Supplementary Material for the Granger Causality with PARAFAC Algorithm used in the paper "Tensor Analysis and Fusion of Multimodal Brain Images"

Esin Karahan, Pedro A. Rojas-Lopez, Maria L. Bringas-Vega, Pedro A. Valdes-Hernandez, Pedro A. Valdes-Sosa

This document is prepared for the explanation of the tensor regression with PARAFAC software presented in http://www.cneuro.cu/software/tensor and http://neurosignal.boun.edu.tr/software/tensor .

I. Notation

In its general form an N-Dimensional tensor (or an N order tensor) is represented by calligraphic uppercase letters (e.g. $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$). For a matrix $\mathbf{X}$, $\mathbf{X}^T$ denotes the transpose and for a tensor $\mathcal{X}$, $\mathcal{X}^*$ denotes the permutation of its modes. Permutation will be clear in the context. $\mathbf{I}$ is the identity matrix, $\mathcal{I}$ is the identity tensor with 1's on the super-diagonal and $\mathbf{1}$ is the column vector of 1's.

Definition: Tensor contraction is a generalization of matrix multiplication for tensors. Consider a tensor $\mathcal{X}$ of size $I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M$ and $\mathcal{Y}$ of size $J_1 \times \cdots \times J_M \times K_1 \times \cdots \times K_P$. Multiplication over common dimensions $J_1 \times \cdots \times J_M$ will give:

$$(\mathcal{X} \bullet_{\{j_1,\cdots,j_M\}} \mathcal{Y})(i_1,\cdots,i_N,k_1,\cdots,k_P)$$
$$= \sum_{j_1 \cdots j_M =1}^{J_1 \cdots J_M} \mathcal{X}(i_1,\cdots,i_N,j_1,\cdots,j_M)\mathcal{Y}(j_1,\cdots,j_M,k_1,\cdots,k_P)$$

Definition: Mode-*n* unfolding of a tensor is the transformation of the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ to a matrix $\mathcal{X}_{(\mathrm{n})} \in \mathbb{R}^{I_n \times I_1 \ldots I_{n-1} I_{n+1} \ldots I_N}$ where mode-*n* fibers are arranged to be columns of the resulting matrix. Tensor element $(i_1, i_2, ..., i_N)$ corresponds to matrix element (i_n, j), where $j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (\mathrm{i}_k - 1) J_k$ with $J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m$.

Definition: Khatri-Rao product is the columnwise Kronecker product . Let $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$, Khatri-Rao product $\mathbf{A} \odot \mathbf{B}$ defined as follows:

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{A}(:,1) \otimes \mathbf{B}(:,1) \;\; \mathbf{A}(:,2) \otimes \mathbf{B}(:,2) \cdots \mathbf{A}(:,K) \otimes \mathbf{B}(:,K)]$$

Definition: Inner product is an operation between two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ that share the same order and dimensions yielding a scalar:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = (\mathcal{X} \bullet_{\{I_1,\cdots,I_N\}} \mathcal{Y}) = \sum_{i_1,\cdots,i_N=1}^{I_1 \cdots I_N} \mathcal{X}(i_1,\cdots,i_N)\mathcal{Y}(i_1,\cdots,i_N)$$

Definition: PARAFAC decomposition

Given $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, PARAFAC decomposition is formulated as:

$$\mathcal{X} = [\![\lambda; \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!] \qquad (1)$$

where $\mathbf{U}_n \in \mathbb{R}^{I_n \times R}$ for $n = 1,...,N$ are the factor matrices or signatures and R is the model order.

PARAFAC decomposition can be expressed in different ways (Kolda, 2006):

Expression 1: Elementwise notation

$$\boldsymbol{\mathcal{X}}(i_1,\ldots,i_N) = \sum_{r=1}^{R} \mathbf{U}_1(\mathrm{i}_1,\mathrm{r})\ldots\mathbf{U}_N(\mathrm{i}_N,\mathrm{r})$$

Expression 2: Sum of outer products

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{U}_1(:,r) \circ \ldots \circ \mathbf{U}_N(:,r)$$

Expression 3: Matricized notation

$$\boldsymbol{\mathcal{X}}_{(\mathrm{n})} = \mathbf{U}_n(\mathbf{U}_N \odot \ldots. \odot \mathbf{U}_{n+1} \odot \mathbf{U}_{n-1} \odot \ldots \odot \mathbf{U}_1)^T$$

Factors of the PARAFAC decomposition $\mathbf{U}_1$ to $\mathbf{U}_N$ can be estimated by using Alternating Least Squares (ALS) algorithm. ALS is an iterative algorithm such that each factor is estimated sequentially by fixing the others. $\mathbf{U}_n$ is estimated by:

$$\hat{\mathbf{U}}_n = \arg\min_{\mathbf{U}_n} \left\{ \left\| \boldsymbol{\mathcal{X}} - [\![\mathbf{U}_1,\ldots,\mathbf{U}_n,\ldots,\mathbf{U}_N]\!] \right\|_2^2 \right\}$$

By using expression 3, this is equivalent to:

$$\hat{\mathbf{U}}_n = \operatorname{argmin}_{\mathbf{U}_n} \left\{ \left\| \boldsymbol{\mathcal{X}}_{(\mathrm{n})} - \mathbf{U}_n(\mathbf{U}_N \odot \ldots. \odot \mathbf{U}_{n+1} \odot \mathbf{U}_{n-1} \odot \ldots \odot \mathbf{U}_1)^T \right\|_2^2 \right\}$$

The closed form solution of $\hat{\mathbf{U}}_n$ is found as:

$$\hat{\mathbf{U}}_n = \boldsymbol{\mathcal{X}}_{(\mathrm{n})}(\mathbf{U}_N \odot \ldots. \odot \mathbf{U}_{n+1} \odot \mathbf{U}_{n-1} \odot \ldots \odot \mathbf{U}_1)^{\dagger}$$

II. TENSOR REGRESSION AND DECOMPOSITION

Given $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M}$ and a linear map $\boldsymbol{\mathcal{A}} : \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N} \rightarrow \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M}$ with $\prod_{m=1}^{M} J_m \le \prod_{n=1}^{N} I_n$, we want to estimate the tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ satisfying $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{A}} \bullet_{\{i_1,\cdots,i_K\}} \boldsymbol{\mathcal{X}}$ with $K \le N$ and its factors are defined as $\boldsymbol{\mathcal{X}} = [\![\lambda; \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!]$. This formulation guarantees $\boldsymbol{\mathcal{X}}$ to be low-rank.

For the tensor MAR described in the paper, N=3, M=2. $\boldsymbol{\mathcal{Y}}$ is the fMRI data sampled from I_{Cx} voxels of the cortical grid at $I_{T\delta}$ time samples. The lagged data values are collected in the tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_T \times I_{lag} \times I_{Cx}}$ and connectivity tensor is denoted as $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_{lag} \times I_{Cx} \times I_{Cx}}$. We will give the estimation of parameters for the general case.

The objective function for this problem is defined as:

$$\underset{\mathcal{X},\mathbf{U}_1,\ldots,\mathbf{U}_N}{\text{minimize}} \quad \frac{1}{2}\left\|\mathcal{Y}-\mathcal{A}\bullet\mathcal{X}\right\|_2^2 \qquad \text{s.t. } \left\|\mathcal{X}-[\![\lambda;\mathbf{U}_1,\mathbf{U}_2,\ldots,\mathbf{U}_N]\!]\right\|_2^2 \tag{2}$$

We relax the objective function by introducing penalty functions and further we impose certain constraints on the factors of the tensor decomposition:

$$\underset{\mathcal{X},\mathbf{U}_1,\ldots,\mathbf{U}_N}{\text{minimize}} \quad \frac{1}{2}\left\|\mathcal{Y}-\mathcal{A}\bullet\mathcal{X}\right\|_2^2+\frac{\alpha_{CP}}{2}\left\|\mathcal{X}-[\![\lambda;\mathbf{U}_1,\mathbf{U}_2,\ldots,\mathbf{U}_N]\!]\right\|_2^2+\sum_{n=1}^{N}\sum_{l=1}^{L}P_{nl}(\alpha_{nl},\mathbf{U}_n) \tag{3}$$

where α_{CP} is the penalization variable for $\mathcal{X}$ to be a multiway array, $P_{nl}(\alpha_{nl},\mathbf{U}_n)$ is the lth penalty function on the factor $\mathbf{U}_n$ with the penalty parameter α_{nl}.

We will use Alternating Direction Method of Multipliers (ADMM) which is preferable to others for large-scale optimization problems (Boyd, Parikh, Chu, Peleato, & Eckstein, 2010). In this method global problem is divided into local ones by using variable splitting and augmented Lagrangian.

In (3) we introduce a new variable $\mathcal{Z}\in\mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$ for variable splitting between first and second term of the objective function:

$$\underset{\mathcal{X},\mathbf{U}_1,\ldots,\mathbf{U}_N}{\text{minimize}} \quad \frac{1}{2}\left\|\mathcal{Y}-\mathcal{A}\bullet\mathcal{X}\right\|_2^2+\frac{\alpha_{CP}}{2}\left\|\mathcal{Z}-[\![\lambda;\mathbf{U}_1,\mathbf{U}_2,\ldots,\mathbf{U}_N]\!]\right\|_2^2+\sum_{n=1}^{N}\sum_{l=1}^{L}P_{nl}(\alpha_{nl},\mathbf{U}_n) \tag{4}$$
$$\text{s. t.} \qquad \mathcal{X}-\mathcal{Z}=0$$

The augmented Lagrangian is given as:

$$\underset{\mathcal{X},\mathcal{Z},\mathbf{U}_1,\ldots,\mathbf{U}_N}{\text{minimize}}\,\mathcal{L}(\mathcal{X},\mathcal{Z},\mathbf{U}_1,\mathbf{U}_2,\ldots,\mathbf{U}_N)=\frac{1}{2}\left\|\mathcal{Y}-\mathcal{A}\bullet\mathcal{X}\right\|_2^2+\frac{\alpha_{CP}}{2}\left\|\mathcal{Z}-[\![\lambda;\mathbf{U}_1,\mathbf{U}_2,\ldots,\mathbf{U}_N]\!]\right\|_2^2+\sum_{n=1}^{N}\sum_{l=1}^{L}P_{nl}(\alpha_{nl},\mathbf{U}_n)$$
$$+\frac{\rho}{2}\left\|\mathcal{X}-\mathcal{Z}\right\|_2^2+\langle\mathcal{W},\mathcal{X}-\mathcal{Z}\rangle$$

where $\mathcal{W}\in\mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$ is the Lagrange multiplier. ADMM algorithm is given as follows:

$$\mathcal{X}^{k+1}=\underset{\mathcal{X}}{\text{argmin}}\,\mathcal{L}(\mathcal{X},\mathcal{Z}^k,\mathbf{U}_1^k,\mathbf{U}_2^k,\ldots,\mathbf{U}_N^k)$$
$$\mathcal{Z}^{k+1}=\underset{\mathcal{Z}}{\text{argmin}}\,\mathcal{L}(\mathcal{X}^{k+1},\mathcal{Z},\mathbf{U}_1^k,\mathbf{U}_2^k,\ldots,\mathbf{U}_N^k)$$
$$\mathbf{U}_1^{k+1}=\underset{\mathbf{U}_1}{\text{argmin}}\,\mathcal{L}(\mathcal{X}^{k+1},\mathcal{Z}^{k+1},\mathbf{U}_1,\mathbf{U}_2^k,\ldots,\mathbf{U}_N^k)$$
$$\vdots$$
$$\mathbf{U}_N^{k+1}=\underset{\mathbf{U}_N}{\text{argmin}}\,\mathcal{L}(\mathcal{X}^{k+1},\mathcal{Z}^{k+1},\mathbf{U}_1^{k+1},\mathbf{U}_2^{k+1},\ldots,\mathbf{U}_N)$$
$$\mathcal{W}^{k+1}=\mathcal{W}^k+\rho(\mathcal{X}^{k+1}-\mathcal{Z}^{k+1})$$

1. **ADMM update for $\mathcal{X}$:**

$\mathcal{X}$ is estimated by:

$$\mathcal{X}=\underset{\mathcal{X}}{\text{argmin}} \quad \frac{1}{2}\left\|\mathcal{Y}-\mathcal{A}\bullet\mathcal{X}\right\|_2^2+\frac{\rho}{2}\left\|\mathcal{X}-\mathcal{Z}\right\|_2^2+\langle\mathcal{W},\mathcal{X}-\mathcal{Z}\rangle$$

Since the objective function is differentiable, the derivative with respect to $\mathcal{X}$ will be:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{X}} = -\mathcal{A}^* \bullet (\mathcal{Y} - \mathcal{A} \bullet \mathcal{X}) + \rho(\mathcal{X} - \mathcal{Z}) + \mathcal{W}$$

By setting the gradient to zero, $\mathcal{X}$ is found as:

$$\mathcal{X} = (\mathcal{A}^* \bullet \mathcal{A} + \rho \mathcal{I})^{-1} \bullet (\mathcal{A}^* \bullet \mathcal{Y} + \rho \mathcal{Z} - \mathcal{W})$$

If the dimensions of $\mathcal{A} \bullet \mathcal{A}^*$ is smaller than $\mathcal{A}^* \bullet \mathcal{A}$, the formulation below can be used:

$$\mathcal{X} = (\mathcal{Z} - \frac{1}{\rho}\mathcal{W}) + \frac{1}{\rho}\mathcal{A}^* \bullet (\frac{1}{\rho}\mathcal{A} \bullet \mathcal{A}^* + \mathcal{I})^{-1} \bullet (\mathcal{Y} - \mathcal{A} \bullet (\mathcal{Z} - \frac{1}{\rho}\mathcal{W}))$$

2. **ADMM update for $\mathcal{Z}$:**

$\mathcal{Z}$ is estimated by:

$$\mathcal{Z} = \underset{\mathcal{Z}}{\operatorname{argmin}} \frac{\alpha_{CP}}{2} \left\| \mathcal{Z} - [\![\lambda_{CP}; \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!] \right\|_2^2 + \frac{\rho}{2} \left\| \mathcal{X} - \mathcal{Z} \right\|_2^2 + \langle \mathcal{W}, \mathcal{X} - \mathcal{Z} \rangle$$

Call $[\![\lambda_{CP}; \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!]$ as $\mathcal{Q}$. Since the objective function is differentiable:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{Z}} = \alpha_{CP}(\mathcal{Z} - \mathcal{Q}) - \rho(\mathcal{X} - \mathcal{Z}) - \mathcal{W}$$

$\mathcal{Z}$ is found as:

$$\mathcal{Z} = \frac{1}{\alpha_{CP} + \rho}(\alpha_{CP}\mathcal{Q} + \rho \mathcal{X} + \mathcal{W})$$

3. **ADMM update for $\mathbf{U}_n$:**

$$\mathbf{U}_n = \underset{\mathbf{U}_n}{\operatorname{argmin}} \quad \frac{\alpha_{CP}}{2} \left\| \mathcal{Z} - [\![\lambda_{CP}; \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!] \right\|_2^2 + \sum_{n=1}^{N} \sum_{l=1}^{L} P_{nl}(\alpha_{nl}, \mathbf{U}_n)$$

At this step any type of solver for tensor decompositions can be used.

For a given $\mathbf{U}_n$ several combinations of penalization functions can be used. We now relate to a combination containing orthogonality, nonnegativity, smoothness and sparsity penalizations.

That is:

$$\sum_{l=1}^{L} P_{nl}(\alpha_{nl}, \mathbf{U}_n) = \alpha_{orth} \left\| \mathbf{U}_n{}^T \mathbf{U}_n - \mathbf{I} \right\|_2^2 + \frac{1}{2}\alpha_{sm} \left\| \mathbf{L}\mathbf{U}_n \right\|_2^2 + \alpha_{sp} \left\| \mathbf{U}_n \right\|_1$$

s.t. $\mathbf{U}_n \geq 0$

where α_{orth} is the penalization parameter for the orthogonality constraint, α_{sm} is for the smoothness and α_{sp} is for the sparsity.

We use Hierarchical Alternating Least Squares (HALS) algorithm to estimate the factors of the PARAFAC decomposition. The HALS algorithm is a modified ALS algorithm in which at each step of ALS, only one column of a factor is estimated by fixing the rest of the columns of the factor. Moreover we make use of the matricized notation for PARAFAC with $\mathbf{G}_n$=$(\mathbf{U}_N \odot .... \odot \mathbf{U}_{n+1} \odot \mathbf{U}_{n-1} \odot ... \odot \mathbf{U}_1)$ to take the objective function as:

$$\mathbf{U}_n = \underset{\mathbf{U}_n}{\operatorname{argmin}} \quad \frac{\alpha_{CP}}{2}\left\|\mathcal{Z}_{(n)} - \mathbf{U}_n\mathbf{G}_n{}^T\right\|_2^2 + \sum_{l=1}^{L} P_{nl}(\alpha_{nl},\mathbf{U}_n)$$

Since at each step of the algorithm we estimate only a column of $\mathbf{U}_n$, we have:

$$\mathbf{U}_n(:,j) = \underset{\mathbf{U}_n(:,k)}{\operatorname{argmin}} \quad \frac{\alpha_{CP}}{2}\left\|\tilde{\mathcal{Z}}_{(n)} - \mathbf{U}_n(:,j)\mathbf{G}_n(:,j)^T\right\|_2^2 + \sum_{l=1}^{L} P_{nl}(\alpha_{nl},\mathbf{U}_n)$$

where $\tilde{\mathcal{Z}}_{(n)} = \mathcal{Z}_{(n)} - \sum_{i\neq j}^{R} \mathbf{U}_n(:,i)\mathbf{G}_n(:,i)^T$

The orthogonality constraint on the nonnegative signatures can be imposed column-wise (Kimura, Tanaka, & Kudo, 2014). The reason for this is that for a nonnegative matrix $\mathbf{X} \in \mathbb{R}^{I\times J}$, orthogonality condition $\mathbf{X}^T\mathbf{X} = \mathbf{I}_J$ can be replaced by $2J$ column-wise coefficients $\mathbf{X}(:,j)^T\,\mathbf{X}(:,j) = 1$ and $\sum_{k\neq j}^{J} \mathbf{X}(:,k)^T\,\mathbf{X}(:,j) = 0$ for $j = 1,2,\ldots,J$.

We can now impose orthogonality conditions as $\sum_{i\neq j}^{R} \mathbf{U}_n(:,i)^T\mathbf{U}_n(:,j) = 0$ for $j = 1,2,\ldots,R$ where R is the number of components in the PARAFAC decomposition. We assume that the factors are normalized. We denote the orthogonality constraint as: $\mathbf{U}_n^{(j)T}\mathbf{U}_n(:,j) = 0$

With this, the factor is estimated by:

$$\hat{\mathbf{U}}_n(:,j) = \underset{\mathbf{U}_n(:,k)}{\operatorname{argmin}} \quad \frac{\alpha_{CP}}{2}\left\|\tilde{\mathcal{Z}}_{(n)} - \mathbf{U}_n(:,j)\mathbf{G}_n(:,j)^T\right\|_2^2 + \alpha_{orth}\mathbf{U}_n^{(j)T}\mathbf{U}_n(:,j) + \frac{1}{2}\alpha_{sm}\left\|\mathbf{L}\mathbf{U}_n(:,j)\right\|_2^2 + \alpha_{sp}\left\|\mathbf{U}_n(:,j)\right\|_1$$

The gradient is found as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_n(:,j)} = \left\{-\alpha_{CP}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j) + \alpha_{CP}\mathbf{U}_n(:,j)\mathbf{G}_n(:,j)^T\,\mathbf{G}_n(:,j) + \alpha_{orth}\mathbf{U}_n^{(j)} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\mathbf{U}_n(:,j) + \alpha_{sp}\mathbf{1}\right\} \quad (5)$$

Due to the scaling on the factors, (5) becomes:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_n(:,j)} = \left\{-\alpha_{CP}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j) + (\alpha_{CP}\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L})\mathbf{U}_n(:,j) + \alpha_{orth}\mathbf{U}_n^{(j)} + \alpha_{sp}\mathbf{1}\right\} \qquad (6)$$

By solving (6) and forcibly keeping the nonnegativity, we obtain the update rule under the assumption of normalization of $\mathbf{U}_n(:,j)^T\,\mathbf{U}_n(:,j) = 1$ as post-processing:

$$\hat{\mathbf{U}}_n(:,j) = \left[\left(\alpha_{CP}\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\left(\alpha_{CP}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j) - \alpha_{orth}\mathbf{U}_n^{(j)} - \alpha_{sp}\mathbf{1}\right)\right]_+$$

where $[a]_+ = \left\{ \begin{matrix} a & if\, a \geq 0 \\ 0 & else \end{matrix} \right\}$

In order to set the value of the α_{orth} parameter we can left multiply the (6) by $\left(\mathbf{U}_n^{(j)T}\right)\left(\alpha_{CP}\mathbf{I}+\alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}$ and take advantage that $\mathbf{U}_n^{(k)T}\mathbf{U}_n(:,j)=0$ we have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_n(:,j)} = \left\{ \begin{matrix} -\alpha_{CP}\left(\mathbf{U}_n^{(j)T}\right)\left(\alpha_{CP}\mathbf{I}+\alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j)+\alpha_{orth}\left(\mathbf{U}_n^{(j)T}\right)\left(\alpha_{CP}\mathbf{I}+\alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\left(\mathbf{U}_n^{(j)}\right) \\ +\alpha_{sp}\left(\mathbf{U}_n^{(j)T}\right)\left(\alpha_{CP}\mathbf{I}+\alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1} \end{matrix} \right\} \tag{7}$$

If (7) is solved for α_{orth}, we will get:

$$\alpha_{orth} = \left\{ \frac{\mathbf{U}_n^{(j)T}\left(\alpha_{CP}\mathbf{I}+\alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\left(\alpha_{CP}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j)-\alpha_{sp}\mathbf{1}\right)}{\mathbf{U}_n^{(j)T}\left(\alpha_{CP}\mathbf{I}+\alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{U}_n^{(j)}} \right\}$$

For the case when orthogonality is not required for the factor matrix $\mathbf{U}_n$, the penalization function will be:

$$\sum_{l=1}^{L} P_{nl}(\alpha_{nl},\mathbf{U}_n) = \frac{1}{2}\alpha_{sm}\left\|\mathbf{L}\mathbf{U}_n\right\|_2^2 + \alpha_{sp}\left\|\mathbf{U}_n\right\|_1$$

$\mathbf{U}_n$ is estimated by:

$$\hat{\mathbf{U}}_n = \underset{\mathbf{U}_n}{\operatorname{argmin}} \; \frac{\alpha_{CP}}{2}\left\|\mathcal{Z}_{(n)} - \mathbf{U}_n\mathbf{G}_n{}^T\right\|_2^2 + \frac{1}{2}\alpha_{sm}\left\|\mathbf{L}\mathbf{U}_n\right\|_2^2 + \alpha_{sp}\left\|\mathbf{U}_n\right\|_1 \tag{8}$$

Since in (8) $\mathbf{U}_n$ is multiplied by $\mathbf{G}_n$ from the right and by $\mathbf{L}$ from the left, estimation of $\mathbf{U}_n$ is not easy. One method is the vectorization of (8) and use Kronecker products which is not favorable since the scale of the problem will be high. We prefer to use ADMM to split the quadratic and penalization functions as follows:

$$\begin{aligned} &\underset{\mathbf{U}_n,\mathbf{V}}{\text{minimize}} \quad \frac{\alpha_{CP}}{2}\left\|\mathcal{Z}_{(n)} - \mathbf{U}_n\mathbf{G}_n{}^T\right\|_2^2 + \frac{1}{2}\alpha_{sm}\left\|\mathbf{L}\mathbf{V}\right\|_2^2 + \alpha_{sp}\left\|\mathbf{V}\right\|_1 \\ &\text{s. t.} \quad \mathbf{U}_n - \mathbf{V} = 0 \end{aligned}$$

The augmented Lagrangian is given as:

$$\underset{\mathbf{U}_n,\mathbf{V}}{\text{minimize}}\,\mathcal{L}(\mathbf{U}_n,\mathbf{V}) = \frac{\alpha_{CP}}{2}\left\|\mathcal{Z}_{(n)} - \mathbf{U}_n\mathbf{G}_n{}^T\right\|_2^2 + \frac{1}{2}\alpha_{sm}\left\|\mathbf{L}\mathbf{V}\right\|_2^2 + \alpha_{sp}\left\|\mathbf{V}\right\|_1 + \frac{\tau}{2}\left\|\mathbf{U}_n - \mathbf{V}\right\|_2^2 + \left\langle \mathbf{W},\mathbf{U}_n - \mathbf{V}\right\rangle$$

where **W** is the Lagrange multiplier.

ADMM update for $\mathbf{U}_n$:

$\mathbf{U}_n$ is estimated by:

$$\underset{\mathbf{U}_n}{\text{minimize}}\, \boldsymbol{\mathcal{L}}(\mathbf{U}_n) = \frac{\alpha_{CP}}{2}\left\|\boldsymbol{\mathcal{Z}}_{(n)} - \mathbf{U}_n\mathbf{G}_n{}^T\right\|_2^2 + \frac{\tau}{2}\left\|\mathbf{U}_n - \mathbf{V}\right\|_2^2 + \left\langle \mathbf{W}, \mathbf{U}_n - \mathbf{V}\right\rangle$$

The gradient of the objective function is:

$$\frac{\partial\boldsymbol{\mathcal{L}}}{\partial\mathbf{U}_n(:,j)} = -\alpha_{CP}\boldsymbol{\mathcal{Z}}_{(n)}\mathbf{G}_n + \alpha_{CP}\mathbf{U}_n\mathbf{G}_n{}^T\mathbf{G}_n + \tau(\mathbf{U}_n - \mathbf{V}) + \mathbf{W} \tag{9}$$

Then $\mathbf{U}_n$ is found by setting (9) to zero:

$$\hat{\mathbf{U}}_n = \left(\alpha_{CP}\boldsymbol{\mathcal{Z}}_{(n)}\mathbf{G}_n + \tau\mathbf{V} - \mathbf{W}\right)\left(\alpha_{CP}\mathbf{G}_n{}^T\mathbf{G}_n + \tau\mathbf{I}\right)^{-1}$$

Note that since the dimensions of $\mathbf{G}_n{}^T\mathbf{G}_n$ is $R \times R$, it is easy to find the inverse of the matrix.

ADMM update for $\mathbf{V}$:

$\mathbf{V}$ is estimated by:

$$\underset{\mathbf{V}}{\text{minimize}}\, \boldsymbol{\mathcal{L}}(\mathbf{V}) = \frac{\tau}{2}\left\|\mathbf{U}_n - \mathbf{V}\right\|_2^2 + \left\langle \mathbf{W}, \mathbf{U}_n - \mathbf{V}\right\rangle + \frac{1}{2}\alpha_{sm}\left\|\mathbf{L}\mathbf{V}\right\|_2^2 + \alpha_{sp}\left\|\mathbf{V}\right\|_1$$

(10)

We used proximal gradient ascent for the estimation of the **V** as described in (Beck & Teboulle, 2009). Proximal map of the function $P(x)$ is defined as $\text{prox}_P(y,\lambda) = \underset{x}{\text{argmin}} \left\{\frac{1}{2}\left\|x - y\right\|_2^2 + \lambda P(x)\right\}$. By using the quadratic approximation of (10), we found **V** as:

$$\hat{\mathbf{V}} = \text{prox}_P\left(\mathbf{V} + \frac{1}{L_V}\left(\mathbf{W} + \tau\mathbf{U}_n - (\tau\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L})\mathbf{V}\right), \frac{\lambda_{sp}}{L_V}\right)$$

L_V is the Lipschitz constant found as (Beck & Teboulle, 2009) :

$$L_V = \max\text{eig}\left(\tau\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\right)$$

In our problem we used the proximal operator of L1 norm, $P(x) = |x|_1$ which gives the estimation of x as follows $\text{prox}_P(y,\lambda) = \text{sign}(x)\cdot(|x|\text{-}\lambda)_+$.

Finally the Lagrange multiplier **W** is updated by:

$$\mathbf{W} = \mathbf{W} + \tau(\mathbf{U}_n - \mathbf{V})$$

The algorithm for the problem in (2) is summarized in **Algorithm** 1.

Algorithm 1 ADMM for tensor regression

```
Inputs : $\mathcal{Y}, \mathcal{A}, R, \alpha_{sm}, \alpha_{sp}, \alpha_{CP}, \rho$
Outputs : $\mathcal{X}, \mathbf{U}_1, \ldots, \mathbf{U}_N$
Initialize : $\{\mathbf{U}_1, \ldots, \mathbf{U}_N\}$
  for $k = 1,2, \ldots,$ do
      if size($\mathcal{A}_{(i)}$,1) > size($\mathcal{A}_{(i)}$,2)
          $\mathcal{X}^{(k+1)} = (\mathcal{A}^* \bullet \mathcal{A} + \rho\mathcal{I})^{-1} \bullet (\mathcal{A}^* \bullet \mathcal{Y} + \rho\mathcal{Z}^{(k)} - \mathcal{W}^{(k)})$
      else
          $\mathcal{X}^{(k+1)} = (\mathcal{Z}^{(k)} - \frac{1}{\rho}\mathcal{W}^{(k)}) + \frac{1}{\rho}\mathcal{A}^* \bullet (\frac{1}{\rho}\mathcal{A} \bullet \mathcal{A}^* + \mathcal{I})^{-1} \bullet (\mathcal{Y} - \mathcal{A} \bullet (\mathcal{Z}^{(k)} - \frac{1}{\rho}\mathcal{W}^{(k)}))$
     end if
     $\mathcal{Z}^{(k+1)} = \frac{1}{\alpha_{CP} + \rho}(\alpha_{CP}\mathcal{Q}^{(k)} + \rho\mathcal{X}^{(k+1)} + \mathcal{W}^{(k)})$
     repeat until convergence
     for $n = 1,2,\ldots,N,$ do
       for $j = 1,2,\ldots,R,$ do
             if $\mathrm{U}_n$ orthogonal
                $\hat{\mathbf{U}}_n(:,j) = \left[\left(\alpha_{CP}\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\left(\alpha_{CP}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j) - \alpha_{orth}\mathbf{U}_n^{(j)} - \alpha_{sp}\mathbf{1}_{I_n}\right)\right]_+$
                $\alpha_{orth} = \left\{\frac{\mathbf{U}_n^{(j)T}\left(\alpha_{CP}\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\left(\alpha_{CP}\tilde{\mathcal{Z}}_{(n)}\mathbf{G}_n(:,j) - \alpha_{sp}\mathbf{1}\right)}{\mathbf{U}_n^{(j)T}\left(\alpha_{CP}\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\right)^{-1}\mathbf{U}_n^{(j)}}\right\}$
             else
                $L_V = \max \operatorname{eig}\left(\tau\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L}\right)$
                repeat until convergence
                   $\hat{\mathbf{U}}_n = \left(\alpha_{CP}\mathcal{Z}_{(n)}\mathbf{G}_n + \tau\mathbf{V} - \mathbf{W}\right)\left(\alpha_{CP}\mathbf{G}_n{}^T\mathbf{G}_n + \tau\mathbf{I}\right)^{-1}$
                   $\hat{\mathbf{V}} = \operatorname{prox}_P\left(\mathbf{V} + \frac{1}{L_V}\left(\mathbf{W} + \tau\mathbf{U}_n - (\tau\mathbf{I} + \alpha_{sm}\mathbf{L}^T\mathbf{L})\mathbf{V}\right), \frac{\lambda_{sp}}{L_V}\right)$
                   $\mathbf{W} = \mathbf{W} + \tau(\mathbf{U}_n - \mathbf{V})$
                end repeat
             end if
       end for
     end for
     end repeat
    $\mathcal{Q}^{(k+1)} = [\![\lambda_{CP}; \mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_N]\!]$
    $\mathcal{W}^{(k+1)} = \mathcal{W}^{(k)} + \rho(\mathcal{X}^{(k+1)} - \mathcal{Z}^{(k+1)})$
 end for
```

References

Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202. doi:10.1137/080716542

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, *3*(1), 1–122. doi:10.1561/2200000016

Kimura, K., Tanaka, Y., & Kudo, M. (2014). A Fast Hierarchical Alternating Least Squares Algorithm for Orthogonal Nonnegative Matrix Factorization. In *JMLR: Workshop and Conference Proceedings* (pp. 129–141).

Kolda, T. G. (2006). *Multilinear operators for higher-order decompositions*. *SAND2006-2081*. Albuquerque, NM, Livermore, CA. Retrieved from http://www.osti.gov/bridge/product.biblio.jsp?osti_id=923081