Bayesian Tensor on Tensor Regression

Kunbo Wang & Yanxun Xu*

Department of Applied Mathematics and Statistics, Johns Hopkins University

Contact Information:
Department of Applied Mathematics and Statistics
Johns Hopkins University
3400 North Charles Street, Baltimore, MD, 21218

Email: kwang45@jhu.edu
yanxun.xu@jhu.edu

Overview

We propose a **Bayesian framework of regression model** to predict a multidimensional array (tensor) of arbitrary dimensions from another multidimensional array of arbitrary dimensions. The framework is based on the contracted product of tensors, and **Tucker decomposition** of regression coefficient tensor. Proper prior distributions are given to factor matrices of Tucker decomposition as well as core tensor, resulting in full posterior conditional distributions given in closed form formulas. Metropolis-Hastings method is used to choose the dimensions of core tensor. Posterior predictive distribution is given via Gibbs sampling method. A fast computing method is also given to speed up computation.

Model Framework

The Bayesian Tensor on Tensor Regression Model:

$$\mathbb{Y} = \langle \mathbb{X}, \mathbb{B} \rangle_L + \mathbb{E} \tag{1}$$

Tensor Predictor, Coefficient and Response:
$\mathbb{X} \in \mathbb{R}^{N \times P_1 \times \cdots \times P_L}$, $\mathbb{B} \in \mathbb{R}^{P_1 \times \cdots \times P_L \times Q_1 \times \cdots \times Q_M}$, and $\mathbb{Y} \in \mathbb{R}^{N \times Q_1 \times \cdots \times Q_M}$
Tensor Error: $\mathbb{E} \in \mathbb{R}^{N \times Q_1 \times \cdots \times Q_M}$ with each element mean zero random variable.
Contracted Tensor Product $\langle \mathbb{X}, \mathbb{B} \rangle_L$:

$$\hat{\mathbb{Y}} = \langle \mathbb{X}, \mathbb{B} \rangle_L \in \mathbb{R}^{N \times Q_1 \times \cdots \times Q_M}$$

with

$$\hat{\mathbb{Y}}_{[n,q_1,\cdots,q_M]} = \sum_{p_1=1}^{P_1} \cdots \sum_{p_L=1}^{P_L} \mathbb{X}_{[n,p_1,\cdots,p_L]} \mathbb{B}_{[p_1,\cdots,p_L,q_1,\cdots,q_M]}$$

Tucker Decomposition of Coefficient Tensor $\mathbb{B}$:

$$\mathbb{B} = [\![\mathbb{G}; \mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(L)}, \mathbf{V}^{(1)}, \cdots, \mathbf{V}^{(M)}]\!] \tag{2}$$

$\mathbf{U}^{(l)} \in \mathbb{R}^{P_l \times R_l}$, factor matrix related to mode l in $\mathbb{B}$ and $\mathbb{X}$
$\mathbf{V}^{(m)} \in \mathbb{R}^{Q_m \times S_m}$, factor matrix related to mode m in $\mathbb{B}$, and $\mathbb{Y}$
$\mathbb{G}$, core tensor of size $R_1 \times \cdots \times R_L \times S_1 \times \cdots \times S_M$, stored structure information.

Prior Information on Core Tensor and Factor Matrices:

$$\begin{aligned} vec\, \mathbf{U}^{(l)} &\sim N(\boldsymbol{\mu}_{U_l}, \boldsymbol{\Sigma}_{U_l}) \\ vec\, \mathbf{V}^{(m)} &\sim N(\boldsymbol{\mu}_{V_m}, \boldsymbol{\Sigma}_{V_m}) \\ vec\, \mathbb{G} &\sim N(\boldsymbol{\mu}_G, \boldsymbol{\Sigma}_G) \\ \mathbb{E}_{(1)} &\sim N(\mathbf{0}, \sigma^2 \boldsymbol{\Sigma}_E), \quad \sigma^2 \sim IG(\alpha, \beta) \end{aligned}$$

Given normal prior on all the factor matrices and core tensor, full posterior can be shown in closed form.

Posterior Inference

Posterior inference of $\mathbf{U}^{(l)}$:

$$\mathbb{B}_{(-)} := \mathbb{G} \times_2 \mathbf{U}^{(2)} \cdots \times_L \mathbf{U}^{(L)} \times_{L+1} \mathbf{V}^{(1)} \cdots \times_{L+M} \mathbf{V}^{(M)}$$
$$\mathbb{C} := \langle \mathbb{B}_{(-)}, \mathbb{X} \rangle_{P_2,\cdots,P_N},\ \mathbb{C}_{(\mathcal{R}\times\mathcal{C})} \in \mathbb{R}^{N \prod_{m=1}^{M} Q_m \times R_1 P_1} \text{ matricization of } \mathbb{C}$$

$$vec\mathbb{Y} = \mathbb{C}_{(\mathcal{R}\times\mathcal{C})} \times\ vec\mathbf{U}^{(1)} + vec\mathbb{E} \tag{3}$$

Then the posterior distribution of $\mathbf{U}^{(l)}$ is also multivariate normal.

Posterior inference of $\mathbf{V}^{(m)}$:

$$\mathbb{B}_{(-)} := \mathbb{G} \times_1 \mathbf{U}^{(1)} \cdots \times_L \mathbf{U}^{(L)} \times_{L+2} \mathbf{V}^{(2)} \cdots \times_{L+M} \mathbf{V}^{(M)}$$
$$\mathbb{D} := \langle \mathbb{B}_{(-)}, \mathbb{X} \rangle_{P_1,\cdots,P_L},\ \mathbb{D}_{(\mathcal{R}\times\mathcal{C})} \in \mathbb{R}^{N \prod_{m=2}^{M} Q_m \times S_1} \text{ matricization of } \mathbb{D}$$

$$\mathbb{Y}_{(2)} = \mathbf{V}^{(1)} \times \left(\mathbb{D}_{(\mathcal{R}\times\mathcal{C})}\right)^T + \mathbb{E}_{(2)} \tag{4}$$

Then the posterior distribution of $\mathbf{V}^{(m)}$ is also multivariate normal.

Posterior inference of $\mathbb{G}$:

$$\mathbf{U} := \mathbf{U}^{(L)} \otimes \cdots \otimes \mathbf{U}^{(1)}$$
$$\mathbf{V} := \mathbf{V}^{(M)} \otimes \cdots \otimes \mathbf{V}^{(1)}$$

$$\mathbb{Y}_{(1)} = (\mathbb{X}_{(1)}\mathbf{U})\mathbb{G}_{(\mathcal{R}\times\mathcal{C})}\mathbf{V}^T + \mathbf{E}_{(1)} \tag{5}$$

By changing variable, we can derive the posterior of $\mathbb{G}$.

MCMC Sampling

For a core tensor $\mathbb{G}$ with fixed dimensions, we can use Gibbs sampling method to update $\mathbf{U}^{(l)}$, $\mathbf{V}^{(m)}$, and $\mathbb{G}$. We need to select the **dimensions of core tensor** $\mathbb{G}$.

Metropolis-Hastings to generate dimensions of $\mathbb{G}$. For example:

$$\theta_1 = (g_1^{(n)}, g_2^{(n)}, \cdots, g_k^{(n)}),\ \theta_2 = (\tilde{g}_1^{(n)}, g_2^{(n)}, \cdots, g_k^{(n)})$$

Accept θ_2 with accept rate $R = \min\{1, r(\theta_1, \theta_2)\}$ where

$$r(\theta_1, \theta_2) = \frac{Pr(\theta_2 \mid \mathbb{Y}, \sigma^2)\ q(\theta_1 \mid \theta_2)}{Pr(\theta_1 \mid \mathbb{Y}, \sigma^2)\ q(\theta_2 \mid \theta_1)}$$

Fast Computing Algorithm: Use maximum a posteriori probability (MAP) estimators instead of generating factor matrices, and core tensor.

$$vec\mathbf{U}_{MAP}^{(1)} = \left(\mathbb{C}_{(\mathcal{R}\times\mathcal{C})}^T \mathbb{C}_{(\mathcal{R}\times\mathcal{C})} + \sigma^2 \boldsymbol{\Sigma}_U^{-1}\right)^{-1} \mathbb{C}_{(\mathcal{R}\times\mathcal{C})}^T vec\mathbf{Y}$$
$$\mathbf{V}_{MAP}^{(1)} = \left(\mathbb{D}_{(\mathcal{R}\times\mathcal{C})}^T \mathbb{D}_{(\mathcal{R}\times\mathcal{C})} + \sigma^2 \mathbf{I}_{S_1}\right)^{-1} \mathbb{D}_{(\mathcal{R}\times\mathcal{C})}^T \mathbb{Y}_{(2)}$$
$$vec\mathbb{G}_{(\mathcal{R}\times\mathcal{C})_{MAP}} = \left((\mathbf{I}_S \otimes (\mathbb{X}_{(1)}\mathbf{U}))^T (\mathbf{I}_S \otimes (\mathbb{X}_{(1)}\mathbf{U})) + \sigma^2 \mathbf{I}_{QS}\right)^{-1} (\mathbf{I}_S \otimes (\mathbb{X}_{(1)}\mathbf{U}))^T vec\tilde{\mathbf{Y}}$$

With this small modification, simulation study shows that we can save approximate 90% of the running time for most cases.

Simulation and Real Data Analysis

Simulation Study:

Consider the true dimension of core tensor $\mathbb{G}$ being θ^*, consider signal to noise ratio (SNR) being 5 or 50, and consider sample size N being 150 or 500.

Consider $\mathbb{X} \in \mathbb{R}^{N \times P_1 \times P_2}$, $\mathbb{Y} \in \mathbb{R}^{N \times Q_1 \times Q_2}$, and $P_1, P_2, Q_1, Q_2 = 15, 20, 5, 10$

Compare relative prediction error ($\mathbf{RPE} = \frac{\left\|\mathbb{Y}_{new} - \langle \mathbb{X}_{new}, \hat{\mathbb{B}} \rangle_L\right\|_F^2}{\|\mathbb{Y}_{new}\|_F^2}$) of our model with OLS method, and CP tensor regression method in Lock (2018).

	Tucker Model	CP model	OLS model
N=150,SNR=5	**0.198(0.014)**	0.252(0.033)	0.750(0.023)
N=150,SNR=50	**0.023(0.001)**	0.083(0.029)	0.530(0.031)
N=500,SNR=50	**0.020(2.1e-04)**	0.034(8.9e-03)	0.049(0.0018)

Table 1: RPE (Sd Error) for some simulation results, $\theta^* = (6, 6, 2, 2)$

Real Data Analysis:

We apply our model to predict different attributes of frontalized facial image, such as smiling, using the Labeled Faces in the Wild database. Each image has 90×90 pixels resulting in $\mathbb{X} \in \mathbb{R}^{1000 \times 90 \times 90 \times 3}$, and attributes $\mathbb{Y} \in \mathbb{R}^{1000 \times 73}$.

	Tucker Model	CP model
CR	**0.952**	0.934
RPE	**0.368**	0.568

Table 2: RPE and 95% credible interval coverage ratio (CR) for two models

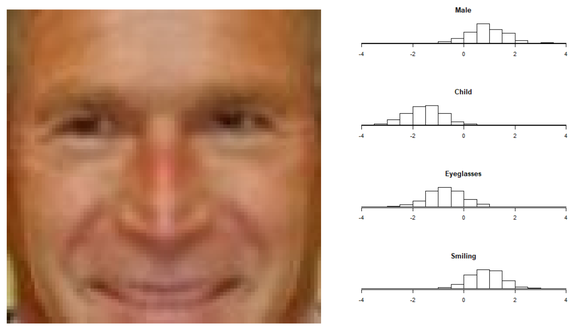

Figure 1: An example of test image and the corresponding posterior predictive values for four selected characteristics

Conclusions

- A Bayesian tensor regression model to predict one tensor from another tensor.
- Our model is based on Tucker Decomposition.
- Metropolis-Hastings method is used to choose dimensions of core tensor.
- A fast computing algorithm is given using MAP instead of random generation.