MACHINE LEARNING Science and Technology

OPEN ACCESS

RECEIVED
10 March 2020

REVISED
6 June 2020

ACCEPTED FOR PUBLICATION
7 August 2020

PUBLISHED
1 December 2020

TOPICAL REVIEW

Randomized algorithms for fast computation of low rank tensor ring model

Salman Ahmadi-Asl[1], Andrzej Cichocki[1,2], Anh Huy Phan[1], Maame G Asante-Mensah[1], Mirfarid Musavian Ghazani[1], Toshihisa Tanaka[3] and Ivan Oseledets[1]

[1] Skolkovo Institute of Science and Technology (SKOLTECH), CDISE, Moscow, Russian Federation
[2] Nicolaus Copernicus University, 87-100 Torun, Poland
[3] Tokyo University of Agriculture and Technology, Tokyo, Japan

E-mail: s.asl@skoltech.ru

Keywords: Tensor Ring-Tensor Train (TR-TT) decompositions, randomized algorithm, random projection

Abstract

Randomized algorithms are efficient techniques for big data tensor analysis. In this tutorial paper, we review and extend a variety of randomized algorithms for decomposing large-scale data tensors in Tensor Ring (TR) format. We discuss both adaptive and nonadaptive randomized algorithms for this task. Our main focus is on the random projection technique as an efficient randomized framework and how it can be used to decompose large-scale data tensors in the TR format. Simulations are provided to support the presentation and efficiency, and performance of the presented algorithms are compared.

1. Introduction

Tensor decompositions have found numerous applications such as in signal processing, machine learning and scientific computing [1–9]. Handling very large-scale data tensors is a challenging task due to high computational complexity and memory requirements. Tucker decomposition [10–12] can resolve this problem by compressing a given data tensor by a smaller core tensor and factor matrices. However, the core tensor suffers from the phenomenon known as *curse of dimensionality* which means that the number of parameters for its representation is exponentially increased as the order of the core tensor is increased [13, 14]. To tackle this difficulty, alternative tensor representations have been introduced. *Tensor networks* are effective tools to cope with this difficulty by approximating a given data tensor by a series of inter-connected smaller low order core tensors [2, 3, 15]. For example, Tensor Train/Tensor Ring (TT/TR) decompositions [16–20] are special cases of Hierarchical Tucker decomposition [21, 22] which are two simple but powerful tensor networks representing the original tensor as a train and a ring (chain) of inter-connected third order tensors, respectively. The TT decomposition is known as *Matrix Product States* (MPS) in quantum physics [23–26]. The TT and the TR decompositions have found many applications in scientific computing and machine learning communities such as computing extreme singular values and computing pseudoinverse of very large matrices [27, 28], reducing number of parameters in deep neural networks (DNNs) [29–33], tensor completion [34–39], machine learning [40–44], quantum chemistry [45], solving high-dimensional PDEs [46], low rank approximation of large sparse matrices [47]. Other related tensor networks are *Projected Entangled Pair States* (PEPS) [48], the *Multi-scale Entanglement Renormalization Ansatz* (MERA) [49] and the *Tree Tensor Network* (TTN) [50]. While the Tucker decomposition suffers from the curse of dimensionality, recently an efficient algorithm has been proposed in [51] to compute a Tucker decomposition whose core tensor stored in the TT format to avoid this difficulty. The idea is based on decomposing a tensor into the TT format followed by a conversion into the Tucker format whose core tensor is stored in the TT format. Deterministic algorithms for computing the TR decomposition involves computation of a series of the SVD of unfolding matrices. Clearly this is computationally prohibitive for large-scale data tensors and requires large memory and computational complexity. Randomized algorithms are effective techniques to cope with this problem by reducing the computational complexity of the deterministic algorithms and also reducing the communication among different levels of memory hierarchy. In this paper, we review a variety of randomized algorithms for computing the TR decomposition of large-scale data tensors.

Our main contributions in this paper are as follows:

- Extending random projection technique for fast TR decomposition (Algorithms 7 and 10),
- Extending randomized algorithms for fast TR decomposition with permutation of core tensors (Algorithms 13 and 14),
- Extending randomized rank-revealing algorithm for fast TR decomposition (Algorithm 7),
- Applying randomized algorithms for fast TR completion task (Example 3 in section 5).

Before starting the next section, we introduce some concepts and notations used throughout the paper. Tensors, matrices and vectors are denoted by underlined bold upper case, bold upper case and bold lower case letters as $\underline{\mathbf{X}}$, $\mathbf{X}$, $\mathbf{x}$, respectively. The notations 'T' and 'Tr' denote the transpose and the trace of a matrix. The Frobenius norm of a tensor is denoted by $\|.\|_F$. Slices are matrices taken from tensors produced by fixing all but two indices. Slices $\mathbf{X}(:,i_2,:)$ of a 3rd-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times I_3}$, are called lateral slices. We can multiply tensors and matrices. For example, the tensor-matrix multiplication of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and a matrix $\mathbf{Q} \in \mathbb{R}^{K\times I_n}$ along mode n is denoted by $\underline{\mathbf{X}} \times_n \mathbf{Q} \in \mathbb{R}^{I_1\times\cdots\times I_{n-1}\times K\times I_{n+1}\times\cdots\times I_N}$ and defined as follows

$$(\underline{\mathbf{X}} \times_n \mathbf{Q})_{i_1\cdots i_{n-1}\,k\,i_{n+1}\cdots i_N} = \sum_{i_n=1}^{I_N} x_{i_1 i_2\cdots i_N}\, q_{k\,i_n}, \quad k = 1,2,\ldots,K. \tag{1}$$

The same definition can be presented for tensor-vector multiplication. Based on the definition of the tensor-matrix multiplication, when a tensor is multiplied by a vector, the resulting tensor has a mode of size 1. In order to remove the mentioned mode and reduce the order of the resulting tensor, we use a special notation $\bar{\times}$. For example, for $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and $\mathbf{y} \in \mathbb{R}^{I_n}$, we have $\underline{\mathbf{X}}\bar{\times}_n\mathbf{y} \in \mathbb{R}^{I_1\times\cdots\times I_{n-1}\times I_{n+1}\cdots\times I_N}$. A tensor can be reshaped to a matrix and vice versa. These procedures are called matricization (unfolding or flattening) and tensorization, respectively. For a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, the n-unfolding of the tensor $\underline{\mathbf{X}}$, is denoted by $\mathbf{X}_{\langle n\rangle} \in \mathbb{R}^{I_1\cdots I_n\times I_{n+1}\cdots I_N}$, and its components are defined as follows

$$\mathbf{X}_{\langle n\rangle}\left(\overline{i_1\ldots i_n}, \overline{i_{n+1}\ldots i_N}\right) = \underline{\mathbf{X}}(i_1,i_2,\ldots,i_N),$$

where $\overline{i_1 i_2\ldots i_N} = \sum\limits_{k=1}^{N}(i_k-1)J_k,\ J_k = \prod\limits_{m=1}^{k-1} I_m$. A special case of the n-unfolding with only one index for the first coordinate is called mode-n unfolding and is denoted by $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n\times\prod\limits_{i\neq n} I_i}$.

If a tensor is multiplied by a matrix along a specific mode, then its mode-n unfolding can be computed as follows

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}}\times_n\mathbf{A} \;\Leftrightarrow\; \mathbf{Y}_{(n)} = \mathbf{A}\mathbf{X}_{(n)}.$$

For a given data matrix $\mathbf{X}$, operator $\mathrm{SVD}_\delta(\mathbf{X})$ denotes the truncated SVD of $\mathbf{X}$, i.e.

$$\min_{\mathbf{Y}}\ \mathrm{rank}(\mathbf{Y}),\ \text{s.t.}\ \|\mathbf{X}-\mathbf{Y}\|_F < \delta,$$

and the corresponding minimal rank is denoted by $\mathrm{rank}_\delta(\mathbf{X})$.

The Tucker decomposition of a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ admits the following model [10–12]

$$\underline{\mathbf{X}} \cong \underline{\mathbf{S}} \times_1 \mathbf{Q}^{(1)} \times_2 \mathbf{Q}^{(2)} \cdots \times_N \mathbf{Q}^{(N)}, \tag{2}$$

where $\underline{\mathbf{S}} \in \mathbb{R}^{K_1\times K_2\times\cdots\times K_N}$ is the core tensor and the matrices $\mathbf{Q}^{(n)} \in \mathbb{R}^{I_n\times K_n},\ K_n \leq I_n,\ n = 1,2,\ldots,N$ are factor matrices. A shorthand notation for the Tucker decomposition is $\underline{\mathbf{X}} \cong \left[\!\left[\underline{\mathbf{S}};\mathbf{Q}^{(1)},\mathbf{Q}^{(2)},\ldots,\mathbf{Q}^{(N)}\right]\!\right]$.

The N-tuple $(K_1,K_2,\ldots,K_N)$ is called multilinear or Tucker rank. Higher order SVD (HOSVD) [52] is a special Tucker decomposition in which the factor matrices are orthogonal.

The organization of this paper is structured as follows. In section 2, basic randomized algorithms for low rank matrix approximation are introduced. The TR model, its properties and basic algorithms are described in section 3. Adaptive and non-adaptive randomized variants of the algorithms presented in section 3 are elaborated in section 4. Simulations are provided in section 5 to support the presentation and the conclusion is give in section 6.

2. Basic randomized algorithms for large-scale matrices

Randomized algorithms are efficient techniques for computing low rank matrix approximation. The principle idea behind this framework is capturing the range (column space) of a matrix and making reduction in the data matrix in this manner. Note that if the number of rows of a matrix is more than its columns then we should capture its row space. The randomized approaches can be categorized into next three groups:

- **Random projection.** In the random projection approach, a matrix is multiplied by a random matrix such as Gaussian, uniform or Bernoulli matrices[3] on the right-hand side [53]. This procedure may be expensive. In order to accelerate the computation procedure, we can use structured random matrices such sparse random matrices [54, 55], subsampled random Fourier transform [56, 57], subsampled Hadamard transforms, sequence of Givens rotations [57, 58] which are established techniques and have been used extensively in the literature [59].
- **Sampling.** In the sampling techniques [60], a part of columns of the original matrix is selected and reduction is made in the data matrix in this manner.
- **Count-Sketch.** The count-sketch approach [61–63] first labels the columns of the data matrix uniformly. This is also called Hashing. Then, the columns with the same labels are grouped together. Afterwards, the columns of each group are multiplied with ± 1 uniformly[4] and they are summed as representative columns.

In this paper, we only consider the random projection technique although the sampling and the count-sketch techniques are also applicable. We mainly focus on standard and distributed data tensors which means that they can be stored in Random Access Memory (RAM) on a single workstation or distributed among several disks, respectively. We only make a few comments on streaming data sets.

2.1. Fast SVD

After making enough reduction in the data matrix via one of the above-mentioned randomized dimensionality reduction techniques, the original data matrix is projected onto the low rank approximation obtained in the first step. To be more precise, let $\mathbf{X} \in \mathbb{R}^{I\times J}$ be a given data matrix with $\text{rank}(\mathbf{X}) = R \ll \min(I,J)$. In the first step, the matrix $\mathbf{X}$ is multiplied by a random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{J\times R}$ as

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\Omega} \in \mathbb{R}^{I\times R}.$$

In the next step, an orthogonal projector onto the column space of the matrix $\mathbf{X}$, i.e. $\mathbf{Q}\mathbf{Q}^T$, is computed where $\mathbf{Q}$ is an orthonormal basis for the range of $\mathbf{Y}$. The orthonormal basis $\mathbf{Q}$ can be computed through the QR decomposition $\mathbf{Y} = \mathbf{QR}$, where $\mathbf{Q} \in \mathbb{R}^{I\times R}$, and $\mathbf{R} \in \mathbb{R}^{R\times R}$. Here, we have

$$\mathbf{X} \cong \mathbf{Q}\mathbf{Q}^T\mathbf{X},$$

and the compressed matrix $\mathbf{B} = \mathbf{Q}^T\mathbf{X} \in \mathbb{R}^{R\times J}$ is of smaller size than $\mathbf{X}$. The SVD of the original matrix $\mathbf{X}$ is recovered from the SVD of $\mathbf{B} = \mathbf{USV}^T$ as follows

$$\mathbf{X} \cong \mathbf{QU}\boldsymbol{\Sigma}\mathbf{V}^T.$$

This procedure is summarized as follows:

Remark 1. *(**Oversampling technique**) The oversampling technique can be used to improve the solution accuracy of the randomized algorithms. In the oversampling technique, we use additional random vectors (for example $R+P$ random vectors instead of R random vectors) in the first step, i.e. the dimensionality reduction step. In practice, typically $P = 5$ or $P = 10$ is enough to achieve reasonable solutions [53].*

Remark 2. *(**Power iteration technique**) The power iteration technique, is used when the singular values of a data matrix do not decay very fast. Here, we exploit matrix $\mathbf{Z} = \left(\mathbf{X}\mathbf{X}^T\right)^q\mathbf{X}$ (q is a non-negative integer number) instead of the matrix $\mathbf{X}$ and the randomized algorithms are applied to this new matrix. Considering the SVD, $\mathbf{X} = \mathbf{USV}^T$, we have $\left(\mathbf{X}\mathbf{X}^T\right)^q\mathbf{X} = \mathbf{U}\mathbf{S}^{2q+1}\mathbf{V}^T$, and it is seen that the left and right singular vectors of the new matrix $\mathbf{Z}$ are the same as those of the matrix $\mathbf{X}$ but the singular values of $\mathbf{Z}$ have faster decay rate. This can improve the solution accuracy obtained by the randomized algorithms. One should avoid constructing the matrix $\mathbf{Z}$ explicitly because of instability issues and it should be computed iteratively using QR decomposition [53]. It was experimentally confirmed that the power iteration $q = 1$ or $q = 2$ is often enough in practice for achieving accurate solutions [53].*

[3] Matrices whose components are taken from Gaussian, uniform or Bernoulli distributions.
[4] This is also called Rademacher distribution.

Algorithm 1: Randomized SVD algorithm with oversampling and power iteration [53]

```
Input: A data matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, target rank $R$, oversampling $P$ and power iteration $q$
Output: SVD factor matrices $\mathbf{U} \in \mathbb{R}^{I \times R}, \mathbf{S} \in \mathbb{R}^{R \times R}$ and $\mathbf{V} \in \mathbb{R}^{R \times J}$
1 Generate a random matrix $\mathbf{\Omega} \in \mathbb{R}^{J \times (P+R)}$ with a prescribed probability distribution
2 Form $\mathbf{Y} = \left(\mathbf{X}\mathbf{X}^T\right)^q \mathbf{X}\mathbf{\Omega} \in \mathbb{R}^{I \times R}$
3 Compute QR decomposition: $\mathbf{Y} = \mathbf{Q}\mathbf{R}$
4 Compute: $\mathbf{B} = \mathbf{Q}^T\mathbf{X} \in \mathbb{R}^{R \times J}$
5 Compute an SVD, $\mathbf{B} = \overline{\mathbf{U}}\,\overline{\mathbf{S}}\,\overline{\mathbf{V}}^T$
6 $\widetilde{\mathbf{U}} = \mathbf{Q}\overline{\mathbf{U}}$
7 $\mathbf{U} = \widetilde{\mathbf{U}}(:, 1:R), \mathbf{S} = \overline{\mathbf{S}}(1:R, 1:R),\ \mathbf{V} = \overline{\mathbf{V}}(:, 1:R)$
```

- **Reduction**. Replacing an extremely large-scale data matrix with a new one of smaller size compared with the original one capturing either the column or the row space of the original data matrix as much as possible.
- **SVD computation**. Applying deterministic algorithms, e.g. truncated SVD to the reduced data matrix and finding its low rank approximation[5].
- **Recovery**. Recovering the SVD of the original data matrix from the SVD of the compressed one.

The basic form of the randomized SVD algorithm equipped with the oversampling and the power iteration strategies is outlined in Algorithm 1.

2.2. Two-sided randomized algorithm

It is also possible to make reduction on both dimensions of a data matrix when both of them are large. This can be done by simultaneous multiplying a given data matrix with two random matrices from the left and right hand sides. Algorithm 2 outlines the structure of such a randomized algorithm. Both Algorithms 1 and 2 are randomized multi-pass SVD algorithms because both of them need to access (pass) the original data matrix $\mathbf{X}$ two times in Lines 2 and 4. These algorithms can be modified to become single-pass algorithms. To this end, Line 4 in both algorithms can be replaced by alternative representations. For Algorithm 1, we can consider [66]

$$\mathbf{B} \cong (\mathbf{\Omega}_2\mathbf{Q})^{\dagger}\mathbf{W}, \quad \mathbf{W} = \mathbf{\Omega}_2\mathbf{X}, \quad \mathbf{\Omega}_2 \in \mathbb{R}^{R \times I}, \tag{3}$$

and for Algorithm 2, we can consider

$$\mathbf{B} \cong \left(\mathbf{\Omega}_2\mathbf{Q}^{(1)}\right)^{\dagger}\mathbf{W}\left(\mathbf{Q}^{(2)\,T}\mathbf{\Omega}_1\right)^{\dagger}, \quad \mathbf{W} = \mathbf{\Omega}_2\mathbf{X}\mathbf{\Omega}_1. \tag{4}$$

The benefit of these approaches is that they avoid computation of the terms $\mathbf{Q}^T\mathbf{X}$ and $\mathbf{Q}^{(1)\,T}\mathbf{X}\mathbf{Q}^{(2)}$ which may be computationally expensive, especially when the data matrix is stored out-of-core and the cost of communication may exceed our main computations. Instead, in formulations (3) and (4), the original data matrix $\mathbf{X}$ is sketched using the random projection technique and the corresponding matrix $\mathbf{B}$ is obtained by solving some well-conditioned overdetermined linear least-squares problems [66]. The matrix multiplication by a random matrix can be performed relatively fast by employing structured random matrices. We should note that this strategy passes the original data matrix only once because all sketching procedures can be done in the first pass over the raw data[6]. Other types of single-pass techniques can be found in [53, 67–70].

2.3. Randomized matrix rank-revealing (RR) algorithms

In randomized Algorithms 1 and 2, we need an estimation of the matrix rank in advance which may be a difficult task. *Randomized rank-revealing (RR)* or equivalently *randomized fixed-precision algorithms* are able to retrieve the rank of a given data matrix and also the corresponding low-rank matrix approximation automatically. In practice, we use randomized RR Algorithm 3 proposed in [71] which is a modification of the RR algorithm developed in [72]. The operator 'orth' in Lines 6, 8, 9, 11 computes an orthonormal basis for the range of a given data matrix. Also, in the first step, the matrices $\mathbf{Q}$ and $\mathbf{B}$ are empty and they are

[5] In [64], randomization is also used in the second step. This is considered as a randomized algorithm with two-step randomization for Nyström kernel matrix approximation. In the first step, authors use a sub-sampling approach after which they apply randomized SVD instead of deterministic SVD.

[6] The MATLAB codes of several randomized algorithms are provided in https://github.com/wangshusen/RandMatrixMatlab and https://github.com/XuFengthucs/fSVT.

Algorithm 2: Two-Sided Matrix Randomized SVD [53, 65]

```
Input: A data matrix $\mathbf{X} \in \mathbb{R}^{I\times J}$, and target rank $R$
Output: SVD factor matrices $\mathbf{U} \in \mathbb{R}^{I\times R}, \mathbf{S} \in \mathbb{R}^{R\times R}$ and $\mathbf{V} \in \mathbb{R}^{R\times J}$
1 Draw random matrices of prescribed sizes $\mathbf{\Omega}_1 \in \mathbb{R}^{J\times R}$, $\mathbf{\Omega}_2 \in \mathbb{R}^{I\times R}$
2 Compute $\mathbf{Y}_1 = \mathbf{X}\mathbf{\Omega}_1 \in \mathbb{R}^{I\times R}$ and $\mathbf{Y}_2 = \mathbf{X}^T\mathbf{\Omega}_2 \in \mathbb{R}^{J\times R}$
3 Compute QR decompositions $\mathbf{Y}_1 = \mathbf{Q}^{(1)}\mathbf{R}_1$, $\mathbf{Y}_2 = \mathbf{Q}^{(2)}\mathbf{R}_2$
4 Compute $\mathbf{B} = \mathbf{Q}^{(1)\,T}\mathbf{X}\mathbf{Q}^{(2)} \in \mathbb{R}^{R\times R}$
5 Compute an SVD, $\mathbf{B} = \overline{\mathbf{U}}\,\overline{\mathbf{S}}\,\overline{\mathbf{V}}^T$
6 Set $\mathbf{U} = \mathbf{Q}^{(1)}\overline{\mathbf{U}}$, $\mathbf{S} = \overline{\mathbf{S}}$ and $\mathbf{V} = \mathbf{Q}^{(2)}\overline{\mathbf{V}}$
```

Algorithm 3: Randomized Matrix Rank-Revealing Algorithm [71, 72]

```
   Input: A data matrix $\mathbf{X} \in \mathbb{R}^{I\times J}$, approximation error bound $\varepsilon$, block size $b$ and power iteration $q$
   Output: $\mathbf{Q} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{R\times J}$ such that $\|\mathbf{X} - \mathbf{QB}\|_F < \varepsilon$
1     $\mathbf{Q} = []$, $\mathbf{B} = []$
2     $E = \|\mathbf{X}\|_F^2$
3     $i = 0$
4  while $E > \varepsilon^2$ do
5     $\mathbf{\Omega}_i = \mathrm{randn}(J, b)$
6     $\mathbf{Q}_i = \mathrm{orth}(\mathbf{X}\mathbf{\Omega}_i - \mathbf{Q}(\mathbf{B}\mathbf{\Omega}_i))$
7     for $l = 1, 2, \ldots, q$ do
8         $\mathbf{Q}_i = \mathrm{orth}(\mathbf{X}^T\mathbf{Q}_i)$
9         $\mathbf{Q}_i = \mathrm{orth}(\mathbf{X}\mathbf{Q}_i)$
10    end
11    $\mathbf{Q}_i = \mathrm{orth}(\mathbf{Q}_i - \mathbf{Q}(\mathbf{Q}^T\mathbf{Q}_i))$
12    $\mathbf{B}_i = \mathbf{Q}_i^T\mathbf{X}$
13    $\mathbf{Q} = [\mathbf{Q}, \mathbf{Q}_i]$
14    $\mathbf{B} = \begin{bmatrix} \mathbf{B} \\ \mathbf{B}_i \end{bmatrix}$
15    $E = E - \|\mathbf{B}_i\|_F^2$
16    $i = i + 1$
17 end
```

updated sequentially. The algorithm requires an approximation error bound ε, block size b and the power iteration q. For more details on theoretical results of this algorithm[7], we refer to [71, 72].

3. Basic tensor ring (TR) decomposition

Tensor Chain (TC) or Ring (TR) decomposition [17–20] is a tensor network representing a tensor as a ring (chain) of 3rd-order tensors (see figure 1). A special case of the TR decomposition with condition $R_0 = R_N = 1$ is called the Tensor Train (TT) decomposition [16] because it represents a tensor as a train of 3rd-order tensors. For simplicity of presentation, throughout the paper, we only focus on the TR decomposition and all materials naturally hold true for the TT decomposition.

Let $\underline{\mathbf{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, then the TR decomposition of the tensor $\underline{\mathbf{X}}$ admits the following model

$$\underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) \cong \sum_{r_0=1}^{R_0} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} \widehat{\underline{\mathbf{X}}}^{(1)}(r_0, i_1, r_1) \ldots \widehat{\underline{\mathbf{X}}}^{(N)}(r_{N-1}, i_N, r_0), \tag{5}$$

where $\widehat{\underline{\mathbf{X}}}^{(n)} \in \mathbb{R}^{R_{n-1}\times I_n\times R_n}$, $n = 1, 2, \ldots, N$ are called core tensors and the $(N-1)$-tuple $(R_0, R_1, \ldots, R_{N-1})$ is called TR-ranks. Note that in the TR decomposition, we have $R_0 = R_N$ and it is also shown in [20] that the TR-ranks satisfy $R_0R_n \leq \mathrm{rank}\left(\mathbf{X}_{\langle n\rangle}\right)$ for $n = 1, 2, \ldots, N$. Equation (5), is called component-wise TR representation and an equivalent slice-wise representation is

[7] The MATLAB implementation of this algorithm and related efficient randomized algorithms can be found in https://github.com/WenjianYu/randQB_auto.

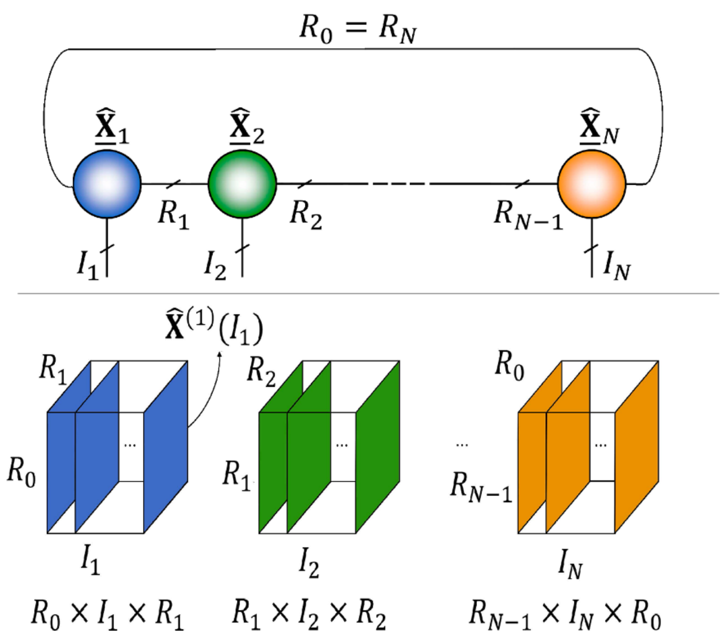


Figure 1. Graphical illustrations of the TR decomposition, Tensor network representation (top), slice-wise representation (bottom) For $R_0 = R_N = 1$, the TR decomposition is simplified into the TT decomposition.

$$\underline{\mathbf{X}}(i_1, i_2, \ldots, i_N) = \mathrm{Tr}\left(\widehat{\mathbf{X}}^{(1)}(i_1) \cdots \widehat{\mathbf{X}}^{(N)}(i_N)\right). \tag{6}$$

Here, $\widehat{\mathbf{X}}^{(n)}(i_n)$ are $R_{n-1} \times R_n$ lateral slices of the core tensors $\widehat{\mathbf{X}}^{(n)}$ for $n = 1, 2, \ldots, N$. In view of (5), introducing an auxiliary index r_0 makes it possible to consider the TR decomposition as a linear combination of R_0 terms of the TT decomposition with partially shared cores. Generally the topological structure of tensor networks can be changed. For example, the TT and TR decompositions can be converted to each other[8] [73, 74].

There are several efficient algorithms for computation of the TR decomposition such as *Sequential SVDs algorithm*, *Alternating Least-Squares (ALS) algorithm*, *ALS with adaptive ranks*[9] and *Block-wise ALS algorithm* [20, 73]. Note that in the ALS-type algorithms, at each iteration, all core tensors are kept fixed except one and the corresponding core tensor is updated. Moreover, the fixed core tensor is alternatively changed and this justifies the name ALS. A closely related algorithm is *modified ALS* (MALS) or *Density Matrix Renormalization Group* (DMRG) algorithm [18, 73, 75–77], where consecutive core tensors are updated simultaneously.

Here, we introduce the TR-SVD algorithm [20] for computing the TR decomposition. It is summarized in Algorithm 4. This algorithm is robust because it relies on the SVD where at each iteration of the algorithm the truncated SVD of the unfolding matrices are computed. It is worth mentioning that the TR-SVD algorithm with initial rank $R_0 = 1$ is equivalent to TT-SVD algorithm [16]. The idea of cross/skeleton or equivalently CUR decomposition [78–81] has been used for the TT decomposition [82–84] which can be naturally used for the TR decomposition [19]. For ALS and MALS-types algorithms see [18, 20, 85, 86]. The TT decomposition can also be computed by Tucker-2 model which is also applicable for the TR decomposition, (see [2, 85]).

Remark 3. *The TR-ranks obtained by Algorithm 4 may not be optimal and often rounding algorithms [16, 87] are used to find a decomposition with lower TR-ranks. Unlike the TT format, the rounding algorithms for mathematical operations in the TR format is more complicated [88]. The TR decomposition as a tensor network contains loop. Hence, it is in general not closed in the Zariski topology [73], section 3,[14, 89, 90]. This means that a sequence of tensors in the TR format may not necessarily converge to a tensor in the TR format, for a detailed theoretical justification see [14, 89, 90]. This may cause instability issues when one wants to find the best TR approximation [18, 73, 91, 92]. That is, the best TR approximation of a given tensor with predefined TR-ranks may not exist and*

[8] The MATLAB codes for converting the TR to the TT and vice versa are available at https://github.com/oscarmickelin/tensor-ring-decomposition.

[9] It is also called incremental rank algorithm.

Algorithm 4: TR-SVD algorithm [20]

```
Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a prescribed approximation error bound $\varepsilon$, and initial rank $R_0$ as a divisor of $\mathrm{rank}_\delta\left(\mathbf{X}_{\langle 1 \rangle}\right)$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that $\left\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\right\|_F \leq \varepsilon \|\underline{\mathbf{X}}\|_F$ and the TR-ranks $(R_0, R_1, \ldots, R_{N-1})$
1  Compute $\delta = \frac{\varepsilon \|\underline{\mathbf{X}}\|_F}{\sqrt{N}}$
2  $\mathbf{C} = \mathrm{reshape}\left(\underline{\mathbf{X}}, [I_1, I_2 I_3 \ldots I_N]\right)$
3  $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \mathrm{SVD}_\delta(\mathbf{C})$
4  Set $R_0 R_1 = \mathrm{rank}(\mathbf{S})$
5  $\widehat{\underline{\mathbf{X}}}^{(1)} = \mathrm{permute}\left(\mathrm{reshape}\left(\mathbf{U}, [I_1, R_0, R_1]\right), [2, 1, 3]\right)$
6  $\underline{\mathbf{C}} = \mathrm{permute}\left(\mathrm{reshape}\left(\mathbf{S}\mathbf{V}^T, \left[R_0, R_1, \prod_{j=2}^N I_j\right]\right), [2, 3, 1]\right)$
7  for $n = 2, \ldots, N-1$ do
8      $\mathbf{C} = \mathrm{reshape}\left(\underline{\mathbf{C}}, \left[R_{n-1} I_n, \frac{\mathrm{numel}(\underline{\mathbf{C}})}{R_{n-1} I_n}\right]\right)$
9      $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \mathrm{SVD}_\delta(\mathbf{C})$
10     $R_n = \mathrm{rank}(\mathbf{S})$
11     $\widehat{\underline{\mathbf{X}}}^{(n)} = \mathrm{reshape}\left(\mathbf{U}, [R_{n-1}, I_n, R_n]\right)$
12     $\underline{\mathbf{C}} = \mathrm{reshape}\left(\mathbf{S}\mathbf{V}^T, \left[R_n, \prod_{j=n+1}^N I_j, R_0\right]\right)$
13 end
14 $\widehat{\underline{\mathbf{X}}}^{(N)} = \mathrm{reshape}\left(\underline{\mathbf{C}}, [R_{N-1}, I_N, R_0]\right)$
```

it can be arbitrarily approximated well by the TR decomposition with lower TR-ranks. In contrast, the best low rank TT decomposition is a well-posed problem [16].

Let τ be a cyclic permutation of the dimensions of a data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and produce a new reshaped data tensor $\underline{\mathbf{X}}^\tau \in \mathbb{R}^{I_{\tau(1)} \times I_{\tau(2)} \times \cdots \times I_{\tau(N)}}$ which is equivalent to

$$\underline{\mathbf{X}}^\tau(i_1, i_2, \ldots, i_N) = \underline{\mathbf{X}}\left(i_{\tau(1)}, i_{\tau(2)}, \ldots, i_{\tau(N)}\right).$$

Assume that the TR representation of the tensor $\underline{\mathbf{X}}^\tau$ is as follows

$$\underline{\mathbf{X}}^\tau(i_1, \ldots, i_N) = \sum_{r_0=1}^{R'_0} \cdots \sum_{r_{N-1}=1}^{R'_{N-1}} \widehat{\underline{\mathbf{X}}}^{(1)}\left(r_0, i_{\tau(1)}, r_1\right) \ldots \widehat{\underline{\mathbf{X}}}^{(N)}\left(r_{N-1}, i_{\tau(N)}, r_0\right)$$

$$= \sum_{r_0=1}^{R'_0} \cdots \sum_{r_{N-1}=1}^{R'_{N-1}} \widehat{\underline{\mathbf{X}}}^{(\tau^{-1}(1))}\left(r_0, i_1, r_1\right) \ldots \widehat{\underline{\mathbf{X}}}^{(\tau^{-1}(N))}\left(r_{N-1}, i_N, r_0\right). \tag{7}$$

where τ^{-1} is the inverse of the cyclic permutation τ. Since there are no boundary on the corner core tensors, the decomposition is invariant to cyclic permutation. However, in practice, the TR-ranks of the permuted tensor $\underline{\mathbf{X}}^\tau$ may be different from the TR-ranks of the tensor $\underline{\mathbf{X}}$ and each cyclic permutation of indices may provide different TR-ranks.

It turns out that two main issues underlying the compression performance of the TR decomposition are

- Cyclic shifts which leads to a suboptimal model
- Initial rank R_0 of the first core tensor

More precisely, choosing different cyclic shifts and an initial rank R_0 may lead to different TR decompositions with different number of parameters. In particular, for different initial ranks, it is shown that it is impossible to find a common minimal rank, see ([87], Proposition 2.2). These facts imply that to find a TR decomposition with suboptimal TR-ranks, it is necessary to check all cyclic shifts and possible initial ranks. This procedure is called *Reduced storage TR-SVD* [87] and is summarized in Algorithm 5. Clearly this algorithm is computationally expensive for high order tensors and because of this issue, a heuristic algorithm called *Heuristic TR-SVD* is developed [87] in which the procedure of initial rank and cyclic permutation selections are performed heuristically. This procedure is summarized in Algorithm 6 and it essentially consists of two parts as follows:

Algorithm 5: TR-SVD with all possible cyclic permutations (see also [87])

```
Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a prescribed approximation error bound $\varepsilon$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that $\left\| \underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}} \right\| \leq \varepsilon \|\underline{\mathbf{X}}\|_F$ and the TR-ranks $(R_0, R_1, \ldots, R_{N-1})$
1  Set $\tau = (1, N, N-1, \ldots, 2)$
2  for $n = 1, 2, \ldots, N$ do
3      Set $\gamma_n = \tau^{n-1}$
4      $E_n$ = Compute all divisors of $\mathrm{rank}_\delta\left(\mathbf{X}^{\gamma_n}_{\langle 1 \rangle}\right)$
5      for All $r \in E_n$ do
6          Set $R_0 = r$
7          Apply Algorithm 4 to the permuted tensor $\underline{\mathbf{X}}^{\gamma_n}$ with initial rank $R_0$ and obtain core tensors $\widetilde{\underline{\mathbf{X}}}^{(n)}\left(i_{\gamma_n^{-1}(k)}\right)$,
           $k = 1,2,\ldots,N$
8      end
9  end
10 Find the optimal TR decomposition with the least number of parameters corresponding to the cyclic permutation $\gamma_{n^*}$ and the initial rank $R_0$, i.e. core tensors $\widetilde{\underline{\mathbf{X}}}^{(k)}\left(i_{\gamma_{n^*}^{-1}(k)}\right)$ for $k = 1, 2, \ldots, N$
11 Set core tensors $\widehat{\underline{\mathbf{X}}}^{(k)}(i_k) \equiv \widetilde{\underline{\mathbf{X}}}^{(\gamma_{n^*}(k))}(i_k)$, $k = 1, 2, \ldots, N$
```

- Preprocessing part to find a sub-optimal cyclic shift and an initial rank R_0 (in heuristic manner),
- Applying the TR-SVD with those parameters obtained from the first step.

Assume $\tau = (1, N, N-1, \ldots, 2)$ and consider all cyclic permutations produced by the generator τ as $\gamma_n = \tau^{n-1}$ for $n = 1, 2, \ldots, N$.

In the heuristic algorithm, firstly a cyclic permutation γ_{n_*} is chosen by solving the following minimization problem

$$\widehat{R}_{\gamma_{n_*}} = \underset{n=1,2,\ldots,N}{\arg\min} \ \mathrm{rank}\left(\mathbf{X}^{\gamma_n}_{\langle 2 \rangle}\right). \tag{8}$$

Afterwards, the initial rank R_0 corresponding to the cyclic permutation γ_{n^*} selected in the first step, is found by solving the following minimization problem [87]

$$\underset{R_0 | \mathrm{rank}_\delta\left(\mathbf{X}^{\gamma_{n_*}}_{\langle 1 \rangle}\right)}{\arg\min} \quad g, \tag{9}$$

where

$$g = \left| \widehat{R}_{\gamma_{\{n_* - 1\}}} - \underbrace{\frac{\mathrm{rank}_\delta\left(\mathbf{X}^{\gamma_{n_*}}_{\langle 1 \rangle}\right)}{R_0}}_{R_1} \right| + \left| \widehat{R}_{\gamma_{n_*}} - R_0 \right|. \tag{10}$$

The main algorithms proposed so far are deterministic using truncated or economic SVD which are quite expensive for big data matrices. Next we present the randomized variants of the mentioned algorithms.

4. Randomized tensor ring (TR) decomposition

The main computationally expensive part of Algorithms 4-6 is computation of the truncated SVD of the unfolding matrices. Exploiting the randomized algorithms can speed-up these algorithms for the TR decomposition. Following this strategy, in this section, randomized algorithms for the TR decomposition are introduced.

As mentioned, we only focus on the random projection technique. The sampling and the count-sketch strategies can be applied straightforwardly. For example, in [19], the cross decomposition was used instead of the SVD. Here, columns and rows are sampled in heuristic ways. Also in [93], the sampling technique is used within the ALS algorithm which scales linearly in the tensor order. The problem is treated as a tensor with missing components (only $\mathcal{O}(N)$ known components) and the ALS-TR algorithm applied to this uncompleted data tensor to simultaneously recover the data tensor and also decompose it into the TR format.

Recently, in order to reduce the high computational cost of the standard TT-SVD algorithm, two types of randomized algorithms have been proposed in [94–96] as follows

Algorithm 6: Simplified TR-SVD (see also [87])

```
Input: A tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a prescribed approximation error bound $\varepsilon$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that $\left\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\right\|_F \leq \varepsilon \|\underline{\mathbf{X}}\|_F$ and the TR-ranks $\{R_0, R_1, \ldots, R_{N-1}\}$
1 Set $\tau = (1, N, N-1, \ldots, 2)$
2 for $n = 1, 2, \ldots, N$ do
3     Set $\gamma_n = \tau^{n-1}$
4     $\widehat{R}_{\gamma_n} = \text{rank}\left(\mathbf{X}^{\gamma_n}_{\langle 2 \rangle}\right)$
5 end
6 $\widehat{R}_{\gamma_{n^*}} = \underset{n=1,2,\ldots,N}{\arg\min} \ \widehat{R}_{\gamma_n}$
7 Compute $R_0^*$ by solving optimization problem (9)
8 Apply Algorithm 4 to the permuted tensor $\underline{\mathbf{X}}^{\gamma_{n^*}}$ with the initial rank $R_0^*$ and obtain core tensors
  $\widetilde{\underline{\mathbf{X}}}^{(n)}\left(i_{\gamma^{-1}_{n^*}(n)}\right), n = 1, 2, \ldots, N$
9 Set core tensors $\widehat{\underline{\mathbf{X}}}^{(n)}(i_n) \equiv \widetilde{\underline{\mathbf{X}}}^{(\gamma_{n^*}(n))}(i_n)$, $n = 1, 2, \ldots, N$
```

- Random projection TT algorithm,
- Adaptive randomized TT algorithm.

The random projection TT algorithm is a variant of the TT-SVD algorithm where at each iteration of the algorithm, random projection technique is used to find low rank approximations of unfolding matrices. This procedure is outlined in Algorithm 7.

We discuss this idea for the general setting of the TR decomposition. To be more clear, we explain one iteration of Algorithm 7. In the first iteration of Algorithm 7, $\mathbf{C}_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3 \cdots I_N}$ is the mode-1 unfolding matrix of the Nth-order tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Considering random Gaussian matrix $\boldsymbol{\Omega}$ of conforming dimension and taking into account the oversampling P, we have

$$\boldsymbol{\Omega} \in \mathbb{R}^{I_2 I_3 \cdots I_N \times (R_0 R_1 + P)} \quad \Rightarrow \quad \mathbf{C}_{(1)} \boldsymbol{\Omega} \in \mathbb{R}^{I_1 \times (R_0 R_1 + P)}.$$

In order to find an orthonormal basis for the range of matrix $\mathbf{C}_{(1)}\boldsymbol{\Omega}$, the QR decomposition of mentioned matrix is computed as follows

$$\mathbf{C}_{(1)} \boldsymbol{\Omega} = \mathbf{Q}\,\mathbf{R}, \quad \mathbf{Q} \in \mathbb{R}^{I_1 \times (R_0 R_1 + P)}.$$

The first $R_0 R_1$ columns of matrix $\mathbf{Q}$ are taken as orthonormal basis for range of $\mathbf{C}_{(1)}$ or

$$\mathbf{Q}^{(1)} = \mathbf{Q}\,(:, 1 : R_0 R_1)\,.$$

Since $\mathbf{Q}^{(1)}\,\mathbf{Q}^{(1)\,T}$ is an orthogonal projection onto the range of $\mathbf{C}_{\langle 1 \rangle}$, we have

$$\mathbf{C}_{(1)} \approx \mathbf{Q}^{(1)} \left(\mathbf{Q}^{(1)\,T} \mathbf{C}_{(1)}\right), \quad \mathbf{Q}^{(1)} \in \mathbb{R}^{I_1 \times R_0 R_1}.$$

Two terms $\mathbf{Q}^{(1)}$ and $\mathbf{Q}^{(1)\,T}\mathbf{C}_{(1)}$ are reshaped into tensors of conforming dimensions in the following manners (see figure 2 for graphical illustration)

$$\text{Reshaping \ \& \ Permutation}: \quad \mathbf{Q}^{(1)} \quad \Rightarrow \quad \widehat{\underline{\mathbf{X}}}^{(1)} \in \mathbb{R}^{R_0 \times I_1 \times R_1},$$

$$\text{Tensorization}: \quad \mathbf{Q}^{(1)\,T}\mathbf{C}_{(1)} \quad \Rightarrow \quad \underline{\mathbf{C}} \times_1 \mathbf{Q}^{(1)\,T} \in \mathbb{R}^{R_0 R_1 \times I_2 \times \cdots \times I_N}.$$

In the next step, the tensor $\underline{\mathbf{C}} \times_1 \mathbf{Q}^{(1)\,T}$ is reshaped to a 3rd order tensor as

$$\underline{\mathbf{C}} = \text{permute}\left(\text{reshape}\left(\underline{\mathbf{C}} \times_1 \mathbf{Q}^{(1)\,T}, [R_0, R_1, I_2 \ldots I_N]\right), [2, 3, 1]\right),$$
$$\underline{\mathbf{C}} = \text{reshape}\left(\underline{\mathbf{C}}, [R_1 I_2, I_3 \ldots I_N, R_0]\right),$$

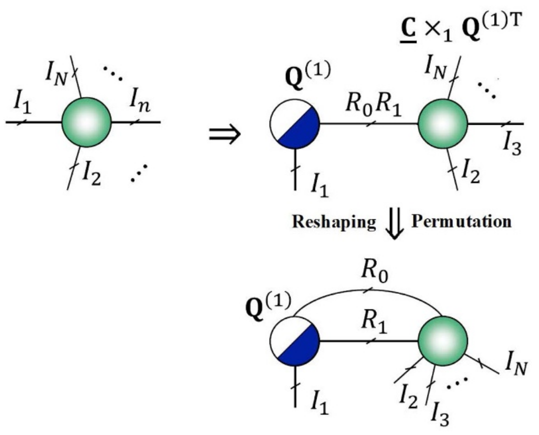


Figure 2. The first step of randomized projection TR-SVD algorithm.

and the procedure is continued with the tensor $\underline{\mathbf{C}}$. In general, in the nth iteration of Algorithm 7, the following reshaped 3rd order tensor is considered

$$\underline{\mathbf{C}} \Leftarrow \text{reshape}\left(\underline{\mathbf{C}}, [R_{n-1}I_n, I_{n+1}I_{n+2}\dots I_N, R_0]\right).$$

Similar to the procedure discussed above, first its mode-1 unfolding, i.e. $\mathbf{C}_{(1)} \in \mathbb{R}^{R_{n-1}I_n \times I_{n+1}\dots I_N R_0}$ is computed. Then from the following randomized low rank matrix approximation

$$\mathbf{C}_{(1)} \approx \mathbf{Q}^{(n)}\left(\mathbf{Q}^{(n)\,T}\mathbf{C}_{(1)}\right), \quad \mathbf{Q}^{(n)} \in \mathbb{R}^{I_nR_{n-1}\times R_n}. \tag{11}$$

the nth core tensor $\underline{\widehat{\mathbf{X}}}^{(n)}$ can be computed as

$$\textit{Reshaping \& Permutation}: \quad \mathbf{Q}^{(n)} \quad \Rightarrow \quad \underline{\widehat{\mathbf{X}}}^{(n)} \in \mathbb{R}^{R_{n-1}\times I_n \times R_n},$$

$$\text{Tensorization}: \mathbf{Q}^{(n)\,T}\mathbf{C}_{(1)} \;\Rightarrow\; \underline{\mathbf{C}} \times_1 \mathbf{Q}^{(n)\,T} \in \mathbb{R}^{R_n \times I_{n+1}\dots I_N \times R_0}.$$

Remark 4. *Algorithm 7 can be equipped with the power iteration technique when the data tensor is corrupted by a level of noise or equivalently the singular values of the unfolding matrices do not decay very fast and oversampling technique may not provide satisfactory approximations. To that end, in Algorithm 7 we replace* $\mathbf{Z} = \mathbf{C}\boldsymbol{\Omega}$ *in Lines 3 and 11, by* $\mathbf{Z} = \left(\mathbf{C}\mathbf{C}^T\right)^q \mathbf{C}\boldsymbol{\Omega}$.

Algorithm 7 needs an estimation of the TR-ranks in advance because it is necessary to have estimation of the unfolding matrices ranks for the projection step in Lines 2 and 11. This imposes a restriction on it because we may not have any information about the TR-ranks. It is of interest to choose the TR-ranks of tensors adaptively during running the algorithm.

Estimating the TR-rank R_n in (11) is equivalent to finding an orthogonal matrix $\mathbf{Q}^{(n)} \in \mathbb{R}^{R_{n-1}I_n \times R_n}$ satisfying

$$\mathbf{C}_{(1)} \approx \left(\mathbf{Q}^{(n)}\mathbf{Q}^{(n)\,T}\right)\mathbf{C}_{(1)} \quad \Rightarrow \underline{\mathbf{C}} \approx \underline{\mathbf{C}} \times_1 \left(\mathbf{Q}^{(n)}\mathbf{Q}^{(n)\,T}\right).$$

This can be equivalently reformulated as the following problem:

Problem 1. Suppose that $\underline{\mathbf{C}} \in \mathbb{R}^{I_nR_{n-1}\times I_{n+1}\times\cdots\times I_N}$ and ε is a prescribed approximation error bound. The objective is to find a columnwise orthogonal matrix $\mathbf{Q}^{(n)} \in \mathbb{R}^{I_nR_{n-1}\times R_n}$ with $R_n \leq I_nR_{n-1}$, such that

$$\left\|\underline{\mathbf{C}} - \underline{\mathbf{C}} \times_1 \left(\mathbf{Q}^{(n)}\mathbf{Q}^{(n)\,T}\right)\right\|_F = \left\|\underline{\mathbf{C}} \times_1 \left(\mathbf{I} - \mathbf{Q}^{(n)}\mathbf{Q}^{(n)\,T}\right)\right\|_F \leq \epsilon, \tag{12}$$

where $\mathbf{I}$ is the identity matrix of size $R_{n-1}I_n \times R_{n-1}I_n$. Problem 1, can be solved by Algorithm 8 or 9, for a detailed study on these algorithms see [53, 94]. Note that in Algorithm 8 a stopping criterion can be either a predefined maximum number of iterations or a predefined approximation error bound. An adaptive randomized algorithm based on this idea is summarized in Algorithm 10. At each iteration of Algorithm 10,

Algorithm 7: Random projection TR-SVD algorithm (see also [94])

```
Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a prescribed approximation error bound $\varepsilon$, and TR-ranks
$(R_0, R_1, \ldots, R_{N-1})$, oversampling $P$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that
$\left\| \underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}} \right\|_F \leq \varepsilon \|\underline{\mathbf{X}}\|_F$
1 $\mathbf{C} = \mathrm{reshape}\,(\underline{\mathbf{X}}, [I_1, I_2 I_3 \ldots I_N])$
2 Compute $\mathbf{Z} = \mathbf{C}\boldsymbol{\Omega}$ where $\boldsymbol{\Omega} \in \mathbb{R}^{I_2 I_3 \cdots I_N \times (R_0 R_1 + P)}$
3 Compute $\mathbf{Q}$ as a columnwise orthogonal basis of $\mathbf{Z}$ by using the QR decomposition
4 Let $\mathbf{Q}^{(1)} = \mathbf{Q}\,(:, 1 : R_0 R_1)$
5 $\widehat{\underline{\mathbf{X}}}^{(1)} = \mathrm{permute}\left(\mathrm{reshape}\left(\mathbf{Q}^{(1)}, [I_1, R_0, R_1]\right), [2, 1, 3]\right)$
6 Compute $\underline{\mathbf{C}} = \underline{\mathbf{X}} \times_1 \mathbf{Q}^{(1)T}$
7 $\underline{\mathbf{C}} = \mathrm{permute}\,(\mathrm{reshape}\,(\underline{\mathbf{C}}, [R_0, R_1, I_2 \ldots I_N]), [2, 3, 1])$
8 $\underline{\mathbf{C}} = \mathrm{reshape}\,(\underline{\mathbf{C}}, [R_1 I_2, I_3 \ldots I_N, R_0])$
9 for $n = 2, \ldots, N-1$ do
10    $\mathbf{C} = \mathrm{reshape}\,(\underline{\mathbf{C}}, [R_{n-1} I_n, I_{n+1} \ldots I_N R_0])$
11    Compute $\mathbf{Z} = \mathbf{C}\boldsymbol{\Omega}$ where $\boldsymbol{\Omega} \in \mathbb{R}^{I_{n+1} I_{n+2} \cdots I_N R_0 \times (R_n + P)}$
12    Compute $\mathbf{Q}$ as a columnwise orthogonal basis of $\mathbf{Z}$ by using the QR decomposition;
13    Let $\mathbf{Q}^{(n)} = \mathbf{Q}\,(:, 1 : R_n)$
14    $\widehat{\underline{\mathbf{X}}}^{(n)} = \mathrm{reshape}\left(\mathbf{Q}^{(n)}, [R_{n-1}, I_n, R_n]\right)$
15    Compute $\underline{\mathbf{C}} = \underline{\mathbf{C}} \times_1 \mathbf{Q}^{(n)T}$
16    $\underline{\mathbf{C}} = \mathrm{reshape}\,(\underline{\mathbf{C}}, [R_{n-1} I_n,\ I_{n+1} \ldots I_N,\ R_0])$
17 end
18 $\widehat{\underline{\mathbf{X}}}^{(N)} = \mathrm{reshape}\,(\mathbf{C}, [R_{N-1}, I_N, R_0])$
```

Algorithm 8: Pseudocode for solving **Problem 1** [53, 94]

```
Input: A data tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_n R_{n-1} \times I_{n+1} \times \cdots \times I_N}$
Output: A columnwise orthogonal matrix $\mathbf{Q}^{(n)} \in \mathbb{R}^{I_n R_{n-1} \times R_n}$ satisfying (12)
1 $k = 1,\ \mathbf{Q}^{(0)} = 0$
2 while a stopping criterion is not satisfied do
3    Draw $N - n$ standard Gaussian vectors $\boldsymbol{\omega}_m \in \mathbb{R}^{I_m}$ with $m = n+1, n+2, \ldots, N$
4    Compute $\mathbf{y}_k = \underline{\mathbf{C}}\ \bar{\times}_{n+1}\ \boldsymbol{\omega}_{n+1}\ \bar{\times}_{n+2} \cdots \bar{\times}_N\ \boldsymbol{\omega}_N$
5    if $k > 1$ then
6       Compute $\mathbf{y}_k = \left(\mathbf{I} - \mathbf{Q}^{(k-1)}\,\mathbf{Q}^{(k-1)\,T}\right)\mathbf{y}_k$
7    end
8    Normalize $\mathbf{q}_k = \frac{\mathbf{y}_k}{\|\mathbf{y}_k\|_2}$ and form $\mathbf{Q}^{(k)} = \left[\mathbf{Q}^{(k-1)}, \mathbf{q}_k\right]$
9    $k = k + 1$
10 end
```

Problem 1 is numerically solved and both an estimation rank R_n and also corresponding columnwise orthogonal matrix $\mathbf{Q}^{(n)}$ are computed.

It is also possible to use randomized RR Algorithm 3 (also see Algorithm 5 in [97]). within the TR-SVD. This procedure is summarized in Algorithm 7 and we refereed to it as Randomized RRTR-SVD algorithm. Please note that in Lines 3 and 9 of Algorithm 7, by Rank-Revealing Algorithm, we mean Algorithm 3. Recently, this idea has been used for the Tucker decomposition [98] and here we utilize it for the TR decomposition.

Following the idea of computation of CANDECOMP/PARAFAC decomposition (CPD) [99–101] with a prior fast randomized HOSVD compression [102], in [103], a randomized algorithm was proposed for computation of the TR decomposition based on a prior Tucker compression. The idea is utilizing a randomized Tucker decomposition in the first step as a preprocessing step after which the deterministic algorithms such as Algorithm 4, Algorithm 5, Algorithm 6 or TR-ALS Algorithm [20] can be applied to the smaller Tucker core tensor.

The randomized Tucker compression is summarized in Algorithm 12 and the TR decomposition with a prior Tucker compression is outlined in Algorithm 13 (see figure 3 for graphical illustration). Please note that Algorithms 12 is also called randomized Sequentially Truncated HOSVD (r-STHOSVD) [98, 104]. A main drawback with Algorithm 13, is that it needs an estimation of the multilinear rank of the original data tensor

Algorithm 9: Pseudocode for solving **Problem 1** (See also [53, 94])

```
Input: A data tensor $\underline{\mathbf{C}} \in \mathbb{R}^{I_n R_{n-1} \times I_{n+1} \times \cdots \times I_N}$, an integer $P$, an approximation error bound $\varepsilon$, a Boolean flag "take max" and maximum number of iterations $Iter_{\max}$
Output: A columnwise orthogonal matrix $\mathbf{Q}^{(n)} \in \mathbb{R}^{I_n R_{n-1} \times R_n}$ satisfying (12)
1  Draw $N-n$ independent families $\left\{\boldsymbol{\omega}_m^{(p)} \in \mathbb{R}^{I_m} : p = 1, 2, \ldots, P\right\}$ of standard Gaussian vectors where $m = n+1, n+2, \ldots, N$
2  Compute $\mathbf{y}_p = \underline{\mathbf{C}} \bar{\times}_{n+1} \boldsymbol{\omega}_{n+1}^{(p)} \bar{\times}_{n+2} \cdots \bar{\times}_N \boldsymbol{\omega}_N^{(p)}$ with $p = 1, 2, \ldots, P$
3  Start with an empty basis matrix $\mathbf{Q}^{(0)}$ and set $k = 0$
4  while $\max\{\|\mathbf{y}_{k+1}\|_2, \|\mathbf{y}_{k+1}\|_2, \ldots, \|\mathbf{y}_{k+P}\|_2\} > \varepsilon$ or $k < Iter_{\max}$ do
5      Set $k = k+1$
6      if the value of 'take max' is 'True' then
7          Choose $k_0 \in \{k+1, \ldots, k+P\}$ such that
           $\|\mathbf{y}_{k_0}\|_2 = \max\{\|\mathbf{y}_{k+1}\|_2, \|\mathbf{y}_{k+2}\|_2, \ldots, \|\mathbf{y}_{k+P}\|_2\}$
           $\mathbf{y}_k = \left(\mathbf{I} - \mathbf{Q}^{(k-1)} \mathbf{Q}^{(k-1)\,T}\right) \mathbf{y}_{k_0}$ if $k > 1$
8      else
9          $\mathbf{y}_k = \left(\mathbf{I} - \mathbf{Q}^{(k-1)} \mathbf{Q}^{(k-1)\,T}\right) \mathbf{y}_k$ if $k > 1$
10     end
11     Compute $\mathbf{q}_k = \frac{\mathbf{y}_k}{\|\mathbf{y}_k\|_2}$ and form $\mathbf{Q}^{(k)} = \left[\mathbf{Q}^{(k-1)}, \mathbf{q}_k\right]$
12     Draw $N-n$ standard Gaussian vectors $\boldsymbol{\omega}_m \in \mathbb{R}^{I_m}$ where $m = n+1, n+2, \ldots, N$
13     Compute $\mathbf{y}_{k+P} = \left(\mathbf{I} - \mathbf{Q}^{(k)} \mathbf{Q}^{(k)\,T}\right) (\underline{\mathbf{C}} \bar{\times}_{n+1} \boldsymbol{\omega}_{n+1} \bar{\times}_{n+2} \cdots \bar{\times}_N \boldsymbol{\omega}_N)$
14     for $i = k+1, k+2, \ldots, k+P-1$ do
15         $\mathbf{y}_i = \mathbf{y}_i - (\mathbf{q}_k^T \mathbf{y}_i)\, \mathbf{q}_k$
16     end
17 end
18 Set $\mathbf{Q} = \mathbf{Q}^{(n)}$ and $R_n$ as the number of all columns of $\mathbf{Q}^{(n)}$.
```

which may be difficult. However, an adaptive algorithm, e.g. Algorithm 3, can be used in Algorithm 12 to find the factor matrices and their corresponding multilinear rank automatically[98]. This procedure is summarized in Algorithm 14.

It has been shown in [105], that the computational complexity of the TT-SVD algorithm for decomposing an Nth-order tensor of size $I \times I \times \cdots \times I$ and the TT-ranks $(R, R, \ldots, R)$ is $\mathcal{O}\left(I^N R^2\right)$, due to the computation of N SVD of the unfolding matrices, (Theorem 2.1 in Page 2136). Since the computational complexity of the TR-SVD algorithm is the same as the TT-SVD algorithm, we have the same complexity for the TR-SVD algorithm. The idea of decomposing tensors in the TT format with a prior Tucker compression was first proposed in [16]. The computational complexity of TR-SVD (with a prior Tucker decomposition) of an Nth-order tensor of size $I \times I \times \cdots \times I$, with the Tucker rank $\left(\tilde{R}, \tilde{R}, \ldots, \tilde{R}\right)$ and the TR-ranks $(R, R, \ldots, R)$ is $\mathcal{O}\left(I^N \tilde{R} + \tilde{R}^N R^2\right)$.

The first term is the cost for multiplying the unfolding matrices with random matrices which is the most expensive operation. The second term is for the TR decomposition of the core Tucker tensor. So, if $\tilde{R} < R^2$, then the approach of the TR decomposition with a prior Tucker compression is cheaper.

We should emphasize that the TR decomposition with a prior Tucker compression is applicable when

- The underlying data tensor is of small order (up to 5) otherwise the curse of dimensionality occurs.
- The Tucker core tensor admits a low multilinear rank.

Concerning Algorithms 5 and 6, the complexity is more involved because they need several permutations of modes. They are not applicable directly to very large-scale tensors and a prior Tucker compression can somewhat reduce the computational complexity. Please note that all algorithms discussed in this paper achieve a suboptimal compression ratio and developing algorithms for finding the optimal model is a challenging topic that needs to be investigated. For example, in [85], novel algorithms are developed for the TT decomposition.

Remark 5. *For decomposing a streaming data tensor in the TT format, specialized algorithms should be used. Recently a streaming TT decomposition is developed in [106] with applications in cyber-physical big data. Further applications of the streaming TT decomposition in DNNs can be found in [107].*

Remark 6. *Recently an efficient technique called TT-HSVD was proposed in [108, 109] for decomposing tensors in the TT format in which the core tensors can be computed in parallel. This is in contrast to the TT-SVD in which at*

Algorithm 10: Adaptive Random projection TR-SVD algorithm (see also [94])

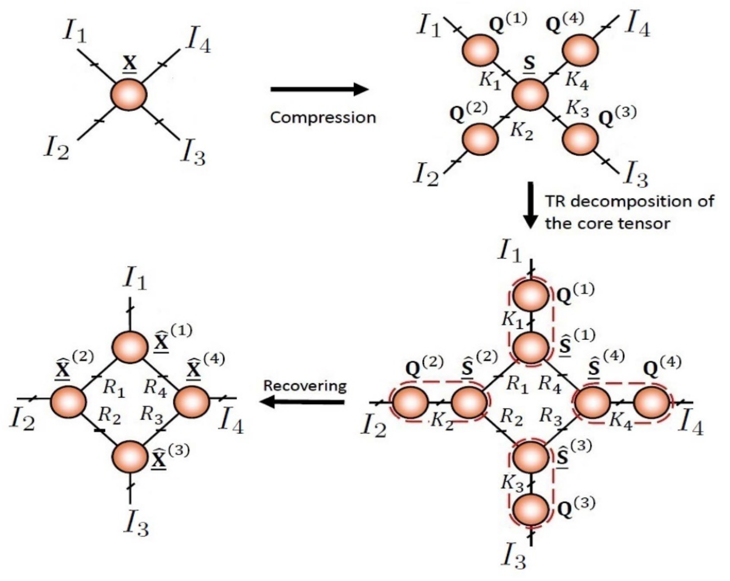
```
Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a prescribed approximation error bound $\varepsilon$, a positive number $P$ and an
initial rank $R_0$ as a divisor of $\mathrm{rank}_\delta\left(\mathbf{X}_{\langle 1 \rangle}\right)$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that
$\left\| \underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}} \right\|_F \leq \varepsilon \|\underline{\mathbf{X}}\|_F$
1  Apply Algorithm 3 to the $\mathbf{C}_{\langle 1 \rangle}$ and generate the columnwise orthogonal matrix $\mathbf{Q}^{(1)} \in \mathbb{R}^{I_1 \times R_0 R_1}$
2  $\widehat{\underline{\mathbf{X}}}^{(1)} = \mathrm{permute}\left(\mathrm{reshape}\left(\mathbf{Q}^{(1)}, [I_1, R_0, R_1]\right), [2, 1, 3]\right)$
3  Compute $\underline{\mathbf{C}} = \underline{\mathbf{X}} \times_1 \mathbf{Q}^{(1)\,T}$
4  $\underline{\mathbf{C}} = \mathrm{permute}\left(\mathrm{reshape}\left(\underline{\mathbf{C}}, [R_0, R_1, I_2 \ldots I_N]\right), [2, 3, 1]\right)$
5  $\underline{\mathbf{C}} = \mathrm{reshape}\left(\underline{\mathbf{C}}, [R_1 I_2, I_3 \ldots I_N, R_0]\right)$
6  for $n = 2, \ldots, N-1$ do
7      $\underline{\mathbf{C}} = \mathrm{reshape}\left(\underline{\mathbf{C}}, [R_{n-1} I_n,\ I_{n+1} \ldots I_N R_0]\right)$
8      Apply Algorithm 8 or 9 to the tensor $\underline{\mathbf{C}}$ for solving Problem 1 and generate the columnwise orthogonal
         matrix $\mathbf{Q}^{(n)} \in \mathbb{R}^{I_n R_{n-1} \times R_n}$
9      $\widehat{\underline{\mathbf{X}}}^{(n)} = \mathrm{reshape}\left(\mathbf{Q}^{(n)}, [R_{n-1}, I_n, R_n]\right)$
10     Compute $\underline{\mathbf{C}} = \underline{\mathbf{C}} \times_1 \mathbf{Q}^{(n)\,T}$
11     $\underline{\mathbf{C}} = \mathrm{reshape}\left(\underline{\mathbf{C}}, [R_{n-1} I_n,\ I_{n+1} \ldots I_N,\ R_0]\right)$
12 end
13 $\widehat{\underline{\mathbf{X}}}^{(N)} = \mathrm{reshape}\left(\mathbf{C}, [R_{N-1}, I_N, R_0]\right)$
```

Algorithm 11: Randomized RRTR-SVD Algorithm

```
Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, a prescribed approximation error bound $\varepsilon$, a power iteration $q$ and an
initial rank $R_0$ as a divisor of $\mathrm{rank}_\delta\left(\mathbf{X}_{\langle 1 \rangle}\right)$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that
$\left\| \underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}} \right\|_F \leq \varepsilon \|\underline{\mathbf{X}}\|_F$ and the TR-ranks are $\{R_0, R_1, \ldots, R_{N-1}\}$
1  Compute $\delta = \frac{\varepsilon \|\underline{\mathbf{X}}\|_F}{\sqrt{N}}$
2  $\mathbf{C} = \mathrm{reshape}\left(\underline{\mathbf{X}}, [I_1, I_2 I_3 \ldots I_N]\right)$
3  $[\mathbf{Q}, \mathbf{B}, \widetilde{R}_1] = \text{Rank-Revealing Algorithm}(\mathbf{C}, \epsilon, b, q)$
4  $R_1 = \widetilde{R}_1 / R_0$
5  $\widehat{\underline{\mathbf{X}}}^{(1)} = \mathrm{permute}\left(\mathrm{reshape}\left(\mathbf{Q}(:, 1:R_0), [I_1, R_0, R_1]\right) [2, 1, 3]\right)$
6  $\underline{\mathbf{C}} = \mathrm{permute}\left(\mathrm{reshape}\left(\mathbf{B}, \left[R_0, R_1, \prod_{j=2}^{N} I_j\right]\right), [2, 3, 1]\right)$
7  for $n = 2, \ldots, N-1$ do
8      $\mathrm{C} = \mathrm{reshape}\left(\underline{\mathbf{C}}, \left[R_{n-1} I_n, \frac{\mathrm{numel}(\underline{\mathbf{C}})}{R_{n-1} I_n}\right]\right)$
9      $[\mathbf{Q}, \mathbf{B}, R_n] = \text{Rank-Revealing Algorithm}(\mathbf{C}, \epsilon, b, q)$
10     $\widehat{\underline{\mathbf{X}}}^{(n)} = \mathrm{reshape}\left(\mathbf{Q}(:, 1:R_n), [R_{n-1}, I_n, R_n]\right)$
11     $\underline{\mathbf{C}} = \mathrm{reshape}\left(\mathbf{B}(1:R_n, :), \left[R_n, \prod_{j=n+1}^{N} I_j, R_0\right]\right)$
12 end
13 $\widehat{\underline{\mathbf{X}}}^{(N)} = \mathrm{reshape}\left(\underline{\mathbf{C}}, [R_{N-1}, I_N, R_0]\right)$
```

each step just one core tensor is computed. This technique can be generalized for the TR decomposition employing randomization for further acceleration in computations.

Remark 7. *Consider Algorithm 1 which is the basic form of randomized algorithms for low rank approximation of matrices [53]. It has been recently shown in that for large sparse matrices, Algorithm 1 or its variants can be totally improved by prior transformation of a given data matrix into the TT matrix format (equivalently Matrix Product Operator (MPO))[105], and performing all computations in the TT matrix format. This algorithm is called tensor train randomized SVD (TTrSVD) algorithm and simulations have shown that for some experiments, it can achieve more than 10 times speed up for the TT decomposition compared with tensor-based alternating least squares SVD (ALS-SVD) [27] and modified alternating least squares SVD (MALS-SVD) matrix approximation methods [27] even with better accuracy.*



I_1
I_4
I_2
I_3
$\underline{\mathbf{X}}$
Compression
$\mathbf{Q}^{(1)}$
$\mathbf{Q}^{(4)}$
$\underline{\mathbf{S}}$
K_1
K_4
$\mathbf{Q}^{(2)}$
K_2
K_3
$\mathbf{Q}^{(3)}$
TR decomposition of the core tensor
$\widehat{\underline{\mathbf{S}}}^{(1)}$
$\widehat{\underline{\mathbf{S}}}^{(2)}$
$\widehat{\underline{\mathbf{S}}}^{(3)}$
$\widehat{\underline{\mathbf{S}}}^{(4)}$
R_1 R_4 R_2 R_3
Recovering
$\widehat{\underline{\mathbf{X}}}^{(1)}$
$\widehat{\underline{\mathbf{X}}}^{(2)}$
$\widehat{\underline{\mathbf{X}}}^{(3)}$
$\widehat{\underline{\mathbf{X}}}^{(4)}$

Figure 3. Randomized TR decomposition with a prior Tucker compression.

Algorithm 12: Pseudocode for the Tucker compression

Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a Tucker rank $(K_1, K_2, \ldots, K_N)$

Output: Approximate Tucker decomposition $\underline{\mathbf{X}} \cong \left[\left[\underline{\mathbf{S}}; \mathbf{Q}^{(1)}, \mathbf{Q}^{(2)}, \ldots, \mathbf{Q}^{(N)}\right]\right]$

```
1 $\underline{\mathbf{S}} = \underline{\mathbf{X}}$
2 for $n = 1, 2, \ldots, N$ do
3     $[H_n, \sim] = \mathrm{Size}(\mathbf{S}_{(n)})$
4     Draw a random matrix $\mathbf{\Omega}_n \in \mathbb{R}^{K_n \times H_n}$
5     $\mathbf{Y} = \mathbf{S}_{(n)} \mathbf{\Omega}_n$
6     $\left[\mathbf{Q}^{(n)}, \sim\right] = \mathrm{qr}(\mathbf{Y})$
7     $\underline{\mathbf{S}} = \underline{\mathbf{S}} \times_n \mathbf{Q}^{(n)\,T}$
8 end
```

Algorithm 13: Randomized TR decomposition with a prior Tucker compression (see also [103])

Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, an approximate multilinear rank $(K_1, K_2, \ldots, K_N)$ and approximate TR-ranks $\{R_0, R_1, \ldots, R_{N-1}\}$

Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$, such that $\left\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\right\| \leq \varepsilon \|\underline{\mathbf{X}}\|$

1 Apply Algorithm 12 to the data tensor $\underline{\mathbf{X}}$ to compress it in the Tucker model with Tucker rank multilinear rank $(K_1, K_2, \ldots, K_N)$ and obtain $\left[\left[\underline{\mathbf{S}}, \mathbf{Q}^{(1)}, \mathbf{Q}^{(2)}, \ldots, \mathbf{Q}^{(N)}\right]\right]$

2 Apply Algorithm 4, Algorithm 6, Algorithm 5 or TR-ALS Algorithm [20] to the compressed data tensor $\underline{\mathbf{S}}$ and obtain $\ll \widehat{\underline{\mathbf{S}}}^{(1)}, \widehat{\underline{\mathbf{S}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{S}}}^{(N)} \gg$

3 Recover the TR cores of the original data tensor from the TR cores of the compressed data tensor, $\widehat{\underline{\mathbf{X}}}^{(n)} = \widehat{\underline{\mathbf{S}}}^{(n)} \times_2 \mathbf{Q}^{(n)}$, $n = 1, 2, \ldots, N$

4 Return $\ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$

5. Simulations

In this section, we evaluate the presented randomized algorithms for computation of the TR decomposition of synthetic and real data tensors. All numerical simulations were conducted on a laptop computer with 2.60 GHz Intel(R) Core(TM) i7-5600U processor and 8GB memory. The evaluation measures of the algorithms are *compression ratio* and *relative error*. The compression ratio is defined as

Algorithm 14: Adaptive randomized TR decomposition with a prior Tucker compression (see also [103])

```
Input: A data tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a prescribed approximation error bound $\varepsilon$
Output: Approximative representation of the tensor $\underline{\mathbf{X}}$ in the TR format $\widehat{\underline{\mathbf{X}}} = \ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$,
such that $\left\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\right\| \leq \varepsilon \|\underline{\mathbf{X}}\|$ and the TR- ranks $\{R_0, R_1, \ldots, R_{N-1}\}$
1 Apply the r-SHOSVD algorithm [98] (also see Algorithm 5 in [97]) to compress the data tensor in the Tucker format and obtain $\left[\left[\underline{\mathbf{S}}; \mathbf{Q}^{(1)}, \mathbf{Q}^{(2)}, \ldots, \mathbf{Q}^{(N)}\right]\right]$
2 Apply Algorithm 4, Algorithm 6, Algorithm 5 or TR-ALS Algorithm [20] to the compressed data tensor and obtain $\underline{\mathbf{S}}$ represented as $\ll \widehat{\underline{\mathbf{S}}}^{(1)}, \widehat{\underline{\mathbf{S}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{S}}}^{(N)} \gg$
3 Recover the TR cores of the original data tensor from the TR cores of the compressed data tensor, $\widehat{\underline{\mathbf{X}}}^{(n)} = \widehat{\underline{\mathbf{S}}}^{(n)} \times_2 \mathbf{Q}^{(n)}$, $n = 1, 2, \ldots, N$
4 Return $\ll \widehat{\underline{\mathbf{X}}}^{(1)}, \widehat{\underline{\mathbf{X}}}^{(2)}, \ldots, \widehat{\underline{\mathbf{X}}}^{(N)} \gg$
```

Table 1. Example 1. Compression ratio of the TR-SVD algorithm for different possible initial ranks R_0 for a random tensor of size $70 \times 70 \times 70 \times 70$ and TR-rank $\{5, 3, 5, 7\}$ with optimal compression 2.9155e-04.

Initial rank R_0	TR-ranks	Compression ratio
1	{1, 15, 25, 35}	0.0038
3	{3, 5, 75, 105}	0.0250
5	{5, 3, 45, 63}	0.0096
15	{15, 1, 15, 21}	0.0019

$$\text{Compression ratio} = \frac{C_2}{C_1},$$

where C_1, C_2 are the number of components of the original data tensor and its approximation in the TR format, respectively. Also, the relative error is defined as

$$e = \frac{\left\|\widetilde{\underline{\mathbf{X}}} - \underline{\mathbf{X}}\right\|_F}{\|\underline{\mathbf{X}}\|_F},$$

where $\widetilde{\underline{\mathbf{X}}}$ and $\underline{\mathbf{X}}$ are approximate and exact data tensors, respectively.

Example 1. In the first experiment, the performance and accuracy of the randomized TR algorithms are compared for synthetic data. We set the power iteration as $q = 1$ and the oversampling as $P = 5$, within the randomized algorithms. Consider a 4th-order random tensor $\underline{\mathbf{X}} \in \mathbb{R}^{70 \times 70 \times 70 \times 70}$ with exact TR-ranks $(5, 3, 5, 7)$. We applied the deterministic and randomized TR algorithms (Algorithms 5–7, 7, 10, 13, 14) to the tensor $\underline{\mathbf{X}}$. The possible initial rank, i.e. R_0, for the TR decomposition were $R_0 = 1, 3, 5, 15$ and the best compression was achieved for $R_0 = 15$. The compression ratio of the TR-SVD achieved using different initial ranks R_0 are reported in table 1. The same compression was achieved using the randomized TR-SVD but with better running time reported in table 2. From table 1, it is seen that the best compression ratio was achieved by initial rank $R_0 = 15$.

Algorithms 5–6 and their variants with a prior Tucker compression (Algorithm 13 or 14) were able to find the TR-ranks $\{15, 1, 15, 21\}$ but with higher computational costs. The running time and relative error of algorithms are reported in table 2. For a prior reduction in the Tucker format, the tensor was compressed to a tensor of size $(15, 15, 35, 35)$ using the r-STHOSVD algorithm. From table 2, it is seen that the randomized algorithms have better running time compared to the deterministic counterparts. Note that the last two algorithms in table 2 are non-adaptive and we gave the true TR-ranks to the algorithms. This is why they achieved a better compression ratio.

In a second simulation, we generated a 5th tensor of size $30 \times 30 \times 30 \times 30 \times 30$ with TR-ranks $(5, 3, 5, 3, 5)$. Here, we wanted to highlight the performance of TR decomposition with a prior Tucker compression when it is combined with Algorithms 5 and 6. Note that for the Tucker compression step, we used multilinear rank $(15, 15, 15, 15, 25)$. The results of this experiment are reported in table 3. The results show significant speed-up of the deterministic algorithms using the randomization technique.

Example 2. In this example, we consider the highly oscillating function $f(x) = (x+1)\sin\left(100(x+1)^2\right)$ depicted in figure 4. It was shown in [110] and [111] that the discretized form of such functions after a tensorization can be compressed in the TR format. Discretization of this function over the range $[-1, 1]$ with

Table 2. Example 1. Comparison results for the data tensor of size $70 \times 70 \times 70 \times 70$ and TR-ranks $(5,3,5,7)$ with optimal compression 2.9155e-04.

Algorithm	Relative Error	CPU Time	Compression ratio
Algorithm 4 (with initial $R_0 = 15$)	7.3475e-06	4.887479	0.0019
Algorithm 7 (with initial $R_0 = 15$)	5.0265e-06	**1.443426**	0.0019
Algorithm 7 with a prior Tucker compression (Algorithm 14)	1.2578e-06	**2.4633**	0.0019
Algorithm 5	4.2254e-06	265.4523	0.0019
Algorithm 5 with a prior Tucker compression (Algorithm14)	5.4346e-06	**2.6749**	0.0019
Algorithm 6	6.6723e-06	72.3477	0.0019
Algorithm 6 with a prior Tucker compression (Algorithm 14)	1.2578e-06	**2.4633**	0.0019
Algorithm 10	2.1165e-06	**3.3244**	0.0019
Algorithm 7	2.0875e-06	**3.2679**	2.9155e-04
Algorithm 13	2.5686e-06	**2.7645**	2.9155e-04

Table 3. Example 1. Comparison results for the data tensor of size $30 \times 30 \times 30 \times 30 \times 30$ and TR-ranks $(5,3,5,3,5)$ with optimal compression 1.0494e-04.

Algorithm	Relative Error	CPU Time	Compression ratio
Algorithm 5	7.6743e-04	139.6735	6.5598e-04
Algorithm 5 with a prior Tucker compression (Algorithm 14)	4.3423e-04	**11.7935**	6.5598e-04
Algorithm 6	2.2911e-04	19.3457	6.5598e-04
Algorithm 6 with a prior Tucker compression (Algorithm 14)	5.5467e-04	**5.3598**	6.5598e-04

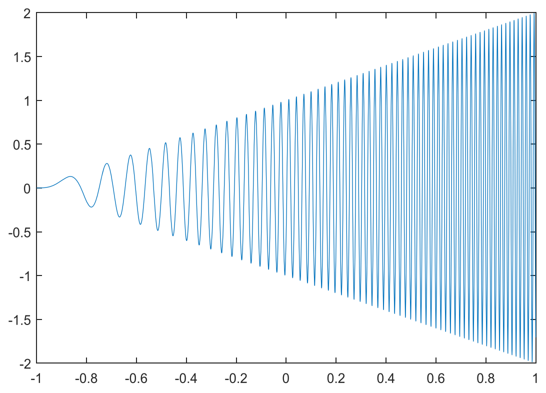


Figure 4. The highly oscillating function $f(x) = (x+1)\sin\left(100(x+1)^2\right)$ over $[-1,1]$. Discretization of this function with step-size $\frac{1}{2^{24}}$ followed by a tensorization has low TR-ranks [110, 111]

Table 4. Example 2. Comparison results for the highly oscillating function $f(x) = (x+1)\sin\left(100(x+1)^2\right)$.

Algorithm	Relative Error	CPU Time	Compression ratio
Algorithm 5	5.3849e-06	803.9073	1.4782e-04
Algorithm 5 with a prior Tucker compression (Algorithm 14)	5.8412e-06	**10.1231**	1.4782e-04
Algorithm 6	5.3859e-06	360.2503	1.4782e-04
Algorithm 5 with a prior Tucker compression (Algorithm 14)	7.1662e-06	**8.4494**	1.4782e-04

step-size $\frac{1}{2^{24}}$ leads to a long vector of size 33554433. We remove the first component of this long vector so that the new vector can be reshaped to a 4th-order tensor of size $128 \times 256 \times 32 \times 32$. This tensor has true multilinear rank $(3,7,21,21)$ which can be computed via the MATLAB function '`mlrank.m`' included in the tensorlab toolbox [112]. The MATLAB function '`mlrankest.m`' gave the numerical multilinear rank $(2,4,16,16)$ but we used the former in our compression step. We again used Algorithm 12 for the Tucker compression step with the multilinear rank $(3,7,21,21)$. In this experiment, we highlighted the importance of prior Tucker compression in the data tensor in reducing the running time of TR algorithms with cyclic permutations of the cores. The results of this simulation are reported in table 4. From the results, the superiority of the randomized algorithms compared to deterministic counterparts is visible. Also, all algorithms achieved the same compression ratio.

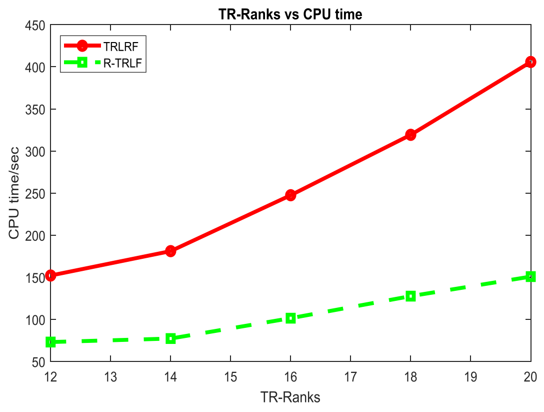


Figure 5. The running time of the TRLRF and the R-TRLRF algorithms for 'Giant' image and TR-ranks 20.

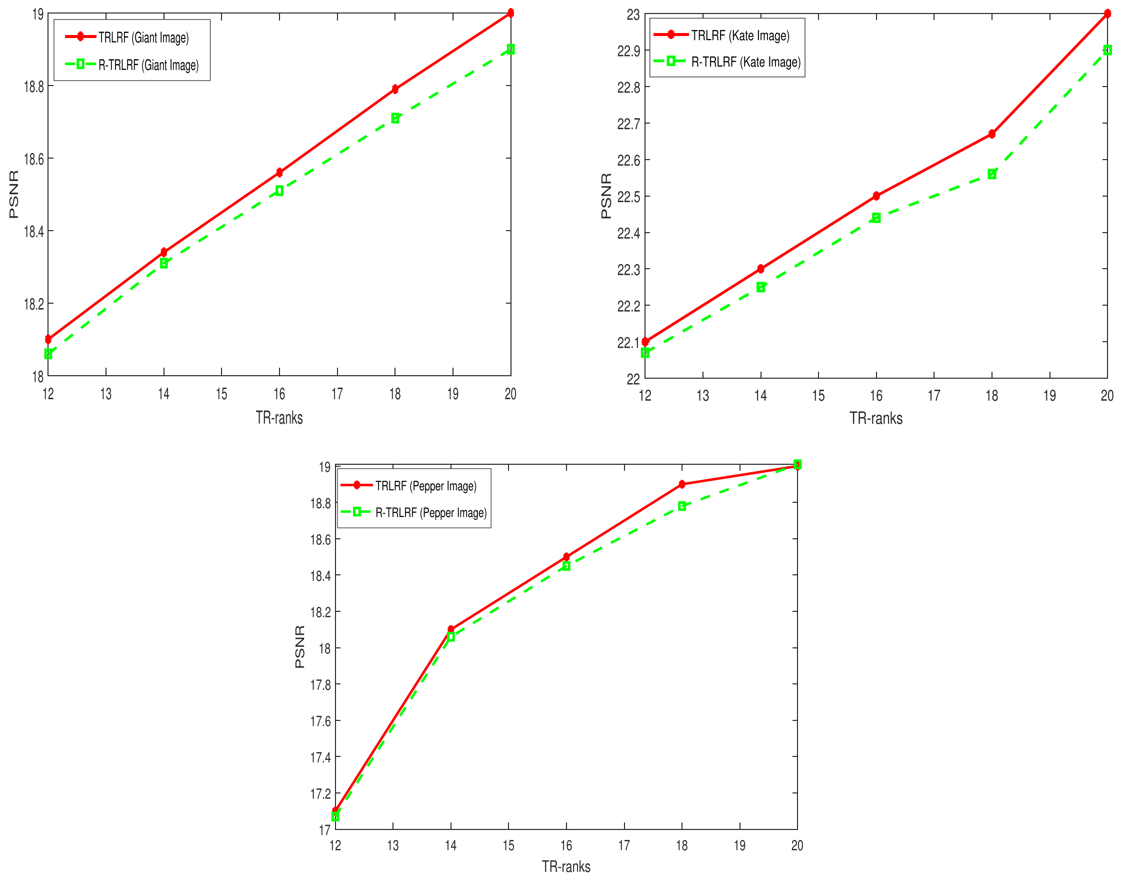


Figure 6. PSNR comparison results of the TRLRF and the R-TRLRF algorithms.

Example 3. In this simulation, we are dealing with the tensor completion problem. The problem of recovering a given data tensor from its partially observed components is known as tensor completion [113, 114]. The TR model has recently been used for recovering the missing elements of incomplete higher-order data tensors in [34, 115–117]. Here, we used the TR low rank factors (TRLRF) algorithm proposed in [116, 117] because as reported in [116, 117], it was the most stable and efficient algorithm in terms of sensitivity to TR-ranks and reconstruction error compared to others. The algorithm is based on nuclear norm minimization formulation and it uses the ADMM algorithm [118] to solve the underlying optimization problem[10]. It requires to compute the SVD of some unfolding matrices in Lines 86–88 of the MATLAB code '`TRLRF.m`'. We modified it to a randomized algorithm by replacing the SVD with Algorithm 3

[10] The MATLAB implementations of it is available in the GitHub repository https://github.com/yuanlonghao/TRLRF.

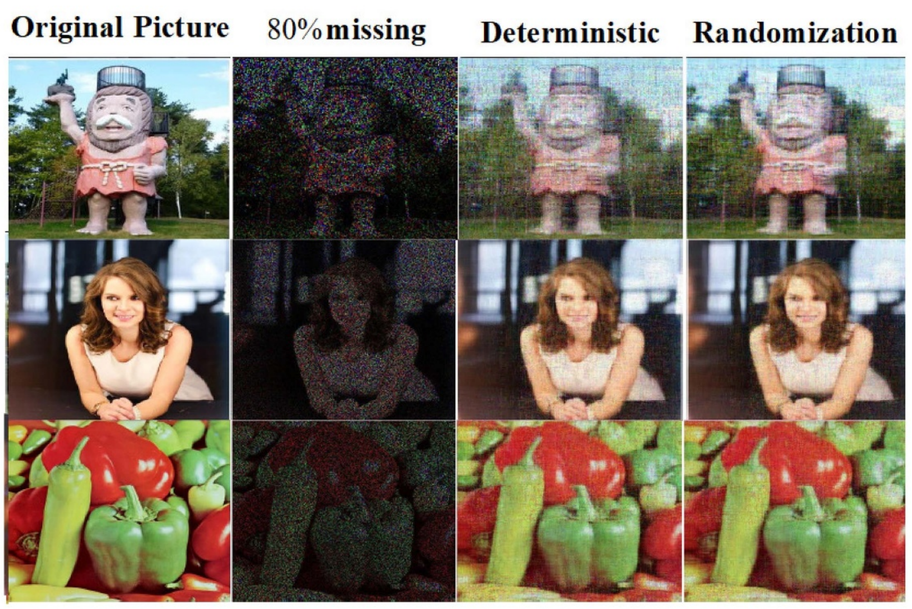


Figure 7. The reconstructed images using the TRLRF algorithm and its randomized variant for TR-rank 20.

where the block size was $b = 100$ and the power iteration was $q = 1$. We call it randomized TRLRF (R-TRLRF) algorithm.

Consider three benchmark images ('Giant' , 'Kate' , 'Peppers') depicted in figure 7, respectively (images in the first column). The "Giant' was of size $256 \times 256 \times 3$, while the 'Kate' and 'Peppers' were of size $512 \times 512 \times 3$. We reshaped 'Giant' , 'Kate' , 'Peppers' to 7th-order tensors of sizes $8 \times 8 \times 4 \times 8 \times 8 \times 4 \times 3$, $16 \times 8 \times 4 \times 16 \times 8 \times 4 \times 3$ and $16 \times 8 \times 4 \times 16 \times 8 \times 4 \times 3$, respectively. We removed randomly 80% of the pixels of the mentioned images and the TRLRF and the R-TRLRF algorithms were applied to the images with missing pixels with varying TR-ranks $R_1 = R_2 = \cdots = R_7 = R$, for $R = 12, 13, \ldots, 20$. The running times of the TRLRF and the R-TRLRF algorithms for different TR-ranks and all images were almost the same and we only report it for the 'Giant' image and TR-ranks 20 in figure 5. From figure 5, the scalability of the R-TRLRF algorithm compared to the TRLRF algorithm is visible. Our simulations confirmed the R-TRLRF algorithm achieves roughly the same accuracy as the TRLRF algorithm while it is computationally much cheaper. The PSNR and the RSE of recovered images obtained by the TRLRF algorithm and the R-TRLRF algorithm are reported in figures 6,
where RSE $= \frac{\|\underline{\mathbf{X}}-\widehat{\underline{\mathbf{X}}}\|_F}{\|\underline{\mathbf{X}}\|_F}$, PSNR $= 10\log_{10}\left(255^2/\mathrm{MSE}\right)$, and MSE $= \left\|\underline{\mathbf{X}} - \widehat{\underline{\mathbf{X}}}\right\|_F^2 /\mathrm{num}\left(\underline{\mathbf{X}}\right)$. Note 'num' denotes the number of parameters of a given data tensor. Here again, it is seen that the R-TRLRF algorithm outperforms the TRLRF algorithm. We visualize the results of recovered images by the TRLRF algorithm and the R-TRLRF algorithm in figure 7 for TR-rank 20.

6. Conclusion

In this paper, we reviewed and extended a variety of randomized algorithms for computation of the Tensor Ring (TR) decomposition which to some extent can be applied to the TT decomposition with $R_0 = 1$. We discussed both adaptive and non-adaptive randomized algorithms for this task. Our main focus was on the random projection technique and used it to speed-up the decomposition of a tensor in the TT/TR format. Simulations were performed on synthetic and real data-sets to support the presentation.

Acknowledgment

The authors would like to thank two anonymous reviewers for their constructive comments which have greatly improved the quality of the paper. This research was also partially supported by the Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001).

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study. The analyzed data are available online.

The data of images are available at https://github.com/wangwenqi1990/TensorRingCompletion and https://drive.google.com/file/d/15xk67wYZ9GI2Kn93g_aaA4CsmgnqGlHE/view.

The synthetic data were generated using functions of the TR toolbox https://github.com/oscarmickelin/tensor-ring-decomposition.

ORCID iD

Salman Ahmadi-Asl https://orcid.org/0000-0002-2614-0146

References

[1] Cichocki A, Mandic D, Phan A H, Caiafa C, Zhou G, Zhao Q and De Lathauwer L 2015 Tensor decompositions for signal processing applications: from two-way to multiway component analysis *IEEE Signal Process. Mag.* **32** 145–63

[2] Cichocki A, Lee N, Oseledets I, Phan A-H, Zhao Q and Mandic D P 2016 Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions *Found. Trends Mach. Learning* **9** 249–429

[3] Cichocki A, Phan A-H, Zhao Q, Lee N, Oseledets I, Sugiyama M and Mandic D P 2017 Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives *Found. Trends Mach. Learn.* **9** 431–673

[4] Kolda T G and Bader B W 2009 Tensor decompositions and applications *SIAM Rev.* **51** 455–500

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

[6] Sidiropoulos N D, De Lathauwer L, Fu X, Huang K, Papalexakis E E and Faloutsos C 2017 Tensor decomposition for signal processing and machine learning *IEEE Trans. Signal Process.* **65** 3551–82

[7] Papalexakis E E, Faloutsos C and Sidiropoulos N D 2017 Tensors for data mining and data fusion: Models, applications and scalable algorithms *ACM Trans. Intell. Syst. Technol.* **8** 16

[8] Acar E and Yener B 2008 Unsupervised multiway data analysis: A literature survey *IEEE Trans. Knowl. Data Eng.* **21** 6–20

[9] Phan A H and Cichocki A 2010 Tensor decompositions for feature extraction and classification of high dimensional datasets *NOLTA* **1** 37–68

[10] Tucker L R 1963 Implications of factor analysis of three-way matrices for measurement of change *Problems in Measuring Change* vol 15 pp 122–37

[11] Tucker L R 1964 The extension of factor analysis to three-dimensional matrices Contributions to Mathematical Psychology Frederiksen N and Gulliksen H Eds (New York: Holt, Rinehart and Winston)

[12] Tucker L R 1966 Some mathematical notes on three-mode factor analysis *Psychometrika* **31** 279–311

[13] Beylkin G and Mohlenkamp M J 2005 Algorithms for numerical analysis in high dimensions *SIAM J. Sci. Comput.* **26** 2133–59

[14] Landsberg J M 2012 *Tensors: Geometry and Applications* (American Mathematical Society)

[15] Orús R 2014 A practical introduction to tensor networks: matrix product states and projected entangled pair states *Ann. Phys.* **349** 117–58

[16] Oseledets I V 2011 Tensor-train decomposition *SIAM J. Sci. Comput.* **33** 2295–2317

[17] Khoromskij B N 2011 $\mathcal{O}(dlogn)$-quantics approximation of n-d tensors in high-dimensional numerical modeling *Constr. Approx.* **34** 257–80

[18] Espig M, Hackbusch W, Handschuh S and Schneider R 2011 Optimization problems in contracted tensor networks *Comput. Vis. Sci.* **14** (6) 271–85

[19] Espig M, Naraparaju K K and Schneider J 2012 A note on tensor chain approximation *Comput. Vis. Sci.* **15** 331–44

[20] Zhao Q, Zhou G, Xie S, Zhang L and Cichocki A 2016 Tensor ring decomposition arXiv: 1606.05535

[21] Hackbusch W and Kühn S 2009 A new scheme for the tensor representation *J. Fourier Analy. Appl.* **15** 706–22

[22] Grasedyck L 2010 Hierarchical singular value decomposition of tensors *SIAM J. Matrix Anal. Appl.* **31** 2029–54

[23] Affleck I, Kennedy T, Lieb E H and Tasaki H 1988 Valence bond ground states in isotropic quantum antiferromagnets *Condensed Matter Physics and Exactly Soluble Models* (Berlin: Springer) pp 253–304

[24] White S R 1992 Density matrix formulation for quantum renormalization groups *Phys. Rev. Lett.* **69** 2863

[25] Perez-Garcia D, Verstraete F, Wolf M M and Cirac J I 2006 Matrix product state representations *Quantum Information and Computation* **7** 401

[26] Schollwöck U 2011 The density-matrix renormalization group in the age of matrix product states *Ann. Phys.* **326** 96–192

[27] Lee N and Cichocki A 2015 Estimating a few extreme singular values and vectors for large-scale matrices in tensor train format *SIAM J. Matrix Anal. Appl.* **36** 994–1014

[28] Lee N and Cichocki A 2016 Regularized computation of approximate pseudoinverse of large matrices using low-rank tensor train decompositions *SIAM J. Matrix Anal. Appl.* **37** 598–623

[29] Novikov A, Podoprikhin D, Osokin A and Vetrov D 2015 Tensorizing neural networks *Advances in Neural Information Processing Systems (NIPS)* pp 442–50

[30] Tjandra, A, Sakti S and Nakamura S 2017 Compressing recurrent neural network with tensor train *IEEE Int. Conf. in Neural Networks (IJCNN)* pp 4451–8

[31] Pan Y, Xu J, Wang M, Wang F, Bai K and Xu Z 2018 Compressing recurrent neural networks with tensor ring for action recognition CoRR arXiv: 1811.07503

[32] Wang W, Sun Y, Eriksson B, Wang W and Aggarwal V 2018 Wide compression: tensor ring nets *Proc. of the Conf. on Computer Vision and Pattern Recognition* pp 9329–38

[33] Cheng Z, Li B, Fan Y and Bao Y 2020 A novel rank selection scheme in tensor ring decomposition based on reinforcement learning for deep neural networks *ICASSP 2020-2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* IEEE pp 3292–6

[34] Wang W, Aggarwal V and Aeron S 2017 Efficient low rank tensor ring completion *Proc. of the IEEE Int. Conf. on Computer Vision* pp 5697–705

[35] Yuan L, Cao J, Wu Q and Zhao Q 2018 Higher-dimension tensor completion via low-rank tensor ring decomposition arXiv: 1807.01589

[36] Yuan L, Zhao Q, Gui L and Cao J 2019 High-order tensor completion via gradient-based optimization under tensor train format *Signal Process. Image Commun.* **73** 53–61

[37] Bengua J A, Phien H N, Tuan H D and Do M N 2017 Efficient tensor completion for color image and video recovery: low-rank tensor train *IEEE Trans. Image Process.* **26** 2466–79

[38] Asif M S and Prater-Bennette A 2020 Low-rank tensor ring model for completing missing visual data *ICASSP 2020-2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* IEEE pp 5415–19

[39] Huang H, Liu Y, Liu J and Zhu C 2020 Provable tensor ring completion *Signal Process.* **171** 107486

[40] Yang Y, Krompass D and Tresp V 2017 Tensor-train recurrent neural networks for video classification *Proc. of the 34th Int. Conf. on Machine Learning* vol 70 JMLR. org pp 3891–900

[41] Kuznetsov M, Polykovskiy D, Vetrov D P and Zhebrak A 2019 A prior of a googol gaussians: a tensor ring induced prior for generative models *Advances in Neural Information Processing Systems (NIPS)* pp 4104–14

[42] Stoudenmire E and Schwab D J 2016 Supervised learning with tensor networks *Advances in Neural Information Processing Systems (NIPS)* pp 4799–807

[43] Liu Y, Yao Q and Li Y 2020 Generalizing tensor decomposition for N-way relational knowledge bases *Proc. of The Conf. 2020* pp 1104–14

[44] Zhao Q, Sugiyama M, Yuan L and Cichocki A 2019 Learning efficient tensor representations with ring-structured networks *ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* IEEE pp 8608–12

[45] Khoromskaia V and Khoromskij B N 2015 Tensor numerical methods in quantum chemistry: from Hartree–Fock to excitation energies *Phys. Chem. Chem. Phys.* **17** 31491–509

[46] Veit A and Scott L R 2017 Using the tensor-train approach to solve the ground-state eigenproblem for hydrogen molecules *SIAM J. Sci. Comput.* **39** B190–B220

[47] Batselier K, Yu W, Daniel L and Wong N 2018 Computing low-rank approximations of large-scale matrices with the tensor network randomized SVD *SIAM J. Matrix Anal. Appl.* **39** 1221–44

[48] Verstraete F, Murg V and Cirac J I 2008 Matrix product states, projected entangled pair states and variational renormalization group methods for quantum spin systems *Adv. Phys.* **57** 143–224

[49] Vidal G 2008 Class of quantum many-body states that can be efficiently simulated *Phys. Rev. Lett.* **101** 110501

[50] Shi Y-Y, Duan L-M and Vidal G 2006 Classical simulation of quantum many-body systems with a tree tensor network *Phys. Rev.* A **74** 022320

[51] Batselier K, Cichocki A and Wong N 2019 Meracle: constructive layer-wise conversion of a tensor train into a mera arXiv: 1912.09775

[52] De Lathauwer L, De Moor B and Vandewalle J 2000 A multilinear singular value decomposition *SIAM J. Matrix Anal. Appl.* **21** 1253–78

[53] Halko N, Martinsson P-G and Tropp J A 2011 Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions *SIAM Rev.* **53** 217–88

[54] Achlioptas D 2003 Database-friendly random projections: Johnson-Lindenstrauss with binary coins *J. Comput. Syst. Sci.* **66** 671–87

[55] Li P, Hastie T J and Church K W 2006 Very sparse random projections Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining ACM 287–96

[56] Woolfe F, Liberty E, Rokhlin V and Tygert M 2008 A fast randomized algorithm for the approximation of matrices *Appl. Comput. Harmon. Anal.* **25** 335–66

[57] Liberty E and Zucker S 2009 Accelerated dense random projections Citeseer

[58] Ailon N and Liberty E 2009 Fast dimension reduction using Rademacher series on dual BCH codes *Discrete Comput. Geom.* **42** 615

[59] Rokhlin V and Tygert M 2008 A fast randomized algorithm for overdetermined linear least-squares regression *Proc. Natl Acad. Sci.* **105** 13212–17

[60] Mahoney M W *et al* 2011 Randomized algorithms for matrices and data *Found. Trends Mach. Learn.* **3** 123–224

[61] Charikar M, Chen K and Farach-Colton M 2002 Finding frequent items in data streams *Int. Coll. on Automata, Languages and Programming* (Berlin: Springer) pp 693–703

[62] Pham N and Pagh R 2013 Fast and scalable polynomial kernels via explicit feature maps *Proc. of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* ACM 239–47

[63] Pagh R 2013 Compressed matrix multiplication *ACM Trans. Comput. Theory* **5** 9

[64] Li M, Bi W, Kwok J T and Lu B-L 2014 Large-scale Nyström kernel matrix approximation using randomized svd *IEEE Trans. Neur. Net. Learn. Syst.* **26** 152–64

[65] Kaloorazi M F and de Lamare R C 2018 Compressed randomized UTV decompositions for low-rank matrix approximations *IEEE J. Sel. Top. Signal Process.* **12** 1155–69

[66] Tropp J A, Yurtsever A, Udell M and Cevher V 2017 Practical sketching algorithms for low-rank matrix approximation *SIAM J. Matrix Anal. Appl.* **38** 1454–85

[67] Woodruff D P *et al* 2014 Sketching as a tool for numerical linear algebra *Found. Trends Theor. Comput. Sci.* **10** 1–157

[68] Boutsidis C, Woodruff D P and Zhong P 2016 Optimal principal component analysis in distributed and streaming models *Proc. of the Forty-Eighth Annual ACM Symposium on Theory of Computing* ACM pp 236–49

[69] Upadhyay J 2018 The price of privacy for low-rank factorization *Advances in Neural Information Processing Systems (NIPS)* pp 4176–87

[70] Bajaj C, Wang Y and Wang T 2019 Sketchycoresvd Sketchysvd from random subsampling of the data matrix arXiv: 1907.13634

[71] Yu W, Gu Y and Li Y 2018 Efficient randomized algorithms for the fixed-precision low-rank matrix approximation *SIAM J. Matrix Anal. Appl.* **39** 1339–59

[72] Martinsson P-G and Voronin S 2016 A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices *SIAM J. Sci. Comput.* **38** S485–S507

[73] Handschuh S 2015 Numerical methods in tensor networks *PhD thesis* Max-Planck-Institute for Mathematics

[74] Handschuh S 2012 Changing the topology of tensor networks arXiv: 1203.1503

[75] Holtz S, Rohwedder T and Schneider R 2012 The alternating linear scheme for tensor optimization in the tensor train format *SIAM J. Sci. Comput.* **34** A683–A713
[76] Huckle T, Waldherr K and Schulte-Herbrüggen T 2013 Computations in quantum tensor networks *Linear Algebr. Appl.* **438** 750–81
[77] Oseledets I and Khoromskij B 2010 DMRG and QTT approach to high-dimensional quantum molecular dynamics *Tech. rep.* MIS preprint 69
[78] Goreinov S A, Tyrtyshnikov E E and Zamarashkin N L 1997 A theory of pseudoskeleton approximations *Linear Algebr. Appl.* **261** 1–21
[79] Goreinov S A, Zamarashkin N L and Tyrtyshnikov E E 1997 Pseudo-skeleton approximations by matrices of maximal volume *Math. Notes* **62** 515–19
[80] Goreinov S, Oseledets I, Savostyanov D, Tyrtyshnikov E and Zamarashkin N 2010 How to find a good submatrix *Matrix Methods: Theory, Algorithms, Applications* (Singapore: World Scientific)
[81] Tyrtyshnikov E 2000 Incomplete cross approximation in the mosaic-skeleton method *Computing* **64** 367–80
[82] Oseledets I and Tyrtyshnikov E 2010 TT-cross approximation for multidimensional arrays *Linear Algebr. Appl.* **432** 70–88
[83] Savostyanov D and Oseledets I 2011 Fast adaptive interpolation of multi-dimensional arrays in tensor train format *The 2011 Int. Workshop on Multidimensional (nD) Systems* IEEE pp 1–8
[84] Savostyanov D V 2014 Quasioptimality of maximum-volume cross interpolation of tensors *Linear Algebr. Appl.* **458** 217–44
[85] Phan A-H, Cichocki A, Uschmajew A, Tichavsky P, Luta G and Mandic D 2019 Tensor networks for latent variable analysis: novel algorithms for tensor train approximation IEEE Trans. Neur. Netw. Learn. Syst.
[86] Grasedyck L, Kluge M and Kramer S 2015 Variants of alternating least squares tensor completion in the tensor train format *SIAM J. Sci. Comput.* **37** A2424–A2450
[87] Mickelin O and Karaman S 2018 On algorithms for and computing with the tensor ring decomposition arXiv: 1807.02513
[88] Batselier K 2018 The trouble with tensor ring decompositions arXiv: 1811.03813
[89] Khoromskij B N 2012 Tensors-structured numerical methods in scientific computing: Survey on recent advances *Chemometr. Intell. Lab. Syst.* **110** 1–19
[90] Ye K and Lim L-H 2018 Tensor network ranks arXiv: 1801.02662
[91] Dolgov S and Khoromskij B 2013 Two-level QTT-Tucker format for optimized tensor calculus *SIAM J. Matrix Anal. Appl.* **34** 593–623
[92] Chen Z, Li Y and Lu J 2019 Tensor ring decomposition: optimization landscape and one-loop convergence of alternating least squares arXiv: 1905.07101
[93] Khoo Y, Lu J and Ying L 2017 Efficient construction of tensor ring representations from sampling arXiv: 1711.00954
[94] Che M and Wei Y 2018 Randomized algorithms for the approximations of Tucker and the tensor train decompositions *Adv. Comput. Math.* **45** 1–34
[95] Huber B, Schneider R and Wolf S 2017 A randomized tensor train singular value decomposition *Compressed Sensing and its Applications* (Berlin: Springer) pp 261–90
[96] Wolf P T and Johannes A S 2019 Low rank tensor decompositions for high dimensional data approximation, recovery and prediction *PhD thesis* (https://doi.org/10.14279/depositonce-8109)
[97] Ahmadi-Asl S, Cichocki A, Phan A H, Oseledets I, Abukhovich S and Tanaka T 2020 Randomized algorithms for computation of Tucker decomposition and higher order SVD (HOSVD) arXiv: 2001.07124
[98] Minster R, Saibaba A K and Kilmer M E 2020 Randomized algorithms for low-rank tensor decompositions in the tucker format *SIAM J. Math. Data Sci.* **2** 189–215
[99] Hitchcock F L 1927 The expression of a tensor or a polyadic as a sum of products *J. Math. Phys.* **6** 164–89
[100] Hitchcock F L 1928 Multiple invariants and generalized rank of a p-way matrix or tensor *J. Math. Phys.* **7** 39–79
[101] Cattell R B 1944 Parallel proportional profiles and other principles for determining the choice of factors by rotation *Psychometrika* **9** 267–83
[102] Erichson N B, Manohar K, Brunton S L and Kutz J N 2017 Randomized CP tensor decomposition arXiv: 1703.09074
[103] Yuan L, Li C, Cao J and Zhao Q 2019 Randomized tensor ring decomposition and its application to large-scale data reconstruction *ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* IEEE pp 2127–31
[104] Vannieuwenhoven N, Vandebril R and Meerbergen K 2012 A new truncation strategy for the higher-order singular value decomposition *SIAM J. Sci. Comput.* **34** A1027–A1052
[105] Oseledets I V 2010 Approximation of $2^d \times 2^d$ matrices using tensor decomposition *SIAM J. Matrix Anal. Appl.* **31** 2130–45
[106] Liu H, Yang L T, Guo Y, Xie X and Ma J 2018 An incremental tensor-train decomposition for cyber-physical-social big data IEEE Trans. Big Data
[107] Singh A, Aujla G S, Garg S, Kaddoum G and Singh G 2020 Deep learning-based SDN model for internet of things: an incremental tensor train approach *IEEE Internet Things J.* **7** 6302–11
[108] Zniyed Y 2019 Breaking the curse of dimensionality based on tensor train: models and algorithms *PhD thesis* Paris-Saclay
[109] Zniyed Y, Boyer R, de Almeida A L and Favier G 2020 A TT-based hierarchical framework for decomposing high-order tensors *SIAM J. Sci. Comput.* **42** A822–A848
[110] Khoromskij B N 2015 Tensor numerical methods for multidimensional PDEs: theoretical analysis and initial applications *ESAIM: Proc. Surveys* **48** 1–28
[111] Khoromskij B N and Repin S I 2015 A fast iteration method for solving elliptic problems with quasiperiodic coefficients *Russian J. Numerical Anal. Math. Modelling* **30** 329–44
[112] Vervliet N, Debals O, Sorber L, Van Barel M and De Lathauwer L 2016 Tensorlab user guide
[113] Song Q, Ge H, Caverlee J and Hu X 2019 Tensor completion algorithms in big data analytics *ACM Trans. Knowl. Discov. Data* **13** 6
[114] Long Z, Liu Y, Chen L and Zhu C 2019 Low rank tensor completion for multiway visual data *Signal Process.* **155** 301–16
[115] Yuan L, Li C, Mandic D, Cao J and Zhao Q 2019 Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion *Proc. of the Conf. on Artificial Intelligence* vol 33 pp 9151–8
[116] Yuan L, Cao J and Zhao Q 2020 Rank minimization on tensor ring: an efficient approach for tensor decomposition and completion *Mach. Learn.* **109** 1–200
[117] Yuan L, Cao J, Zhao X, Wu Q and Zhao Q 2018 Higher-dimension tensor completion via low-rank tensor ring decomposition *2018 Asia-Pacific Signal and Information Processing Association Annual Summit Conf. (APSIPA ASC)* IEEE pp 1071–6
[118] Boyd S, Parikh N, Chu E, Peleato B and Eckstein J *et al* 2011 Distributed optimization and statistical learning via the alternating direction method of multipliers *Found. Trends Mach. Learn.* **3** 1–122