Practical Sketching-Based Randomized Tensor Ring Decomposition

Yajie Yu[a, b]   Hanyu Li[a, c]

[a] Chongqing University

[b] zqyu@cqu.edu.cn

[c] hyli@cqu.edu.cn, lihy.hy@gmail.com

CQSIAM, June 11, 2022

Outline

1. Introduction
   - Tensor decompositions
   - Algorithms for TR decomposition
   - “Sketching”
2. TR-SRFT-ALS
   - Motivations
   - New findings
   - Algorithm and theoretical analysis
3. TR-TS-ALS
   - New findings
   - Algorithm and theoretical analysis
4. Numerical Results
   - Synthetic data
   - Real data
5. Conclusions

Outline

1 Introduction
- Tensor decompositions
- Algorithms for TR decomposition
- "Sketching"

2 TR-SRFT-ALS
- Motivations
- New findings
- Algorithm and theoretical analysis

3 TR-TS-ALS
- New findings
- Algorithm and theoretical analysis

4 Numerical Results
- Synthetic data
- Real data

5 Conclusions

Tensor

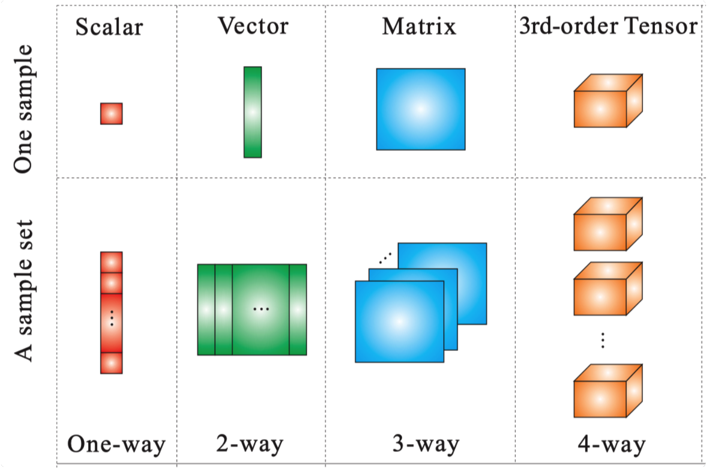


Figure 1: Graphical representation of multiway array (tensor) data.

Tensor

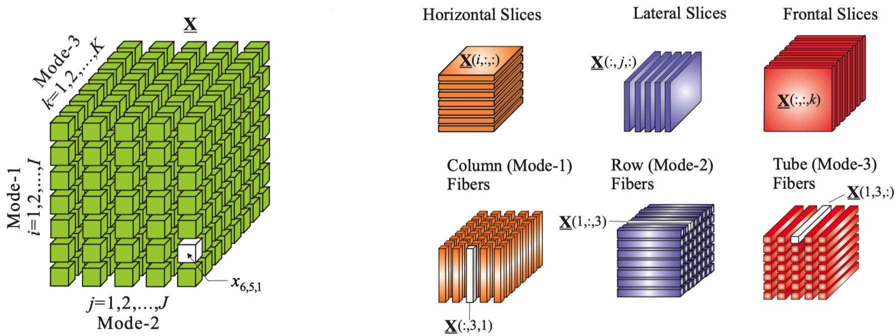


Figure 2: A 3rd-order tensor with entries, slices and fibers.

CP & Tucker decompositions

- CANDECOMP/PARAFAC (CP) decomposition.
  - The CP tensor decomposition aims to approximate an order-N tensor as a sum of R rank-one tensors;
  - $\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} = [[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}]]$;
  - $\mathcal{O}(NIR)$ parameters: is linear to the tensor order N.
- Tucker decomposition
  - The Tucker decomposition decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mode;
  - $\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} = [[\boldsymbol{\mathcal{G}}; \mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}]]$;
  - $\mathcal{O}\left(NIR + R^N\right)$ parameters: is exponential to the tensor order N.
- Some limitations

CP — Its optimization problem is difficult; it is difficult to find the optimal solution and CP-rank (NP-hard);

Tucker — Its number of parameters is exponential to tensor order. (Curse of Dimensionality)

CP & Tucker decompositions

- CANDECOMP/PARAFAC (CP) decomposition.
  - The CP tensor decomposition aims to approximate an order-N tensor as a sum of R rank-one tensors;
  - $\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} = [[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}]]$;
  - $\mathcal{O}(NIR)$ parameters: is linear to the tensor order N.
- Tucker decomposition
  - The Tucker decomposition decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mode;
  - $\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} = [[\boldsymbol{\mathcal{G}}; \mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}]]$;
  - $\mathcal{O}\left(NIR + R^N\right)$ parameters: is exponential to the tensor order N.
- Some limitations

  CP: Its optimization problem is difficult; it is difficult to find the optimal solution and CP-rank (NP-hard);

  Tucker: Its number of parameters is exponential to tensor order. (Curse of Dimensionality)

CP & Tucker decompositions

- CANDECOMP/PARAFAC (CP) decomposition.
  - The CP tensor decomposition aims to approximate an order-N tensor as a sum of R rank-one tensors;
  - $\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} = [[\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}]]$;
  - $\mathcal{O}(NIR)$ parameters: is linear to the tensor order N.
- Tucker decomposition
  - The Tucker decomposition decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mode;
  - $\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)} = [[\boldsymbol{\mathcal{G}}; \mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)}]]$;
  - $\mathcal{O}\left(NIR + R^N\right)$ parameters: is exponential to the tensor order N.
- Some limitations

CP
: Its optimization problem is difficult; it is difficult to find the optimal solution and CP-rank (NP-hard);

Tucker
: Its number of parameters is exponential to tensor order. (Curse of Dimensionality)

Tensor Train (TT) decomposition

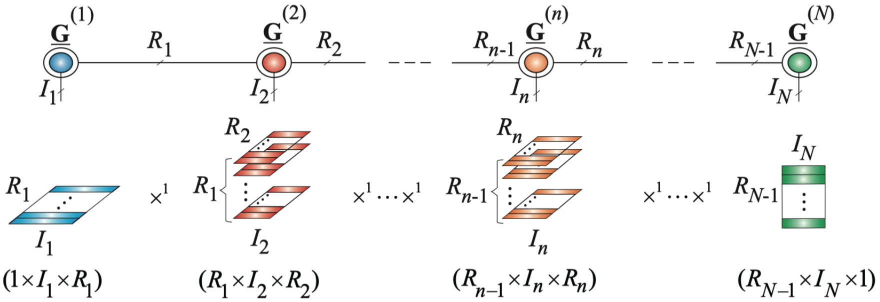


Figure 3: TT/MPS decomposition of an N-th order tensor $\boldsymbol{\mathcal{X}}$.

- Slice representation:

$$\boldsymbol{\mathcal{X}}(i_1, \cdots, i_N) = \mathbf{G}_1(i_1)\mathbf{G}_1(i_2)\cdots\mathbf{G}_N(i_N)$$

Tensor Train (TT) decomposition

- Limitations of TT decomposition:
  - The constraint on TT-ranks, i.e., $R_1 = R_{N+1} = 1$, leads to the limited representation ability and flexibility;
  - TT-ranks always have a fixed pattern, i.e., smaller for the border cores and larger for the middle cores, which might not be the optimum for specific data tensor;
  - The multilinear products of cores in TT decomposition must follow a strict order such that the optimized TT cores highly depend on the permutation of tensor dimensions. Hence, finding the optimal permutation remains a challenging problem.

Tensor Ring (TR) decomposition

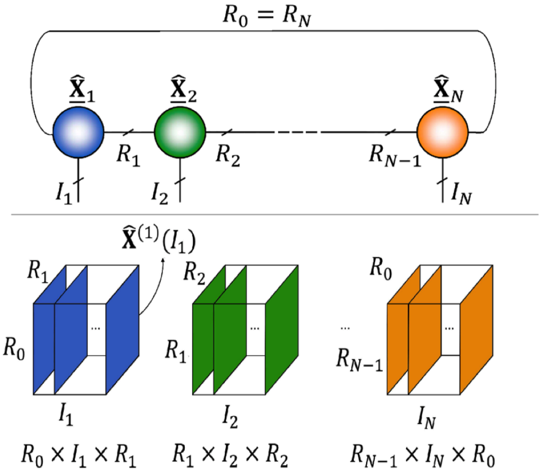


Figure 4: TR decomposition of an N-th order tensor $\boldsymbol{\mathcal{X}}$.

Tensor Ring (TR) decomposition

- Scalar representation:

$$\boldsymbol{\mathcal{X}}(i_1, \cdots, i_N) = \sum_{r_1, \cdots, r_N = 1}^{R_1, \cdots, R_N} \prod_{n=1}^{N} \boldsymbol{\mathcal{G}}_n(r_n, i_n, r_{n+1}); \quad R_1 = R_{N+1}$$

- Slice representation:

$$\boldsymbol{\mathcal{X}}(i_1, \cdots, i_N) = \mathsf{Tr}\{\mathbf{G}_1(i_1)\mathbf{G}_1(i_2) \cdots \mathbf{G}_N(i_N)\};$$

- Tensor representation:

$$\boldsymbol{\mathcal{X}} = \mathsf{Tr}\left(\mathbf{G}_1 \times^1 \mathbf{G}^2 \times^1 \cdots \times^1 \mathbf{G}_N\right);$$

- $\mathcal{O}\left(NIR^2\right)$ parameters: is linear to the tensor order N.

Tensor Ring (TR) decomposition

- Advantages of TR decomposition:
    - TR model has a more generalized and powerful representation ability than TT model, due to relaxation of the strict condition $R_1 = R_{N+1} = 1$ in TT decomposition. In fact, TT decomposition can be viewed as a special case of TR model; Overcome the first limitation of TT decomposition.
    - TR model is more flexible than TT model, because TR-ranks can be equally distributed in the cores; Overcome the second limitation of TT decomposition.
    - The multilinear products of cores in TR decomposition don't need a strict order, i.e., the circular dimensional permutation invariance. Overcome the third limitation of TT decomposition.
    - TR-ranks are usually smaller than TT-ranks because TR model can be represented as a linear combination of TT decompositions whose cores are partially shared.
- Batselier K. (2018). The Trouble with Tensor Ring Decompositions. arXiv:1811. 03813.

Classical algorithms for TR decomposition

Algorithm 1 TR-SVD [ZZX+16]

function $[\{\boldsymbol{\mathcal{G}}_n\}_{n=1}^N, R_1, \cdots, R_N] = \text{TR-SVD}(\boldsymbol{\mathcal{X}}, \varepsilon_p)$

  Compute truncation threshold δ_k for $k = 1$ and $k > 1$

  Choose one mode as the start point (e.g., the first mode) and obtain the 1-unfolding matrix $\mathbf{X}_{<1>}$

  Low-rank approximation by applying δ_1-truncated SVD: $\mathbf{X}_{<1>} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal + \mathbf{E}_1$

  Split ranks R_1, R_2 by

$$\min_{R_1, R_2} \|R_1 - R_2\|, \ s.t. \ \text{rank}_{\delta_1}(\mathbf{X}_{<1>})$$

  $\boldsymbol{\mathcal{G}}_1 \leftarrow \text{permute}(\text{shape}(\mathbf{U}, [I_1, R_1, R_2]), [2, 1, 3])$

  $\boldsymbol{\mathcal{G}}^{>1} \leftarrow \text{permute}(\text{shape}(\mathbf{\Sigma}\mathbf{V}^\intercal, [R_1, R_2, \prod_{j=2}^d]), [2, 3, 1])$

  for $k = 2, \cdots, N-1$ **do**

    $\boldsymbol{\mathcal{G}}^{>k-1} = \text{reshape}(\boldsymbol{\mathcal{G}}^{>k-1}, [R_k I_k, I_{k+1} \cdots I_N R_1])$

    Compute δ_k-truncated SVD:

$$\boldsymbol{\mathcal{G}}^{>k-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal + \mathbf{E}_k$$

    $R_{k+1} \leftarrow \text{rank}_{\delta_k}(\boldsymbol{\mathcal{G}}^{>k-1})$

    $\boldsymbol{\mathcal{G}}_k \leftarrow \text{shape}(\mathbf{U}, [R_k, I_k, R_{k+1}])$

    $\boldsymbol{\mathcal{G}}^{>k} \leftarrow \text{shape}(\mathbf{\Sigma}\mathbf{V}^\intercal, [R_{k+1}, \prod_{j=k+1}^N I_j, R_1])$

  end for

  return $\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_N$ and the TR-rank $R_1, \cdots, R_N$.

end function

Classical algorithms for TR decomposition

Algorithm 2 TR-ALS [ZZX$^+$16] [1]

function $\{\boldsymbol{\mathcal{G}}_n\}_{n=1}^{N} = \mathrm{TR\text{-}ALS}(\boldsymbol{\mathcal{X}}, R_1, \cdots, R_N)$
    Initialize cores $\boldsymbol{\mathcal{G}}_2, \cdots, \boldsymbol{\mathcal{G}}_N$
    repeat
        for $n = 1, \cdots, N$ **do**
            Compute $\mathbf{G}_{[2]}^{\neq n}$ from cores
            Update $\boldsymbol{\mathcal{G}}_n = \arg\min_{\boldsymbol{\mathcal{Z}}} \|\mathbf{G}_{[2]}^{\neq n} \mathbf{Z}_{(2)}^{\mathsf{T}} - \mathbf{X}_{[n]}^{\mathsf{T}}\|_F$
        end for
    until termination criteria met
    return $\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_N$
end function

[ZZX$^+$16] Zhao, Q., Zhou, G., Xie, S., & Zhang, L., Cichocki, A. (2016). Tensor Ring Decomposition. ArXiv:1606.05535.

[1] More details: (1) ALS with adaptive ranks and (2) block-wise ALS

Randomized algorithms for TR decomposition

Algorithm 3 rTR-ALS [YLCZ19]

function $\{\boldsymbol{\mathcal{G}}_n\}_{n=1}^N = \text{TR-rALS}(\boldsymbol{\mathcal{X}}, R_1, \cdots, R_N, K_1, \cdots, K_N)$
    for $n = 1, \cdots, N$ **do**
        Create matrix $\mathbf{M} \in \mathbb{R}_{i\neq n} I_i \times K_n$ following the Gaussian distribution.
        Compute $\mathbf{Y} = \mathbf{X}_{(n)}\mathbf{M}$ ▷ random projection
        $[\mathbf{Q}_n, \ ] = QR(\mathbf{Y})$ ▷ economy QR decomposition
        $\boldsymbol{\mathcal{P}} \leftarrow \boldsymbol{\mathcal{X}} \times_n \mathbf{Q}_n^\intercal$
    end for
    Obtain TR factors $[\boldsymbol{\mathcal{Z}}_n]$ of $\boldsymbol{\mathcal{P}}$ by TR-ALS or TR-SVD
    for $n = 1, \cdots, N$ **do**
        $\boldsymbol{\mathcal{G}}_n = \boldsymbol{\mathcal{Z}}_n \times_2 \mathbf{Q}_n$
    end for
    return $\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_N$
end function

[YLCZ19] Yuan, L., Li, C., Cao, J., & Zhao, Q. (2019). Randomized Tensor Ring Decomposition and its Application to Large-scale Data Reconstruction. ICASSP, 2127–2131.

[ACP+20] Ahmadi-Asl, S., Cichocki, A., Phan, A. H., Asante-Mensah, M. G., Ghazani, M. M., Tanaka, T., & Oseledets, I. (2020). Randomized algorithms for fast computation of low rank tensor ring model. Machine Learning: Science and Technology, 2(1), 011001.

Randomized algorithms for TR decomposition

Algorithm 4 TR-ALS-Sampled [MB21]

function $\{\boldsymbol{\mathcal{G}}_n\}_{n=1}^N = \text{TR-ALS-SAMPLED}(\boldsymbol{\mathcal{X}}, R_1, \cdots, R_N)$
  Initialize cores $\boldsymbol{\mathcal{G}}_2, \cdots, \boldsymbol{\mathcal{G}}_N$
  Using the leverage scores to compute distributions $\mathbf{p}^{(2)}, \cdots, \mathbf{p}^{(N)}$ without explicitly forming the subchain unfold matrix.
  repeat
    for $n = 1, \cdots, N$ **do**
      Set sample size J
      Draw sampling matrix $\mathbf{S} \sim \mathcal{D}(J, \mathbf{q}^{\neq n})$
      Compute $\hat{\boldsymbol{\mathcal{G}}}^{\neq n} = \text{SST}(\texttt{idxs}, \boldsymbol{\mathcal{G}}_{n+1}, \boldsymbol{\mathcal{G}}_N, \boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_{n-1})$ and $\hat{\mathbf{G}}_{[2]}^{\neq n}$
      Compute $\hat{\mathbf{X}}_{[n]}^{\mathsf{T}} = \mathbf{S}\mathbf{X}_{[n]}^{\mathsf{T}}$
      Update $\boldsymbol{\mathcal{G}}_n = \arg\min_{\boldsymbol{\mathcal{Z}}} \|\hat{\mathbf{G}}_{[2]}^{\neq n}\mathbf{Z}_{(2)}^{\mathsf{T}} - \hat{\mathbf{X}}_{[n]}^{\mathsf{T}}\|_F$
      Update n-th distribution $\mathbf{p}^{(n)}$
    end for
  until termination criteria met
  return $\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_N$
end function

[MB21] Malik, O. A., & Becker, S. (2021, July). A sampling-based method for tensor ring decomposition. In International Conference on Machine Learning (pp. 7400-7411). PMLR.

Randomized algorithms for TR decomposition

Algorithm 5 Sampled Subchain Tensor (SST) [MB21]

function $\mathcal{G}_S^{\neq n} = \mathrm{SST}(\texttt{idxs}, \mathcal{G}_{n+1}, \mathcal{G}_N, \mathcal{G}_1, \mathcal{G}_{n-1})$ ▷ $\mathcal{G}_n \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$
▷ $\texttt{idxs} \in \mathbb{R}^{m \times (N-1)}$ is from the set of tuples $\{i_{n+1}^{(j)}, \cdots, i_N^{(j)}, i_1^{(j)}, \cdots, i_{n-1}^{(j)}\}$ for $j \in [m]$
▷ idxs is retrieved from the sampling matrix $\mathbf{S} \in \mathbb{R}^{m \times \prod_{k \neq n} I_k}$ or the specific sampling with given probabilities
    Let $\mathcal{G}_S^{\neq n}$ be a tensor of size $R_{n+1} \times m \times R_n$, where every lateral slice is an $R_{n+1} \times R_n$ identity matrix
    for $k = n+1, \cdots, N, 1, \cdots, n-1$ **do**
        $\mathcal{G}_{(k)S}^{\neq n} \leftarrow \mathcal{G}_k(:, \texttt{idxs}(:, k), :)$
        $\mathcal{G}_S^{\neq n} \leftarrow \mathcal{G}_S^{\neq n} \circledast_2 \mathcal{G}_{(k)S}^{\neq n}$ ▷ see Definition 3.2 for $\circledast_2$.
    end for
    return $\mathcal{G}_S^{\neq n}$
end function

Some sketching techniques

- Randomized Algorithms
  - Sampling
    - Uniform
    - Importance
      - Based on norm
      - Based on leverage scores
  - Projection
    - Gaussian
      - Kronecker Gaussian
      - Khatri-Rao Gaussian
    - SRFT/SRHT
      - Kronecker FJLT
      - Khatri-Rao FJLT
    - CountSketch
      - TensorSketch
      - Higher-order CountSketch

SRFT

Definition 1.1 (SRFT)

The **SRFT** is constructed as a matrix of the form

$$\mathbf{\Phi} = \mathbf{S}\mathcal{F}\mathbf{D},$$

where

- $\mathbf{S} \in \mathbb{R}^{m \times N} = m$ random rows of the $N \times N$ identity matrix;
- $\mathcal{F} \in \mathbb{C}^{N \times N} =$ (unitary) discrete Fourier transform of dimension N;
- $\mathbf{D} \in \mathbb{R}^{N \times N} =$ diagonal matrix with diagonal entries drawn uniformly from $\{+1, -1\}$.

Kronecker SRFT (KSRFT)

Definition 1.2 (KSRFT)

The **KSRFT** is constructed as a matrix of the form

$$\mathbf{\Phi} = \mathbf{S}\left(\bigotimes_{j=D}^{1} \mathcal{F}_j \mathbf{D}_j\right),$$

where

- $\mathbf{S} \in \mathbb{R}^{m \times N} = m$ random rows of the $N \times N$ identity matrix with $N = \prod_{i=1}^{D} n_j$;
- $\mathcal{F}_j \in \mathbb{C}^{n_j \times n_j} =$ (unitary) discrete Fourier transform of dimension n_j;
- $\mathbf{D}_j \in \mathbb{R}^{n_j \times n_j} =$ diagonal matrix with diagonal entries drawn uniformly from $\{+1, -1\}$.

[BBK18] Battaglino, C., Ballard, G., & Kolda, T. G. (2018). A Practical Randomized CP Tensor Decomposition. SIMAX, 39(2), 876-901.
[JKW20] Jin, R., Kolda, T. G., & Ward, R. (2021). Faster Johnson–Lindenstrauss transforms via kronecker products. Information and Inference: A Journal of the IMA, 10(4), 1533-1562.

CountSketch

Definition 1.3 (CountSketch)

The **CountSketch** is constructed as a matrix of the form

$$\mathbf{\Phi} = \mathbf{\Omega}\mathbf{D},$$

where

- $\mathbf{\Omega} \in \mathbb{R}^{m\times N}$ = a matrix with $\mathbf{\Omega}(j,i) = 1$ if $j = h(i)$, $\forall i \in [N]$ and $\mathbf{\Omega}(j,i) = 0$ otherwise, where $h : [N] \to [m]$ is a hash map such that $\forall i \in [N]$ and $\forall j \in [m]$, $\Pr[h(i) = j] = 1/m$;
- $\mathbf{D} \in \mathbb{R}^{N\times N}$ = diagonal matrix with diagonal entries drawn uniformly from $\{+1, -1\}$.

[CW17] Clarkson K L, & Woodruff D P. (2017). Low-rank approximation and regression in input sparsity time. Journal of the ACM, 63(6), 1-45.

TensorSketch

Definition 1.4 (TensorSketch)

The order N **TensorSketch** matrix $\mathbf{T} = \mathbf{\Omega D} \in \mathbb{R}^{m \times \prod_{i=1}^N I_i}$ is defined based on two hash maps H and S defined below,

$$H : [I_1] \times [I_2] \times \cdots \times [I_N] \to [m] : (i_1, \ldots, i_N) \mapsto \left(\sum_{n=1}^{N} (H_n(i_n) - 1) \mod m \right) + 1,$$

$$S : [I_1] \times [I_2] \times \cdots \times [I_N] \to \{-1, 1\} : (i_1, \ldots, i_N) \mapsto \prod_{n=1}^{N} S_n(i_n),$$

where each H_n for $n \in [N]$ is a 3-wise independent hash map that maps $[I_n] \to [m]$, and each S_n is a 4-wise independent hash map that maps $[I_n] \to \{-1, 1\}$. A hash map is k-wise independent if any designated k keys are independent random variables. Specifically, the two matrices $\mathbf{\Omega}$ and $\mathbf{D}$ are defined based on H and S, respectively, as follows,

- $\mathbf{\Omega} \in \mathbb{R}^{m \times \prod_{i=1}^N I_i}$ is a matrix with $\mathbf{\Omega}(j, i) = 1$ if $j = H(i)\ \forall i \in \left[\prod_{i=1}^N I_i\right]$, and $\mathbf{\Omega}(j, i) = 0$ otherwise,
- $\mathbf{D} \in \mathbb{R}^{\prod_{i=1}^N I_i \times \prod_{i=1}^N I_i}$ is a diagonal matrix with $\mathbf{D}(i, i) = S(i)$.

Above we use the notation $H(i) = H(\overline{i_1 i_2 \cdots i_N})$ and $S(i) = S(\overline{i_1 i_2 \cdots i_N})$, where $\overline{i_1 i_2 \cdots i_N}$ denotes the **big-endian convention**.

Outline

1 Introduction
- Tensor decompositions
- Algorithms for TR decomposition
- "Sketching"

2 TR-SRFT-ALS
- Motivations
- New findings
- Algorithm and theoretical analysis

3 TR-TS-ALS
- New findings
- Algorithm and theoretical analysis

4 Numerical Results
- Synthetic data
- Real data

5 Conclusions

Motivation: CP-ALS

- Classical CP

CP-ALS

$$\arg\min_{\mathbf{A}_n} \|\mathbf{Z}^{(n)}\mathbf{A}_n^\intercal - \mathbf{X}_{(n)}^\intercal\|_F.$$

$$\mathbf{Z}^{(n)} = \mathbf{A}_N \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n+1} \odot \cdots \odot \mathbf{A}_1.$$

- Randomized CP in [BBK18] [2]

Rand-CP

$$\arg\min_{\mathbf{A}_n} \|\mathbf{S}\left(\bigotimes_{j=N, j\neq n}^{1} \mathcal{F}_j\mathbf{D}_j\right)\mathbf{Z}^{(n)}\mathbf{A}_n^\intercal - \mathbf{S}\left(\bigotimes_{j=N, j\neq n}^{1} \mathcal{F}_j\mathbf{D}_j\right)\mathbf{X}_{(n)}^\intercal\|_F.$$

$$\hat{\mathbf{Z}}^{(n)} = \left(\bigotimes_{j=N, j\neq n}^{1} \mathcal{F}_j\mathbf{D}_j\right)\mathbf{Z}^{(n)} = \bigodot_{j=N, j\neq n}^{1}(\mathcal{F}_j\mathbf{D}_j\mathbf{A}_j).$$

[2] [BBK18] Battaglino, C., Ballard, G., & Kolda, T. G. (2018). A Practical Randomized CP Tensor Decomposition. SIAM Journal on Matrix Analysis and Applications, 39(2), 876-901.

Ideas

- Original problem: TR-ALS

$$\arg\min_{\mathbf{G}_{n(2)}} \|\mathbf{G}_{[2]}^{\neq n}\mathbf{G}_{n(2)}^{\intercal} - \mathbf{X}_{[n]}^{\intercal}\|_F. \tag{2.1}$$

- Reduced problem: Sketched TR-ALS

$$\arg\min_{\mathbf{G}_{n(2)}} \left\|\mathcal{S}\mathbf{G}_{[2]}^{\neq n}\mathbf{G}_{n(2)}^{\intercal} - \mathcal{S}\mathbf{X}_{[n]}^{\intercal}\right\|_F.$$

- Ideas
  - Avoid forming $\mathcal{S}$ explicitly.
  - Avoid forming $\mathbf{G}_{[2]}^{\neq n}$ explicitly.
  - Avoid the classical matrix multiplication of $\mathcal{S}$ and $\mathbf{G}_{[2]}^{\neq n}$ directly.

New findings

1 Mixing the rows of $\mathbf{G}_{[2]}^{\neq n}$ is equivalent to mixing the lateral slides of $\boldsymbol{\mathcal{G}}^{\neq n}$, i.e., $\mathcal{S}\mathbf{G}_{[2]}^{\neq n} = (\boldsymbol{\mathcal{G}}^{\neq n} \times_2 \mathcal{S})_{[2]}$.

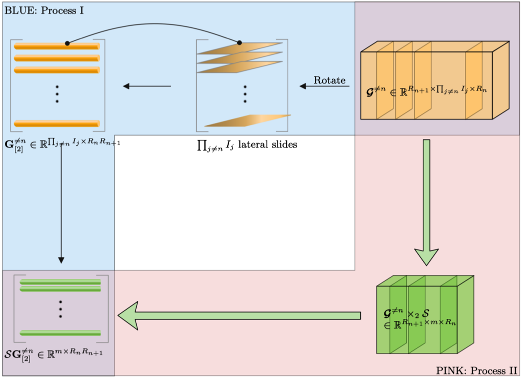


Figure 5: Illustration of the transformation from Process I to Process II.

2 $\boldsymbol{\mathcal{G}}^{\neq n}$ may be written as a Kronecker-like or KR-like product of TR-cores.

New definition

Definition 2.1 (Subchain product)

Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times J_1 \times K}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{K \times J_2 \times I_2}$ be two 3-order tensors, and $\mathbf{A}(j_1)$ and $\mathbf{B}(j_2)$ be the j_1-th and j_2-th lateral slices of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$, respectively. The mode-2 **subchain product** of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ is a tensor of size $I_1 \times J_1 J_2 \times I_2$ denoted by $\boldsymbol{\mathcal{A}} \boxtimes_2 \boldsymbol{\mathcal{B}}$ and defined as

$$(\boldsymbol{\mathcal{A}} \boxtimes_2 \boldsymbol{\mathcal{B}})(\overline{j_1 j_2}) = \boldsymbol{\mathcal{A}}(j_1)\boldsymbol{\mathcal{B}}(j_2).$$

That is, with respect to the correspondence on indices, the lateral slices of $\boldsymbol{\mathcal{A}} \boxtimes_2 \boldsymbol{\mathcal{B}}$ are the classical matrix products of the lateral slices of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$. The mode-1 and mode-3 subchain products can be defined similarly.

Therefore, $\boldsymbol{\mathcal{G}}^{\neq n}$ can be rewritten as

$$\boldsymbol{\mathcal{G}}^{\neq n} = \boldsymbol{\mathcal{G}}_{n+1} \boxtimes_2 \cdots \boxtimes_2 \boldsymbol{\mathcal{G}}_N \boxtimes_2 \boldsymbol{\mathcal{G}}_1 \boxtimes_2 \cdots \boxtimes_2 \boldsymbol{\mathcal{G}}_{n-1}. \tag{2.2}$$

New proposition

$$\begin{aligned}\mathcal{S}\mathbf{G}_{[2]}^{\neq n} &= (\boldsymbol{\mathcal{G}}^{\neq n} \times_2 \mathcal{S})_{[2]} \\ &= ((\boldsymbol{\mathcal{G}}_{n+1} \boxtimes_2 \cdots \boxtimes_2 \boldsymbol{\mathcal{G}}_N \boxtimes_2 \boldsymbol{\mathcal{G}}_1 \boxtimes_2 \cdots \boxtimes_2 \boldsymbol{\mathcal{G}}_{n-1}) \times_2 \mathcal{S})_{[2]}\end{aligned}$$

Proposition 2.2

Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times J_1 \times K}$ *and* $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{K \times J_2 \times I_2}$ *be two 3-order tensors, and* $\mathbf{A} \in \mathbb{R}^{R_1 \times J_1}$ *and* $\mathbf{B} \in \mathbb{R}^{R_2 \times J_2}$ *be two matrices. Then*

$$(\boldsymbol{\mathcal{A}} \times_2 \mathbf{A}) \boxtimes_2 (\boldsymbol{\mathcal{B}} \times_2 \mathbf{B}) = (\boldsymbol{\mathcal{A}} \boxtimes_2 \boldsymbol{\mathcal{B}}) \times_2 (\mathbf{B} \otimes \mathbf{A}).$$

Idea on algorithm

- Choose the "$\mathcal{S}$".
  - Let $\mathcal{S} = \mathbf{S}\mathcal{F}\mathbf{D}$, where

$$\mathcal{F} = \left(\bigotimes_{j=n-1,\cdots,1,N,\cdots,n+1} \mathcal{F}_j \right), \ \mathbf{D} = \left(\bigotimes_{j=n-1,\cdots,1,N,\cdots,n+1} \mathbf{D}_j \right).$$

That is,

$$\mathcal{S} = \mathbf{S} \left(\bigotimes_{j=n-1,\cdots,1,N,\cdots,n+1} \mathcal{F}_j \mathbf{D}_j \right).$$

  - Thus,

$$\underset{\mathbf{G}_{n(2)}}{\arg\min} \left\| \mathbf{S}\mathcal{F}\mathbf{D}\mathbf{G}_{[2]}^{\neq n} \mathbf{G}_{n(2)}^{\intercal} - \mathbf{S}\mathcal{F}\mathbf{D}\mathbf{X}_{[n]}^{\intercal} \right\|_F, \tag{2.3}$$

[BBK18] Battaglino, C., Ballard, G., & Kolda, T. G. (2018). A Practical Randomized CP Tensor Decomposition. SIAM Journal on Matrix Analysis and Applications, 39(2), 876-901.

Details

- The first term in eq. (2.3), $\mathbf{S}\mathcal{F}\mathbf{D}\mathbf{G}_{[2]}^{\neq n}$:

Step 1 (Mixing step) Using Proposition 2.2 and eq. (2.2)

$$\begin{aligned}\hat{\mathcal{G}}^{\neq n} &= \mathcal{G}^{\neq n} \times_2 \mathcal{F}\mathbf{D} \\ &= (\mathcal{G}_{n+1} \times_2 (\mathcal{F}_{n+1}\mathbf{D}_{n+1}))\boxtimes_2 \\ &\cdots \boxtimes_2 (\mathcal{G}_N \times_2 (\mathcal{F}_N\mathbf{D}_N)) \boxtimes_2 (\mathcal{G}_1 \times_2 (\mathcal{F}_1\mathbf{D}_1))\boxtimes_2 \\ &\cdots \boxtimes_2 (\mathcal{G}_{n-1} \times_2 (\mathcal{F}_{n-1}\mathbf{D}_{n-1})).\end{aligned}$$

i.e. $\mathcal{F}\mathbf{D}\mathbf{G}_{[2]}^{\neq n} = \hat{\mathbf{G}}_{[2]}^{\neq n}$.

Step 2 (Sampling step) According to the sampling method in Algorithm 5, we have

$$\begin{aligned}\hat{\mathcal{G}}^{\neq n} \times_2 \mathbf{S} = &(\mathcal{G}_{n+1} \times_2 (\mathbf{S}_{n+1}\mathcal{F}_{n+1}\mathbf{D}_{n+1}))\circledast_2 \\ &\cdots \circledast_2 (\mathcal{G}_N \times_2 (\mathbf{S}_N\mathcal{F}_N\mathbf{D}_N)) \circledast_2 (\mathcal{G}_1 \times_2 (\mathbf{S}_1\mathcal{F}_1\mathbf{D}_1))\circledast_2 \\ &\cdots \circledast_2 (\mathcal{G}_{n-1} \times_2 (\mathbf{S}_{n-1}\mathcal{F}_{n-1}\mathbf{D}_{n-1})),\end{aligned}$$

using Proposition 3.3, we have $\mathbf{S} = \left(\bigodot_{\substack{j=n-1,\cdots,1,\\ N,\cdots,n+1}} \mathbf{S}_j^{\mathsf{T}}\right)^{\mathsf{T}}$

[MB21] Malik, O. A., & Becker, S. (2021, July). A sampling-based method for tensor ring decomposition. In International Conference on Machine Learning (pp. 7400-7411). PMLR.

Details

- The second term in eq. (2.3), $\mathbf{S}\mathcal{F}\mathbf{D}\mathbf{X}_{[n]}^{\intercal}$:
  - Let $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{X}} \times_1 \mathcal{F}_1\mathbf{D}_1 \times_2 \mathcal{F}_2\mathbf{D}_2 \cdots \times_N \mathcal{F}_N\mathbf{D}_N$.
  - The second term is equivalent to

$$\mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal}(\mathbf{D}_n\mathcal{F}_n^*)^{\intercal}.$$

- Rewrite eq. (2.3) as

$$\underset{\mathbf{G}_{n(2)}}{\arg\min} \left\| \left(\mathbf{S}\hat{\mathbf{G}}_{[2]}^{\neq n}\right) \mathbf{G}_{n(2)}^{\intercal} - \left(\mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal}\right) (\mathbf{D}_n\mathcal{F}_n^*)^{\intercal} \right\|_F.$$

Algorithm

Algorithm 6 TR-SRFT-ALS (Proposal)

function $\{\boldsymbol{\mathcal{G}}_n\}_{n=1}^{N} = \text{TR-SRFT-ALS}(\boldsymbol{\mathcal{X}}, R_1, \cdots, R_N, m)$ ▷ $\boldsymbol{\mathcal{G}}_n \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$; $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
▷ $(R_1, \cdots, R_N)$ are the TR-ranks
▷ m is the uniform sampling size

  Initialize cores $\boldsymbol{\mathcal{G}}_2, \cdots, \boldsymbol{\mathcal{G}}_N$
  Define random sign-flip operators $\mathbf{D}_j$ and FFT matrices $\mathcal{F}_j$, for $j \in [N]$
  Mix cores: $\hat{\boldsymbol{\mathcal{G}}}_n \leftarrow \boldsymbol{\mathcal{G}}_n \times_2 \mathcal{F}_n \mathbf{D}_n$, for $n = 2, \cdots, N$
  Mix tensor: $\hat{\boldsymbol{\mathcal{X}}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathcal{F}_1 \mathbf{D}_1 \times_2 \mathcal{F}_2 \mathbf{D}_2 \cdots \times_N \mathcal{F}_N \mathbf{D}_N$
  repeat
    for $n = 1, \cdots, N$ **do**
      Define sampling operator $\mathbf{S} \in \mathbb{R}^{m \times \prod_{j \neq n} I_j}$
      Retrieve `idxs` from $\mathbf{S}$
      $\hat{\boldsymbol{\mathcal{G}}}_S^{\neq n} = \text{SST}(\texttt{idxs}, \hat{\boldsymbol{\mathcal{G}}}_{n+1}, \cdots, \hat{\boldsymbol{\mathcal{G}}}_N, \hat{\boldsymbol{\mathcal{G}}}_1, \cdots, \hat{\boldsymbol{\mathcal{G}}}_{n-1})$
      $\hat{\mathbf{X}}_{S[n]}^{\intercal} \leftarrow \mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal} (\mathbf{D}_n \mathcal{F}_n^*)^{\intercal}$
      Update $\boldsymbol{\mathcal{G}}_n = \arg\min_{\boldsymbol{\mathcal{Z}}} \|\hat{\mathbf{G}}_{S[2]}^{\neq n} \mathbf{Z}_{(2)}^{\intercal} - \hat{\mathbf{X}}_{S[n]}^{\intercal}\|_F$ subject to $\boldsymbol{\mathcal{G}}_n$ being real-valued
      $\hat{\boldsymbol{\mathcal{G}}}_n \leftarrow \boldsymbol{\mathcal{G}}_n \times_2 \mathcal{F}_n \mathbf{D}_n$
    end for
  until termination criteria met
  return $\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_N$
end function

Premix

- $$\arg\min_{\mathbf{G}_{n(2)}} \| \left(\mathbf{S}\hat{\mathbf{G}}_{[2]}^{\neq n}\right) \mathbf{G}_{n(2)}^{\intercal} - \left(\mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal}\right) (\mathbf{D}_n \mathcal{F}_n^{*})^{\intercal} \|_F.$$
- Rewrite it as
$$\arg\min_{\mathbf{G}_{n(2)}} \| \left(\mathbf{S}\hat{\mathbf{G}}_{[2]}^{\neq n}\right) \mathbf{G}_{n(2)}^{\intercal} (\mathcal{F}_n \mathbf{D}_n)^{\intercal} - \mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal} \|_F,$$
- Let $\hat{\mathbf{G}}_{n(2)} = \mathcal{F}_n \mathbf{D}_n \mathbf{G}_{n(2)}$
$$\arg\min_{\hat{\mathbf{G}}_{n(2)}} \| \left(\mathbf{S}\hat{\mathbf{G}}_{[2]}^{\neq n}\right) \hat{\mathbf{G}}_{n(2)}^{\intercal} - \left(\mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal}\right) \|_F.$$
- Solve the problem above to get $\hat{\mathcal{G}}_n$ first and then recover the original cores $\mathcal{G}_n$.

Algorithm

Algorithm 7 TR-SRFT-ALS-Premix (Proposal)

function $\{\mathcal{G}_n\}_{n=1}^N = \text{TR-SRFT-ALS-PREMIX}(\mathcal{X}, R_1, \cdots, R_N, m)$ ▷ $\mathcal{G}_n \in \mathbb{C}^{R_n \times I_n \times R_{n+1}}$; $\mathcal{X} \in \mathbb{C}^{I_1 \times \cdots \times I_N}$

▷ $(R_1, \cdots, R_N)$ are the TR-ranks
▷ m is the uniform sampling size

  Define random sign-flip operators $\mathbf{D}_j$ and FFT matrices $\mathcal{F}_j$, for $j \in [N]$
  Mix tensor: $\hat{\mathcal{X}} \leftarrow \mathcal{X} \times_1 \mathcal{F}_1\mathbf{D}_1 \times_2 \mathcal{F}_2\mathbf{D}_2 \cdots \times_N \mathcal{F}_N\mathbf{D}_N$
  Initialize cores $\hat{\mathcal{G}}_2, \cdots, \hat{\mathcal{G}}_N$
  repeat
    for $n = 1, \cdots, N$ **do**
      Define sampling operator $\mathbf{S} \in \mathbb{R}^{m \times \prod_{j \neq n} I_j}$
      Retrieve `idxs` from $\mathbf{S}$
      $\hat{\mathcal{G}}_S^{\neq n}$ = SST(`idxs`, $\hat{\mathcal{G}}_{n+1}, \cdots, \hat{\mathcal{G}}_N, \hat{\mathcal{G}}_1, \cdots, \hat{\mathcal{G}}_{n-1}$)
      $\hat{\mathbf{X}}_{S[n]}^{\intercal} \leftarrow \mathbf{S}\hat{\mathbf{X}}_{[n]}^{\intercal}$
      Update $\hat{\mathcal{G}}_n = \arg\min_{\mathcal{Z}} \|\hat{\mathbf{G}}_{S[2]}^{\neq n}\mathbf{Z}_{(2)}^{\intercal} - \hat{\mathbf{X}}_{S[n]}^{\intercal}\|_F$
    end for
  until termination criteria met
  for $n = 1, \cdots, N$ **do**
    Unmix cores: $\mathcal{G}_n \leftarrow \hat{\mathcal{G}}_n \times_2 \mathbf{D}_n\mathcal{F}_n^*$
  end for
  return $\mathcal{G}_1, \cdots, \mathcal{G}_N$
end function

Some remarks

- Like the algorithms for CP decomposition given in [BBK18][3], but with new tensor product and property;
- Compared with the method in [MB21][4], our method may work better for some special data, such as for the data with core tensors may include outliers;
- $\mathbf{F}_j\mathbf{D}_j$ can be any suitable randomized matrices: CountSketch, rTR-ALS[5], unified form.

[3]Battaglino, C., Ballard, G., & Kolda, T. G. (2018). A Practical Randomized CP Tensor Decomposition. SIAM Journal on Matrix Analysis and Applications, 39(2), 876-901.

[4]Malik, O. A., & Becker, S. (2021, July). A sampling-based method for tensor ring decomposition. In International Conference on Machine Learning (pp. 7400-7411). PMLR.

[5]Yuan, L., Li, C., Cao, J., & Zhao, Q. (2019). Randomized Tensor Ring Decomposition and its Application to Large-scale Data Reconstruction. ICASSP, 2127–2131.

Illustration

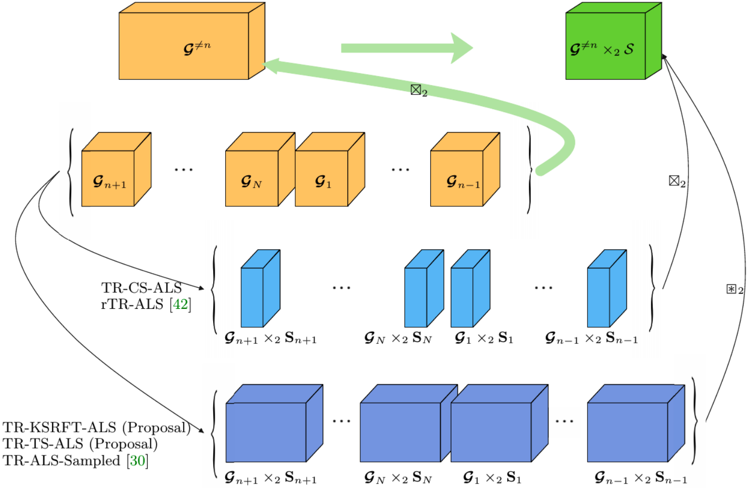


Figure 6: Illustration of how to efficiently construct $\mathcal{G}^{\neq n} \times_2 \mathcal{S}$ by sketching the core tensors.

Theoretical analysis

Theorem 2.3

For the subchain unfolded matrix $\mathbf{G}_{[2]}^{\neq n} \in \mathbb{R}^{\prod_{j\neq n} I_j \times R_n R_{n+1}}$ *and* $\mathbf{X}_{[n]}^{\intercal} \in \mathbb{R}^{\prod_{j\neq n} I_j \times I_n}$ *in eq. (2.1), denote* $rank(\mathbf{G}_{[2]}^{\neq n}) = r \leq R_n R_{n+1}$ *and fix* $\varepsilon, \eta \in (0,1)$ *such that* $\prod_{j\neq n} I_j \lesssim 1/\varepsilon^r$ *with integer* $r \geq 2$*. Then a sketching matrix* $\mathcal{S}$ *used in Algorithm 6 and Algorithm 7, i.e.,*

$$\mathcal{S} = \left(\bigodot_{\substack{j=n-1,\cdots,1,\\ N,\cdots,n+1}} \mathbf{S}_j^{\intercal} \right)^{\intercal} \left(\bigotimes_{\substack{j=n-1,\cdots,1,\\ N,\cdots,n+1}} (\mathcal{F}_j \mathbf{D}_j) \right) \in \mathbb{C}^{m \times \prod_{j\neq n} I_j}$$

with

$$m = \mathcal{O}\left(\varepsilon^{-1} r^{2(N-1)} \log^{2N-3}(\frac{r}{\varepsilon}) \log^4(\frac{r}{\varepsilon} \log(\frac{r}{\varepsilon})) \log \prod_{j\neq n} I_j \right)$$

is sufficient to output

$$\tilde{\mathbf{G}}_{n(2)}^{\intercal} = \underset{\mathbf{G}_{n(2)}^{\intercal} \in \mathbb{R}^{R_n R_{n+1} \times I_n}}{\arg\min} \|\mathcal{S}\mathbf{G}_{[2]}^{\neq n}\mathbf{G}_{n(2)}^{\intercal} - \mathcal{S}\mathbf{X}_{[n]}^{\intercal}\|_F,$$

such that

$$\mathbf{Pr}\left(\|\mathbf{G}_{[2]}^{\neq n}\tilde{\mathbf{G}}_{n(2)}^{\intercal} - \mathbf{X}_{[n]}^{\intercal}\|_F = (1 \pm \mathcal{O}(\varepsilon)) \min \|\mathbf{G}_{[2]}^{\neq n}\mathbf{G}_{n(2)}^{\intercal} - \mathbf{X}_{[n]}^{\intercal}\|_F \right) \geq 1 - \eta - 2^{-\Omega(\log \prod_{j\neq n} I_j)}.$$

Outline

1 Introduction
- Tensor decompositions
- Algorithms for TR decomposition
- “Sketching”

2 TR-SRFT-ALS
- Motivations
- New findings
- Algorithm and theoretical analysis

3 TR-TS-ALS
- New findings
- Algorithm and theoretical analysis

4 Numerical Results
- Synthetic data
- Real data

5 Conclusions

TensorSketch

Definition 3.1 (TensorSketch for Subchain Product)

The order N **TensorSketch** matrix $\mathbf{T} = \mathbf{\Omega}\mathbf{D} \in \mathbb{R}^{m \times \prod_{i=1}^{N} I_i}$ is defined based on two hash maps H and S defined below,

$$H : [I_1] \times [I_2] \times \cdots \times [I_N] \to [m] : (i_1, \dots, i_N) \mapsto \left(\sum_{n=1}^{N} (H_n(i_n) - 1) \mod m \right) + 1,$$

$$S : [I_1] \times [I_2] \times \cdots \times [I_N] \to \{-1, 1\} : (i_1, \dots, i_N) \mapsto \prod_{n=1}^{N} S_n(i_n),$$

where each H_n for $n \in [N]$ is a 3-wise independent hash map that maps $[I_n] \to [m]$, and each S_n is a 4-wise independent hash map that maps $[I_n] \to \{-1, 1\}$. A hash map is k-wise independent if any designated k keys are independent random variables. Specifically, the two matrices $\mathbf{\Omega}$ and $\mathbf{D}$ are defined based on H and S, respectively, as follows,

- $\mathbf{\Omega} \in \mathbb{R}^{m \times \prod_{i=1}^{N} I_i}$ is a matrix with $\mathbf{\Omega}(j, i) = 1$ if $j = H(i)$ $\forall i \in \left[\prod_{i=1}^{N} I_i\right]$, and $\mathbf{\Omega}(j, i) = 0$ otherwise,
- $\mathbf{D} \in \mathbb{R}^{\prod_{i=1}^{N} I_i \times \prod_{i=1}^{N} I_i}$ is a diagonal matrix with $\mathbf{D}(i, i) = S(i)$.

Above we use the notation $H(i) = H(\overline{i_1 i_2 \cdots i_N})$ and $S(i) = S(\overline{i_1 i_2 \cdots i_N})$, where $\overline{i_1 i_2 \cdots i_N}$ denotes the **little-endian convention**.

Related works

- Malik, O. A., & Becker, S. (2020). Fast randomized matrix and tensor interpolative decomposition using CountSketch. Advances in Computational Mathematics, 46(6), 76.
  - $\mathbf{P} = \mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(N)}$ for $n \in [N]$.
  - $\mathbf{TP} = \mathsf{FFT}^{-1}\left(\circledast_{n=1}^{N} \mathsf{FFT}\left(\mathbf{S}^{(n)}\mathbf{A}^{(n)}\right)\right)$.
- Malik, O. A., & Becker, S. (2018). Low-Rank Tucker Decomposition of Large Tensors Using TensorSketch. Advances in Neural Information Processing Systems, 31.
  - $\mathbf{P} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)}$ for $n \in [N]$.
  - $\mathbf{TP} = \mathsf{FFT}^{-1}\left(\left(\bigodot_{n=1}^{N}\left(\mathsf{FFT}\left(\mathbf{S}^{(n)}\mathbf{A}^{(n)}\right)\right)^{\intercal}\right)^{\intercal}\right)$.
- Pagh Rasmus. (2013). Compressed matrix multiplication. ACM Transactions on Computation Theory (TOCT).
- Diao, H., Song, Z., Sun, W.,& Woodruff, D. (2018). Sketching for Kronecker Product Regression and P-splines. International Conference on Artificial Intelligence and Statistics, 1299–1308.
- What about $\mathbf{TG}_{[2]}^{\neq n}$? Recall that

$$\boldsymbol{\mathcal{G}}^{\neq n} = \boldsymbol{\mathcal{G}}_{n+1} \boxtimes_2 \cdots \boxtimes_2 \boldsymbol{\mathcal{G}}_N \boxtimes_2 \boldsymbol{\mathcal{G}}_1 \boxtimes_2 \cdots \boxtimes_2 \boldsymbol{\mathcal{G}}_{n-1}.$$

New definition

Definition 3.2 (Slices-Hadamard product)

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times J \times K}$ and $\mathcal{B} \in \mathbb{R}^{K \times J \times I_2}$ be two 3-order tensors, and $\mathbf{A}(j)$ and $\mathbf{B}(j)$ are the j-th lateral slices of $\mathcal{A}$ and $\mathcal{B}$, respectively. The mode-2 **slices-Hadamard product** of $\mathcal{A}$ and $\mathcal{B}$ is a tensor of size $I_1 \times J \times I_2$ denoted by $\mathcal{A} \circledast_2 \mathcal{B}$ and defined as

$$(\mathcal{A} \circledast_2 \mathcal{B})(j) = \mathcal{A}(j)\mathcal{B}(j).$$

That is, the j-th lateral slice of $\mathcal{A} \circledast_2 \mathcal{B}$ is the classical matrix product of the j-th lateral slices of $\mathcal{A}$ and $\mathcal{B}$. The mode-1 and mode-3 slices-Hadamard product can be defined similarly.

New Propositions

Proposition 3.3

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times J_1 \times K}$ *and* $\mathcal{B} \in \mathbb{R}^{K \times J_2 \times I_2}$ *be two 3-order tensors, and* $\mathbf{A} \in \mathbb{R}^{M \times J_1}$ *and* $\mathbf{B} \in \mathbb{R}^{M \times J_2}$ *be two matrices. Then*

$$(\mathcal{A} \times_2 \mathbf{A}) \circledast_2 (\mathcal{B} \times_2 \mathbf{B}) = (\mathcal{A} \boxtimes_2 \mathcal{B}) \times_2 (\mathbf{B}^\intercal \odot \mathbf{A}^\intercal)^\intercal.$$

Proposition 3.4

Let $\mathbf{S}_n = \mathbf{\Omega}_n \mathbf{D}_n \in \mathbb{R}^{m \times I_n}$*, where* $\mathbf{\Omega}_n \in \mathbb{R}^{m \times I_n}$ *and* $\mathbf{D}_n \in \mathbb{R}^{I_n \times I_n}$ *are defined based on* H_n *and* S_n *in Definition 3.1. Let* $\mathbf{T} \in \mathbb{R}^{m \times \prod_{i=1}^{N} I_N}$ *be defined in Definition 3.1 and* $\mathcal{P} = \mathcal{A}^{(1)} \boxtimes_2 \mathcal{A}^{(2)} \boxtimes_2 \cdots \boxtimes_2 \mathcal{A}^{(N)}$ *with* $\mathcal{A}^{(n)} \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$ *for* $n \in [N]$*. Then*

$$\mathcal{P} \times_2 \mathbf{T} = FFT^{-1}\left(\circledast_2 {}_{n=1}^{N} FFT\left(\mathcal{A}^{(n)} \times_2 \mathbf{S}_n, [\,], 2 \right), [\,], 2 \right).$$

Algorithm

Algorithm 8 TR-TS-ALS (Proposal)

function $\{\boldsymbol{\mathcal{G}}_n\}_{n=1}^N = \text{TR-TS-ALS}(\boldsymbol{\mathcal{X}}, R_1, \cdots, R_N, m)$ ▷ $\boldsymbol{\mathcal{G}}_n \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$; $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
▷ $(R_1, \cdots, R_N)$ are the TR-ranks
▷ m is the embedding size
  Define $\mathbf{S}_j$, i.e., the CountSketch, based on H_n and S_n in Definition 3.1, for $j \in [N]$
  for $n = 1, \cdots, N$ **do**
    Build the TensorSketch $\mathbf{T}_{\neq n} \in \mathbb{R}^{m \times \prod_{j \neq n} I_j}$
    Compute the sketch of $X_{[n]}^{\intercal}$: $\hat{\mathbf{X}}_{[n]}^{\intercal} \leftarrow \mathbf{T}_{\neq n} \mathbf{X}_{[n]}^{\intercal}$
  end for
  Initialize cores $\boldsymbol{\mathcal{G}}_2, \cdots, \boldsymbol{\mathcal{G}}_N$
  repeat
    for $n = 1, \cdots, N$ **do**
      Compute $\hat{\boldsymbol{\mathcal{G}}}^{\neq n} = \boldsymbol{\mathcal{G}}^{\neq n} \times_2 \mathbf{T}_{\neq n} = \text{FFT}^{-1}\left(\circledast_{2\ j=n+1,\cdots,N}^{\ 1,\cdots,n-1} \text{FFT}\left(\boldsymbol{\mathcal{G}}_j \times_2 \mathbf{S}_n, [\,], 2\right), [\,], 2\right)$
      Update $\boldsymbol{\mathcal{G}}_n = \arg\min_{\boldsymbol{\mathcal{Z}}} \|\hat{\mathbf{G}}_{[2]}^{\neq n} \mathbf{Z}_{(2)}^{\intercal} - \hat{\mathbf{X}}_{[n]}^{\intercal}\|_F$
    end for
  until termination criteria met
  return $\boldsymbol{\mathcal{G}}_1, \cdots, \boldsymbol{\mathcal{G}}_N$
end function

Theoretical analysis

Theorem 3.5

For the subchain unfolded matrix $\mathbf{G}_{[2]}^{\neq n} \in \mathbb{R}^{\prod_{j\neq n} I_j \times R_n R_{n+1}}$ *and* $\mathbf{X}_{[n]}^{\intercal} \in \mathbb{R}^{\prod_{j\neq n} I_j \times I_n}$ *in eq.* (2.1)*, fix* $\varepsilon, \eta \in (0,1)$*. Then a TensorSketch* $\mathbf{T}_{\neq n}$ *used in Algorithm 8 with*

$$m = \mathcal{O}\left(((R_n R_{n+1} \cdot 3^{N-1})((R_n R_{n+1} + 1/\varepsilon^2)/\eta\right),$$

is sufficient to output

$$\tilde{\mathbf{G}}_{n(2)}^{\intercal} = \underset{\mathbf{G}_{n(2)}^{\intercal} \in \mathbb{R}^{R_n R_{n+1} \times I_n}}{\arg\min} \|\mathbf{T}_{\neq n} \mathbf{G}_{[2]}^{\neq n} \mathbf{G}_{n(2)}^{\intercal} - \mathbf{T}_{\neq n} \mathbf{X}_{[n]}^{\intercal}\|_F,$$

such that

$$\mathbf{Pr}\left(\|\mathbf{G}_{[2]}^{\neq n} \tilde{\mathbf{G}}_{n(2)}^{\intercal} - \mathbf{X}_{[n]}^{\intercal}\|_F = (1 \pm \mathcal{O}(\varepsilon)) \min \|\mathbf{G}_{[2]}^{\neq n} \mathbf{G}_{n(2)}^{\intercal} - \mathbf{X}_{[n]}^{\intercal}\|_F\right) \geq 1 - \eta.$$

Outline

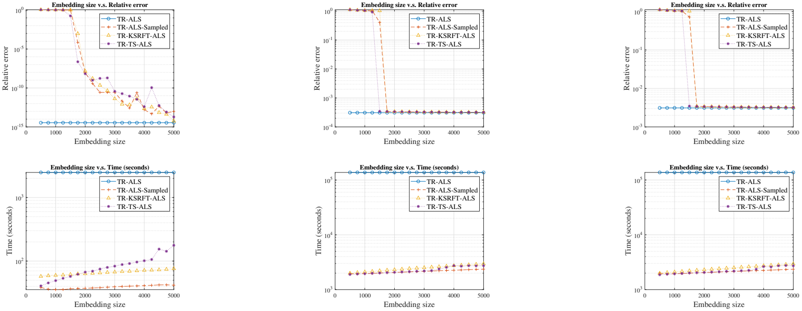
1. Introduction
   - Tensor decompositions
   - Algorithms for TR decomposition
   - "Sketching"
2. TR-SRFT-ALS
   - Motivations
   - New findings
   - Algorithm and theoretical analysis
3. TR-TS-ALS
   - New findings
   - Algorithm and theoretical analysis
4. Numerical Results
   - Synthetic data
   - Real data
5. Conclusions

The first experiment

- generate_low_rank_tensor($sz, ranks, noise, large\_elem$)
  - Create 3 cores of size $R_{true} \times I \times R_{true}$ with entries drawn independently from a standard normal distribution.
  - Set $large\_elem$ to increase the coherence;
  - $R_{true} = 10$;
  - sz = [I,I,I]= [500,500,500];
  - $ranks = R$;
  - $large\_elem$ = 20;
  - $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}_{ture} + noise \left(\frac{\|\boldsymbol{\mathcal{X}}_{ture}\|}{\|\boldsymbol{\mathcal{N}}\|} \right) \boldsymbol{\mathcal{N}}$.

[MB21] Malik, O. A., & Becker, S. (2021, July). A sampling-based method for tensor ring decomposition. In International Conference on Machine Learning (pp. 7400-7411). PMLR.

The first experiment



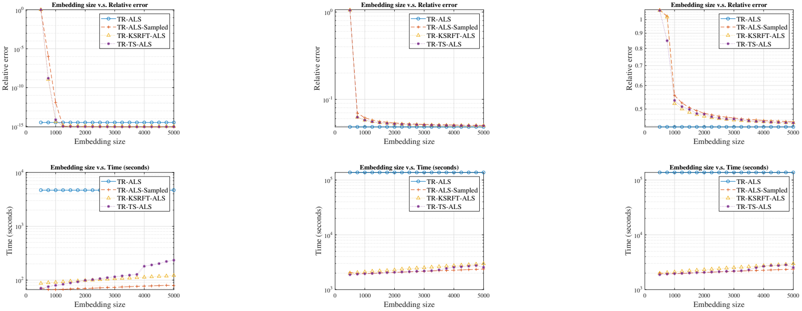
(a) $noise = 0$ (b) $noise = 0.01$ (c) $noise = 0.1$

Figure 7: Embedding sizes v.s. relative errors and running time (seconds) of the first synthetic experiment with true and target ranks $R_{true} = R = 10$ and different noises.

The second experiment

- generate_sparse_low_rank_tensor($sz, ranks, density, noise$)
  - Create 3 cores of size $R_{true} \times I \times R_{true}$ with non-zero entries drawn from a standard normal distribution;
  - $R_{true} = 10$;
  - $sz = [I,I,I]$= [500,500,500];
  - $ranks = R$;
  - $density =$ 0.05;

The second experiment



(a) $noise = 0$ (b) $noise = 0.01$ (c) $noise = 0.1$

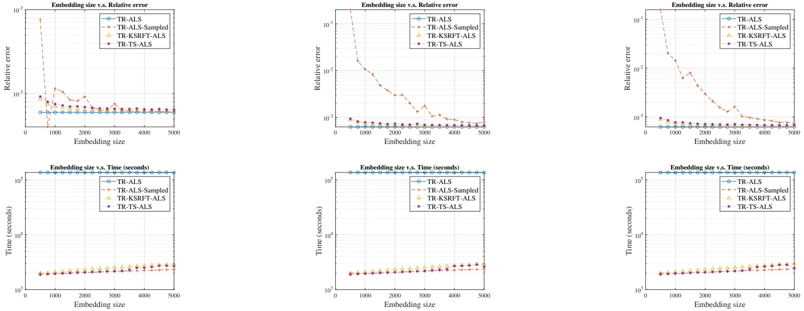
Figure 8: Embedding sizes v.s. relative errors and running time (seconds) of the second synthetic experiment with true and target ranks $R_{true} = R = 10$ and different noises.

The third experiment

- generate_sptr_tensor($sz, ranks, noise, spread, magnitude$)
  - Create 3 cores of size $R_{true} \times I \times R_{true}$ with entries drawn independently from a standard normal distribution;
  - $spread$: How many non-zeros elements are added to each of these first three columns;
  - $magnitude$: Those non-zero elements are chosen;
  - $R_{true} = 10$;
  - sz = [I,I,I]= [500,500,500];
  - $ranks = R$;

[LK20] Larsen, B. W., & Kolda T. G. (2020). Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition. arXiv:2006.16438.

The third experiment



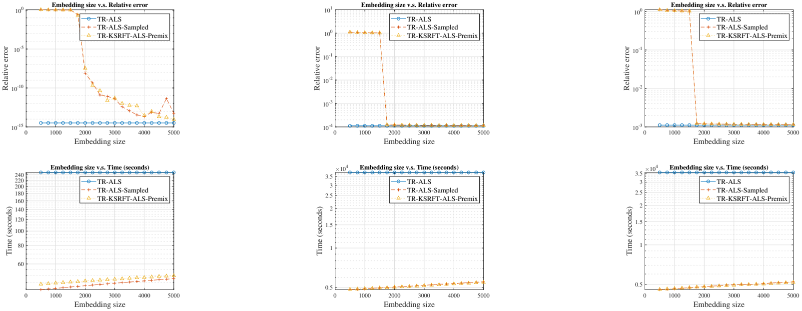
(a) $noise = 0$ (b) $noise = 0.01$ (c) $noise = 0.1$

Figure 9: Embedding sizes v.s. relative errors and running time (seconds) of the third synthetic experiment with true and target ranks $R_{true} = R = 10$ and different noises.

The forth experiment

- generate_complex_low_rank_tensor($sz, ranks, noise, large\_elem$)
  - Create 3 cores of size $R_{true} \times I \times R_{true}$ with entries drawn independently from a standard normal distribution and add imaginary part;
  - Set $large\_elem$ to increase the coherence;
  - $R_{true} = 10$;
  - sz = [I,I,I]= [500,500,500];
  - $ranks = R$;
  - $large\_elem$ = 20;

The forth experiment



(a) $noise = 0$ (b) $noise = 0.01$ (c) $noise = 0.1$

Figure 10: Embedding sizes v.s. relative errors and running time (seconds) of the fourth synthetic experiment with true and target ranks $R_{true} = R = 10$ and different noises.

Real data

Dataset	Size	Type
Indian Pines	$145 \times 145 \times 220$	Hyperspectral
SalinasA.	$83 \times 86 \times 224$	Hyperspectral
C1-vertebrae	$512 \times 512 \times 47$	CT Images
Uber.Hour[6]	$183 \times 1140 \times 1717$	Sparse
Uber.Date	$24 \times 1140 \times 1717$	Sparse

Table 1: Size and type of real datasets.

[6] Larsen, B. W., & Kolda T. G. (2020). Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition. arXiv:2006.16438.

Real data

Method	Indian Pines ($R = 20$)			SalinasA. ($R = 15$)			C1-vertebrae ($R = 25$)		
	Error	Time	num	Error	Time	num	Error	Time	num
TR-ALS	0.0263	32.9536		0.0066	4.0225		0.0804	409.7951	
TR-ALS-Sampled	0.0289	13.7424	120	0.0069	2.4166	54	0.0882	128.3391	228
TR-SRFT-ALS	0.0289	12.3571	53	0.0073	1.8510	23	0.0883	101.7646	88
TR-SRFT-ALS (No pre-time)		11.9446			1.7093			101.4037	
TR-TS-ALS	0.0289	12.0229	73	0.0073	2.2868	30	0.0883	156.5089	217

Method	Uber.Hour ($R = 15$)			Uber.Date ($R = 18$)		
	Error	Time	num	Error	Time	num
TR-ALS	0.7530	869.1631		0.3864	1452.1900	
TR-ALS-Sampled	0.8246	64.7240	230	0.4226	159.1936	320
TR-SRFT-ALS	0.8272	39.0307	40	0.4246	51.3584	46
TR-SRFT-ALS (No pre-time)		21.9817			48.9433	
TR-TS-ALS	0.8274	45.3829	47	0.4239	113.8542	147

Outline

1 Introduction
- Tensor decompositions
- Algorithms for TR decomposition
- "Sketching"

2 TR-SRFT-ALS
- Motivations
- New findings
- Algorithm and theoretical analysis

3 TR-TS-ALS
- New findings
- Algorithm and theoretical analysis

4 Numerical Results
- Synthetic data
- Real data

5 Conclusions

Conclusions

1. We propose two randomized algorithms for TR decomposition, TR-SRFT-ALS and TR-TS-ALS.
2. We propose two new tensor products and find their interesting properties.
3. Numerical experiments are provided to test the proposed methods.

Thanks!

References I

[ACP+20] S. Ahmadi-Asl, A. Cichocki, A. H. Phan, M. G. Asante-Mensah, M. M. Ghazani, T. Tanaka, and I. Oseledets, *Randomized algorithms for fast computation of low rank tensor ring model*, Machine Learning: Science and Technology **2** (2020), no. 1, 011001 (en).

[BBK18] C. Battaglino, G. Ballard, and T. G. Kolda, *A Practical Randomized CP Tensor Decomposition*, SIAM Journal on Matrix Analysis and Applications **39** (2018), no. 2, 876–901.

[CW17] K. L. Clarkson and D. P. Woodruff, *Low-Rank Approximation and Regression in Input Sparsity Time*, Journal of the ACM **63** (2017), no. 6, 54:1–54:45.

[JKW20] R. Jin, T. G. Kolda, and R. Ward, *Faster Johnson-Lindenstrauss Transforms via Kronecker Products*, Information and Inference: A Journal of the IMA (2020), iaaa028, available at 1909.04801 (en).

[LK20] B. W. Larsen and T. G. Kolda, *Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition*, arXiv:2006.16438 (2020), available at 2006.16438.

References II

[MB21] O. A. Malik and S. Becker, *A sampling-based method for tensor ring decomposition*, Proceedings of the 38th international conference on machine learning, 2021, pp. 7400–7411.

[YLCZ19] L. Yuan, C. Li, J. Cao, and Q. Zhao, *Randomized Tensor Ring Decomposition and Its Application to Large-scale Data Reconstruction*, ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 2127–2131.

[ZZX+16] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, *Tensor Ring Decomposition*, arXiv:1606.05535 [cs] (2016), available at 1606.05535.