \frac{\beta_1}{\beta_2}$, $\beta_1 = \frac{\nu + n}{2}$, and $\beta_2 = \frac{\nu + \mu^T \Sigma_p^{-1} \mu + \text{tr}(\Sigma_p^{-1} \Psi)}{2}$.

- **M-step**
  - Maximize the expected log-likelihood: $\max_{\mathcal{U}} E_q[\log p(\mathcal{Y}, Z, \mathcal{M}, \eta|\mathcal{U}) p(\mathcal{U})]$. 
  - After eliminating constant terms, need to solve
    $$\min_{\mathcal{U}} f(\mathcal{U}) = \tau \left\| \left[ E_q[\mathcal{M}] ; (\Sigma^{(1)})^{-\frac{1}{2}}, \ldots, (\Sigma^{(K)})^{-\frac{1}{2}} \right] \right\|^2$$
    $$+ \sum_{k=1}^{K} \frac{n}{n_k} \log |\Sigma^{(k)}| + \tau \text{tr} \left( \Sigma_p^{-1} \Psi \right),$$
    where $\tau = E_q[\eta]$, and $\Sigma^{(k)} = k(U^{(k)}, U^{(k)})$ is a function of $U^{(k)}$. 

Inference (Probit noise)

• M-step (continued)

  • Gradient descent:

  \[
  \frac{\partial f}{\partial u_{ij}^{(k)}} = \frac{n}{n_k} \text{tr} \left( (\Sigma^{(k)})^{-1} \frac{\partial \Sigma^{(k)}}{\partial u_{ij}^{(k)}} \right) + \tau \mu^T \Delta^{(k)} \mu + \tau \text{tr} \left( \Delta^{(k)} \Psi \right)
  \]

  \[
  \Delta^{(k)} = (\Sigma^{(1)})^{-1} \otimes \ldots \otimes (\Sigma^{(k-1)})^{-1} \otimes (\Sigma^{(k)})^{-1} \otimes (\Sigma^{(k+1)})^{-1} \otimes \ldots \otimes (\Sigma^{(K)})^{-1}
  \]

• With an $\ell_1$ penalty on $f(\mathcal{U})$, optimize using a projected scaled subgradient L-BFGS algorithm (Schmidt, 2010).
Inference

- Gaussian noise
  - Follows the same format as the probit noise case.
  - $\mathbb{E}_q[Z]$ is replaced by $Y_\odot$.
  - No need to update $q(Z)$.
  - The variational EM algorithm is applied only to the observed entries $M_\odot$ and $[\Sigma_p]_\odot$. 
Prediction

- **Probit noise**
  - Given a missing value index \( i = (i_1, ..., i_K) \), the predictive distribution (intractable) is
    \[
    p(y_i | \mathcal{Y}) \approx \int p(y_i | m_i)p(m_i | M, \eta)q(\mathcal{M})q(\eta)d\mathcal{M}d\eta
    \]
    
  - To make this tractable, replace \( q(\eta)d\eta \) with the mode of its approximate posterior \( \tau^* = (\beta_1 - 1)/\beta_2 \) to obtain
    \[
    \int p(y_i = 1 | z_i)p(z_i | m_i)p(m_i | M, \tau^*)q(\mathcal{M})d\mathcal{M}dz_idm_idM
    = \int \delta(z_i > 0)\mathcal{N}(z_i | \mu_i(1), \nu_i^2(1))dz_i = \Phi(\frac{\mu_i(1)}{\nu_i(1)}) \tag{24}
    \]
    where
    \[
    k(i, j) = \prod_{k=1}^{K} \Sigma^{(k)}(u_{ik}^{(k)}, u_{jk}^{(k)}), \quad k = [k(i, j)]_{j \in \Theta}^T
    \]
    \[
    \mu_i(\rho) = k^T(\Sigma_p + \rho^2 \tau^* I)^{-1}\text{vec} (\mathcal{Y})
    \]
    \[
    \nu_i^2(\rho) = 1 + \frac{1}{\tau^*}[k(i, i) - k^T(\Sigma_p + \rho^2 \tau^* I)^{-1}k]
    \]

- **Gaussian noise**: \( p(y_i | \mathcal{Y}_\oplus) = \mathcal{N}(\mu_i(\sigma), \nu_i(\sigma)) \).
Efficient Algorithm

• Above optimization has $O(n^3)$ and $O(n^2)$ time and space complexities for each EM iteration ($n = \prod n_k$).
• Bottlenecks: $\text{tr}(\Sigma_p^{-1} \mathbf{Y})$, $\text{tr}(\Delta^{(k)} \mathbf{Y})$, and $\mathbf{Y} \text{vec}(\mathbb{E}_q[\mathcal{Z}])$.
• For efficient computation, let $\Sigma^{(k)} = V^{(k)} \Lambda^{(k)} V^{(k)T}$ and $\mathbb{E}_q[\eta] = 1$. ($\mathbb{E}_q[\eta] \neq 1$ case is a straightforward extension).
• Above optimization has $O(n^3)$ and $O(n^2)$ time and space complexities for each EM iteration.
• Represent $\mathbf{Y}$ as

$$\mathbf{Y} = \Sigma_p (I + \Sigma_p)^{-1} = V \Lambda V^T$$

where

$$V = V^{(1)} \otimes \ldots \otimes V^{(K)}$$

$$\Lambda = \Lambda^{(1)} \otimes \ldots \otimes \Lambda^{(K)} (I + \Lambda^{(1)} \otimes \ldots \otimes \Lambda^{(K)})^{-1}$$

• Can now compute the eigendecomposition of $\mathbf{Y}$ from $\Sigma^{(k)}$. 
Efficient algorithm

- Compute $\text{tr}(\Sigma_p^{-1} \mathbf{Y})$ and $\text{tr}(\Delta^{(k)} \mathbf{Y})$
  
  $$\text{tr}(\Sigma_p^{-1} \mathbf{Y}) = \text{tr}(\Sigma_p^{-1} \mathbf{V}^T \Lambda \mathbf{V}) = \text{tr}(\Lambda \mathbf{V} \Sigma_p^{-1} \mathbf{V}^T) = \text{diag}(\mathbf{V} \Sigma_p^{-1} \mathbf{V}^T) \text{diag}(\Lambda) = \mathbf{d}_1 \otimes \ldots \otimes \mathbf{d}_K \text{diag}(\Lambda) = \mathbf{d}_1 \otimes \ldots \otimes \mathbf{d}_K \text{vec} (\mathcal{D}) = \mathcal{D} \times_1 \mathbf{d}_1 \ldots \times_K \mathbf{d}_K,$$

  where $\mathbf{d}_k = \text{diag}(\mathbf{V}^{(k)} (\Sigma^{(k)})^{-1} \mathbf{V}^{(k)^T})^T$.

- If we take one gradient descent step for each M step, then
  $$\mathbf{d}_k = \text{diag} \left( (\Delta^{(k)})^{-1} \right)^T$$

- $O(n)$ complexity in both time and space.

- Similar technique can be used to compute $\text{tr}(\Delta^{(k)} \mathbf{Y})$.

- Compute $\mathbf{Y} \text{vec}(\mathbb{E}_q [\mathcal{Z}])$
  
  $$\mathbf{Y} \text{vec}(\mathbb{E}_q [\mathcal{Z}]) = \mathbf{V} \Lambda \mathbf{V}^T \text{vec}(\mathbb{E}_q [\mathcal{Z}]) = \mathbf{V} \Lambda \text{vec} \left( [\mathbb{E}_q [\mathcal{Z}] ; \mathbf{V}^{(1)^T}, \ldots, \mathbf{V}^{(K)^T}] \right) = \mathbf{V} \text{vec}(\mathcal{G} \odot \mathcal{D})$$

  $$= \text{vec}([\mathcal{G} \odot \mathcal{D} ; \mathbf{V}^{(1)^T}, \ldots, \mathbf{V}^{(K)^T}]).$$

  $$\mathcal{G} = [\mathbb{E}_q [\mathcal{Z}] ; \mathbf{V}^{(1)^T}, \ldots, \mathbf{V}^{(K)^T}].$$

  - Time complexity: $O(\sum_{k=1}^K n_k^3 + n_k n) = O(n^{1+1/K})$ for a uniform tensor

  - Space complexity: $O(n + \sum_{k=1}^K n_k^2) = O(n)$ for a uniform tensor
Results on continuous tensor data

• Setting: 3 continuous chemometrics datasets
  • *amino* (5 × 51 × 201), *bread* (10 × 11 × 8), *flow injection* (12 × 100 × 89)
  • Random split via 5-fold cross validation (4 for training; 1 for testing). 10 different partitions generated for each dataset.
  • \( r = 3, \; \nu = 10, \; k(u^{(k)}_{ik}, u^{(k)}_{jk}) = \exp(-\gamma\|u^{(k)}_{ik} - u^{(k)}_{jk}\|_t), \) where \( t = 1, 2 \) and \( \gamma \) is selected from \([0.01: 0.05: 1]\) by 5-fold cross validation. \( \lambda \) is chosen from \( \{1, 10, 100\} \).
• Results (MSE ± SE)

<table>
<thead>
<tr>
<th>Data</th>
<th>amino</th>
<th>flow injection</th>
<th>bread</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>0.053±0.002</td>
<td>0.051±0.005</td>
<td>0.238±0.001</td>
</tr>
<tr>
<td>TD</td>
<td>0.054±0.002</td>
<td>0.051±0.003</td>
<td>0.248±0.001</td>
</tr>
<tr>
<td>HOSVD</td>
<td>0.053±0.002</td>
<td>0.052±0.004</td>
<td>0.259±0.001</td>
</tr>
<tr>
<td>NCP</td>
<td>0.057±0.005</td>
<td>0.110±0.023</td>
<td>0.233±0.001</td>
</tr>
<tr>
<td>PTD</td>
<td>0.054±0.002</td>
<td>0.048±0.002</td>
<td>0.240±0.001</td>
</tr>
<tr>
<td>WCP</td>
<td>0.049±0.004</td>
<td>0.079±0.011</td>
<td>0.246±0.003</td>
</tr>
<tr>
<td>InfTucker(^{ap})</td>
<td><strong>0.047±0.003</strong></td>
<td><strong>0.049±0.002</strong></td>
<td><strong>0.232±0.001</strong></td>
</tr>
<tr>
<td>InfTucker(^{tp})</td>
<td><strong>0.047±0.003</strong></td>
<td><strong>0.046±0.002</strong></td>
<td><strong>0.225±0.001</strong></td>
</tr>
</tbody>
</table>
Results on binary tensor data

• Setting: 3 (sparse) binary social network datasets
  • *Enron* (203 × 203 × 200; 0.01% nonzero),
  • *Digg1* (581 × 124 × 48; 0.024% nonzero)
  • *Digg2* (22 × 109 × 330 × 30; 0.002% nonzero)
• Choose *r* from \{3, 5, 8, 10, 15, 20\} based on cross-validation
• Evaluate all these approaches by area-under-curve (AUC) values averaged over 50 runs (larger AUC ⇒ better).

• Results

![Graphs showing AUC values for Enron, Digg1, and Digg2 datasets](image-url)