Block Tensor Train Decomposition for Missing Value Imputation

Namgil Lee*
* Kangwon National University, Chuncheon, South Korea
E-mail: namgil.lee@kangwon.ac.kr Tel: +82-33-250-8433

***Abstract*—We propose a new method for imputation of missing values in large scale matrix data based on a low-rank tensor approximation technique called the block tensor train (TT) decomposition. Given sparsely observed data points, the proposed method iteratively computes the soft-thresholded singular value decomposition (SVD) of the underlying data matrix with missing values. The SVD of matrices is performed based on a low-rank block TT decomposition for large scale data matrices with a low-rank tensor structure. Experimental results on simulated data demonstrate that the proposed method can estimate a large amount of missing values accurately compared to a matrix-based standard method.**

I. Introduction

In this paper, we propose a new method for estimating missing values in large scale matrix data

$$\mathbf{Y} \in \mathbb{R}^{I \times J_1 J_2 \cdots J_N}, \tag{1}$$

by extending standard iterative soft-thresholded singular value decomposition (SVD) to block tensor train (TT) decomposition as an efficient low-rank data structure. The SVD is a mathematical technique which factorizes a matrix into the product of matrices of left singular vectors, right singular vectors, and singular values. In statistics and data mining, the SVD has been used as a key numerical technique for multivariate data analyses such as principal component analysis (PCA) and multiple regression analysis. The SVD has a wide range of applications in areas such as ecology [4] and recommender systems [7], [10].

A tensor refers to a multi-dimensional array, which can be considered as a generalization of vectors and matrices. Tensor decomposition, like matrix SVD, has been developed for a wide scope of applications in signal processing, machine learning, chemometrics, and neuroscience [9]. Traditional tensor decompositions include Candecomp/Parafac (CP) decomposition and Tucker decomposition; see, e.g., [9]. Modern tensor decompositions have been developed more recently to cope with the problem called as the curse-of-dimensionality, which means an exponential rate of increase in the storage and computational costs as the dimensionality of tensors increases [5], [8].

The tensor train (TT) decomposition is one of the modern tensor decompositions which generalize the matrix SVD to higher-order (i.e., multi-dimensional) tensors [16]. Modern tensor decompositions such as the TT decomposition applies not only to higher order tensors, but also to large scale vectors and matrices, by transforming the vectors and matrices into higher-order tensors via a tensorization process [2]. Once the large scale vectors and matrices have been decomposed by TT decomposition, algebraic operations such as the matrix-by-vector multiplication can be performed much efficiently with logarithmically scaled computational costs [16].

In this work, we focus on the so-called block TT decomposition, which is one of the TT representations of large scale matrices. The block TT decomposition has been used for approximately computing a set of eigenvectors or singular vectors of large scale matrices [3], [11], [12], [13]. Compared to the other TT representations for matrices, the block TT decomposition is suitable for approximating low-rank data matrices, which is a typical assumption of standard imputation methods based on matrix SVD.

For SVD of large data matrices with missing values, [15] proposed an iterative soft-thresholding algorithm. In that algorithm, a convex optimization problem is suggested and solved by an iterative process consisting of soft-thresholding SVD and missing data imputation. The SVD plays a crucial role for estimating missing values of large matrices based on its low-rank constraints and soft-thresholding regularization. We remark that other SVD-based missing data imputation algorithms have been developed in recent decades, see, e.g., references in [4], [15].

In this study, we propose an iterative soft-thresholding algorithm based on the block TT decomposition. In the proposed algorithm, the missing values in large scale matrix data are estimated based on soft-thresholded SVD, where the set of either the left or right singular vectors are represented by low-rank block TT decomposition. Moreover, in order for an efficient computation with sparsely observed data during iteration process, we propose an efficient procedure for computing a multiplication of a sparse matrix with a large matrix in block TT format. In addition, similarly as in [15], we consider that an estimate of the underlying data matrix $\mathbf{Y} \in \mathbb{R}^{I \times J_1 \cdots J_N}$ is represented as a low-rank matrix using block TT format plus a sparse matrix format as follows:

$$\mathbf{Y} \approx \widehat{\mathbf{Y}} = \mathbf{X} + \mathbf{R}_\Omega,$$

where $\mathbf{R}_\Omega$ is the sparse matrix of residuals for the observed data values. By using algebraic operations based on the above format together with the proposed efficient sparse matrix-by-block TT multiplication, the proposed block TT-based method can carry out missing data imputation fast and accurately.

II. Block Tensor Train (TT) Decomposition for Singular Value Decomposition (SVD) of Large Matrices

A. Notations

In this work, scalars, vectors, and matrices are denoted by lowercase $(a, b, \ldots)$, lowercase bold $(\mathbf{a}, \mathbf{b}, \ldots)$, and uppercase bold letters $(\mathbf{A}, \mathbf{B}, \ldots)$. A tensor is a multi-dimensional array which generalizes vectors and matrices. The size of a tensor can be written as $I_1 \times I_2 \times \cdots \times I_N$, where N is called the order of the tensor. Tensors are denoted by uppercase calligraphic letters $(\mathcal{A}, \mathcal{B}, \ldots)$.

For a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its $(i_1, i_2, \ldots, i_N)$th element is denoted by $(\mathcal{A})_{i_1, i_2, \ldots, i_N}$ or $\mathcal{A}(i_1, i_2, \ldots, i_N)$. For $n = 1, \ldots, N$, the mode-n unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a matrix $\mathbf{A}_{(n)} \equiv [\mathcal{A}]_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$ with entries

$$\left(\mathbf{A}_{(n)}\right)_{i_n, k_n} = (\mathcal{A})_{i_1, i_2, \ldots, i_N}$$

and $k_n = 1 + \sum_{m \neq n} (i_m - 1) \prod_{p=1, p \neq n}^{m-1} I_p$ for all $i_m = 1, \ldots, I_m$, $m = 1, \ldots, N$.

The mode-$(M, 1)$ product of two tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$ and $\mathcal{B} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with $I_M = J_1$ is defined by

$$\mathcal{A} \bullet \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{M-1} \times J_2 \times \cdots \times J_N}$$

with entries

$$(\mathcal{A} \bullet \mathcal{B})_{i_1, \ldots, i_{M-1}, j_2, \ldots, j_N} = \sum_{i_M = 1}^{I_M} (\mathcal{A})_{i_1, \ldots, i_M} (\mathcal{B})_{i_M, j_2, \ldots, j_N}.$$

Note that, in the case that $M = N = 2$, the mode-$(2, 1)$ product of two matrices $\mathbf{A}$ and $\mathbf{B}$ is same to the matrix-by-matrix multiplication as $\mathbf{A} \bullet \mathbf{B} = \mathbf{AB}$. It also has the commutativity as $(\mathcal{A} \bullet \mathcal{B}) \bullet \mathcal{C} = \mathcal{A} \bullet (\mathcal{B} \bullet \mathcal{C})$. See, e.g., [14] for further properties.

B. Block TT Decomposition for Matrices

We consider that a large "tall-and-skinny" matrix $\mathbf{V} \in \mathbb{R}^{J_1 \cdots J_N \times R_X}$ is reshaped and permuted into a tensor $\mathcal{V}$ of size $J_1 \times \cdots \times J_n \times R_X \times J_{n+1} \times \cdots \times J_N$. The block-$n$ tensor train (TT) decomposition of $\mathbf{V}$ is defined by a product of a series of low-order tensors as

$$\mathbf{V} \approx \mathcal{V} = \mathcal{V}_1 \bullet \mathcal{V}_2 \bullet \cdots \bullet \mathcal{V}_N, \tag{2}$$

where $\mathcal{V}_m \in \mathbb{R}^{R_{m-1} \times J_m \times R_m}$ $(m \neq n)$ are third-order tensors, $\mathcal{V}_n \in \mathbb{R}^{R_{n-1} \times J_n \times R_X \times R_n}$ is a fourth-order tensor. The tensors $\mathcal{V}_1, \ldots, \mathcal{V}_N$ are called the TT-cores and $R_1, \ldots, R_{N-1}$ are called the TT-ranks. We assume that $R_0 = R_N = 1$.

Note that when the large matrix $\mathbf{V}$ is decomposed by the block TT decomposition, the storage cost reduces from $\mathcal{O}(J^N R)$ to $\mathcal{O}(NJR^2)$, where $J = \max(\{J_m\})$ and $R = \max(\{R_m\}, R_X)$. See, e.g., [16], for further properties of TT decomposition and algebraic operations.

Orthogonalization of TT-cores is an important concept for applications to numerical optimization. A third-order TT-core $\mathcal{V}_m \in \mathbb{R}^{R_{m-1} \times J_m \times R_m}$ is called left-orthogonalized if the row vectors of $[\mathcal{V}_m]_{(3)} \in \mathbb{R}^{R_m \times R_{m-1} J_m}$ are orthonormal, i.e.,

$$[\mathcal{V}_m]_{(3)} [\mathcal{V}_m]_{(3)}^\top = \mathbf{I}_{R_m},$$

and it is called right-orthogonalized if the row vectors of $[\mathcal{V}_m]_{(1)} \in \mathbb{R}^{R_{m-1} \times J_m R_m}$ are orthonormal [6], [14].

Moreover, we define the partial products of TT-cores of a block-n TT decomposition in (2), by $\mathcal{V}_{<n} = \mathcal{V}_1 \bullet \cdots \bullet \mathcal{V}_{n-1} \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times R_X}$ and $\mathcal{V}_{>n} = \mathcal{V}_{n+1} \bullet \cdots \bullet \mathcal{V}_N \in \mathbb{R}^{R_X \times J_{n+1} \times \cdots \times J_N}$. It has been known that the matrix $\mathbf{V} \in \mathbb{R}^{J_1 \cdots J_N \times R_X}$ in (2) can be represented as a product of two matrices as

$$\mathbf{V} = \mathbf{V}_{\neq n} \mathbf{V}_n, \tag{3}$$

where $\mathbf{V}_{\neq n} = [\mathcal{V}_{>n}]_{(1)}^\top \otimes \mathbf{I}_{J_n} \otimes [\mathcal{V}_{<n}]_{(n)}^\top \in \mathbb{R}^{J_1 \cdots J_N \times R_{n-1} J_n R_n}$ is called the frame matrix and $\mathbf{V}_n = [\mathcal{V}_n]_{(3)}^\top \in \mathbb{R}^{R_{n-1} J_n R_n \times R_X}$ is the mode-3 unfolding of the nth TT-core [11], [14]. It has also been known that the frame matrix consists of orthonormal columns if the TT-cores $\mathcal{V}_1, \ldots, \mathcal{V}_{n-1}$ are left-orthogonalized and the TT-cores $\mathcal{V}_{n+1}, \ldots, \mathcal{V}_N$ are right-orthogonalized [11], [14].

C. SVD Based on Block TT Decomposition

In this section, we describe the proposed SVD algorithm based on block TT decomposition, which will be the most crucial step in the missing data imputation procedure in the next section.

The goal is to compute the R_X largest singular values and the corresponding left/right singular vectors of a large matrix $\widehat{\mathbf{Y}} \in \mathbb{R}^{I \times J_1 J_2 \cdots J_N}$ as

$$\widehat{\mathbf{Y}} \approx \mathbf{U} \mathbf{S} \mathbf{V}^\top, \tag{4}$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{R_X}] \in \mathbb{R}^{I \times R_X}$ is the matrix of the left singular vectors, $\mathbf{S} = \mathrm{diag}(s_1, \ldots, s_{R_X})$ is the diagonal matrix of the R_X largest singular values, and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{R_X}] \in \mathbb{R}^{J_1 \cdots J_N \times R_X}$ is the matrix of the right singular vectors. In this study, we consider that $\mathbf{V}$ is represented by a block-n TT decomposition with TT-cores $\mathcal{V}_1, \ldots, \mathcal{V}_N$ as in (2).

The above problem of computing the largest singular values and the corresponding singular vectors can be expressed as the maximization problem given as [13]

$$\begin{aligned} \underset{\mathbf{U}, \mathbf{V}}{\text{maximize}} \quad & \text{trace}\left(\mathbf{U}^\top \widehat{\mathbf{Y}} \mathbf{V}\right) \\ \text{subject to} \quad & \mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I}_{R_X}. \end{aligned} \tag{5}$$

Suppose that $\mathbf{V}$ is in a block-n TT format (2), i.e., the nth TT-core is of fourth-order, and all the TT-cores except the nth one are fixed. Due to the expression in (3), the large scale optimization problem in (5) can be reduced to the smaller scale problem written as

$$\begin{aligned} \underset{\mathbf{U}, \mathbf{V}_n}{\text{maximize}} \quad & \text{trace}\left(\mathbf{U}^\top \widehat{\mathbf{Y}}_n \mathbf{V}_n\right) \\ \text{subject to} \quad & \mathbf{U}^\top \mathbf{U} = \mathbf{V}_n^\top \mathbf{V}_n = \mathbf{I}_{R_X}, \end{aligned} \tag{6}$$

where $\widehat{\mathbf{Y}}_n = \widehat{\mathbf{Y}} \mathbf{V}_{\neq n} \in \mathbb{R}^{I \times R_{n-1} J_n R_n}$.

The proposed algorithm, `bttdSsvdImpute_ALS`, iteratively updates $\mathbf{U}$, $\mathbf{S}$, and the TT-cores $\mathcal{V}_1, \ldots, \mathcal{V}_N$ of $\mathbf{V}$ in (4). One iteration of `bttdSsvdImpute_ALS` is described in Algorithm 1. It is based on the alternating least squares (ALS) approach which updates $\mathbf{U}$ or $\mathbf{V}_n$ at each iteration while other components are fixed. Each update is performed by the power method for computing extreme eigenvectors or singular vectors based on the small optimization problem in (6) [17]. After updating $\mathcal{V}_n$, the block-n TT format is converted to the block-$(n+1)$ or the block-$(n-1)$ TT format by computing SVD for $\mathcal{V}_n$, which helps to keep TT-cores orthogonalized as well.

Algorithm 1: One iteration of `bttdSsvdImpute_ALS`

Input : A matrix $\widehat{\mathbf{Y}} \in \mathbb{R}^{I \times J_1 J_2 \cdots J_N}$, $\mathbf{U}$, $\mathbf{S}$, and $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ in block-1 TT format
Output: Updated $\mathbf{U}$, $\mathbf{S}$, and $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ in block-1 TT format

`// Update` $\mathbf{U}$
Compute $\mathbf{Z} \leftarrow \widehat{\mathbf{Y}}\mathbf{V}$
Compute SVD of $\mathbf{Z}$ as $\mathbf{Z} = \widetilde{\mathbf{U}}\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^\top$
Update $\mathbf{U} \leftarrow \widetilde{\mathbf{U}}$, $\mathbf{S} \leftarrow \widetilde{\mathbf{S}}$, $\mathbf{V}_1 \leftarrow \mathbf{V}_1\widetilde{\mathbf{V}}$
for $n \leftarrow 1, 2, \ldots, N, N-1, \ldots, 1$ **do**
  `// Update` $\mathcal{V}_n$
  Compute $\mathbf{Z} \leftarrow \mathbf{V}_{\neq n}^\top \widehat{\mathbf{Y}}^\top \mathbf{U}$
  Compute SVD of $\mathbf{Z}$ as $\mathbf{Z} = \widetilde{\mathbf{U}}\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^\top$
  Update $[\mathcal{V}_n]_{(3)} \leftarrow \widetilde{\mathbf{U}}$, $\mathbf{S} \leftarrow \widetilde{\mathbf{S}}$, $\mathbf{U} \leftarrow \mathbf{U}\widetilde{\mathbf{V}}$
  if *n_is_increasing* **then**
  `// Convert block-`n` TT into block-`$(n+1)$` TT format`
  Compute SVD of $[\mathcal{V}_n]_{(\{1,2\}\times\{3,4\})} = \widetilde{\mathbf{U}}\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^\top \in \mathbb{R}^{R_{n-1}J_n \times R_X R_n}$
  Update $[\mathcal{V}_n]_{(\{1,2\}\times\{3\})} \leftarrow \widetilde{\mathbf{U}}$, $[\mathcal{V}_{n+1}]_{(\{1\}\times\{3,2,4\})} \leftarrow \widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^\top([\mathcal{V}_{n+1}]_{(1)} \otimes \mathbf{I}_{R_X})$;
  else
  `// Convert block-`n` TT into block-`$(n-1)$` TT format`
  Compute SVD of $[\mathcal{V}_n]_{(\{1,3\}\times\{2,4\})} = \widetilde{\mathbf{U}}\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^\top \in \mathbb{R}^{R_{n-1}R_X \times J_n R_n}$
  Update $[\mathcal{V}_n]_{(\{1\}\times\{2,3\})} \leftarrow \widetilde{\mathbf{V}}^\top$, $[\mathcal{V}_{n-1}]_{(\{1,2,3\}\times\{4\})} \leftarrow (\mathbf{I}_{R_X} \otimes [\mathcal{V}_{n-1}]_{(\{1,2\}\times\{3\})})\widetilde{\mathbf{U}}\widetilde{\mathbf{S}}$;
end

III. An Iterative Soft-thresholding Algorithm for Missing Value Imputation

Let $\mathbf{Y} \in \mathbb{R}^{I \times J_1 J_2 \cdots J_N}$ denote the original data matrix and $\Omega \subset \{1, \ldots, I\} \times \{1, \ldots, J_1 J_2 \cdots J_N\}$ denote the set of indices of the observed entries, i.e., $(\mathbf{Y})_{i,k}$ is observed if and only if $(i,k) \in \Omega$. We define a projection to the set of observed entries by $P_\Omega \mathbf{Y} \in \mathbb{R}^{I \times J_1 J_2 \cdots J_N}$ with

$$(P_\Omega \mathbf{Y})_{i,k} = \begin{cases} (\mathbf{Y})_{i,k} & \text{if } (i,k) \in \Omega \\ 0 & \text{if } (i,k) \notin \Omega. \end{cases} \tag{7}$$

We consider to solve the following nuclear norm regularized minimization problem:

$$\underset{\mathbf{X}}{\text{minimize}} \quad J(\mathbf{X}) = \frac{1}{2}\|P_\Omega \mathbf{Y} - P_\Omega \mathbf{X}\|_F^2 + \lambda \|\mathbf{X}\|_*, \tag{8}$$

where $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} a_{i,j}^2}$ is the Frobenius norm of a matrix $\mathbf{A}$, and $\|\mathbf{X}\|_*$ is called the nuclear norm and is defined by the sum of the singular values of $\mathbf{X}$. An algorithm called `softImpute` has been proposed in [15] and shown to generate estimates which converge to the minimizer of the cost function $J(\mathbf{X})$.

In this paper, we present an extension of the `softImpute`, which is called `bttdSsvdImpute_ALS`. The proposed `bttdSsvdImpute_ALS` generalizes `softImpute` to higher-order tensor structured data by using block TT decomposition, as described in the previous sections.

The proposed algorithm is described in Algorithm 2. Consider that an estimate of the solution to the problem (8) can be written by $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, where $\mathbf{V}$ is in block-n TT format with TT-cores $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$. The missing values are estimated by the corresponding values of $\mathbf{X}$. The estimate of the underlying data matrix $\mathbf{Y}$ can be written as

$$\widehat{\mathbf{Y}} = \mathbf{X} + \mathbf{R}_\Omega, \tag{9}$$

where $\mathbf{R}_\Omega$ is the sparse matrix of residuals defined by

$$\mathbf{R}_\Omega = P_\Omega \mathbf{Y} - P_\Omega \mathbf{X}.$$

The iterations in the proposed algorithm is stopped by the criterion described as follows. At iteration $k = 0, 1, 2, \ldots$, let $\mathbf{X}^{(k)} = \mathbf{U}^{(k)}\mathbf{S}^{(k)}(\mathbf{V}^{(k)})^\top$ denote the estimated solution to the minimization problem. The iteration is stopped if the rate of changes in the singular vectors is sufficiently small as

$$ratio = \frac{\|\mathbf{U}^{(k-1)} - \mathbf{U}^{(k)}\|_F}{\max(\|\mathbf{U}^{(k-1)}\|_F, \|\mathbf{U}^{(k)}\|_F)} \leq \delta_X, \tag{10}$$

where $\delta_X > 0$ is a predefined tolerance parameter.

IV. Experiments

The experiments were conducted using R software version 3.4.3 on a PC with Intel i5-6200 CPU and a memory of 8 GB running Windows 10 Pro operating system.

A. Simulations

Through simulated experiments, we verify convergence of the block TT-based proposed method and compare performances of the proposed method with the matrix-based standard method. Details about the simulated data are described in each subsection.

Algorithm 2: `bttdSsvdImpute_ALS`

Input : $\mathbf{Y} \in \mathbb{R}^{I \times J_1 J_2 \cdots J_N}$, an integer R_X, a parameter $\delta_X > 0$

Output: An approximate solution, $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, to the minimization problem (8), where $\mathbf{U} \in \mathbb{R}^{I \times R_X}$, $\mathbf{S} = \mathrm{diag}(s_1, \ldots, s_{R_X})$, and $\mathbf{V} \approx \{\mathcal{V}_1, \ldots, \mathcal{V}_N\} \in \mathbb{R}^{J_1 \cdots J_N \times R_X}$ in block-1 TT format

Initialize $\mathbf{U} \leftarrow \mathbf{0}$, $\mathbf{S} \leftarrow \mathrm{diag}(1, \ldots, 1)$, TT-cores $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ for block-1 TT format with entries randomly drawn from standard normal distribution.

Left-orthogonalize $\mathcal{V}_2, \ldots, \mathcal{V}_N$. Orthogonalize $\mathcal{V}_1$ so that $[\mathcal{V}_1]_{(3)}[\mathcal{V}_1]_{(3)}^\top = \mathbf{I}_{R_X}$.

$\mathbf{R}_\Omega \leftarrow P_\Omega \mathbf{Y} - P_\Omega \mathbf{X}$. Define $\widehat{\mathbf{Y}} \equiv \mathbf{X} + \mathbf{R}_\Omega$

$iter \leftarrow 0$, $ratio \leftarrow 1$

while $iter \leq max\_iter$ *and* $ratio \geq \delta_X$ **do**

  Update $\mathbf{U}$, $\mathbf{S}$, $\{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ by Algorithm 1.

  Update $\mathbf{R}_\Omega \leftarrow P_\Omega \mathbf{Y} - P_\Omega \mathbf{X}$ and $\widehat{\mathbf{Y}}$.

  Compute $ratio$ by (10).

end

1) Sparsely Observed Random Matrices: We consider a data matrix $\mathbf{Y}$ of size $J \times J^N$, which is considered as mode-1 unfolding of an order-$(N+1)$ tensor $\mathcal{Y} \in \mathbb{R}^{J \times J \times \cdots \times J}$, i.e.,

$$\mathbf{Y} \equiv \mathbf{Y}_{(1)} \in \mathbb{R}^{J \times J^N}.$$

We assume that only the super-diagonal elements of the tensor $\mathcal{Y}$ are observed, so that the index set of the observed values can be written as $\Omega = \{(j,k) \in \{1, \ldots, J\} \times \{1, \ldots, J^N\} \,|\, k = 1 + \sum_{n=1}^{N} (j-1) J^{n-1}, 1 \leq j \leq J\}$. The observed values of $\mathbf{Y}$ were randomly generated from a uniform distribution on $[0, 1]$. For instance, $N = 2$ implies that $\mathbf{Y}$ is a matrix of size $J \times J^2$, and the $(j, j + (j-1)J)$th elements are the observed elements as

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \cdots & * & \cdots & \cdots & \cdots & * \\ * & \cdots & y_{2,2+J} & \cdots & \cdots & \cdots & * \\ \vdots & & \vdots & & \ddots & & \vdots \\ * & \cdots & * & \cdots & \cdots & \cdots & y_{J,J^2} \end{bmatrix} \in \mathbb{R}^{J \times J^2}.$$

In the simulation, the rank of the solution matrix $\mathbf{X}$ were changed by $R_X = 1, 2, \ldots, 5$. The initial values for the TT-ranks of the right singular vectors, $\mathbf{V}$, were set at $R_n = 1$ for all $n = 1, \ldots, N-1$. The TT-cores were initialized randomly by the standard normal distribution and then the right singular vectors were orthogonalized. We defined a maximal TT-rank by $R_{max} = 10$, i.e., $R_X \leq 10$. With $\lambda = 0$, we estimated $\mathbf{X}$ and calculated the cost function value by

$$J(\mathbf{X}) = \frac{1}{2} \|P_\Omega \mathbf{Y} - P_\Omega \mathbf{X}\|_F^2.$$

The tolerance parameter for stopping criterion was set by $\delta_X = 10^{-6}$.

Fig. 1 shows the convergence of the cost function of the solution matrix estimated by the proposed method. We can see that with the rank $R_X = 1$, the convergence speed is relatively slow compared to the other cases of $R_X = 2, 3, 4, 5$. This is because the rank R_X larger than 1 allows TT-ranks to adaptively change during iteration process, while the rank $R_X = 1$ does not. Note that the TT-ranks were initialized by $R_n = 1$ for all n, which means that, even if the TT-ranks did not change at $R_n = 1$ for the case of $R_X = 1$, the sparsely observed values was estimated accurately with $\delta_X = 10^{-6}$. This is a reasonable result because there exists a rank-1 matrix which perfectly fit with the observed values $\mathbf{y} = [y_{1,1}, \ldots, y_{J,J^2}]^\top$, e.g., $\mathbf{X} = [\mathbf{y}, \mathbf{y}, \cdots, \mathbf{y}] = \mathbf{y}\mathbf{1}^\top$, where $\mathbf{1}$ is a vector of ones, and the vector $\mathbf{1}$ has the TT-ranks $R_n = 1$ for all n.

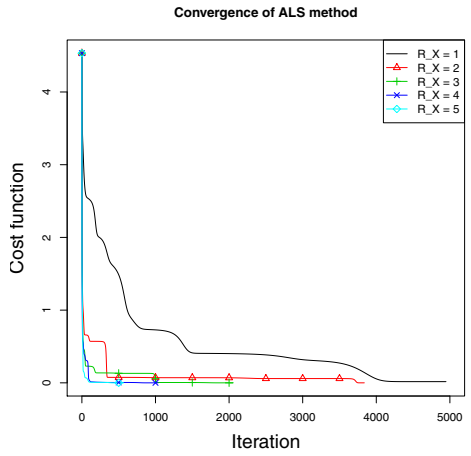

Fig. 1. Convergence in the cost function value of the proposed method. The data is the simulated sparsely observed random matrices of size 20×20^N.

2) Low-rank Matrices: In this section, low-rank data matrices $\mathbf{Y}$ were randomly generated by using block TT decomposition as

$$\mathbf{Y} = \mathbf{U}_0 \mathbf{S}_0 \mathbf{V}_0^\top,$$

where $\mathbf{U}_0 \in \mathbb{R}^{I \times R_Y}$ were generated from standard normal distribution and then orthogonalized, $\mathbf{S}_0 = \mathrm{diag}(s_1, \ldots, s_{R_Y}) \in \mathbb{R}^{R_Y \times R_Y}$ were generated from uniform $[0.5, 1]$ distribution, and TT-cores of block-1 TT tensor $\mathbf{V}_0 \approx \mathcal{V}_1 \bullet \cdots \bullet \mathcal{V}_N \in \mathbb{R}^{I \times R_Y}$ were randomly generated by standard normal distribution and then left-orthogonalized. The TT-ranks of $\mathbf{V}_0$ were set at $R_n = 2$, and other parameters were set at $N \in \{2, 3\}, I = 100, J_n \in \{4, 8, 12, 16, 20\}, R_Y = R_X = 4, \delta_X = 0.0005, \lambda = 0$. The observed values were randomly sampled, and the number of observed values, $|\Omega|$, were set at

$$|\Omega| = \rho I J_1 \cdots J_N.$$

The proportion was set by $\rho \in \{0.2, 0.4\}$ to simulate sparsely observed data matrices. The simulation can be conducted for other values of ρ in future works.

We compared performances of the block TT-based proposed method with the matrix-based method. In the matrix-based

TABLE I
SUMMARY STATISTICS FOR THE DIFFERENCE OF $RRMSE$ VALUES BETWEEN THE BLOCK TT-BASED METHOD AND THE MATRIX-BASED METHOD, FOR THE CASE THAT $N = 2$, $\rho = 0.2$, AND $J_n = 4, 8, 12, 16, 20$.

$N = 2, \rho = 0.2$	J_n				
	4	8	12	16	20
Min.	-0.365	-0.466	-1.179	-0.393	-0.926
1st Qu.	-0.329	-0.042	-0.049	-0.032	-0.021
Median	-0.006	0.705	-0.009	-0.009	-0.015
Mean	0.014	0.626	-0.247	-0.088	0.031
3rd Qu.	0.148	0.943	-0.005	-0.009	-0.002
Max.	0.622	1.991	0.008	0.000	1.118

TABLE II
SUMMARY STATISTICS FOR THE DIFFERENCE OF $RRMSE$ VALUES BETWEEN THE BLOCK TT-BASED METHOD AND THE MATRIX-BASED METHOD, FOR THE CASE THAT $N = 2$, $\rho = 0.4$, AND $J_n = 4, 8, 12, 16, 20$.

$N = 2, \rho = 0.4$	J_n				
	4	8	12	16	20
Min.	-1.415	-0.00020	-0.00102	-0.0018	-0.0023
1st Qu.	-0.175	-0.00001	0.00006	-0.0014	-0.0022
Median	-0.110	0.00002	0.00043	-0.0014	-0.0016
Mean	-0.326	0.00094	0.00023	-0.0012	-0.0013
3rd Qu.	-0.007	0.00006	0.00066	-0.0008	-0.0004
Max.	0.079	0.00484	0.00105	-0.0007	-0.0001

method, the right singular vectors $\mathbf{V}$ are estimated in full matrix format. In the case that $N = 1$, the block TT-based method and the matrix based method are identical. As a performance measure, the relative root mean squared error (RRMSE) was calculated by

$$RRMSE = \frac{\|\mathbf{X} - \mathbf{Y}\|_F}{\|\mathbf{Y}\|_F}.$$

The simulation results for $N = 2$ are illustrated in Tables I and II. Each value in the tables show the difference of the RRMSE values of the block TT based-method and the matrix based-method, i.e.,

$$RRMSE(blockTT) - RRMSE(matrix).$$

The negative value implies that the block TT-based method achieved higher accuracy in the missing data imputation. In Tables I and II, we can see that the higher the size J_n became, the block TT-based method achieved higher accuracy.

The simulation results for $N = 3$ are illustrated in Tables III and IV. We did not run the simulation for large proportion $\rho \geq 0.5$ and large size $J_n \geq 16$ because the number of observed data values increased largely so that computational costs became relatively huge. In the tables, we can see that the proposed block TT-based method was more accurate for relatively large sizes J_n. Moreover, compared to the case of lower order $N = 2$, the proposed block TT-based method achieved higher accuracy for the case of higher order $N = 3$, which implies higher efficiency for higher order tensor structured data. In addition, it is remarkable that even for the cases of high order, large data sizes, and large proportion of missing data (e.g., $\rho \leq 0.2$), the low-rank block TT-based method could recover missing data successfully.

V. CONCLUSIONS AND DISCUSSION

In this work, we proposed a novel method for missing data imputation based on block TT decomposition. Since the block TT decomposition computes SVD of large matrices approximately, the proposed method successfully extended the existing method on low-rank matrix data to low-rank tensor structured data, while potentially inheriting favorable properties such as global convergence. Compared to the matrix-based method, the proposed method could achieve higher accuracy in the cases of higher order ($N > 1$) tensor structured data in the simulated experiments.

TABLE III
SUMMARY STATISTICS FOR THE DIFFERENCE OF $RRMSE$ VALUES BETWEEN THE BLOCK TT-BASED METHOD AND THE MATRIX-BASED METHOD, FOR THE CASE THAT $N = 3$, $\rho = 0.2$, AND $J_n = 4, 8, 12, 16, 20$.

$N = 3, \rho = 0.2$	J_n				
	4	8	12	16	20
Min.	-1.580	-0.0314	-0.0268	-0.030	-0.023
1st Qu.	-0.020	-0.0208	-0.0136	-0.020	-0.023
Median	0.018	-0.0173	-0.0110	-0.020	-0.022
Mean	0.177	-0.0184	-0.0133	-0.020	-0.020
3rd Qu.	0.339	-0.0166	-0.0088	-0.017	-0.020
Max.	2.130	-0.0057	-0.0064	-0.013	-0.012

On the other hand, [1] and [13] have proposed SVD algorithms for large scale matrices based on TT decompositions. However, existing SVD methods based on TT decompositions assume that the large scale matrices are in the TT matrix format, whereas in this work the large scale matrices are in the low-rank matrix plus sparse format.

There are several challenges for missing data imputation based on (block) TT decompositions. (i) Alternating least squares (ALS)-based algorithms often show slow convergence speed, especially for high dimensional data problems. Modified ALS algorithm can be an alternative, but its computational cost is usually higher than that of ALS algorithms [13]. (ii) The computational cost highly depends on the number of observed data values, which requires more efficient way of computing sparse matrix-by-tensor multiplication. In this sense, recently arising new trends such as randomized methods [1] can be a new direction.

TABLE IV
SUMMARY STATISTICS FOR THE DIFFERENCE OF $RRMSE$ VALUES BETWEEN THE BLOCK TT-BASED METHOD AND THE MATRIX-BASED METHOD, FOR THE CASE THAT $N = 3$, $\rho = 0.4$, AND $J_n = 4, 8, 12$.

$N = 3, \rho = 0.4$	J_n		
	4	8	12
Min.	-0.0113	-0.0021	-0.0139
1st Qu.	-0.0017	-0.0009	-0.0069
Median	-0.0005	-0.0004	-0.0043
Mean	-0.0015	-0.0005	-0.0057
3rd Qu.	0.0026	0.0001	-0.0029
Max.	0.0035	0.0006	-0.0004

REFERENCES

[1] K. Batselier, W. Yu, L. Daniel, and N. Wong. Computing low-rank approximations of large-scale matrices with the Tensor Network randomized SVD. arXiv:1707.07803, 2017.

[2] O. Debals and L. De Lathauwer. Stochastic and deterministic tensorization for blind signal separation. In E. Vincent, A. Yeredor, Z. Koldovsky, and P. Tichavský, editors, Proceedings of the 12th International Conference on Latent Variable Analysis and Signal Separation, pages 3–13. Springer International Publishing, 2015.

[3] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov. Computation of extreme eigenvalues in higher dimensions using block tensor train format. Comput. Phys. Comm., 185, 2014.

[4] S. Dray and J. Josse. Principal component analysis with missing values: a comparative survey of methods. Plant Ecology, 216(5):657–667, 2015. doi: 10.1007/s11258-014-0406-z.

[5] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. GAMM-Mitt., 36:53–78, 2013.

[6] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed TT-rank. Numerische Mathematik, 120(4):701–731, 2012.

[7] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. Recommender Systems: An Introduction. Cambridge University Press, First edition, 2010. ISBN: 0521493366.

[8] B. N. Khoromskij. Tensors-structured numerical methods in scientific computing: Survey on recent advances. Chemometr. Intell. Lab. Syst., 110:1–19, 2012.

[9] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. SIAM Review, 51:455–500, 2009.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

[11] D. Kressner, M. Steinlechner, and A. Uschmajew. Low-rank tensor methods with subspace correction for symmetric eigenvalue problems. SIAM Journal on Scientific Computing, 36(5):A2346–A2368, 2014.

[12] O. S. Lebedeva. Tensor conjugate-gradient-type method for Rayleigh quotient minimization in block QTT-format. Russian J. Numer. Anal. Math. Model., 26:465–489, 2011.

[13] N. Lee and A. Cichocki. Estimating a few extreme singular values and vectors for large-scale matrices in tensor train format. SIAM Journal on Matrix Analysis and Applications, 36(3):994–1014, 2015. doi: 10.1137/140983410.

[14] N. Lee and A. Cichocki. Fundamental tensor operations for large-scale data analysis using tensor network formats. Multidimensional Systems and Signal Processing, 29(3):921–960, 2018. doi: 10.1007/s11045-017-0481-0.

[15] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. Journal of Machine Learning Research, 11:2287–2322, 2010.

[16] I. V. Oseledets. Tensor-train decomposition. SIAM Journal on Scientific Computing, 33(5):2295– 2317, 2011. doi: 10.1137/090752286.

[17] Y. Saad. Numerical Methods for Large Eigenvalue Problems, volume 66 of Classics in Applied Mathematics. SIAM, Philadelphia, Revised edition, 2011.