Algorithms for TT Decomposition

February 24, 2022

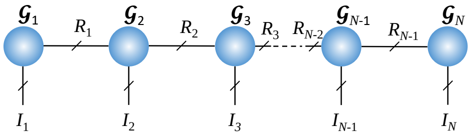
Outlines

- Tucker decomposition with bound constraint
- Two decomposition problems in Tensor-train
- Existing algorithms for TT
- Alternating multi-core update algorithm
- Nested Tucker-2

Part I

TT decomposition

Tensor Train I



$$\mathcal{X} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} \mathcal{G}_1(:, r_1) \circ \mathcal{G}_2(r_1, :, r_2) \circ \cdots \circ \mathcal{G}_N(r_{N-1}, :),$$

or

$$\mathcal{X} = \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N.$$

where $(R_1, R_2, \ldots, R_{N-1})$ represents the TT-rank of $\mathcal{X}$.

Tensor Train II

- TT-decomposition can be computed efficiently, while the ranks can be determined based on an approximation error boundVidal (2003); Oseledets and Tyrtyshnikov (2009)
- TT-decomposition is very suited to higher-order tensors.
- Applications: solving a huge system of linear equations or eigenvalue decomposition of large-scale data Holtz et al. (2012); Kressner et al. (2014), PDE, data completion, modelling in system identification, deep learning.
- Solving the higher order CPD through TT.

TT contraction I

Definition (Tensor train contraction)

The train contraction performs a tensor contraction between the last mode of tensor $\mathcal{A}$ and the first mode of tensor $\mathcal{B}$, where $I_N = J_1$, to yield a tensor $\mathcal{C} = \mathcal{A} \bullet \mathcal{B}$ of size $I_1 \times \cdots \times I_{N-1} \times J_2 \times \cdots \times J_K$, the elements of which are given by $c_{i_1,\ldots,i_{N-1},j_2,\ldots,j_K} = \sum_{i_N=1}^{I_N} a_{i_1,\ldots,i_{N-1},i_N} b_{i_N,j_2,\ldots,j_K}$.

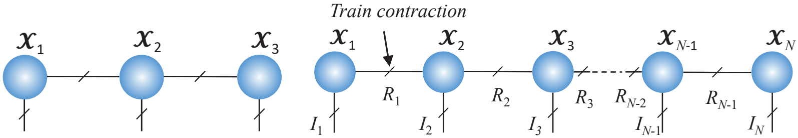


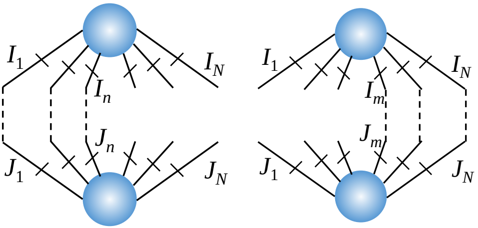
Figure: Graphical illustration of a Tucker-2 tensor (left) and a TT-tensor (right) $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$. A node represents a 3-rd order core tensor $\mathcal{X}_n$.

TT contraction II

Definition (Left and right contractions)

The n-modes left contraction between two tensors $\mathcal{A}$ and $\mathcal{B}$ is denoted by $\mathcal{C}_L = \mathcal{A} \ltimes_n \mathcal{B}$ and computes contraction product between their first n modes.
The right contraction denoted by $\mathcal{C}_R = \mathcal{A} \rtimes_n \mathcal{B}$ computes contraction product between their last n modes.

TT contraction III



(a) Left and right contractions

Figure: Left and right contractions between two tensors.

Orthogonalization I

Given a TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$

Definition (**Left and right orthogonality conditions Holtz et al. (2012); Kressner and Macedo (2014)**)

A core tensor $\mathcal{X}_n$ is left-orthogonal if $\mathcal{X}_n \ltimes_2 \mathcal{X}_n = \mathbf{I}_{R_n}$,
and right orthogonal if $\mathcal{X}_n \rtimes_2 \mathcal{X}_n = \mathbf{I}_{R_{n-1}}$.

Mode-n left orthogonalisation is achieved using the orthogonal Tucker-1 decomposition of $\mathcal{X}_n$ in the form $\mathcal{X}_n = \tilde{\mathcal{X}}_n \bullet \mathbf{L}$

$$\mathcal{X} = \mathcal{X}_1 \bullet \cdots \bullet \mathcal{X}_{n-1} \bullet \tilde{\mathcal{X}}_n \bullet (\mathbf{L} \bullet \mathcal{X}_{n+1}) \bullet \cdots \bullet \mathcal{X}_N.$$

Mode-n right orthogonalisation performs the orthogonal Tucker-1 decomposition $\mathcal{X}_n = \mathbf{R} \bullet \tilde{\mathcal{X}}_n$, and results

$$\mathcal{X} = \mathcal{X}_1 \bullet \cdots \bullet (\mathcal{X}_{n-1} \bullet \mathbf{R}) \bullet \tilde{\mathcal{X}}_n \bullet \mathcal{X}_{n+1} \bullet \cdots \bullet \mathcal{X}_N .$$

Orthogonalization II

Algorithm 1: `Left Orthogonalization for the core` $\mathcal{X}_n$

Input: TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$, mode-$n$
Output: TT-tensor $\mathcal{X}$ has $\mathcal{X}_n \ltimes_2 \mathcal{X}_n = \mathbf{I}_{R_n}$
begin

$[\mathcal{X}_n]^T_{(3)} = \mathbf{Q}\,\mathbf{R}$ `/* QR decomposition of` $[\mathcal{X}_n]^T_{(3)}$ `*/`

$\mathcal{X}_n = \mathtt{reshape}(\mathbf{Q}^T, R_{n-1} \times I_n \times R_n)$

$\mathcal{X}_{n+1} \leftarrow \mathbf{R}^T \bullet \mathcal{X}_{n+1}$

end

Algorithm 2: `Right Orthogonalization for the core` $\mathcal{X}_n$

Input: TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$, mode-$n$
Output: TT-tensor $\mathcal{X}$ has $\mathcal{X}_n \rtimes_2 \mathcal{X}_n = \mathbf{I}_{R_{n-1}}$
begin

$[\mathcal{X}_n]^T_{(1)} = \mathbf{Q}\,\mathbf{R}$ `/* QR decomposition of` $[\mathcal{X}_n]^T_{(1)}$ `*/`

$\mathcal{X}_n = \mathtt{reshape}(\mathbf{Q}^T, R_{n-1} \times I_n \times R_n)$

$\mathcal{X}_{n-1} \leftarrow \mathcal{X}_{n-1} \bullet \mathbf{R}^T$

end

Orthogonalization III

Left orthogonalisation up to mode n performs $(n-1)$ left orthogonalizations of the core tensors to the left of n such that $\mathcal{X}_k \ltimes_2 \mathcal{X}_k = \mathbf{I}_{R_k}$ for $k = 1, 2, \ldots, n-1$.

Right orthogonalisation up to mode n performs $(N-n)$ right orthogonalizations of the core tensors to the right of n such that $\mathcal{X}_k \rtimes_2 \mathcal{X}_k = \mathbf{I}_{R_{k-1}}$ for $k = n+1, n+2, \ldots, N$.

Question:

Does the tensor $\mathcal{X}_{<n}$ (or $\mathcal{X}_{>n}$ hold left-orthogonalization (or right-orthogonalization)?

TT-decomposition

Two approximation problems of a tensor $\mathcal{Y}$ by a tensor $\mathcal{X}$ in the TT-format:

- **The TT-approximation with given TT-ranks**

$$\min \quad D = \|\mathcal{Y} - \mathcal{X}\|_F^2 \ . \tag{1}$$

- **The TT-approximation with a given approximation accuracy (denoising problem)** based on the rank minimisation problem with error bound constraint such that the estimated TT-tensor $\mathcal{X}$ should have minimum number of parameters

$$\min \quad \sum_{n=1}^{N} I_n R_{n-1} R_n \quad \text{s.t.} \ \ \|\mathcal{Y} - \mathcal{X}\|_F^2 \leq \varepsilon^2 \,, \tag{2}$$

where ε^2 represents the approximation accuracy.

TT-SVD and TT-truncation I

TT-SVD Vidal (2003); Oseledets and Tyrtyshnikov (2009); Oseledets (2011) is based on sequential projection and truncated SVD, known as the best algorithm when the data admits the TT model.

Truncated SVD The first core $\mathcal{X}_1$ comprises the R_1 leading singular vectors of the reshaping matrix $\mathbf{Y}_{(1)}$, subject to the error norm being less than ε times the data norm, that is,

$$\|\mathbf{Y}_{(1)} - \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{V}^T\|_F^2 \leq \varepsilon^2\,\|\mathbf{Y}_{(1)}\|_F^2 \tag{3}$$

or $\|\boldsymbol{\sigma}\|_2^2 \geq (1-\varepsilon^2)\|\mathbf{Y}_{(1)}\|_F^2$.

Projection The projected data $\mathrm{diag}(\boldsymbol{\sigma})\mathbf{V}^T$ is then reshaped into a matrix $\mathbf{Y}_2$ of size $(R_1 I_2) \times (I_3 I_4 \cdots I_N)$,

TT-SVD and TT-truncation II

Truncated SVD $\mathcal{X}_2$ is estimated from the leading left singular vectors of $\mathbf{Y}_2$, whereas the rank R_2 is chosen such that the norm of the residual is less than $\sqrt{1-\varepsilon^2}||\mathbf{Y}_2||_F$.

The sequential projection and truncation procedure is repeated in order to find the remaining core tensors.
The algorithm, summarised in Algorithm 3, executes only $(N-1)$ sequential data projections and $(N-1)$ truncated-SVDs in order to estimate N core tensors.

TT-SVD and TT-truncation III

Rounding Operation. For the decomposition with TT-ranks specified, TT-SVD is used for further truncation of tensor given in the TT-format.

In terms of the approximation accuracy Oseledets and Tyrtyshnikov (2009)

$$\|\mathcal{Y} - \mathcal{X}\|_F^2 \leq \sum_{k=1}^{N-1} \varepsilon_k^2 ,$$

where ε_k is the truncation error at the k-th step.

- When the data is subject to small noise and admits the TT-format, the TT-SVD works well
- Less efficient when data is heavily corrupted by noise or when the approximation is with relatively small ranks.

TT-SVD and TT-truncation IV

Algorithm 3: `TT-SVD`

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, TT-rank $(R_1, R_2, \ldots, R_{N-1})$ or approximation accuracy ε

Output: A TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ such that $\min \|\mathcal{Y} - \mathcal{X}\|_F^2$ or $\|\mathcal{Y} - \mathcal{X}\|_F^2 \le \varepsilon^2 \|\mathcal{Y}\|_F^2$

begin

    for $n = 1, \ldots, N-1$ **do**

        $\mathbf{Y} = \mathtt{reshape}(\mathcal{Y}, (I_n R_{n-1}) \times \prod_{k=n+1}^{N} I_k)$

        Truncated SVD $\mathbf{Y} \approx \mathbf{U}\, \mathrm{diag}(\boldsymbol{\sigma})\, \mathbf{V}^T$ with given rank R_n or such that $\|\boldsymbol{\sigma}\|_2^2 \ge (1 - \varepsilon^2)\|\mathbf{Y}\|_F^2$

        $\mathcal{X}_n = \mathtt{reshape}(\mathbf{U}, R_{n-1} \times I_n \times R_n)$

        $\mathcal{Y} \leftarrow \mathrm{diag}(\boldsymbol{\sigma})\, \mathbf{V}^T$

    end

    $\mathcal{X}_N = \mathcal{Y}$

end

First Example for TT-decomposition I

Figure: Benchmark images.

Fitting image by TT-decomposition. For color images of size $256 \times 256 \times 3$, apply horizontal and vertical shifts within a window of $[-2, 2]$ to generate 24 copies, which together with the original images created a tensor of size $25 \times 256 \times 256 \times 3$.
The data were then folded by the Kronecker folding Phan et al. (2012) to yield order-7 tensors, $\mathcal{Y}$, of size $25 \times 4 \times 4 \times 4 \times 4 \times 4 \times 192$.

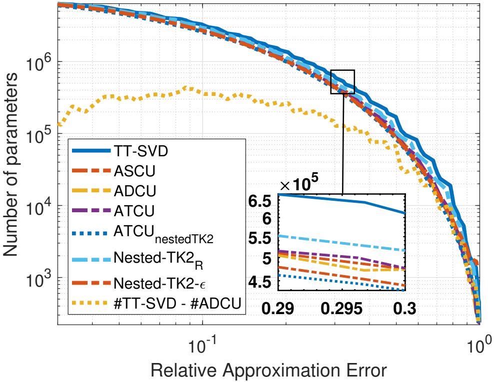
First Example for TT-decomposition II

We compare TT-models when the relative approximation errors, ε, were bounded, that is,

$$||\mathcal{Y} - \mathcal{X}||_F \leq \varepsilon \, ||\mathcal{Y}||_F$$

The number of model parameters exceeded the number of data entries 4915200, e.g., when the approximation error bound $\varepsilon < 0.03$.

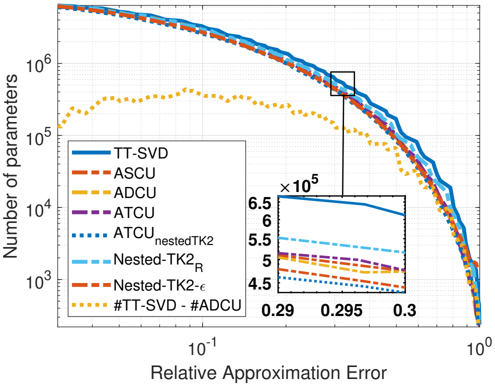
First Example for TT-decomposition



(a) For the decomposition of Lena image

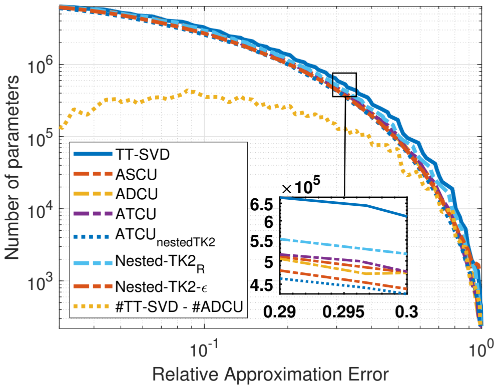
- TT-SVD yield models with higher number of parameters than other algorithms.

First Example for TT-decomposition



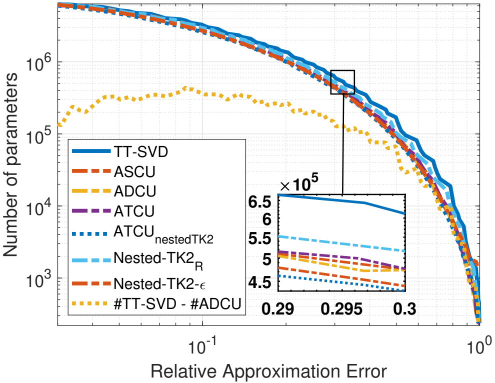
- For the relative aproximation error of 0.0870, TT-SVD yielded a model consisting of **3734573** parameters while the models estimated by ASCU, ADCU and ATCU needed **308964**, **439208**, **439272** fewer parameters to achieve the relative errors of 0.0851, 0.0870 and 0.0869, respectively.

First Example for TT-decomposition



- For the decomposition with low accuracies, the TT-models in all algorithms became relatively low-rank or rank-1; hence there was no much difference in terms of the number of parameters.

First Example for TT-decomposition



- The remarkably high difference exceeding 100000 parameters was observed when the relative approximation errors were lower than 0.4.

TT-SVD

Remark

- For the approximation with a given TT-rank, TT-SVD is not guaranteed to achieve the minimum approximation error.
- For the decomposition with bound constraint, TT-tensors obtained byTT-SVD often exhibit badly unbalaced TT-ranks.
- TT-SVD tends to select higher TT-ranks than needed for the decomposition problem with bound constraint.

Density Matrix Renormalization Group algorithm (DMRG)

- Another algorithm for the TT-decomposition White (1993); Holtz et al. (2012); **?** works as an alternating least squares algorithm.
- Each iteration it solves a minimisation problem over two consecutive core tensors, $\mathcal{X}_n$ and $\mathcal{X}_{n+1}$, by means of SVD, then updates $\mathcal{X}_{n+1}$ and $\mathcal{X}_{n+2}$ and so on.
- Like the TT-SVD, DMRG can determine the TT-ranks based on singular values
- Both algorithms are best suited to the decomposition with given ranks or when the error bound is negligible.

Alternating Multi-Core Update Algorithms I

Lemma (Frobenius norm of a TT-tensorOseledets et al. (2014))

Under the left-orthogonalisation up to $\mathcal{X}_n$, and the right-orthogonalisation up to $\mathcal{X}_m$, where $n \leq m$, the Frobenius norm of a TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ is equivalent to the Frobenius norm of $\mathcal{X}_{n:m}$, that is, $\|\mathcal{X}\|_F^2 = \|\mathcal{X}_{n:m}\|_F^2$.

Proof.

With the left and right orthogonalisations, the two matricizations $[\mathcal{X}_{<n}]_{(n)}^T$ and $[\mathcal{X}_{>m}]_{(1)}^T$ are orthogonal matrices. Hence, $\|\mathcal{X}\|_F^2 = \|[\mathcal{X}_{<n}]_{(n)}^T \bullet \mathcal{X}_{n:m} \bullet [\mathcal{X}_{>m}]_{(1)}\|_F^2 = \|\mathcal{X}_{n:m}\|_F^2$. $\square$

Alternating Multi-Core Update Algorithms II

- Assume that the TT-tensor $\mathcal{X}$ is left-orthogonalised up to $\mathcal{X}_n$ and right-orthogonalized up to $\mathcal{X}_m$, where in our methods m can take one of the values $n, n+1$ or $n+2$.
- Let $\mathcal{X}_{n:m} = \mathcal{X}_n \bullet \mathcal{X}_{n+1} \bullet \cdots \bullet \mathcal{X}_m$, then following Lemma 4, the error function can be written as

$$\begin{aligned} D &= \|\mathcal{Y} - \mathcal{X}\|_F^2 \\ &= \|\mathcal{Y}\|_F^2 + \|\mathcal{X}\|_F^2 - 2\langle \mathcal{Y}, \mathcal{X}\rangle \\ &= \|\mathcal{Y}\|_F^2 + \|\mathcal{X}_{n:m}\|_F^2 - 2\langle \mathcal{T}_{n:m}, \mathcal{X}_{n:m}\rangle \\ &= \|\mathcal{Y}\|_F^2 - \|\mathcal{T}_{n:m}\|_F^2 + \|\mathcal{T}_{n:m} - \mathcal{X}_{n:m}\|_F^2 \end{aligned} \tag{4}$$

where $\mathcal{T}_{n:m}$ is of size $R_{n-1} \times I_n \times \cdots \times I_m \times R_m$ represents a tensor contraction between $\mathcal{Y}$ and $\mathcal{X}$ along all modes but the modes-$(n, n+1, \ldots, m)$

$$\mathcal{T}_{n:m} = (\mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y}) \rtimes_{N-m} \mathcal{X}_{>m} \quad \text{for} \;\; n = 1, 2, \ldots \tag{5}$$

Alternating Multi-Core Update Algorithms III

Sub optimization problem The objective function in (4) indicates that the sub TT-tensor $\mathcal{X}_{n:m}$ is the best approximation to $\mathcal{T}_{n:m}$.
Following on this, we can update $(m - n + 1)$ core tensors $\mathcal{X}_n, \ldots, \mathcal{X}_m$, while fixing the other cores $\mathcal{X}_j$, for $j < n$ or $j > m$.

Single core update , i,e., $m = n$, the algorithm sequentially updates first the core tensors $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{N-1}$, and then $\mathcal{X}_N, \mathcal{X}_{N-1}, \ldots, \mathcal{X}_2$.

DMRG like update , $m = n + 1$, the update can be performed with overlapping core indices, e.g., $(\mathcal{X}_1, \mathcal{X}_2)$, $(\mathcal{X}_2, \mathcal{X}_3)$, $\ldots$, as in the DMRG optimisation scheme White (1993).

Alternating Multi-Core Update Algorithms IV

Algorithm 4: `Alternating Multi-Core Update`

Input: $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, and rank-$(R_1, R_2, \ldots, R_{N-1})$ or bound ε^2,
k : the number of core tensors to be updated per iteration
Output: $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$such that $\min \|\mathcal{Y} - \mathcal{X}\|_F^2$ (or $\|\mathcal{Y} - \mathcal{X}\|_F^2 \leq \varepsilon^2$)
begin
  Initialize $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$, e.g, by rounding $\mathcal{Y}$ **repeat**
    % `Left-to-Right update-------------------------------------`
    for $n = 1, s+1, 2s+1, \ldots$ **do**
      $\mathcal{T}_{n:m} = \mathcal{L}_n \ltimes_{N-m} \mathcal{X}_{>m}$ /* $m = n + k - 1, \mathcal{L}_1 = \mathcal{Y}$ */
      $[\mathcal{X}_n, \ldots, \mathcal{X}_m] = \texttt{bestTT\_approx}(\mathcal{T}_{n:m})$
      for $i = n, n+1, \ldots, n+s-1$ **do**
        $\mathcal{X} = \texttt{Left\_Orthogonalize}(\mathcal{X}, i)$
        $\mathcal{L}_{i+1} = \mathcal{X}_i \ltimes_2 \mathcal{L}_i$ /* `Update left-contracted tensor` */
      end
    end
    % `Right-to-Left update-------------------------------------`
    for $n = \widetilde{N}, \widetilde{N} - s, \widetilde{N} - 2s, \ldots$ **do**
      $\mathcal{T}_{n:m} = \mathcal{L}_n \ltimes_{N-m} \mathcal{X}_{>m}$
      $[\mathcal{X}_n, \ldots, \mathcal{X}_m] = \texttt{bestTT\_approx}(\mathcal{T}_{n:m})$
      for $i = m, m-1, \ldots, m-s+1$ **do**
        $\mathcal{X} = \texttt{Right\_Orthogonalize}(\mathcal{X}, i)$
      end
    end
  until *a stopping criterion is met*

Alternating Single-Core Update I

For this special case, the error function becomes

$$D = \|\mathcal{Y}\|_F^2 - \|\mathcal{T}_n\|_F^2 + \|\mathcal{T}_n - \mathcal{X}_n\|_F^2 \quad \text{for} \quad n = 1, 2, \ldots, N. \tag{6}$$

where $\mathcal{T}_n$ is of size $R_{n-1} \times I_n \times R_n$.
For the **TT-decomposition with given accuracy**, $\mathcal{X}_n$ should have minimum number of parameters, such that

$$\|\mathcal{T}_n - \mathcal{X}_n\|_F^2 \leq \varepsilon_n^2 \tag{7}$$

where $\varepsilon_n^2 = \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{T}_n\|_F^2$ is assumed to be non-negative.

Remark

Alternating Single-Core Update II

- A negative accuracy ε_n^2 indicates that either the rank R_{n-1} or R_n is quite small, and needs to be increased, that is, the core $\mathfrak{X}_{n-1}$ or $\mathfrak{X}_{n+1}$ should be adjusted to have higher ranks.
- Often, the TT-ranks R_n are set to sufficiently high values, and then gradually decrease or at least behave in a non-increasing manner during the update of the core tensors.

Alternating Single-Core Update III

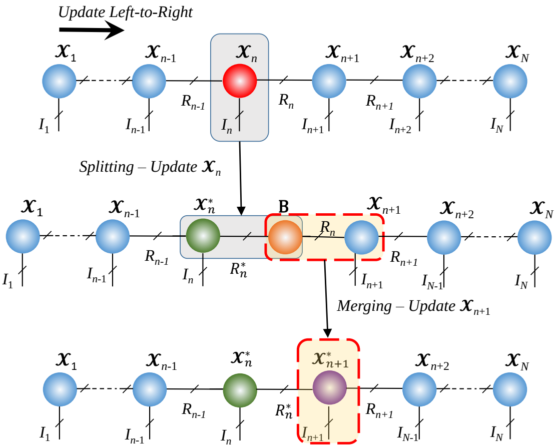


Figure: Update scheme of the ASCU algorithm for the case of two-core update. The core tensor $\mathcal{X}_n$ is split into two core tensors, $\mathcal{X}^*_n$ and **B**, with a minimal rank R^*_n. The core tensor $\mathcal{X}_{n+1}$ is then updated by **B**.

Alternating Single-Core Update IV

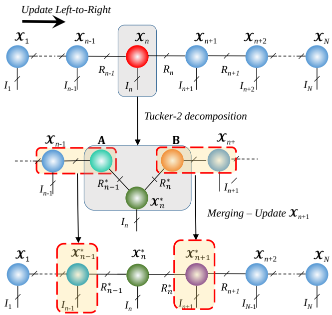


Figure: Update scheme of the ASCU algorithm for the case of three core update. The core tensor $\mathfrak{X}_n$ is approximated by TK2 decomposition, $\mathbf{A} \bullet \mathfrak{X}_n^* \bullet \mathbf{B}$ with minimal ranks, R_{n-1}^* and R_n^*. The two core tensors, $\mathfrak{X}_{n-1}$ and $\mathfrak{X}_{n+1}$, are then updated by $\mathbf{A}$ and $\mathbf{B}$, respectively.

TT-SVD as a variant of ASCU with one update round I

- Consider the approximation of a tensor $\mathcal{Y}$ of size $I_1 \times I_2 \times \cdots \times I_N$ using the ASCU algorithm with one-side rank adjustment at a given accuracy ε^2.
- Horizontal slices of the core tensors $\mathcal{X}_n$ are initialized by unit vectors $\boldsymbol{e}_r$ as $\mathrm{vec}(\mathbf{X}_n(r,:,:)) = \boldsymbol{e}_r$, for $r = 1, 2, \ldots, R_{n-1}$, where $R_n = \prod_{k=n+1}^{N} I_k$, i.e., the mode-1 matricizations of the core tensors are identity matrices, $[\mathcal{X}_n]_{(1)} = \mathbf{I}_{R_{n-1}}$.
- The contracted tensor $\mathcal{T}_1$ is the data $\mathcal{Y}$, and the mode-1 approximation error is simply the global approximation error $\varepsilon_1^2 = \varepsilon^2$. That means, the ASCU estimates the first core tensor $\mathcal{X}_1$ as in TT-SVD.
- Since the core tensors $\mathcal{X}_3, \ldots, \mathcal{X}_N$ are not updated, the contracted tensor $\mathcal{T}_2$ represents the projection of $\mathcal{Y}$ onto the subspace spanned by $\mathcal{X}_1$, implying that ASCU estimates $\mathcal{X}_2$ in a similar way as TT-SVD.

TT-SVD as a variant of ASCU with one update round II

- The difference here is that the mode-2 approximation accuracy ε_2^2 in ASCU is affected by the term $\|\mathcal{Y}\|_F^2 - \|\mathcal{T}_2\|_F^2$ (see (7)) which is only zero or negligible for the exact or high accuracy approximation $\varepsilon \approx 0$.
- The remaining core tensors $\mathcal{X}_3, \ldots, \mathcal{X}_N$ are updated similarly, but with different approximation accuracies.
- Another difference is that TT-SVD estimates the core tensors once, while ASCU runs the right-to-left update after it completes the first round left-to-right update, and so on.
- To summarise, TT-SVD acts as ASCU with one update cycle, but with a different error tolerance. ASCU attains an approximation error closer to the predefined accuracy.

Nested network of TK2 I

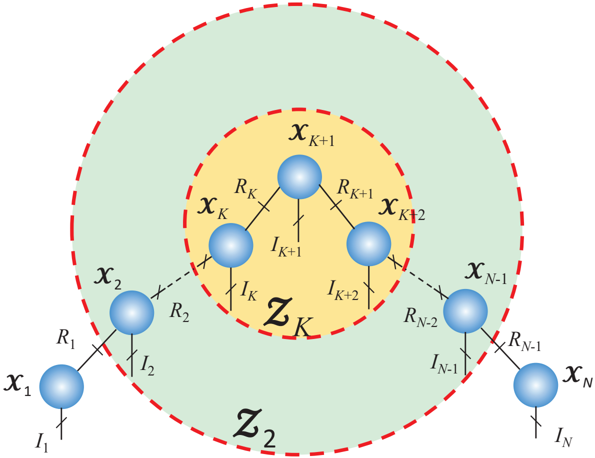


Figure: A nested network of TK2 decompositions forms a TT network.

Nested network of TK2 II

- Consider a TK2 decomposition of $\mathcal{Y}$ which gives two core tensors, $\mathcal{X}_1$ and $\mathcal{X}_N$, for the first and last modes

$$\mathcal{Y} \approx \mathcal{X}_1 \bullet \mathcal{Z}_2 \bullet \mathcal{X}_N \,.$$

  where $\mathcal{Z}_2 = \mathcal{X}_1^T \bullet \mathcal{Y} \bullet \mathcal{X}_N^T$ is of size $R_1 \times I_2 \times I_3 \times \cdots \times I_{N-1} \times R_{N-1}$.
- The matrices are estimated from a matrix of size $I_1 I_N \times I_1 I_N$.
- Next, we estimate two core tensors $\mathcal{X}_2$ of size $R_1 \times I_2 \times R_2$ and $\mathcal{X}_{N-1}$ of size $R_{N-2} \times I_{N-1} \times R_{N-1}$ within TK2 of $\mathcal{Z}_2$

$$\mathcal{Z}_2 \approx \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-2}$$

  where $\mathcal{Z}_3$ is a tensor of size $R_2 \times I_3 \times I_4 \times \cdots \times I_{N-2} \times R_{N-2}$.

Nested network of TK2 III

- It can be verified that estimation of the three core tensors $\mathcal{X}_2$, $\mathcal{Z}_3$, $\mathcal{X}_{N-2}$ within the TT-tensor $\mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1} \bullet \mathcal{X}_N$, while fixing the two orthogonal matrices, $\mathcal{X}_1$ and $\mathcal{X}_N$, becomes the estimation of a TK2 decomposition of $\mathcal{Z}_2$

$$\begin{aligned} &\|\mathcal{Y} - \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1} \bullet \mathcal{X}_N\|_F^2 \\ &\quad = \|\mathcal{Y}\|_F^2 - \|\mathcal{Z}_2\|_F^2 + \|\mathcal{Z}_2 - \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1}\|_F^2. \end{aligned}$$

- Similarly, we perform TK2 decomposition of $\mathcal{Z}_3$ to get the core tensors $\mathcal{X}_3$ and $\mathcal{X}_{N-3}$.

Nested network of TK2 IV

Nested network of TK2 V

Nested TK2 with bound constraint

- In the first layer, $\mathcal{X}_1$ and $\mathcal{X}_N$ are estimated within a smallest TK2 model such that

$$\|\mathcal{Y} - \mathcal{X}_1 \bullet \mathcal{Z}_2 \bullet \mathcal{X}_N\|_F^2 = \|\mathcal{Y}\|_F^2 - \|\mathcal{Z}_2\|_F^2 \leq \varepsilon^2 \,.$$

  This is achieved when $\|\mathcal{Y}\|_F^2 - \|\mathcal{Z}_2\|_F^2$ is close to or attains the bound $\varepsilon_1^2 = \varepsilon^2$ so that $\mathcal{X}_1$ and $\mathcal{X}_N^T$ have small ranks.
- In the second layer, we solve a TK2 with a much smaller bound

$$\|\mathcal{Z}_2 - \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1}\|_F^2 \leq \varepsilon_2^2 = \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{Z}_2\|_F^2 \ll \varepsilon_1^2.$$

  A similar procedure is applied to the core tensors $\mathcal{Z}_3, \mathcal{Z}_4, \ldots$, but the approximation errors are decreasing significantly.

Nested network of TK2 VI

- If the bound is attained in the first layer, i.e., $\varepsilon_2^2 = \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{Z}_2\|_F^2 = 0$, then higher layers solve exact TK2 models. Implying that the factor matrices within TK2 will have full rank or very high rank, i.e., $R_3 \approx R_2 I_2$, $R_{N-1} \approx I_{N-1} R_N$, $R_4 \approx R_3 I_3 \approx R_2 I_2 I_3$.
  In this case, dimensions of the core tensors, especially the central cores, grow dramatically, and as a result the final TT-model is not very compact. This behavior is also observed in the TT-SVD.

In order to deal with the large rank issue, we suggest to scale the error bounds in some first layers to smaller than the required bounds, e.g., by a factor of $\exp(-1 + n/\lfloor \frac{N}{2} \rfloor)$, where $n = 1, 2, \ldots, \lfloor \frac{N}{2} \rfloor$ is the layer index, $\lfloor \frac{N}{2} \rfloor$ greatest integer less than or equal to $N/2$.

Image denoising I

- Color images $\mathcal{T}$ degraded by additive Gaussian noise at SNR = 10 dB
- Constructed tensors, $\mathcal{Y}_{r,c}$, of a size $w \times w \times 3 \times (2d+1) \times (2d+1)$

$$\mathcal{Y}_{r,c}(:,:,:,d+1+i,d+1+j) = \mathcal{T}_{r+i,c+j}$$

comprising $(2d+1)^2$ blocks, around the block $\mathcal{T}_{r,c} = \mathcal{T}(r : r+w-1, c : c+w-1, :)$, where $i,j = -d,\ldots,0,\ldots,d$, and d represents the neighbour width.

Image denoising II

- Each tensor $\mathcal{Y}_{r,c}$ was then approximated with bounded approximation error, where $\delta^2 = 3\sigma^2 w^2 (2d+1)^2$, and σ the noise level.
- For a color image $\mathbf{Y}$ of size $I \times J \times 3$, degraded by additive Gaussian noise, the basic idea behind the proposed method is that for each block of pixels of size $h \times w \times 3$, given by $\mathbf{Y}_{r,c} = \mathbf{Y}(r : r+h-1, c : c+w-1, :)$, a small tensor $\mathcal{Y}_{r,c}$ of size $h \times w \times 3 \times (2d+1) \times (2d+1)$, comprising $(2d+1)^2$ blocks around $\mathbf{Y}_{r,c}$ is constructed, in the form

$$\mathcal{Y}_{r,c}(:,:,:,d+1+i,d+1+j) = \mathbf{Y}_{r+i,c+j}$$

, where $i,j = -d,\ldots,0,\ldots,d$, and d represents the neighbourhood width.

Image denoising III

- Every (r, c)-block $\mathbf{Y}_{r,c}$ is then approximated

$$\|\mathcal{Y}_{r,c} - \mathcal{X}_{r,c}\|_F^2 \leq \varepsilon^2$$

  where ε^2 is the noise level.
- A pixel is then reconstructed as an average of all its approximations by approximated tensors which cover that pixel.
- DCT spatial filtering used as a preprocessing.

Image denoising IV

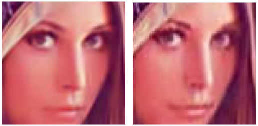
Table: Performance comparison of algorithms in terms of MSE (dB), PSNR (dB) and SSIM for image denoising when SNR = 10 dB.

Algorithms	MSE	PSNR	SSIM	MSE	PSNR	SSIM
	Lena			Pepper		
TT-SVD	35.11	32.68	0.892	40.40	32.07	0.861
TT-ASCU	**27.37**	**33.76**	**0.927**	**31.47**	**33.15**	**0.924**
TT-ADCU	28.04	33.65	0.926	32.09	33.07	0.923
Tucker	34.59	32.74	0.919	38.96	32.23	0.917
K-SVD	34.76	32.72	0.908	35.74	32.60	0.918
	Pens			Barbara		
TT-SVD	44.92	31.61	0.884	32.30	33.04	0.901
TT-ASCU	**36.61**	**32.50**	**0.908**	**24.92**	**34.16**	**0.934**
Tucker	48.56	31.27	0.884	33.20	32.92	0.919
K-SVD	50.04	31.14	0.862	35.41	32.64	0.908
	House			House2		
TT-SVD	23.70	34.38	0.877	41.07	32.00	0.905
TT-ASCU	**19.30**	**35.28**	**0.899**	**38.53**	**32.27**	**0.926**

Image denoising V

(a) Noisy image at SNR = 10 dB

(b) TT-ASCU, MSE = **27.37** dB

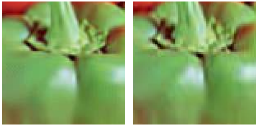


(c) From left to right, EPC(SSIM = 0.924), Tucker(0.919), TT-SVD(0.892), BRTF(0.840) and K-SVD(0.908)

Image denoising VI

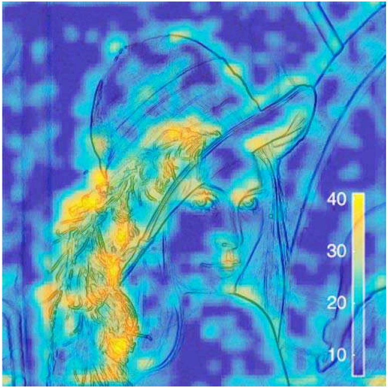
(d) Noisy image at SNR = 10 dB

(e) TT-ASCU, MSE = **31.47** dB

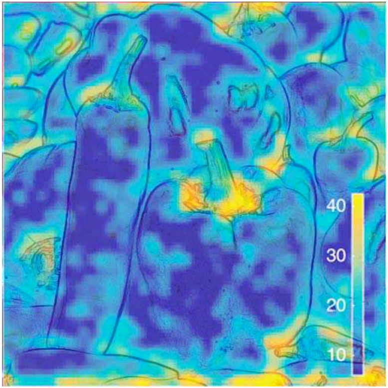


(f) From left to right, EPC(SSIM = 0.926), Tucker(0.917), TT-SVD(0.861), BRTF(0.825) and K-SVD(0.918)

Image denoising VII



(g) Lena



(h) Pepper

Figure: Visualization of the TT-rank maps. Each entry of the map expresses the average of the sum of the TT-ranks of the TT-tensors which cover the corresponding pixel.

Denoising with unknown target ranks I

- Consider noisy signals, $y(t) = x(t) + e(t)$, where $x(t)$ can take one of the following forms

$$x_1(t) = \frac{\sin(2000t^{2/3})}{4t^{1/4}}, \quad x_3(t) = \sin(\frac{5(t+1)}{2})\cos(100(t+1)^2),$$
$$x_2(t) = \sin(t^{-1}), \qquad x_4(t) = \mathrm{sign}(\sin(8\pi t))(1+\sin(80\pi t)),$$

  and $e(t)$ additive Gaussian noise.
- The signals $y(t)$ has length of $K = 2^{22}$, and were tensorized (reshaped) to tensors $\mathcal{Y}$ of order-22 and size $2 \times 2 \times \cdots \times 2$. With this tensorization, the five signals $x_r(t)$ can be well approximated by tensors in the TT-format.

Denoising with unknown target ranks II

Table: The TT-ranks of signals x_r of length $K = 2^{22}$ and of their estimates $\hat{x}_r$ using the TT-SVD and the AMCU algorithms.

Signal	TT-ranks	SAE (dB)	Time (s)
x_1	2-2-3-3-3-3-4-4-5-6-7-8-10-13-19-26-32-16-8-4-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-677-967-789-443-228-115-58-30-16-8-4-2	4.18	9.69
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-1-2-2-3-3-6-11-20-34-32-16-8-4-2	27.49	3.25
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-1-2-2-3-5-8-14-28-49-45-24-16-8-4-2	26.66	2.01
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-1-2-2-4-5-10-13-26-20-40-24-16-8-4-2	27.89	2.54
$\hat{x}_{ATCU_2}$	1-1-1-1-1-1-1-2-2-3-5-8-14-28-48-22-24-16-8-4-2	27.61	2.81
$\hat{x}_{ATCU_1}$	1-1-1-1-1-1-1-2-2-3-5-8-14-26-37-22-24-16-8-4-2	27.64	2.41

Denoising with unknown target ranks III

Table: The TT-ranks of signals x_r of length $K = 2^{22}$ and of their estimates $\hat{x}_r$ using the TT-SVD and the AMCU algorithms.

Signal	TT-ranks	SAE (dB)	Time (s)
x_2	2-4-8-16-32-56-47-38-32-26-22-18-15-13-12-10-8-7-6-4-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-675-959-782-440-226-114-57-28-15-8-4-2	6.18	9.63
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-2-4-8-13-21-35-65-92-54-27-15-8-4-2	22.73	2.08
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-2-3-5-8-12-20-37-71-94-52-26-13-7-4-2	23.10	1.52
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-2-4-4-8-11-22-36-72-85-54-27-15-8-4-2	23.34	1.45
$\hat{x}_{ATCU_2}$	1-1-1-1-1-1-2-3-5-8-12-20-37-70-104-56-28-13-7-4-2	23.11	1.83
$\hat{x}_{ATCU_0}$	1-1-1-1-1-1-1-2-4-7-11-22-37-66-105-54-27-15-8-4-2	23.07	1.67

Denoising with unknown target ranks IV

Signal	TT-ranks	SAE (dB)	Time (s)
x_3	2-2-2-2-2-3-3-3-3-4-4-4-5-6-7-9-12-16-8-4-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-677-966-789-443-228-115-58-29-15-8-4-2	4.41	9.67
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-1-1-1-2-2-2-3-4-7-11-13-8-4-2	31.48	3.10
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-1-1-1-2-2-3-6-10-18-32-16-8-8-4-2	32.47	2.15
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-1-2-1-2-2-4-2-4-6-12-16-16-8-4-2	34.58	2.52
$\hat{x}_{ATCU_2}$	1-1-1-1-1-1-1-1-1-2-2-3-5-8-14-23-8-8-8-4-2	33.52	2.77
$\hat{x}_{ATCU_0}$	1-1-2-1-1-2-1-1-2-1-2-3-2-3-6-6-9-16-8-4-2	31.49	2.58

Denoising with unknown target ranks V

Signal	TT-ranks	SAE (dB)	Time (s)
x_4	2-2-2-2-2-3-3-3-3-3-3-3-3-3-3-3-2-1-1-1		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-111-207-378-653-920-762-433-223-112-56-28-14-7-4-2	4.36	9.69
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-1-1-1-1-2-2-3-3-3-3-2-1-1-1	35.88	2.91
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-1-1-1-1-2-2-2-3-4-8-15-11-7-4-2	35.89	2.17
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-1-2-1-2-1-2-4-8-13-26-24-16-8-4-2	39.35	2.50
$\hat{x}_{ATCU_0}$	1-1-2-1-1-2-1-1-2-1-1-2-2-3-3-3-3-2-1-1-1	36.12	2.61
x_5	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-677-966-788-443-228-115-58-29-15-8-4-2	5.17	9.69
$\hat{x}_{ASCU}$	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	46.04	3.21
$\hat{x}_{ADCU_1}$	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	46.04	3.17
$\hat{x}_{ATCU_0}$	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	46.04	2.76

Part II

Higher Order CPD

Higher Order CPD I

Decomposition of tensors of higher order, e.g., 10, 20 is computationally expensive

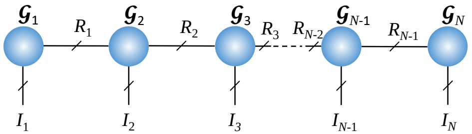
- computational costs of most existing algorithms for CPD increase exponentially with the tensor order
- computational cost for the tensor unfoldings, permutation of tensor entries Phan et al. (2013b); Vannieuwenhoven et al. (2015).

Higher Order CPD II

Existing Methods

- **Reshaping** a higher-order tensor into an order-3 tensor followed by a CP decomposition and calculation of the loading components Phan et al. (2013a); Bhaskara et al. (2014); Chiantini et al. (2017).
- **Compression** using e.g., the Tucker decomposition, can reduce the computational cost of CPD to $\mathcal{O}(NR^{N+1})$. However, only applicable when the estimated rank $R \leq I_n$.

Tensor Train I



Tensor-train (TT) decomposition

$$\mathcal{X} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{N-1}=1}^{R_{N-1}} \mathcal{G}_1(:, r_1) \circ \mathcal{G}_2(r_1, :, r_2) \circ \cdots \circ \mathcal{G}_N(r_{N-1}, :),$$

or

$$\mathcal{X} = \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N.$$

where $(R_1, R_2, \ldots, R_{N-1})$ represents the TT-rank of $\mathcal{X}$.

Tensor Train II

- TT-decomposition can be computed efficiently, while the ranks can be determined based on an approximation error boundVidal (2003); Oseledets and Tyrtyshnikov (2009); Phan et al. (2016)
- TT-decomposition is very suited to higher-order tensors.
- Exist a conversion from TT to CPD for the data which admits the CP model.

Lemma (**TT-representation of a K-tensor Oseledets (2011)**)

A K-tensor $\mathcal{Y} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$ *of rank-R can be expressed in a TT-format as*

$$\mathcal{Y} = \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N$$

where $\mathcal{G}_n$ *are of size* $R \times I_n \times R$, *for* $n = 2, \ldots, N-1$ *and* $\mathcal{G}_1 = \mathbf{A}^{(1)}$, $\mathcal{G}_N = \mathbf{A}^{(N)}$. *Then vertical slices of the core tensors* $\mathcal{G}_n$ *are diagonal matrices*

$$\mathcal{G}_n(:, i, :) = \operatorname{diag}(\mathbf{A}^{(n)}(i, :)) \,.$$

The conversion indicates that

$$\|\mathcal{Y} - \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_N\|_F^2 \leq \|\mathcal{Y} - [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]\|_F^2$$

where a TT-tensor $\mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N$ is of rank-$(R, \ldots, R)$.
For the exact case, i.e., when $\mathcal{Y}$ is of rank-R,

$$\mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!].$$

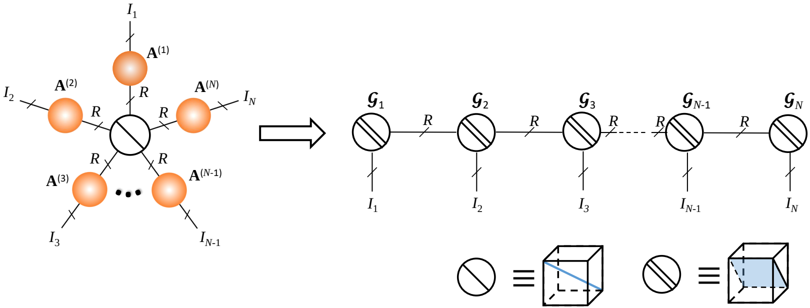


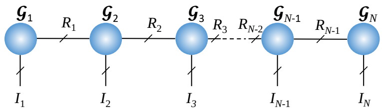
Figure: TT representation of a tensor in the Kruskal format.

Lemma (**Kruskal representation of a TT-tensor**)

A TT-tensor $\mathcal{Y} = \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N$ *of rank-*$(R_0, R_1, R_2, \ldots, R_N)$ *has an equivalent Kruskal-representation with* $R_1 R_2 \cdots R_N$ *rank-1 tensors*

$$\mathcal{Y} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$$

where $\mathbf{A}^{(n)} = [\mathcal{G}_n]_{(2)} \left(\mathbf{1}_{R_{>n}}^T \otimes \mathbf{I}_{R_{n-1}R_n} \otimes \mathbf{1}_{R_{<n-1}}^T\right)$
with $R_{<n} = R_0 R_1 \cdots R_{n-1}$ *and* $R_{>n} = R_{n+1} \cdots R_{N-1} R_N$.



$\equiv$

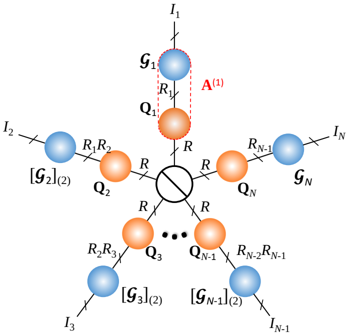


Figure: Kruskal representation of a tensor in the TT-format. $\mathbf{Q}_n = \mathbf{1}_{R_{>n}}^T \otimes \mathbf{I}_{R_{n-1}R_n} \otimes \mathbf{1}_{R_{<n-1}}^T$ represents the dependence matrix. See also Lemma 6.

The Kruskal representation of a TT-tensor usually exceeds the true rank of the tensor $\mathcal{Y}$.

Fast conversion from a TT-tensor to a K-tensor I

Lemma

$\mathcal{Y}$ has a unique CPD given by $[\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$, and a TT representation of rank-$(R, \ldots, R)$, that is

$$\mathcal{Y} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] = \mathbf{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathbf{G}_N. \tag{8}$$

Then

$$\mathcal{G}_n = [\![\mathbf{Q}_n, \mathbf{A}^{(n)}, \mathbf{S}_n]\!],$$

where $\mathbf{Q}_n$ and $\mathbf{S}_n$ are matrices of size $R \times R$ which hold

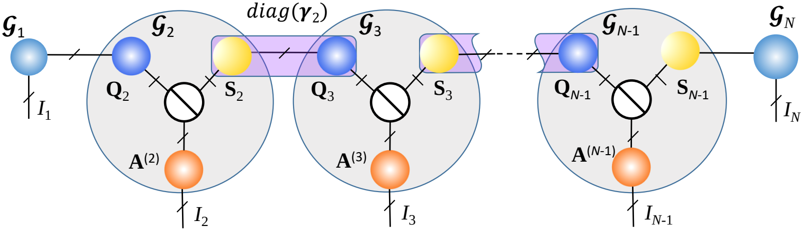
$$\mathbf{A}^{(1)} = \mathbf{G}_1 \, \mathbf{Q}_2 \, \mathrm{diag}(\boldsymbol{\gamma}_1)\,, \tag{9}$$

$$\mathbf{A}^{(N)} = \mathbf{G}_N^T \, \mathbf{S}_{N-1} \, \mathrm{diag}(\boldsymbol{\gamma}_N)\,, \tag{10}$$

$$\mathbf{S}_n \, \mathbf{Q}_{n+1} = \mathrm{diag}(\boldsymbol{\gamma}_n), \quad n = 2, \ldots, N-2, \tag{11}$$

and $\boldsymbol{\gamma}_1 \circledast \boldsymbol{\gamma}_2 \circledast \cdots \circledast \boldsymbol{\gamma}_{N-2} \circledast \boldsymbol{\gamma}_N = \mathbf{1}_R$.

Fast conversion from a TT-tensor to a K-tensor II



- Conversion of a tensor in the TT-format to a K-tensor through CPDs of the 3rd-order core tensors $\mathcal{G}_2, \ldots, \mathcal{G}_{N-1}$
- Big nodes designate the core tensors $\mathcal{G}_n$ and their CPDs, $\mathcal{G}_n = [\![\mathbf{Q}_n, \mathbf{A}^{(n)}, \mathbf{S}_n]\!]$.
- The factor matrix $\mathbf{A}^{(n)}$ can be retrieved from the 2nd factor matrix.

Fast conversion from a TT-tensor to a K-tensor III

Permutation ambiguity

The columns of $\mathbf{A}^{(n)}$ in CPDs of $\mathcal{G}_n$ may not match the ordering of columns of the other factor matrices.
Require appropriate permutations to reorder the columns of $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$.

$$\mathbf{S}_n^T \mathbf{Q}_{n+1} = \operatorname{diag}(\boldsymbol{\gamma}_n)\, \mathbf{P}_n.$$

The K-tensor of $\mathcal{G}_{n+1}$ can be permuted and normalised to give

$$\mathcal{G}_{n+1} = [\![\boldsymbol{\lambda}_{n+1} \circledast \mathbf{P}_n\, \boldsymbol{\gamma}_n ; \mathbf{Q}_{n+1} \mathbf{P}_n^T \operatorname{diag}(\mathbf{1}_R \oslash \boldsymbol{\gamma}_n), \mathbf{A}^{(n+1)}\, \mathbf{P}_n, \mathbf{S}_{n+1}\, \mathbf{P}_n]\!]$$

so that

$$\mathbf{S}_n^T\, \tilde{\mathbf{Q}}_{n+1} = \mathbf{S}_n^T\, \mathbf{Q}_{n+1} \mathbf{P}_n^T \operatorname{diag}(\mathbf{1}_R \oslash \boldsymbol{\gamma}_n) = \mathbf{I}_R\,.$$

Fast conversion from a TT-tensor to a K-tensor IV

Algorithm 5: TT to K-tensor conversion for the exact model

Input: TT-tensor $\mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_N$: $(I_1 \times I_2 \times \cdots \times I_N)$ of rank-$(R, \ldots, R)$

Output: A K-tensor $[\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]$ of rank R

begin

  % Rank-R CPD of $\mathcal{G}_n$ to find $\mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N-1)}$ ---------------------

  for $n = 2, \ldots, N-1$ **do**

    $\mathcal{G}_n \approx [\![\boldsymbol{\lambda}_n; \mathbf{Q}_n, \mathbf{A}^{(n)}, \mathbf{S}_n]\!]$

  end

  for $n = 2, \ldots, N-2$ **do**

    % Seek permutation matrices $\mathbf{P}_n$ -----------------------------

    $\mathbf{S}_n^T \mathbf{Q}_{n+1} \approx \operatorname{diag}(\boldsymbol{\gamma}_n)\mathbf{P}_n$

    % Reorder columns of $\mathbf{A}^{(n+1)}$ ---------------------------------

    $\mathbf{A}^{(n+1)} \leftarrow \mathbf{A}^{(n+1)}\mathbf{P}_n$, $\mathbf{S}^{(n+1)} \leftarrow \mathbf{S}^{(n+1)}\mathbf{P}_n$, $\boldsymbol{\lambda}_n \leftarrow \boldsymbol{\lambda}_n \circledast \mathbf{P}_n\boldsymbol{\gamma}_n$

  end

  $\boldsymbol{\lambda} = \boldsymbol{\lambda}_2 \circledast \boldsymbol{\lambda}_3 \circledast \cdots \circledast \boldsymbol{\lambda}_{N-1}$

  $\mathbf{A}^{(1)} = \mathbf{G}_1 \mathbf{Q}_2$, $\mathbf{A}^{(N)} = \mathbf{G}_N^T \mathbf{S}_{N-1} \operatorname{diag}(\boldsymbol{\lambda})$

end

Iterative Algorithm to Fit a Rank-R tensor to a TT-tensor I

Noisy or inexact model

Exact conversion does not work, but offers a good initial guess.
Need to solve

$$\min \quad D = \frac{1}{2}\|\mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_N - [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!]\|_F^2 . \quad (12)$$

Naive implementation

Iterative Algorithm to Fit a Rank-R tensor to a TT-tensor II

Replace the TT-tensor $\mathcal{G}$ by an equivalent Kruskal tensor $[\![\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!]$

$$\mathbf{U}^{(n)} = [\mathcal{G}_n]_{(2)} \left(\mathbf{1}^T_{R_{>(n)}} \otimes \mathbf{I}_{R_{n-1} R_n} \otimes \mathbf{1}^T_{R_{<n-1}} \right) . \tag{13}$$

New objective function

$$\min \quad D = \frac{1}{2} \| [\![\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!] - [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \|_F^2 .$$

with gradients

$$\frac{\partial D}{\partial \mathbf{A}^{(n)}} = \mathbf{U}^{(n)} \left(\circledast_{k \neq n} \mathbf{U}^{(k)T} \mathbf{A}^{(k)} \right) . \tag{14}$$

The ALS update rule for CPD can be expressed as

$$\mathbf{A}^{(n)} = \mathbf{U}^{(n)} \left(\circledast_{k \neq n} \mathbf{U}^{(k)T} \mathbf{A}^{(k)} \right) \left(\circledast_{k \neq n} \mathbf{A}^{(k)T} \mathbf{A}^{(k)} \right)^{-1} .$$

Expensive because of high number of $R_1 R_2 \cdots R_N$

Iterative Algorithm to Fit a Rank-R tensor to a TT-tensor III

Fast update rule

$$\mathbf{A}^{(n)} = [\mathcal{G}_n]_{(2)} \, (\Psi_{>n} \odot \Psi_{<n}) \, (\Gamma_n^*)^{-1} \, . \tag{15}$$

where contraction matrices, $\Psi_{>n}$ and $\Psi_{<n}$, is computed with a cost of $\mathcal{O}(I_n R^3)$

$$\Psi_{>n} = [\mathcal{G}_{n+1}]_{(1)} \left(\Psi_{>(n+1)} \odot \mathbf{A}^{(n+1)*}\right) , \tag{16}$$

$$\Psi_{<n} = [\mathcal{G}_{n-1}]_{(3)} \left(\mathbf{A}^{(n-1)*} \odot \Psi_{<(n-1)}\right) . \tag{17}$$

Algorithm 6: The TT2CP algorithm

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_d)$, and rank-$R$
Output: N factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$
begin

% **Stage 1:** `TT-decomposition of` $\mathcal{Y}$ `--------------------------------`

1 $\mathcal{Y} \approx \mathcal{X} = \mathcal{G}_1 \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{N-1} \bullet \mathcal{G}_N$

% **Stage 2:** `Convert TT-tensor` $\mathcal{X}$ `to a K-tensor of rank-`R `for the exact model------------------------------------------------------`

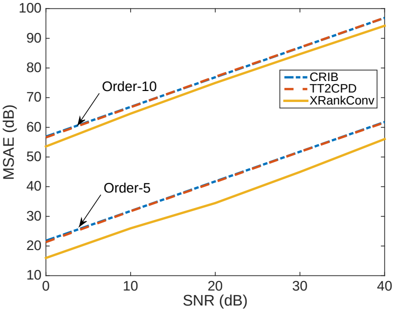
2 $[\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}] = \mathtt{TT\_to\_CPD}(\mathcal{X}, R)$

% **Stage 3:** `Fit a K-tensor of rank-`R `to TT-tensor -------------`

3 $[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}] = \mathtt{TT\_CPD}(\mathcal{X}, [\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}])$

end

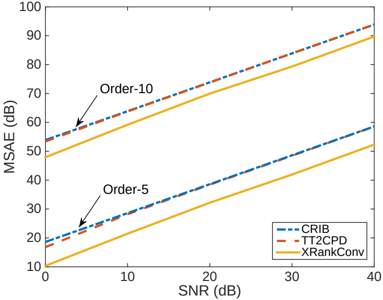
Example (Decomposition of random tensors)



(a) $R = 5$

Figure: The MSAE of components for CPD of order-5 and order-10 tensors of size $5 \times 5 \times \cdots \times 5$.

Example (Decomposition of random tensors)



(a) $R = 10$

Figure: The MSAE of components for CPD of order-5 and order-10 tensors of size $5 \times 5 \times \cdots \times 5$.

Blind identification (BI) in a system of 2 mixtures and R binary signals I

Blind identification

- Given only noisy observations, $\mathbf{X} = \mathbf{HS} + \mathbf{N}$ from R stationary sources, $\mathbf{S}$,
- Estimate the mixing matrix, $\mathbf{H} \in \mathbb{R}^{2 \times R}$, under some mild assumptions,
  i.e., the sources are statistically independent and non-Gaussian,
  their number is known,
  and the matrix **H** has no pairwise collinear columns (see also Yeredor (2000); Comon and Rajih (2006)).

Blind identification (BI) in a system of 2 mixtures and R binary signals II

- Yeredor (2000); Comon and Rajih (2006): Generate a higher-order tensor, $\mathcal{Y}$ from the observations, $\mathbf{X}$, by means of partial derivatives of the second Generalised Characteristic Functions (GCFs) of the observations, $\Phi_{\boldsymbol{x}}(\boldsymbol{u}) = \log \mathrm{E}\left[\exp(\boldsymbol{u}^T \boldsymbol{x})\right]$, at multiple processing points, $\boldsymbol{u}$ of length I

$$\begin{aligned}
\Psi_{\boldsymbol{x}}(\boldsymbol{u}) &= \frac{\partial^N \Phi_{\boldsymbol{x}}(\boldsymbol{u})}{\partial \boldsymbol{u}^N} = \frac{\partial^N \Phi_{\boldsymbol{s}}(\mathbf{H}^T \boldsymbol{u})}{\partial \boldsymbol{u}^N} \\
&= \Psi_{\boldsymbol{s}}(\mathbf{H}^T \boldsymbol{u}) \times_1 \mathbf{H} \times_2 \mathbf{H} \cdots \times_N \mathbf{H},
\end{aligned}$$

  where $\Psi_{\boldsymbol{s}}(\boldsymbol{v})$ are the Nth-order derivatives of $\Phi_{\boldsymbol{s}}(\boldsymbol{v})$ with respect to a vector, $\boldsymbol{v}$, of the length R,

- $\Psi_{\boldsymbol{s}}(\boldsymbol{v})$ is an Nth-order diagonal tensor, because the sources are statistically independent.

Blind identification (BI) in a system of 2 mixtures and R binary signals III

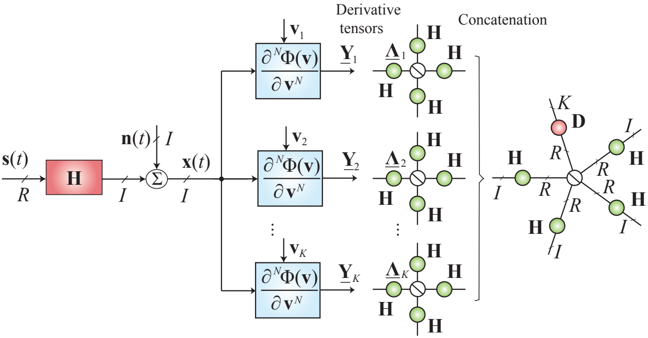


Figure: Blind identification estimates the mixing system **H** from only the knowledge of the noisy observations **X**.

Blind identification (BI) in a system of 2 mixtures and R binary signals IV

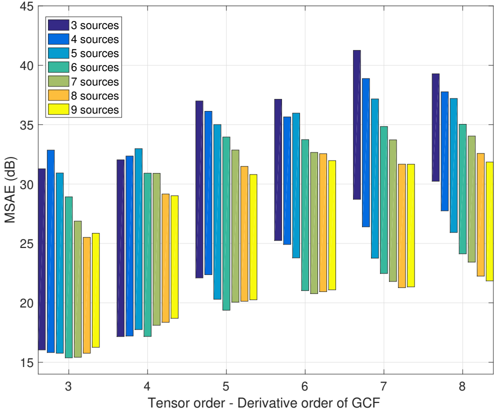


Figure: Mean SAE (in dB) achieved by CPD.

Blind identification (BI) in a system of 2 mixtures and R binary signals V

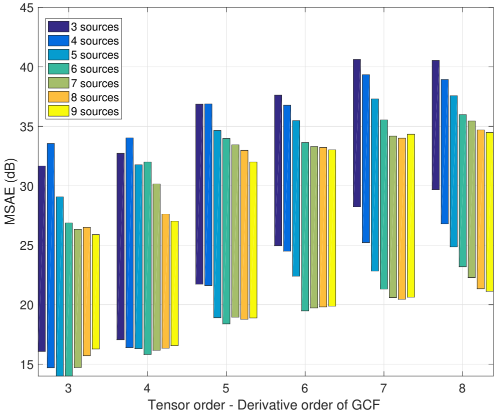


Figure: Mean SAE (in dB) by CPD aided with a prior TT decomposition.

Blind separation of damped sinusoid signals. I

Problem

Consider a noisy signal, $y(t)$, created as a combination of $R = 3$ complex valued damped sinusoids, $x_r(t)$, to yield

$$y(t) = a_1 x_1(t) + a_2 x_2(t) + a_3 x_3(t) + n(t),$$

where

$$x_r(t) = \exp(-i(\omega_r t + \phi_r) - \tau_r t),$$

and $\omega_r = 20\pi r$, $\tau_r = 2r$, $\phi_r = \frac{\pi r}{2R+1}$, $t = 0, 1/300, \ldots, (T-1)/300$, and $T = 413$ samples.

- In order to extract the source, $x_r(t)$, perform two steps: tensorization and tensor decomposition.
  - Construct an order-4 Toeplitz tensor of size $192 \times 16 \times 16 \times 192$

Blind separation of damped sinusoid signals. II

- Reshape it to an order-18 tensor of size $12 \times 2 \times 2 \times \cdots \times 2 \times 12$.
- Each signal $x_r(t)$ yields a tensor $\mathcal{X}_r$ of rank-1,
- The observed signal $y(t)$ yields a tensor of rank-$R = 3$.

Blind separation of damped sinusoid signals. III

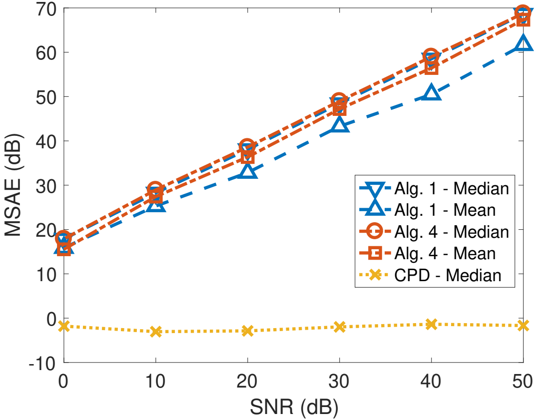


Figure: Mean SAE (in dB) in the estimation of the complex-valued damped sinusoids from a single mixture through CPDs of order-18 tensors.

Blind separation of damped sinusoid signals. IV

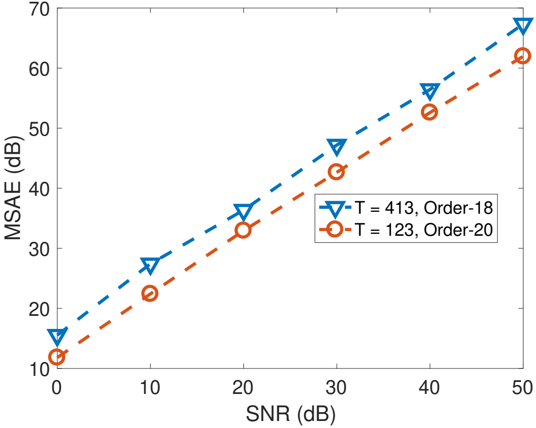


Figure: Mean SAE (in dB) in the estimation of the complex-valued damped sinusoids from a single mixture of length $T = 123$ and $T = 413$.

References I

Bhaskara, A., Charikar, M., Moitra, A., and Vijayaraghavan, A. (2014). Smoothed analysis of tensor decompositions. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 594–603, New York, NY, USA. ACM.

Chiantini, L., Ottaviani, G., and Vannieuwenhoven, N. (2017). Effective criteria for specific identifiability of tensors and forms. *SIAM Journal on Matrix Analysis and Applications*, 38(2):656–681.

Comon, P. and Rajih, M. (2006). Blind identification of under-determined mixtures based on the characteristic function. *Signal Processing*, 86(9):2271 – 2281. Special Section: Signal Processing in {UWB} Communications.

Holtz, S., Rohwedder, T., and Schneider, R. (2012). The alternating linear scheme for tensor optimization in the tensor train format. *SIAM J. Scientific Computing*, 34(2).

References II

Kressner, D. and Macedo, F. (2014). Low-rank tensor methods for communicating Markov processes. In *Quantitative Evaluation of Systems*, pages 25–40. Springer.

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

Oseledets, I. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Oseledets, I., Dolgov, S., Kazeev, V., Savostyanov, D., Lebedeva, O., Zhlobich, P., Mach, T., and Song, L. (2014). TT-Toolbox. https://github.com/oseledets/TT-Toolbox.

Oseledets, I. and Tyrtyshnikov, E. (2009). Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759.

Phan, A.-H., Cichocki, A., Tichavský, P., Mandic, D. P., and Matsuoka, K. (2012). On revealing replicating structures in multiway data: A novel tensor decomposition approach. In *Latent Variable Analysis and Signal Separation*, volume 7191 of *Lecture Notes in Computer Science*, pages 297–305. Springer Berlin Heidelberg.

Phan, A.-H., Cichocki, A., Uschmajew, A., Tichavsky, P., Luta, G., and Mandic, D. (2016). Tensor networks for latent variable analysis. Part I: Algorithms for tensor train decomposition. *ArXiv e-prints*.

Phan, A.-H., Tichavský, P., and Cichocki, A. (2013a). CANDECOMP/PARAFAC decomposition of high-order tensors through tensor reshaping. *IEEE Transactions on Signal Processing*, 61(19):4847–4860.

References IV

Phan, A.-H., Tichavský, P., and Cichocki, A. (2013b). Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. *Signal Processing, IEEE Transactions on*, 61(19):4834–4846.

Vannieuwenhoven, N., Meerbergen, K., and Vandebril, R. (2015). Computing the gradient in optimization algorithms for the cp decomposition in constant memory through tensor blocking. *SIAM Journal on Scientific Computing*, 37(3):C415–C438.

Vidal, G. (2003). Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902.

White, S. (1993). Density-matrix algorithms for quantum renormalization groups. *Physical Review B*, 48(14):10345.

Yeredor, A. (2000). Blind source separation via the second characteristic function. *Signal Processing*, 80(5):897–902.