Parallel Randomized Algorithms for Tucker Decompositions

Rachel Minster, Zitong Li, Qiming Fang, Grey Ballard

Wake Forest University

2/26/22

Acknowledgements to NSF CCF 1942892 for funding
Motivation: Multidimensional data

Multidimensional data appears in many applications:

- Numerical simulations for PDE’s
- Facial recognition
- Hyperspectral imaging

Christophe, Duhamel, IEEE Transactions on Image Processing, 2009
Motivation: Multidimensional data

Multidimensional data appears in many applications:

- Numerical simulations for PDE’s
- Facial recognition
- Hyperspectral imaging

and is often large and difficult to store or compute with

---

Christophe, Duhamel, IEEE Transactions on Image Processing, 2009
Motivation: Multidimensional data

Multidimensional data appears in many applications:

- Numerical simulations for PDE’s
- Facial recognition
- Hyperspectral imaging

and is often large and difficult to store or compute with

Goal: efficiently obtain compressed representation of data

Method: use parallel, randomized algorithms for Tucker decompositions

- can obtain large compression ratios with high accuracy

Christophe, Duhamel, IEEE Transactions on Image Processing, 2009
Contributions

New parallel, randomized algorithms for computing the Tucker decomposition
- Uses a Kronecker product of random matrices to exploit structure
- Significantly reduces computational cost compared to deterministic and randomized counterparts

New parallel method of computing a multi tensor-times-matrix (multi-TTM) product, an “all-at-once” approach

Theoretical error bound for the algorithms
- Tail bound
Tensor-times-matrix (TTM) and Multi-TTM

Key tensor operations:

- **Tensor-times-matrix (TTM):** $\mathcal{X} \times_j U$
  - Tensor multiplied by a matrix in a single mode $j$
  - Computed as matrix multiplication: matrix times unfolded tensor

\[
\mathcal{X} \times_1 U \quad \rightarrow \quad U \quad \cdots \quad X_{(1)} \quad = \quad \cdots
\]
Key tensor operations:

- **Tensor-times-matrix (TTM):** $\mathcal{X} \times_j U$
  - Tensor multiplied by a matrix in a single mode $j$
  - Computed as matrix multiplication: matrix times unfolded tensor

- **Multi-TTM:** $\mathcal{X} \times_1 U_1 \times_2 U_2 \cdots \times_d U_d$ for $d$-mode tensor
  - Can be unfolded in $j$-th mode as
    $U_j X_{(j)} (U_d \otimes U_{d-1} \otimes \cdots \otimes U_{j+1} \otimes U_{j-1} \otimes \cdots \otimes U_1)^\top$
  - with $\otimes$ the Kronecker product
Tucker Format

Approximates tensor $\mathcal{X}$ as

$$\mathcal{X} \approx G \times_1 A_1 \times \cdots \times_d A_d$$

with $G \in \mathbb{R}^{r_1 \times \cdots \times r_d}$, $A_j \in \mathbb{R}^{n_j \times r_j}$

Popular algorithms: Higher Order SVD (HOSVD)\(^1\) and Sequentially Truncated Higher Order SVD (STHOSVD)\(^2\)

---

\(^1\)De Lathauwer, De Moor, Vandewalle, SIAM Journal on Matrix Analysis and Applications, 2000
\(^2\)Vannieuwenhoven, Vandebril, Meerbergen, SIAM Journal on Scientific Computing, 2012
Tucker Format

Approximates tensor $\mathcal{X}$ as

$$\mathcal{X} \approx G \times_1 A_1 \times \cdots \times_d A_d$$

with $G \in \mathbb{R}^{r_1 \times \cdots \times r_d}$, $A_j \in \mathbb{R}^{n_j \times r_j}$

Popular algorithms: Higher Order SVD (HOSVD)$^1$ and Sequentially Truncated Higher Order SVD (STHOSVD)$^2$

General approach:

1. Unfold tensor along mode $j$
2. Compute rank-$r_j$ SVD of mode unfolding
3. Factor matrix $A_j$ formed from left singular vectors
4. Core (or partial core) formed via TTM’s

---

$^1$De Lathauwer, De Moor, Vandewalle, SIAM Journal on Matrix Analysis and Applications, 2000

Tucker Format

General approach:
1. Unfold tensor along mode $j$
2. Compute rank-$r_j$ SVD of mode unfolding
3. Factor matrix $A_j$ formed from left singular vectors
4. Core (or partial core) formed via TTM’s

Our approach:
- Use a randomized algorithm\(^3\) to speed up SVD step
  - Use a Kronecker product of random matrices instead of single random matrix to exploit structure
- Implement in parallel
  - Use a new, faster parallel version of a key operation (multi-TTM) to significantly lower runtime

\(^3\)Ahmadi-Asl, Abukhovich, Asante-Menash, Chichocki, Phan, Tanaka, Oseledets, IEEE Access, 2021
Randomized Range Finder

For a matrix $X$, finds a matrix $Q$ that estimates the range of $X$, or $X \approx QQ^\top X$

Inputs: matrix $X \in \mathbb{R}^{m \times n}$

- target rank $r \leq \text{rank } X$
- oversampling parameter $p$

Main Steps:

1. Draw $\Omega \in \mathbb{R}^{n \times (r+p)}$, a random matrix
2. Form product $Y = X\Omega$
3. Compute thin QR $Y = QR$

Halko, Martinsson, Tropp, SIAM Review, 2011
Randomized Range Finder

For a matrix $X$, finds a matrix $Q$ that estimates the range of $X$, or $X \approx QQ^TX$

Inputs: matrix $X \in \mathbb{R}^{m \times n}$
target rank $r \leq \text{rank } X$
oversampling parameter $p$

Main Steps:

1. Draw $\Omega \in \mathbb{R}^{n \times (r+p)}$, a random matrix
2. Form product $Y = X\Omega$
3. Compute thin QR $Y = QR$

Idea: Use Kronecker product of $k$ random matrices $\Phi_j$ as $\Omega = \Phi_1 \otimes \Phi_2 \otimes \cdots \otimes \Phi_k$ so that

$$Y = X\Omega = X(\Phi_1 \otimes \Phi_2 \otimes \cdots \otimes \Phi_k)$$

takes the form of an unfolded multi-TTM
Randomized HOSVD with Kronecker Product

Inputs: $\mathcal{X} \in \mathbb{R}^{n \times \cdots \times n}$, target rank $(r, \ldots, r)$, oversampling parameter $p$

Main steps:

For modes $j = 1 : d$,

1. **Randomized range finder of unfolding $\mathcal{X}(j)$**
   
   a. Compute $Y(j) = \mathcal{X}(j)\Omega$ via Multi-TTM in all modes but $j$:
   
   $$Y = \mathcal{X} \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}$$

   b. Thin QR of $Y(j) = A_j R$ with $A_j \in \mathbb{R}^{n \times (r+p)}$

End for

3. Form core via multi-TTM: $\mathcal{G} = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top$

4. Truncate down to target rank

Deterministic HOSVD on $\mathcal{G}$, combine factor matrices with $A_j$'s
Randomized HOSVD with Kronecker Product

Inputs: $\mathcal{X} \in \mathbb{R}^{n \times \cdots \times n}$, target rank $(r, \ldots, r)$, oversampling parameter $p$

Main steps:
For modes $j = 1 : d$,

1. **Randomized range finder of unfolding $X_{(j)}$**
   a. Compute $Y_{(j)} = X_{(j)} \Omega$ via Multi-TTM in all modes but $j$:
      $$Y = \mathcal{X} \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}$$
   b. Thin QR of $Y_{(j)} = A_j R$ with $A_j \in \mathbb{R}^{n \times (r+p)}$

End for

3. **Form core via multi-TTM:** $G = \mathcal{X} \times_1 A_1^T \times \cdots \times_d A_d^T$

4. **Truncate down to target rank**
   - Deterministic HOSVD on $G$, combine factor matrices with $A_j$'s
Randomized HOSVD with Kronecker Product

Inputs: \( X \in \mathbb{R}^{n \times \cdots \times n} \), target rank \((r, \ldots, r)\), oversampling parameter \( p \)

Main steps:

For modes \( j = 1 : d \),

1. Randomized range finder of unfolding \( X(j) \)
   - Compute \( Y(j) = X(j)\Omega \) via Multi-TTM in all modes but \( j \):
     \[
     Y = X \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}
     \]
   - Thin QR of \( Y(j) = A_j R \) with \( A_j \in \mathbb{R}^{n \times (r+p)} \)

End for

3. Form core via multi-TTM: \( G = X \times_1 A_1^\top \times \cdots \times_d A_d^\top \)

4. **Truncate down to target rank**
   - Deterministic HOSVD on \( G \), combine factor matrices with \( A_j \)'s
Comparison: algorithm types

Standard approach: one random matrix $\Omega \in \mathbb{R}^{n^{d-1} \times (r+p)}$
- Computing $Y = X(j) \Omega \rightarrow$ one large matrix multiply

Our approach: Kronecker product of random matrices $\Omega = \Phi_1 \otimes \cdots \otimes \Phi_d$
with $\Phi_j \in \mathbb{R}^{n \times s}$, $s^{d-1} = r + p$
- Computing $Y = X(j) \Omega \rightarrow$ one multi-TTM with skinny matrices
Comparison: algorithm types

Standard approach: one random matrix $\Omega \in \mathbb{R}^{n^{d-1} \times (r+p)}$
- Computing $Y = X_{(j)}\Omega$ $\rightarrow$ one large matrix multiply

Our approach: Kronecker product of random matrices $\Omega = \Phi_1 \otimes \cdots \otimes \Phi_d$
with $\Phi_j \in \mathbb{R}^{n \times s}$, $s^{d-1} = r + p$
- Computing $Y = X_{(j)}\Omega$ $\rightarrow$ one multi-TTM with skinny matrices

Two options for our approach:

1. Use an independent products of $\Phi_j$’s per mode

2. Reuse same Kronecker factors $\Phi_j$ in $\Omega_j$ (i.e., $\Omega_1 = \Phi_2 \otimes \cdots \otimes \Phi_d$)
Comparison: algorithm types

Standard approach: one random matrix \( \Omega \in \mathbb{R}^{n^{d-1} \times (r+p)} \)

- Computing \( Y = X_{(j)} \Omega \) → one large matrix multiply

Our approach: Kronecker product of random matrices \( \Omega = \Phi_1 \otimes \cdots \otimes \Phi_d \)
with \( \Phi_j \in \mathbb{R}^{n \times s} \), \( s^{d-1} = r + p \)

- Computing \( Y = X_{(j)} \Omega \) → one multi-TTM with skinny matrices

Two options for our approach:

1. Use an independent products of \( \Phi_j \)'s per mode
   - Generating and storing more random matrices
   - “rKron”

2. Reuse same Kronecker factors \( \Phi_j \) in \( \Omega_j \) (i.e., \( \Omega_1 = \Phi_2 \otimes \cdots \otimes \Phi_d \))
   - Allows for reuse of computations
   - Makes analysis more complicated
   - “rKron-reuse”
Theoretical Bound

Parameters:

- $d$-way tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times \cdots \times n}$
- target rank $(r, r, \ldots, r)$, oversampling parameter $p$
- $\alpha, \beta > 1$ satisfying $n > r + p \geq \frac{\alpha^2 \beta}{(\alpha - 1)^2} (r^2 + r)$
- SRHT-like random matrices: $\Phi = DH$
  - $D$ diagonal Rademacher
  - $H$ randomly sampled columns from Hadamard matrix

Error bound

Except with probability at most $\frac{d}{\beta}$,

$$
\| \mathcal{X} - \hat{\mathcal{X}} \|_F^2 \leq \left(1 + \frac{\alpha n^{2d-2}}{(r + p)^{d-1}} \right) \| \mathcal{X} - \hat{\mathcal{X}}_{\text{HOSVD}} \|_F^2
$$
Theoretical Bound

Error bound

Except with probability at most \( \frac{d}{\beta} \),

\[
\|X - \hat{X}\|_F^2 \leq \left(1 + \frac{\alpha n^{2d-2}}{(r+p)^{d-1}}\right) \|X - \hat{X}_{\text{HOSVD}}\|_F^2
\]

Notes:

- Pessimistic compared to accuracy shown in numerical results
- Uses SRHT-like random matrices that can be represented as a Kronecker product themselves
- Allows for independent product of random matrices per mode, or reuse of same product of random matrices
Numerical Results: Accuracy

Parameters:
- $500 \times 500 \times 500$ tensor with moderately decaying singular values
- target rank $(10, 10, 10)$, oversampling parameter $5$, $s = 4$
- rand-HOSVD: Gaussian random matrix
- rKron, rKron-reuse: SRHT random matrices

Relative Error over 100 trials

<table>
<thead>
<tr>
<th>Method</th>
<th>Relative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>rand-HOSVD</td>
<td>1.05 $\times 10^{-4}$</td>
</tr>
<tr>
<td>rKron</td>
<td>1.1 $\times 10^{-4}$</td>
</tr>
<tr>
<td>rKron-reuse</td>
<td>1.15 $\times 10^{-4}$</td>
</tr>
</tbody>
</table>

[Graph showing relative error for rand-HOSVD, rKron, and rKron-reuse]
Numerical Results: Accuracy

- $500 \times 500 \times 500$ random tensor with true rank $(50, 50, 50)$ and $10^{-4}$ noise
- oversampling parameter $5$, $s \leq 11$
- rand-HOSVD: Gaussian random matrix
- rKron, rKron-reuse: SRHT random matrices

Relative Error with increasing rank
Randomized HOSVD with Kronecker Product

Inputs: $\mathcal{X} \in \mathbb{R}^{n \times \cdots \times n}$, target rank $(r, \ldots, r)$, oversampling parameter $p$

Main steps:

For modes $j = 1 : d$,

1. Randomized range finder of unfolding $X_{(j)}$
   
   a. Multi-TTM in all modes but $j$:
      $$Y = \mathcal{X} \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}$$

   b. Thin QR so that $X_{(j)} \approx A_j A_j^T X_{(j)}$

   End for

3. Form core via multi-TTM: $\mathcal{G} = \mathcal{X} \times_1 A_1^T \times \cdots \times_d A_d^T$

4. Truncate down to target rank
   
   a. Deterministic HOSVD on $\mathcal{G}$, combine factor matrices with $A_j$'s
All-at-once multi-TTM

Goal: compute $\mathbf{Y} = \mathbf{X} \times_1 U_1 \times_2 U_2 \times \cdots \times_k U_k$ for $k \leq d$ matrices

Two approaches based on communication: in sequence and all-at-once
All-at-once multi-TTM

Goal: compute $\mathcal{Y} = \mathcal{X} \times_1 U_1 \times_2 U_2 \times \cdots \times_k U_k$ for $k \leq d$ matrices

Two approaches based on communication: in sequence and all-at-once

Example: 2 modes $\mathcal{X} \times_1 U^\top \times_2 V^\top = U^\top XV$

In sequence$^4$:
- Compute local $U^\top X$, communicate result
- Compute local multiply with $V$, communicate result

$^4$Ballard, Klinvex, Kolda, ACM TOMS, 2020
All-at-once multi-TTM

Goal: compute $\mathcal{Y} = \mathcal{X} \times_1 U_1 \times_2 U_2 \times \cdots \times_k U_k$ for $k \leq d$ matrices

Two approaches based on communication: in sequence and all-at-once

Example: 2 modes $\mathcal{X} \times_1 U^\top \times_2 V^\top = U^\top X V$

In sequence$^4$:
- Compute local $U^\top X$, communicate result
- Compute local multiply with $V$, communicate result

All-at-once:
- Compute local $U^\top X V$
- Communicates final result

$^4$Ballard, Klinvex, Kolda, ACM TOMS, 2020
Comparison: multi-TTM

In-sequence:
- fewer flops, more communication

All-at-once:
- slightly more flops, generally less communication
## Comparison: multi-TTM

### In-sequence:
- fewer flops, more communication
- better choice when matrices are fat

### All-at-once:
- slightly more flops, generally less communication
- better choice when matrices are skinny
In-sequence:
- fewer flops, more communication
- better choice when matrices are fat
- In randomized HOSVD algorithm, use for core multi-TTM
  \[ G = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top \]
- factor matrices \( A_j \) have more \((r + p)\) columns

All-at-once:
- slightly more flops, generally less communication
- better choice when matrices are skinny
- In randomized HOSVD algorithm, use to compute sketch
  \[ Y = \mathcal{X} \times_2 \Phi_2^\top \times \cdots \times_d \Phi_d^\top \]
- random matrices are very skinny (\( s \) columns)
Numerical Results: Parallel Runtime

Parameters:

- 4-way tensor, 250 in each mode
- 16 cores on single multicore server
- Gaussian random matrices

Runtime of multi-TTM methods with increasing number of columns $s$: 

![Chart showing runtime of multi-TTM methods with increasing number of columns $s$.]
Numerical Results: Parallel Runtime

Parameters:
- 4-way tensor, 256 in each mode
- Target rank $(32, 32, 32, 32)$, $s = (3, 3, 4, 4)$
- Gaussian random matrices, rKron-reuse
- On Andes cluster (OLCF)

Runtime of full algorithms with increasing number of cores:
Conclusions

Contributions: new parallel, randomized algorithms for Tucker decompositions

- Use a Kronecker product of random matrices to exploit structure and employ multi-TTM instead of large matrix multiply
- Different versions: re-using or constructing independent Kronecker products
- New method for computing a multi-TTM in parallel
  - An all-at-once approach that can communicate less than standard approach
  - Works well with Kronecker product of random matrices in our Tucker algorithms