Randomized Algorithms for Tensor Decompositions in the Tucker Format

Rachel Minster

Wake Forest University

5/6/22

Acknowledgements to NSF CCF 1942892, DMS 1821149 and DMS 1745654 for funding

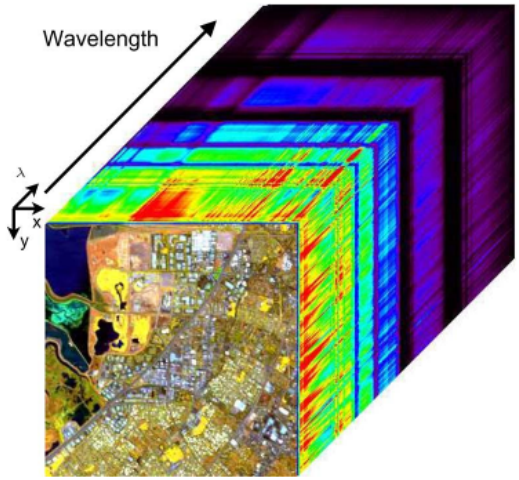
Motivation: Data is often...

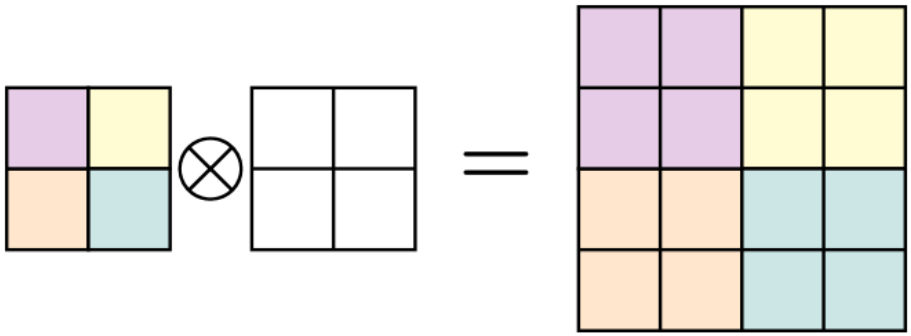
Large-scale

Difficult to store and compute with

Exploit underlying structure - low-rank, multidimensional, sparse, Kronecker,...

Use randomization to increase efficiency





Outline

1. Randomization in NLA
   - Background on randomized matrix techniques
2. Tensors
   - Intro to tensors, tensor operations
   - Tucker decompositions
3. Randomized algorithms for Tucker decompositions
   - Algorithms and analysis
4. Improvements on randomized algorithms for Tucker decompositions
   - Kronecker-structured random matrices
   - Parallel implementation
5. Structure-preserving Tucker decomposition and algorithms
   - Variation on the Tucker decomposition with strong benefits

Outline

1. **Randomization in NLA**
   - Background on randomized matrix techniques
2. Tensors
   - Intro to tensors, tensor operations
   - Tucker decompositions
3. Randomized algorithms for Tucker decompositions
   - Algorithms and analysis
4. Improvements on randomized algorithms for Tucker decompositions
   - Kronecker-structured random matrices
   - Parallel implementation
5. Structure-preserving Tucker decomposition and algorithms
   - Variation on the Tucker decomposition with strong benefits

Randomization in Numerical Linear Algebra

Main problem to solve: Low-rank matrix/tensor approximations

- Principal Component Analysis (PCA), Subset-selection, Clustering

Example applications:

- Machine learning (facial recognition), system identification[1], approximation of kernel interactions[2]

[1] Minster, Saibaba, Kar, Chakrabortty, SIMAX, 2021, [2] Saibaba, Minster, Kilmer, arXiv 2107.13107, 2021

Randomization in Numerical Linear Algebra

Main problem to solve: Low-rank matrix/tensor approximations

- Principal Component Analysis (PCA), Subset-selection, Clustering

Example applications:

- Machine learning (facial recognition), system identification[1], approximation of kernel interactions[2]

Main challenges to address:

- Massive data (may need to use distributed systems)
- Computationally expensive problems

[1] Minster, Saibaba, Kar, Chakrabortty, SIMAX, 2021, [2] Saibaba, Minster, Kilmer, arXiv 2107.13107, 2021

Randomization in Numerical Linear Algebra

Main problem to solve: Low-rank matrix/tensor approximations

- Principal Component Analysis (PCA), Subset-selection, Clustering

Example applications:

- Machine learning (facial recognition), system identification[1], approximation of kernel interactions[2]

Main challenges to address:

- Massive data (may need to use distributed systems)
- Computationally expensive problems

Benefits of randomization:

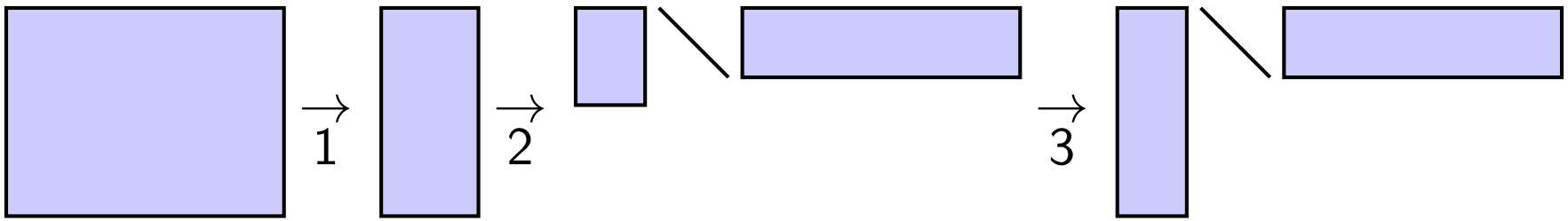
- Provably accurate (average case and long-term behavior)
- More computationally feasible

[1] Minster, Saibaba, Kar, Chakrabortty, SIMAX, 2021, [2] Saibaba, Minster, Kilmer, arXiv 2107.13107, 2021

Randomization for low-rank approximations

Main Idea:

1. Dimension reduction
2. Deterministic computation
3. Recover low-rank approximation of original data

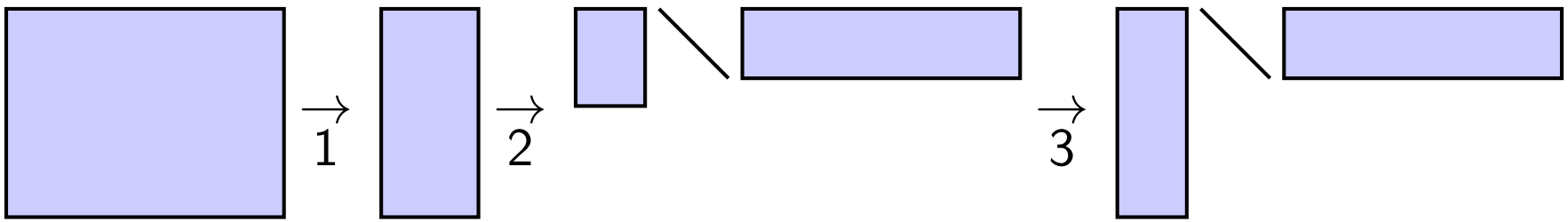


Ahmadi-Asl, Abukhovich, Asante-Mensah, Cichocki, Phan, Tanaka and Oseledets, IEEE Access, 2021

Randomization for low-rank approximations

Main Idea:

1. Dimension reduction
   - **Random Projection**
   - Sampling
2. Deterministic computation
3. Recover low-rank approximation of original data

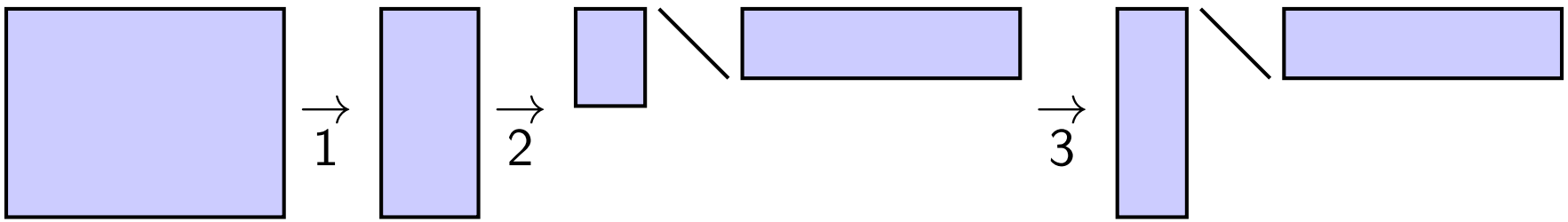


Ahmadi-Asl, Abukhovich, Asante-Mensah, Cichocki, Phan, Tanaka and Oseledets, IEEE Access, 2021

Randomization for low-rank approximations

Main Idea:

1. Dimension reduction
   - **Random Projection**
   - Sampling
2. Deterministic computation
   - QR or SVD
3. Recover low-rank approximation of original data

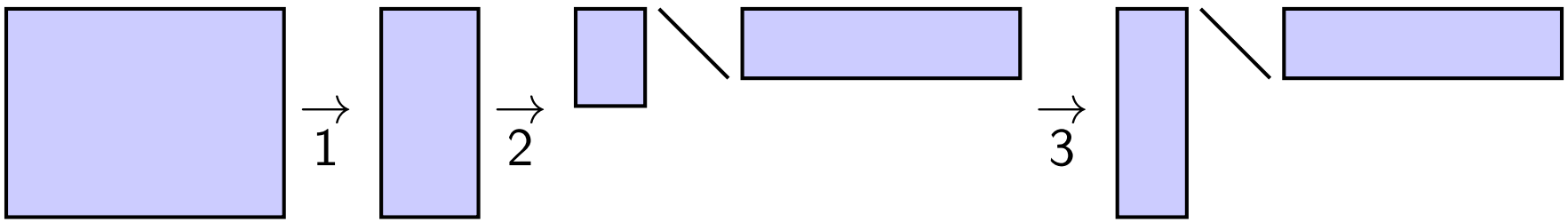


Ahmadi-Asl, Abukhovich, Asante-Mensah, Cichocki, Phan, Tanaka and Oseledets, IEEE Access, 2021

Randomization for low-rank approximations

Main Idea:

1. Dimension reduction
   - **Random Projection**
   - Sampling
2. Deterministic computation
   - QR or SVD
3. Recover low-rank approximation of original data
   - not always needed



Ahmadi-Asl, Abukhovich, Asante-Mensah, Cichocki, Phan, Tanaka and Oseledets, IEEE Access, 2021

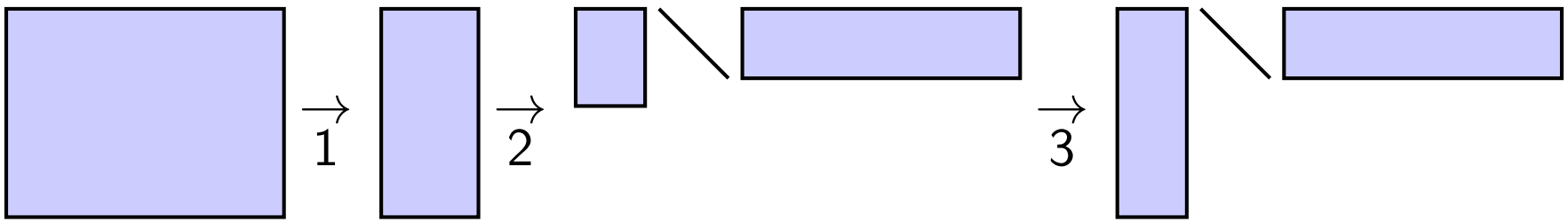
Randomization for low-rank approximations

Main Idea:

1. Dimension reduction
   - **Random Projection**
   - Sampling
2. Deterministic computation
   - QR or SVD
3. Recover low-rank approximation of original data

Methods:

- Randomized Range Finder: $X \approx QQ^\top X$
- Randomized SVD: $X \approx U\Sigma V^\top$



Ahmadi-Asl, Abukhovich, Asante-Mensah, Cichocki, Phan, Tanaka and Oseledets, IEEE Access, 2021

Randomized Range Finder

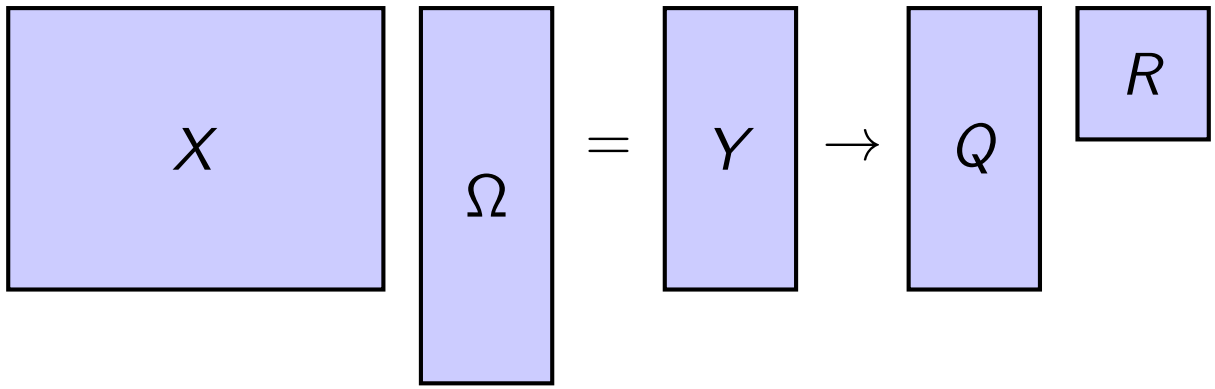
Inputs: $X \in \mathbb{R}^{m \times n}$, target rank $r \leq \text{rank}(X)$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. **Draw $\mathbf{n} \times (\mathbf{r} + \mathbf{p})$ random matrix $\mathbf{\Omega}$**
2. Form sketch $Y = X\Omega$
3. Compute thin QR $Y = QR$

Details:

- e.g. Gaussian, Rademacher, etc.
- Y made up of random linear combinations of the columns of X
- Y is $m \times (r + p)$, tall and skinny



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Range Finder

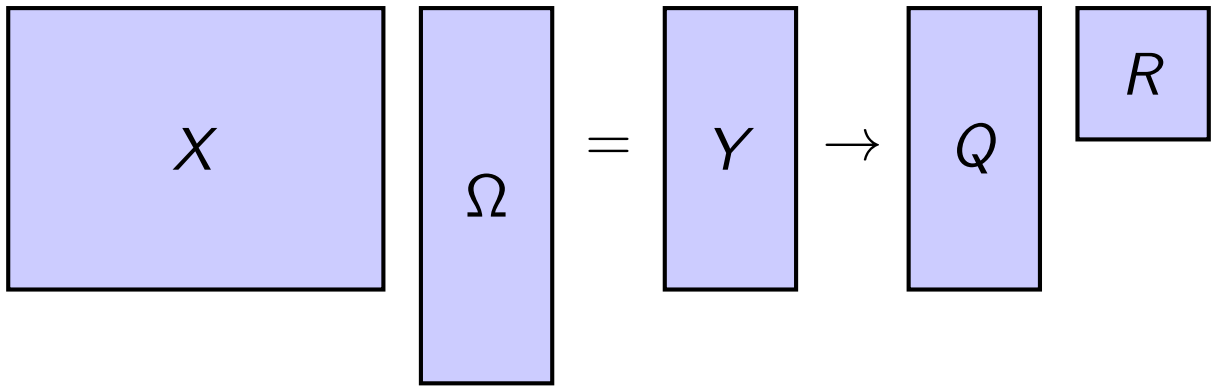
Inputs: $X \in \mathbb{R}^{m \times n}$, target rank $r \leq \text{rank}(X)$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. Draw $n \times (r + p)$ random matrix Ω
2. **Form sketch $\mathbf{Y} = \mathbf{X\Omega}$**
3. Compute thin QR $Y = QR$

Details:

- e.g. Gaussian, Rademacher, etc.
- Y made up of random linear combinations of the columns of X
- Y is $m \times (r + p)$, tall and skinny



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Range Finder

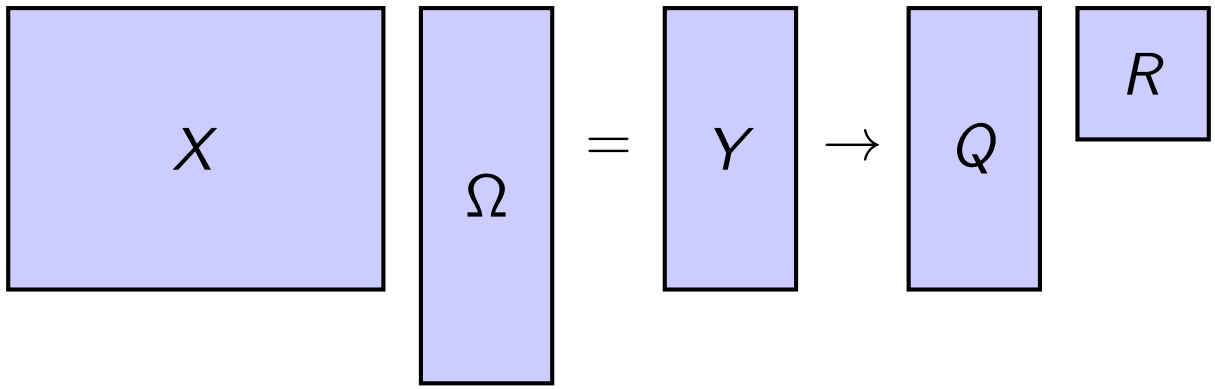
Inputs: $X \in \mathbb{R}^{m \times n}$, target rank $r \leq \text{rank}(X)$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. Draw $n \times (r + p)$ random matrix Ω
2. Form sketch $Y = X\Omega$
3. **Compute thin QR $\mathbf{Y = QR}$**

Details:

- e.g. Gaussian, Rademacher, etc.
- Y made up of random linear combinations of the columns of X
- Y is $m \times (r + p)$, tall and skinny



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Range Finder

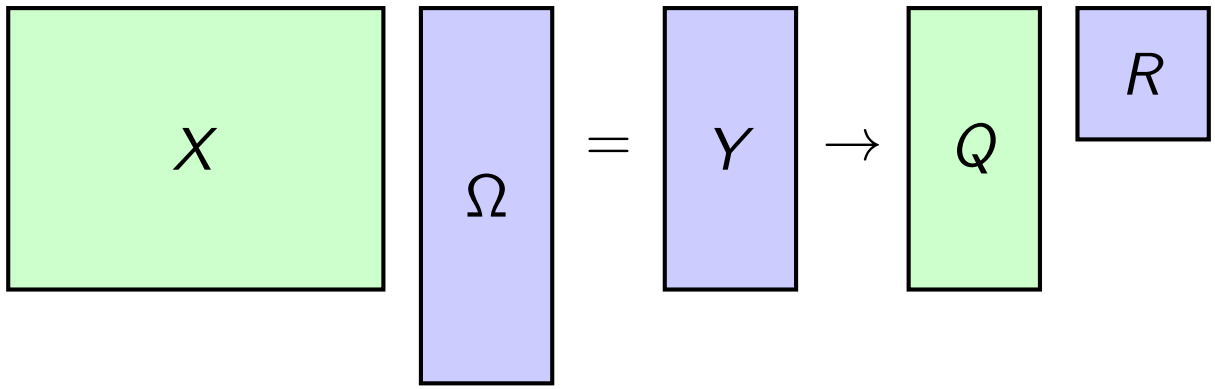
Inputs: $X \in \mathbb{R}^{m \times n}$, target rank $r \leq \text{rank}(X)$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. Draw $n \times (r + p)$ random matrix Ω
2. Form sketch
3. Compute thin QR

Details:

- e.g. Gaussian, Rademacher, etc.
- Y made up of random linear combinations of the columns of X
- Y is $m \times (r + p)$, tall and skinny



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized SVD

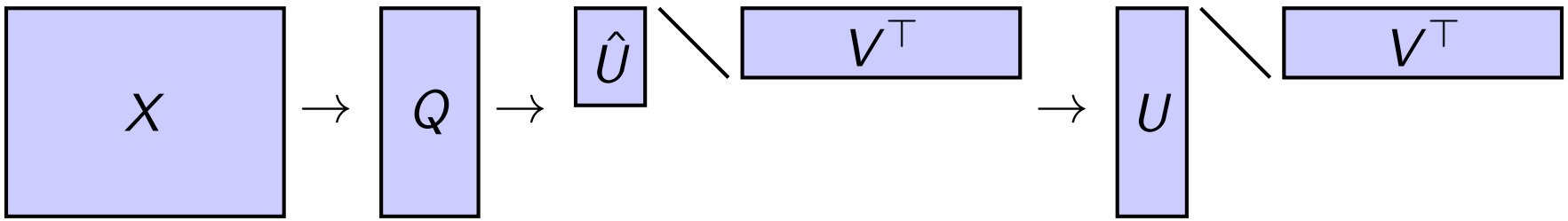
Inputs: $X \in \mathbb{R}^{m \times n}$, $r \leq \operatorname{rank} X$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. **Randomized range finder**
2. Form $B = Q^\top X$
3. Compute truncated (rank-r) SVD $B = \hat{U} \Sigma V^\top$
4. Form approximate left singular vectors $U = Q\hat{U}$

Details:

- Draw random matrix $\Omega \in \mathbb{R}^{n \times (r+p)}$
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
- $\hat{U}$ is $(r + p) \times r$ and needs to be $m \times r$



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized SVD

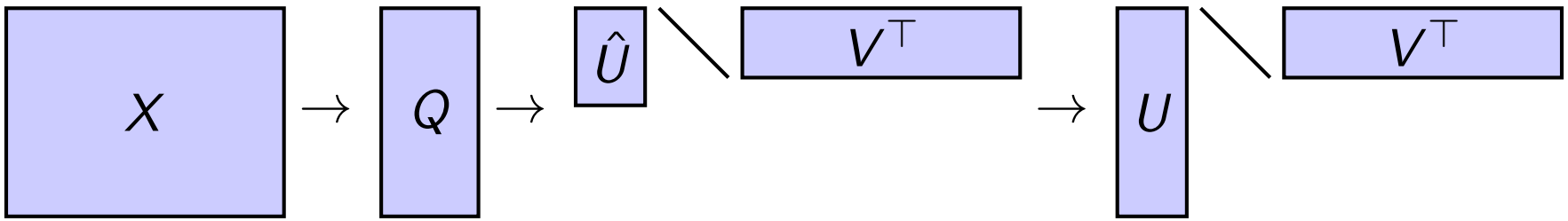
Inputs: $X \in \mathbb{R}^{m \times n}$, $r \leq \operatorname{rank} X$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. Randomized range finder
2. **Form $\mathbf{B} = \mathbf{Q}^\top \mathbf{X}$**
3. Compute truncated (rank-r) SVD $B = \hat{U} \Sigma V^\top$
4. Form approximate left singular vectors $U = Q\hat{U}$

Details:

- Draw random matrix $\Omega \in \mathbb{R}^{n \times (r+p)}$
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
- $\hat{U}$ is $(r + p) \times r$ and needs to be $m \times r$



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized SVD

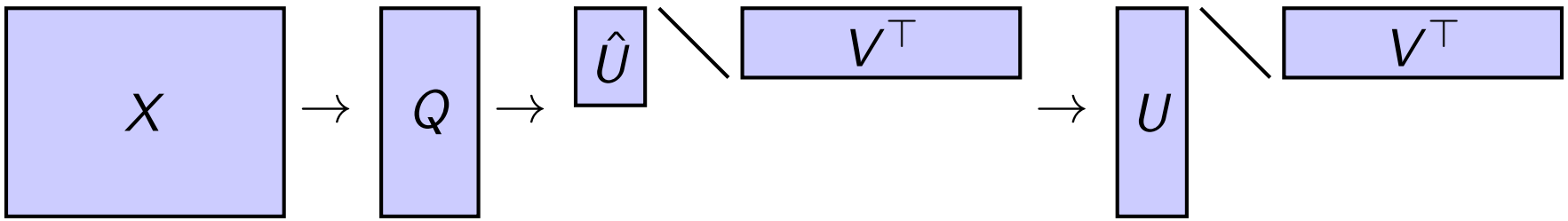
Inputs: $X \in \mathbb{R}^{m \times n}$, $r \leq \operatorname{rank} X$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. Randomized range finder
2. Form $B = Q^\top X$
3. **Compute truncated (rank-r) SVD $\mathbf{B = \hat{U} \Sigma V^\top}$**
4. Form approximate left singular vectors $U = Q\hat{U}$

Details:

- Draw random matrix $\Omega \in \mathbb{R}^{n \times (r+p)}$
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
- $\hat{U}$ is $(r + p) \times r$ and needs to be $m \times r$



Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized SVD

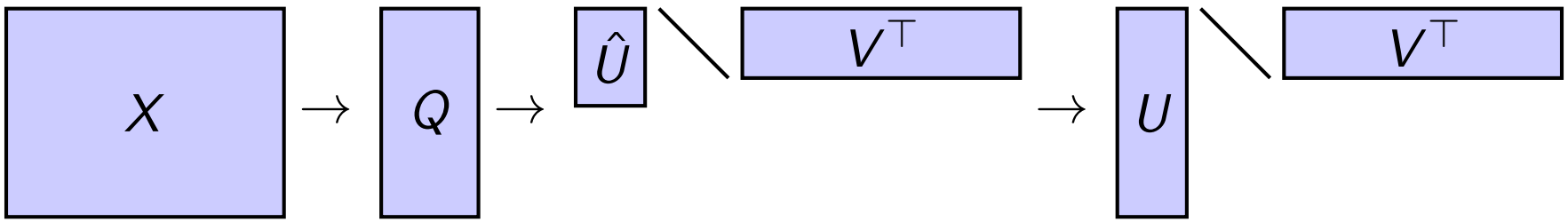
Inputs: $X \in \mathbb{R}^{m \times n}$, $r \leq \operatorname{rank} X$, oversampling parameter p with $r + p < \min\{m, n\}$

Algorithm steps:

1. Randomized range finder
2. Form $B = Q^\top X$
3. Compute truncated (rank-r) SVD $B = \hat{U} \Sigma V^\top$
4. **Form approximate left singular vectors $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$**

Details:

- Draw random matrix $\Omega \in \mathbb{R}^{n \times (r+p)}$
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
- $\hat{U}$ is $(r + p) \times r$ and needs to be $m \times r$



Halko, Martinsson, Tropp, SIAM Review, 2011

RRF and RSVD Analysis

RRF Accuracy:

$$\mathbb{E}_\Omega \|X - QQ^\top X\|_F^2 \leq \left(1 + \frac{r}{p-1}\right) \sum_{j=r+1}^{\min\{m,n\}} \sigma_j^2(X)$$

Halko, Martinsson, Tropp, SIAM Review, 2011

RRF and RSVD Analysis

RSVD Accuracy:

$$\mathbb{E}_\Omega \|X - QQ^\top X\|_F^2 \leq \mathbb{E}_\Omega \|X - U\Sigma V^\top\|_F^2 \leq \left(1 + \frac{r}{p-1}\right) \sum_{j=r+1}^{\min\{m,n\}} \sigma_j^2(X)$$

Zhang, Saibaba, Kilmer, Aeron, Numerical Linear Algebra and Applications, 2018

RRF and RSVD Analysis

RSVD Accuracy:

$$\mathbb{E}_\Omega \|X - QQ^\top X\|_F^2 \leq \mathbb{E}_\Omega \|X - U\Sigma V^\top\|_F^2 \leq \left(1 + \frac{r}{p-1}\right) \sum_{j=r+1}^{\min\{m,n\}} \sigma_j^2(X)$$

Computational Complexity

Randomized Range Finder: $\mathcal{O}(mn(r+p))$

- Dominant costs: forming sketch and thin QR

Randomized SVD: $\mathcal{O}(mn(r+p))$

- Dominant costs: forming sketch, thin QR, forming B, truncated SVD, recovering U

Zhang, Saibaba, Kilmer, Aeron, Numerical Linear Algebra and Applications, 2018

Improvements to Randomized SVD

Many improvements to randomized SVD algorithm for lower complexity or higher accuracy

Examples:

- Subspace iterations[1]
- Adaptive randomized SVD[1]
- RSVDPACK[2]

[1] Halko, Martinsson, Tropp, SIAM Review, 2011,

[2] Voronin, Martinsson, arXiv preprint arXiv:1502.05366, 2015

Improvements to Randomized SVD

Many improvements to randomized SVD algorithm for lower complexity or higher accuracy

Examples:

- Subspace iterations[1]
  - Computing $Y = (XX^\top)^q X\Omega$
  - Higher accuracy, slower and more passes through data
- Adaptive randomized SVD[1]
- RSVDPACK[2]

[1] Halko, Martinsson, Tropp, SIAM Review, 2011,

[2] Voronin, Martinsson, arXiv preprint arXiv:1502.05366, 2015

Improvements to Randomized SVD

Many improvements to randomized SVD algorithm for lower complexity or higher accuracy

Examples:

- Subspace iterations[1]
  - Computing $Y = (XX^\top)^q X\Omega$
  - Higher accuracy, slower and more passes through data
- Adaptive randomized SVD[1]
  - Computes approximation satisfying given error tolerance instead of given rank
  - Better in many practical applications where rank of data is not known
- RSVDPACK[2]

[1] Halko, Martinsson, Tropp, SIAM Review, 2011, [2] Voronin, Martinsson, arXiv preprint arXiv:1502.05366, 2015

Improvements to Randomized SVD

Many improvements to randomized SVD algorithm for lower complexity or higher accuracy

Examples:

- Subspace iterations[1]
  - Computing $Y = (XX^\top)^q X\Omega$
  - Higher accuracy, slower and more passes through data
- Adaptive randomized SVD[1]
  - Computes approximation satisfying given error tolerance instead of given rank
  - Better in many practical applications where rank of data is not known
- RSVDPACK[2]
  - Uses eigenvalue decomposition of B instead of truncated SVD
  - Uses QR of B instead of truncated SVD
  - Much faster, similar accuracy

[1] Halko, Martinsson, Tropp, SIAM Review, 2011, [2] Voronin, Martinsson, arXiv preprint arXiv:1502.05366, 2015

Improvements to Randomized SVD

Many improvements to randomized SVD algorithm for lower complexity or higher accuracy

Examples:

- **Subspace iterations**[1]
  - Computing $Y = (XX^\top)^q X\Omega$
  - Higher accuracy, slower and more passes through data
- Adaptive randomized SVD[1]
  - Computes approximation satisfying given error tolerance instead of given rank
  - Better in many practical applications where rank of data is not known
- RSVDPACK[2]
  - Uses eigenvalue decomposition of B instead of truncated SVD
  - Uses QR of B instead of truncated SVD
  - Much faster, similar accuracy

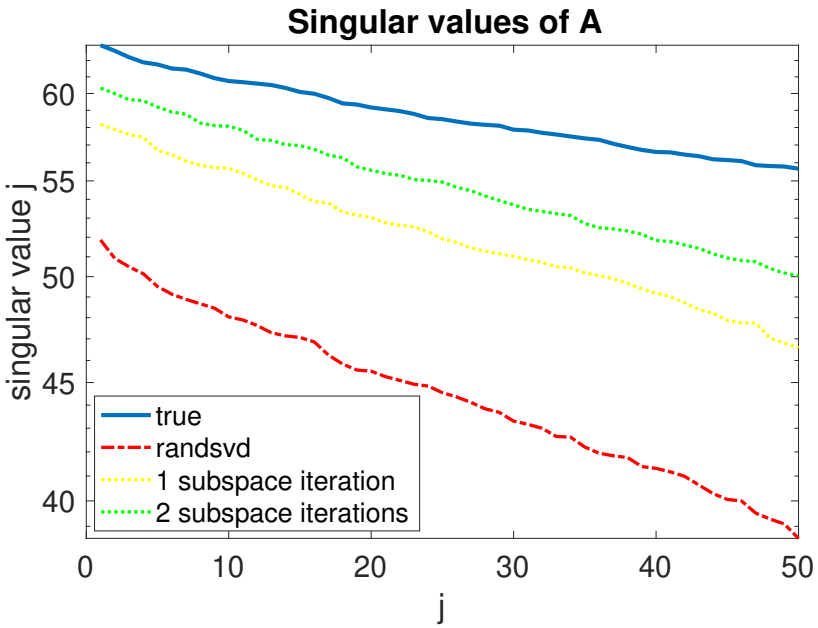
[1] Halko, Martinsson, Tropp, SIAM Review, 2011, [2] Voronin, Martinsson, arXiv preprint arXiv:1502.05366, 2015

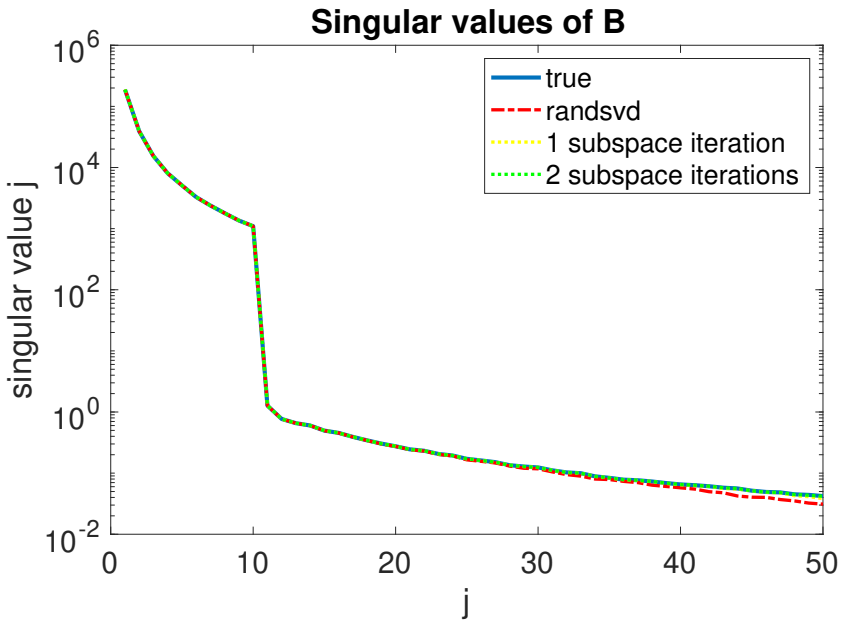
Numerical Results – Singular Values

Matrix A: Random 1000×1000 matrix

Parameters: $r = 50, p = 5$

Matrix B: Synthetic 1000×1000 matrix with 10 large singular values



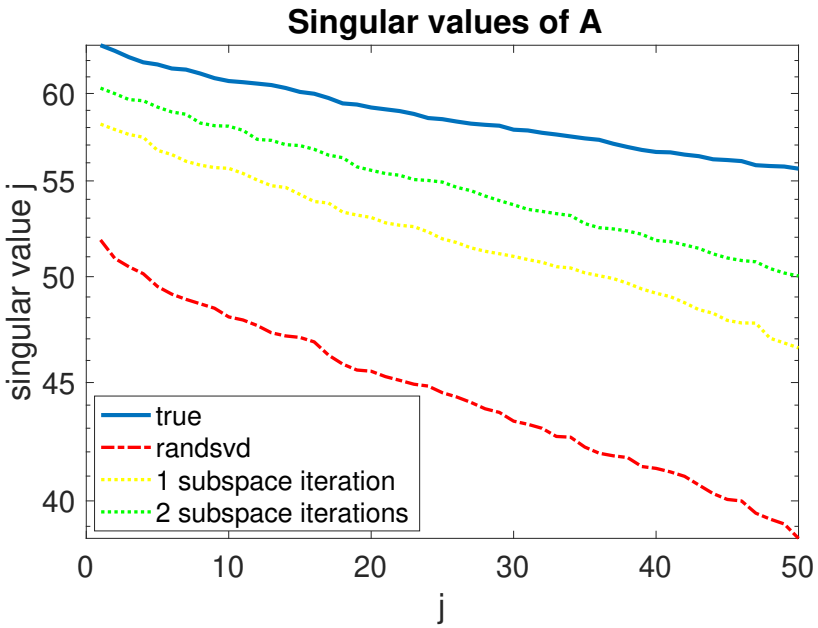


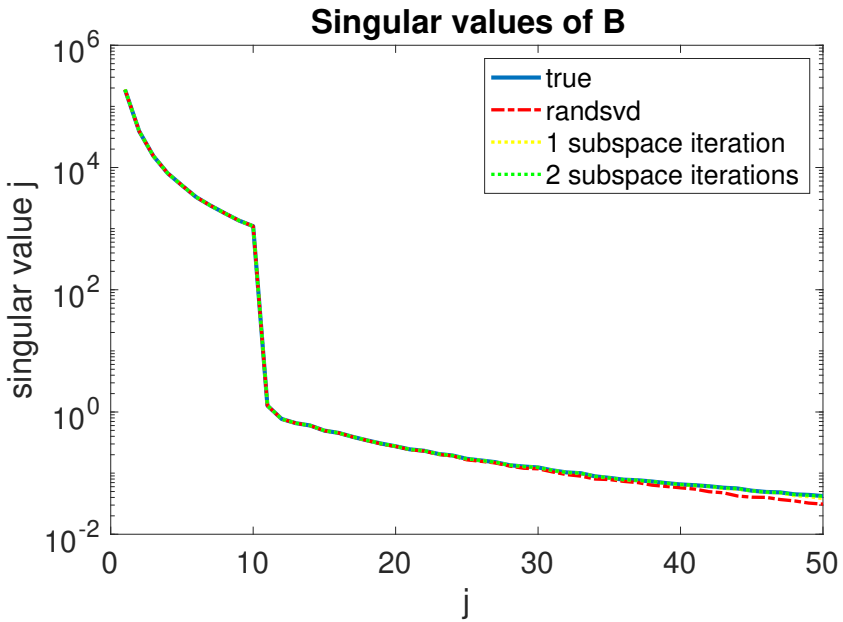
Numerical Results – Singular Values

Matrix A: Random 1000×1000 matrix

Parameters: $r = 50, p = 5$

Matrix B: Synthetic 1000×1000 matrix with 10 large singular values



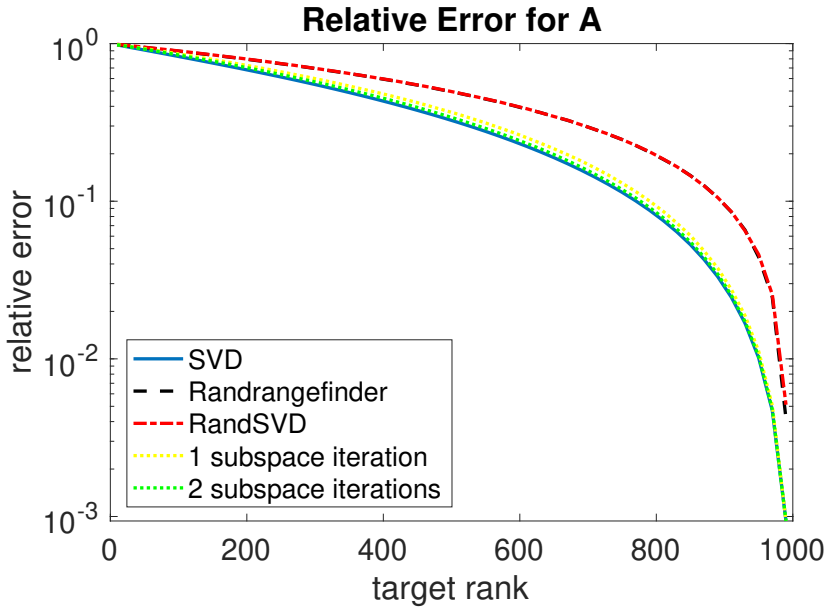


Timing (for A):

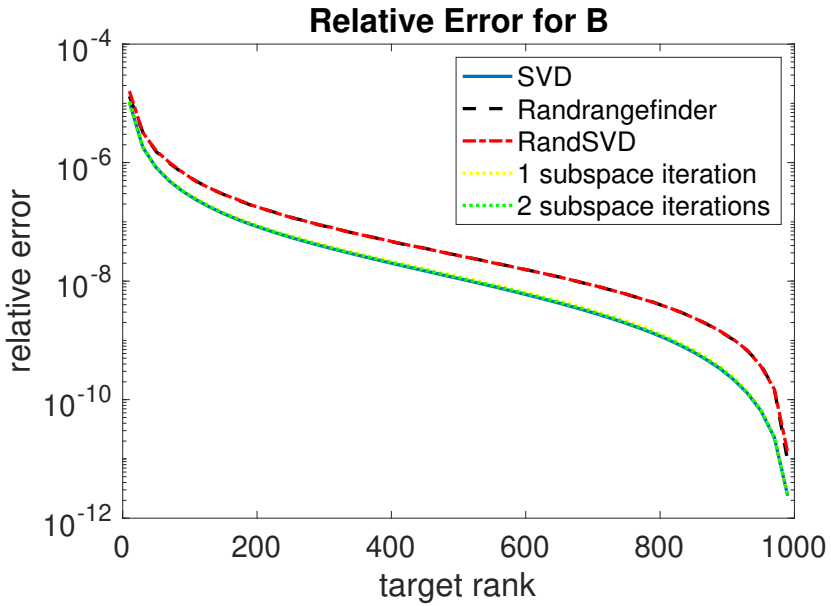
SVD	RRF	RSVD (0)	RSVD (1)	RSVD (2)
0.3424	0.0085	0.0139	0.0185	0.0425

Numerical Results – Relative Error

Matrix A: Random 1000×1000 matrix



Matrix B: Synthetic 1000×1000 matrix with 10 large singular values



Takeaways for Randomization in NLA

Main idea:

Using dimension reduction via randomization can significantly reduce the cost of computing low-rank matrix approximations.

Two methods:

- Randomized range finder: $X \approx QQ^\top X$, rank-$r + p$ approximation
- Randomized SVD: $X \approx U\Sigma V^\top$, rank-r approximation
- Asymptotic cost equivalent, but randomized SVD has more lower-order computations

Works well when singular values decay rapidly

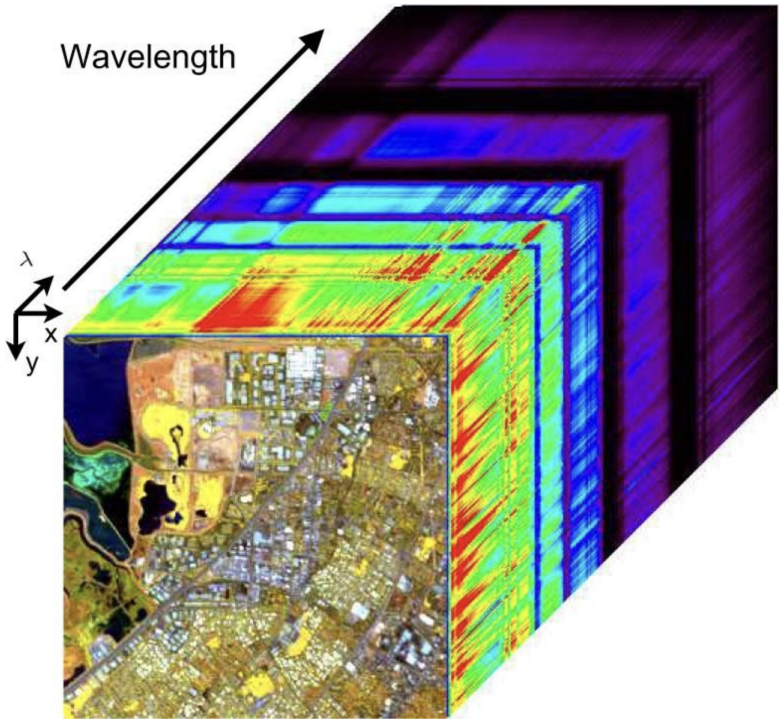
Outline

1. Randomization in NLA
   - Background on randomized matrix techniques
2. **Tensors**
   - Intro to tensors, tensor operations
   - Tucker decompositions
3. Randomized algorithms for Tucker decompositions
   - Algorithms and analysis
4. Improvements on randomized algorithms for Tucker decompositions
   - Kronecker-structured random matrices
   - Parallel implementation
5. Structure-preserving Tucker decomposition and algorithms
   - Variation on the Tucker decomposition with strong benefits

Motivation: Multidimensional data

Multidimensional data (tensors) appears in many applications:

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



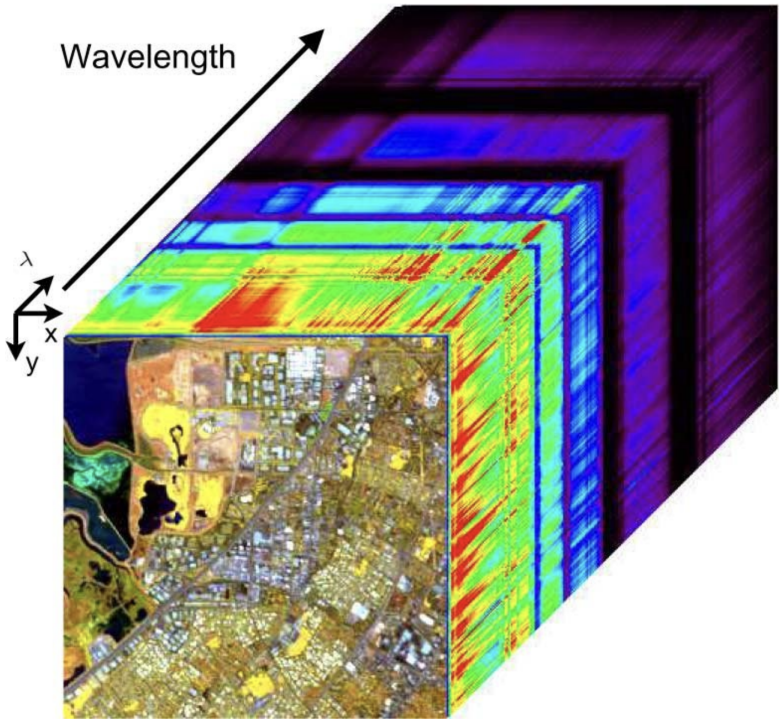
Christophe, Duhamel, IEEE Transactions on Image Processing, 2009

Motivation: Multidimensional data

Multidimensional data (tensors) appears in many applications:

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

and is often large and difficult to store or compute with



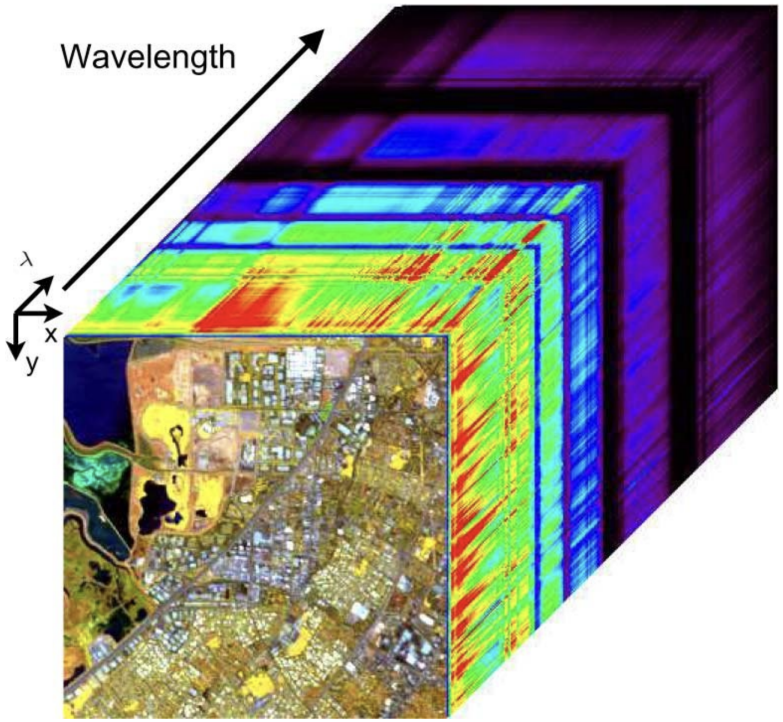
Christophe, Duhamel, IEEE Transactions on Image Processing, 2009

Motivation: Multidimensional data

Multidimensional data (tensors) 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 compress data in tensor form

Method: Use tensor decompositions

- Tucker decomposition: obtain large compression ratios with high accuracy

Christophe, Duhamel, IEEE Transactions on Image Processing, 2009

Matricization

Unfolding a tensor:

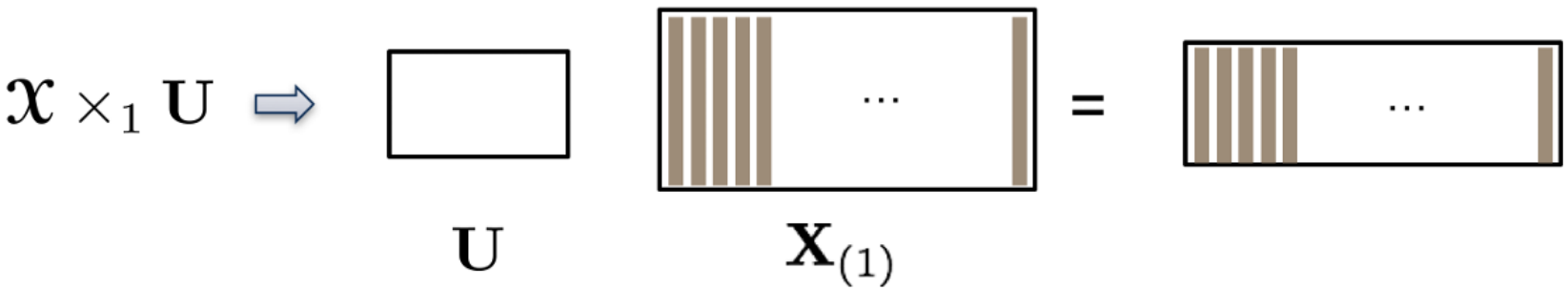
- Can unfold a d-mode tensor into a matrix in d different ways
- For mode j, arrange mode-j fibers (rows, columns, etc.) of $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ to be the columns of the matrix $X_{(j)} \in \mathbb{R}^{n_j \times \prod_{i \neq j} n_i}$

Consider $\mathcal{X}_{::1} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ $\mathcal{X}_{::2} = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$

$$X_{(1)} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix} \quad X_{(2)} = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix} \quad X_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

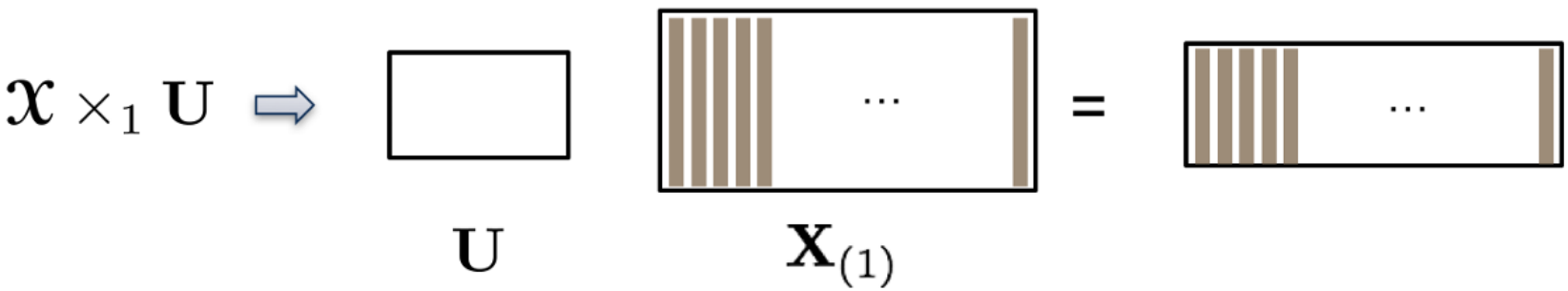
Tensor-times-matrix (TTM) and Multi-TTM

- 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



Tensor-times-matrix (TTM) and Multi-TTM

- 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

$$U_j X_{(j)} (U_d \otimes U_{d-1} \otimes \cdots \otimes U_{j+1} \otimes U_{j-1} \otimes \cdots \otimes U_1)^\top$$

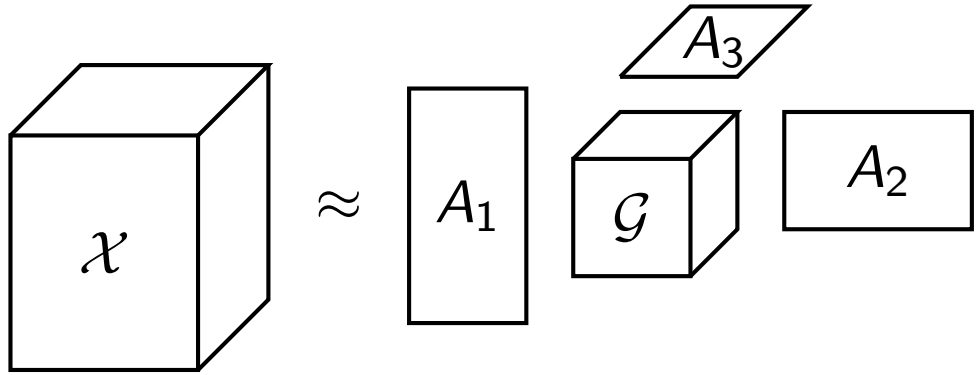
with $\otimes$ the Kronecker product

Tucker Format

Approximates tensor $\mathcal{X}$ as

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

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



Popular algorithms: Higher Order SVD (HOSVD)[1] and Sequentially Truncated Higher Order SVD (STHOSVD)[2]

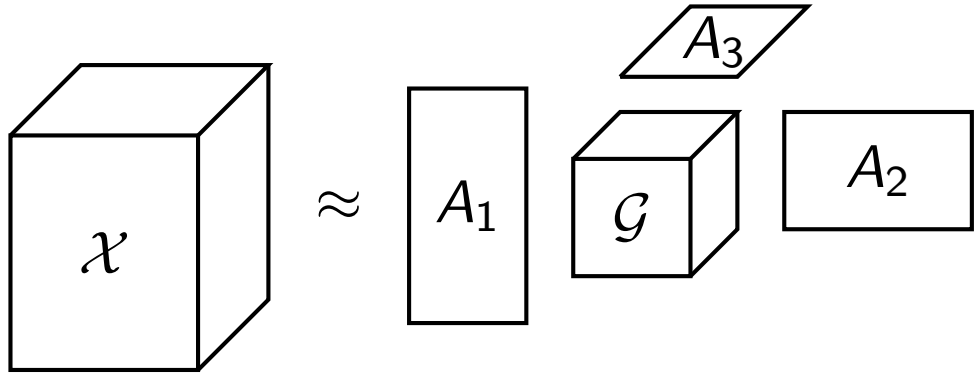
[1] De Lathauwer, De Moor, Vandewalle, SIAM Journal on Matrix Analysis and Applications, 2000
[2] Vannieuwenhoven, Vandebril, Meerbergen, SIAM Journal on Scientific Computing, 2012

Tucker Format

Approximates tensor $\mathcal{X}$ as

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

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



Popular algorithms: Higher Order SVD (HOSVD)[1] and Sequentially Truncated Higher Order SVD (STHOSVD)[2]

General approach for each mode (HOSVD):

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 formed via TTM's

[1] De Lathauwer, De Moor, Vandewalle, SIAM Journal on Matrix Analysis and Applications, 2000
[2] Vannieuwenhoven, Vandebril, Meerbergen, SIAM Journal on Scientific Computing, 2012

Sequentially Truncated HOSVD (STHOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$

Algorithm Steps:

For the current mode,

1. Rank-r SVD of core tensor unfolding
2. Update core tensor

Details:

- $G_{(j)} \approx U_r \Sigma_r V_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = U_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \Sigma_r V_r^\top$)

Vannieuwenhoven, Vandebril, Meerbergen, SISC, 2012

Sequentially Truncated HOSVD (STHOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$

Algorithm Steps:

For the current mode,

1. **Rank-r SVD of core tensor unfolding**
2. Update core tensor

Details:

- $G_{(j)} \approx U_r \Sigma_r V_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = U_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \Sigma_r V_r^\top$)

Vannieuwenhoven, Vandebril, Meerbergen, SISC, 2012

Sequentially Truncated HOSVD (STHOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$

Algorithm Steps:

For the current mode,

1. Rank-r SVD of core tensor unfolding
2. **Update core tensor**

Details:

- $G_{(j)} \approx U_r \Sigma_r V_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = U_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \Sigma_r V_r^\top$)

Vannieuwenhoven, Vandebril, Meerbergen, SISC, 2012

Sequentially Truncated HOSVD (STHOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$

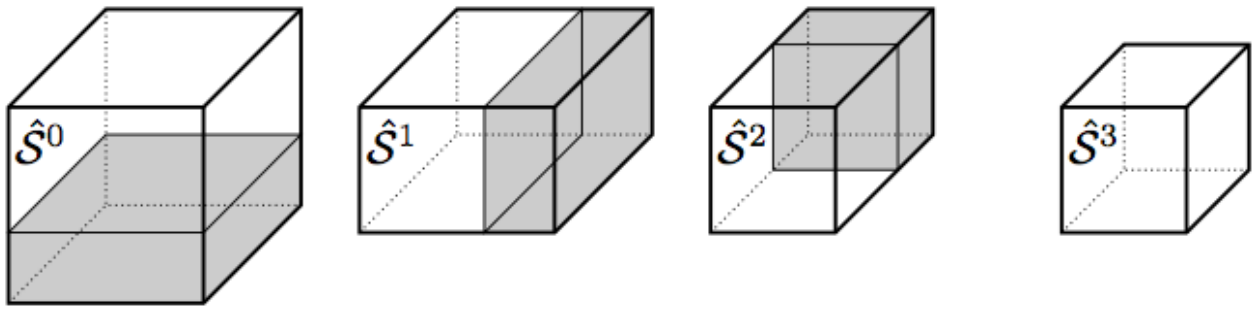
Algorithm Steps:

For the current mode,

1. Rank-r SVD of core tensor unfolding
2. Update core tensor

Details:

- $G_{(j)} \approx U_r \Sigma_r V_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = U_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \Sigma_r V_r^\top$)



Vannieuwenhoven, Vandebril, Meerbergen, SISC, 2012

Outline

1. Randomization in NLA
   - Background on randomized matrix techniques
2. Tensors
   - Intro to tensors, tensor operations
   - Tucker decompositions
3. **Randomized algorithms for Tucker decompositions**
   (Work with Arvind Saibaba and Misha Kilmer)
   - Algorithms and analysis
4. Improvements on randomized algorithms for Tucker decompositions
   - Kronecker-structured random matrices
   - Parallel implementation
5. Structure-preserving Tucker decomposition and algorithms
   - Variation on the Tucker decomposition with strong benefits

Randomized Algorithms for Tucker Decompositions Idea

Both HOSVD/STHOSVD:

- dominant cost is an SVD of $X_{(1)}$
- Cost: $\mathcal{O}(n^{d+1})$

Randomized Algorithms for Tucker Decompositions Idea

Both HOSVD/STHOSVD:

- dominant cost is an SVD of $X_{(1)}$
- Cost: $\mathcal{O}(n^{d+1})$

If we instead use Randomized SVD:

- dominant cost is an SVD of $Q^\top X_{(1)}$
- Cost: $\mathcal{O}(rn^d)$
- Factor matrices approximate singular vectors

Randomized Algorithms for Tucker Decompositions Idea

Both HOSVD/STHOSVD:

- dominant cost is an SVD of $X_{(1)}$
- Cost: $\mathcal{O}(n^{d+1})$

If we instead use Randomized SVD:

- dominant cost is an SVD of $Q^\top X_{(1)}$
- Cost: $\mathcal{O}(rn^d)$
- Factor matrices approximate singular vectors

If we instead use Randomized Range Finder:

- dominant cost is matrix multiplication $X_{(1)}\Omega$
- Cost: $\mathcal{O}(rn^d)$, lower order terms cheaper
- Factor matrices approximate range of $X_{(1)}$, not singular vectors

Randomized Algorithms for Tucker Decompositions Idea

Both HOSVD/STHOSVD:

- dominant cost is an SVD of $X_{(1)}$
- Cost: $\mathcal{O}(n^{d+1})$

If we instead use Randomized SVD:

- dominant cost is an SVD of $Q^\top X_{(1)}$
- Cost: $\mathcal{O}(rn^d)$
- Factor matrices approximate singular vectors

If we instead use Randomized Range Finder:

- dominant cost is matrix multiplication $X_{(1)}\Omega$
- Cost: $\mathcal{O}(rn^d)$, lower order terms cheaper
- Factor matrices approximate range of $X_{(1)}$, not singular vectors

Idea: replace expensive SVD step with cheaper randomized approach

Randomized Tucker (HOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$, oversampling parameter p

Algorithm steps:
For current mode j,

1. **Unfold tensor along mode j**
2. Compute randomized SVD of unfolding
3. Form factor matrix A_j

After all modes processed,

1. Form core $\mathcal{G} = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top$

Details:

- With target rank r_j and oversampling parameter p
- Use independent random matrix for each mode
- Factor matrix made of first r_j left singular vectors (approximate)
- Core essentially formed from right singular vectors scaled by singular values

Randomized Tucker (HOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$, oversampling parameter p

Algorithm steps:
For current mode j,

1. Unfold tensor along mode j
2. **Compute randomized SVD of unfolding**
3. Form factor matrix A_j

After all modes processed,

1. Form core $\mathcal{G} = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top$

Details:

- With target rank r_j and oversampling parameter p
- Use independent random matrix for each mode
- Factor matrix made of first r_j left singular vectors (approximate)
- Core essentially formed from right singular vectors scaled by singular values

Randomized Tucker (HOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$, oversampling parameter p

Algorithm steps:
For current mode j,

1. Unfold tensor along mode j
2. Compute randomized SVD of unfolding
3. **Form factor matrix** A_j

After all modes processed,

1. Form core $\mathcal{G} = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top$

Details:

- With target rank r_j and oversampling parameter p
- Use independent random matrix for each mode
- Factor matrix made of first r_j left singular vectors (approximate)
- Core essentially formed from right singular vectors scaled by singular values

Randomized Tucker (HOSVD)

Inputs: $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, target rank $(r_1, \ldots, r_d)$, oversampling parameter p

Algorithm steps:
For current mode j,

1. Unfold tensor along mode j
2. Compute randomized SVD of unfolding
3. Form factor matrix A_j

After all modes processed,

1. **Form core**
   $\mathcal{G} = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top$

Details:

- With target rank r_j and oversampling parameter p
- Use independent random matrix for each mode
- Factor matrix made of first r_j left singular vectors (approximate)
- Core essentially formed from right singular vectors scaled by singular values

Randomized STHOSVD Algorithm (R-STHOSVD)

New input: processing order ρ

Main Steps:

For current mode j,

1. Randomized SVD of unfolding
2. Update core tensor

Details:

- An independent $\Omega_j \in \mathbb{R}^{n_j \times (r_j+p)}$ generated for each mode
- $G_{(j)} \approx \widehat{U}_r \widehat{\Sigma}_r \widehat{V}_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = \widehat{U}_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \widehat{\Sigma}_r \widehat{V}_r^\top$)

Randomized STHOSVD Algorithm (R-STHOSVD)

New input: processing order ρ

Main Steps:

For current mode j,

1. **Randomized SVD of unfolding**
2. Update core tensor

Details:

- An independent $\Omega_j \in \mathbb{R}^{n_j \times (r_j + p)}$ generated for each mode
- $G_{(j)} \approx \widehat{U}_r \widehat{\Sigma}_r \widehat{V}_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = \widehat{U}_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \widehat{\Sigma}_r \widehat{V}_r^\top$)

Randomized STHOSVD Algorithm (R-STHOSVD)

New input: processing order ρ

Main Steps:

For current mode j,

1. Randomized SVD of unfolding
2. **Update core tensor**

Details:

- An independent $\Omega_j \in \mathbb{R}^{n_j \times (r_j+p)}$ generated for each mode
- $G_{(j)} \approx \widehat{U}_r \widehat{\Sigma}_r \widehat{V}_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = \widehat{U}_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \widehat{\Sigma}_r \widehat{V}_r^\top$)

Randomized STHOSVD Algorithm (R-STHOSVD)

New input: processing order ρ

Main Steps:

For current mode j,

1. Randomized SVD of unfolding
2. Update core tensor

Details:

- An independent $\Omega_j \in \mathbb{R}^{n_j \times (r_j+p)}$ generated for each mode
- $G_{(j)} \approx \widehat{U}_r \widehat{\Sigma}_r \widehat{V}_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = \widehat{U}_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \widehat{\Sigma}_r \widehat{V}_r^\top$)

Results after each mode:

- partially truncated core tensor $\mathcal{G}^{(j)} = \mathcal{X} \times_{k=1}^{j} A_k^\top$

Randomized STHOSVD Algorithm (R-STHOSVD)

New input: processing order ρ

Main Steps:

For current mode j,

1. Randomized SVD of unfolding
2. Update core tensor

Details:

- An independent $\Omega_j \in \mathbb{R}^{n_j \times (r_j+p)}$ generated for each mode
- $G_{(j)} \approx \widehat{U}_r \widehat{\Sigma}_r \widehat{V}_r^\top$
- Factor matrix saved as leading r left singular vectors ($A_j = \widehat{U}_r$)
- Save core tensor as r right singular vectors scaled by singular values ($G_{(j)} \leftarrow \widehat{\Sigma}_r \widehat{V}_r^\top$)

Results after each mode:

- partially truncated core tensor $\mathcal{G}^{(j)} = \mathcal{X} \times_{k=1}^{j} A_k^\top$
- After mode j, we have j-th approximation $\mathcal{X}^{(j)} = \mathcal{G}^{(j)} \times_{k=1}^{j} A_k$
- Alternate version $\mathcal{X}^{(j)} = \mathcal{X} \times_{k=1}^{j} A_k A_k^\top$

Error Analysis for R-STHOSVD

Assumptions (simple case):

- $\widehat{\mathcal{X}} = [\mathcal{G}; A_1, \ldots, A_d]$ the randomized STHOSVD of $\mathcal{X} \in \mathbb{R}^{n \times \cdots \times n}$
- Target rank $\mathbf{r} = (r, r, \ldots, r)$
- Oversampling parameter $p \geq 2$ such that $r + p \leq n$
- Processing order $\rho = [1, 2, \ldots, d]$

Theorem (M, Saibaba, Kilmer)

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2 \leq \left(1 + \frac{r}{p-1}\right) \sum_{j=1}^{d} \sum_{i=r+1}^{n} \sigma_i^2(X_{(j)})$$

Error Analysis for R-STHOSVD

Assumptions (simple case):

- $\widehat{\mathcal{X}} = [\mathcal{G}; A_1, \ldots, A_d]$ the randomized STHOSVD of $\mathcal{X} \in \mathbb{R}^{n \times \cdots \times n}$
- Target rank $\mathbf{r} = (r, r, \ldots, r)$
- Oversampling parameter $p \geq 2$ such that $r + p \leq n$
- Processing order $\rho = [1, 2, \ldots, d]$

Theorem (M, Saibaba, Kilmer)

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2 \leq \left(1 + \frac{r}{p-1}\right) \sum_{j=1}^{d} \sum_{i=r+1}^{n} \sigma_i^2(X_{(j)})$$

Alternate form:

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2 \leq d \left(1 + \frac{r}{p-1}\right) \|\mathcal{X} - \widehat{\mathcal{X}}_{\text{opt}}\|_F^2$$

Error Analysis for R-STHOSVD

Theorem (M, Saibaba, Kilmer)

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F \leq \left(\left(1 + \frac{r}{p-1}\right) \sum_{j=1}^{d} \sum_{i=r+1}^{n} \sigma_i^2(X_{(j)}) \right)^{1/2}$$

Notes:

- Bound is independent of processing order
- Choice of oversampling parameter $p = r + 1$

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F \leq \sqrt{2d} \|\mathcal{X} - \widehat{\mathcal{X}}_{\text{opt}}\|_F$$

Error Analysis for R-STHOSVD

Theorem (M, Saibaba, Kilmer)

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F \leq \left(\left(1 + \frac{r}{p-1}\right) \sum_{j=1}^{d} \sum_{i=r+1}^{n} \sigma_i^2(X_{(j)}) \right)^{1/2}$$

Notes:

- Bound is independent of processing order
- Choice of oversampling parameter $p = r + 1$

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F \leq \sqrt{2d} \|\mathcal{X} - \widehat{\mathcal{X}}_{\text{opt}}\|_F$$

Proof Summary (3 dimensions)

Assumptions:

- $\widehat{\mathcal{X}} = [\mathcal{G}; A_1, A_2, A_3]$ the randomized STHOSVD of $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$
- Target rank $\mathbf{r} = (r, r, r)$
- Oversampling parameter $p \geq 2$
- Processing order $\rho = [1, 2, 3]$

Notation:

- $\mathbb{E}_1 = \mathbb{E}_{\Omega_1}, \ \mathbb{E}_{12} = \mathbb{E}_{\{\Omega_1, \Omega_2\}}$
- partially truncated core tensor $\mathcal{G}^{(j)} = \mathcal{X} \times_{k=1}^{j} A_k^\top$
- j-th approximation $\widehat{\mathcal{X}}^{(j)} = \mathcal{G}^{(j)} \times_{k=1}^{j} A_k$

Proof Summary (3 dimensions)

Assumptions:

- $\widehat{\mathcal{X}} = [\mathcal{G}; A_1, A_2, A_3]$ the randomized STHOSVD of $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$
- Target rank $\mathbf{r} = (r, r, r)$
- Oversampling parameter $p \geq 2$
- Processing order $\rho = [1, 2, 3]$

Notation:

- $\mathbb{E}_1 = \mathbb{E}_{\Omega_1}, \ \mathbb{E}_{12} = \mathbb{E}_{\{\Omega_1, \Omega_2\}}$
- partially truncated core tensor $\mathcal{G}^{(j)} = \mathcal{X} \times_{k=1}^{j} A_k^\top$
- j-th approximation $\widehat{\mathcal{X}}^{(j)} = \mathcal{G}^{(j)} \times_{k=1}^{j} A_k$

Main steps:

1. Linearity of expectations and independence of Ω_j's
2. Rewrite/unfold
3. Bound using randomized matrix results
4. Combine

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \quad \|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2 + \quad \|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2 + \quad \|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

$$\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 = \mathbb{E}_{12}\mathbb{E}_3\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

$$\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 = \mathbb{E}_{12}\mathbb{E}_3\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

$$\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 = \mathbb{E}_{12}\mathbb{E}_3\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2$$

Step 2 Rewrite/unfold:

$$\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 = \|\widehat{\mathcal{X}}^{(2)} \times_3 (I - A_3A_3^\top)\|_F^2$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

$$\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 = \mathbb{E}_{12}\mathbb{E}_3\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2$$

Step 2 Rewrite/unfold:

$$\begin{aligned}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 &= \|\widehat{\mathcal{X}}^{(2)} \times_3 (I - A_3A_3^\top)\|_F^2 \\ &= \|\mathcal{G}^{(2)} \times_1 A_1 \times_2 A_2 \times_3 (I - A_3A_3^\top)\|_F^2\end{aligned}$$

Proof Summary (3 dimensions)

Step 1 Linearity of expectations and independence of Ω_j's

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

$$\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 = \mathbb{E}_{12}\mathbb{E}_3\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2$$

Step 2 Rewrite/unfold:

$$\begin{aligned}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 &= \|\widehat{\mathcal{X}}^{(2)} \times_3 (I - A_3A_3^\top)\|_F^2 \\ &= \|\mathcal{G}^{(2)} \times_1 A_1 \times_2 A_2 \times_3 (I - A_3A_3^\top)\|_F^2 \\ &= \|(I - A_3A_3^\top)G_{(3)}^{(2)}(I \otimes A_2^\top \otimes A_1^\top)\|_F^2\end{aligned}$$

Proof Summary (3 dimensions)

Step 3 Bound using randomized matrix results:

$$\mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}(I \otimes A_2^\top \otimes A_1^\top)\|_F^2 \leq \mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}\|_F^2$$

Proof Summary (3 dimensions)

Step 3 Bound using randomized matrix results:

$$\mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}(I \otimes A_2^\top \otimes A_1^\top)\|_F^2 \leq \mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}\|_F^2$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(G_{(3)}^{(2)})\right]$$

Proof Summary (3 dimensions)

Step 3 Bound using randomized matrix results:

$$\mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}(I \otimes A_2^\top \otimes A_1^\top)\|_F^2 \leq \mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}\|_F^2$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(G_{(3)}^{(2)})\right]$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(X_{(3)})\right]$$

Proof Summary (3 dimensions)

Step 3 Bound using randomized matrix results:

$$\mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}(I \otimes A_2^\top \otimes A_1^\top)\|_F^2 \leq \mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}\|_F^2$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(G_{(3)}^{(2)})\right]$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(X_{(3)})\right]$$

$$= \left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(X_{(3)})$$

Proof Summary (3 dimensions)

Step 3 Bound using randomized matrix results:

$$\mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}(I \otimes A_2^\top \otimes A_1^\top)\|_F^2 \leq \mathbb{E}_{12}\mathbb{E}_3\|(I - A_3A_3^\top)G_{(3)}^{(2)}\|_F^2$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(G_{(3)}^{(2)})\right]$$

$$\leq \mathbb{E}_{12}\left[\left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(X_{(3)})\right]$$

$$= \left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(X_{(3)})$$

Summary:

$$\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)} - \widehat{\mathcal{X}}^{(3)}\|_F^2 \leq \left(1 + \frac{r}{p-1}\right)\sum_{i=r+1}^{n}\sigma_i^2(X_{(3)})$$

Proof Summary (3 dimensions)

Recall sum:

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

Step 4 Combine:

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 \leq \left(1+\frac{r}{p-1}\right)\sum_{j=1}^{3}\sum_{i=r+1}^{n}\sigma_i^2(X_{(j)})$$

Proof Summary (3 dimensions)

Recall sum:

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 = \mathbb{E}_1\|\mathcal{X}-\widehat{\mathcal{X}}^{(1)}\|_F^2+\mathbb{E}_{12}\|\widehat{\mathcal{X}}^{(1)}-\widehat{\mathcal{X}}^{(2)}\|_F^2+\mathbb{E}_{123}\|\widehat{\mathcal{X}}^{(2)}-\widehat{\mathcal{X}}^{(3)}\|_F^2$$

Step 4 Combine:

$$\mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F^2 \leq \left(1+\frac{r}{p-1}\right)\sum_{j=1}^{3}\sum_{i=r+1}^{n}\sigma_i^2(X_{(j)})$$

$$\implies \mathbb{E}_{123}\|\mathcal{X}-\widehat{\mathcal{X}}\|_F \leq \left(\left(1+\frac{r}{p-1}\right)\sum_{j=1}^{3}\sum_{i=r+1}^{n}\sigma_i^2(X_{(j)})\right)^{1/2}$$

Results: Relative Error and Runtime

Test tensor: $\mathcal{X} \in \mathbb{R}^{25\times25\times25\times25\times25}$

- $\mathcal{X}_{ijklm} = \frac{1}{i+j+k+l+m}$
- $\mathcal{X}$ has $9,765,625$ nonzeros

Inputs

- oversampling parameter $p = 5$
- processing order $\rho = [1, 2, 3, 4, 5]$

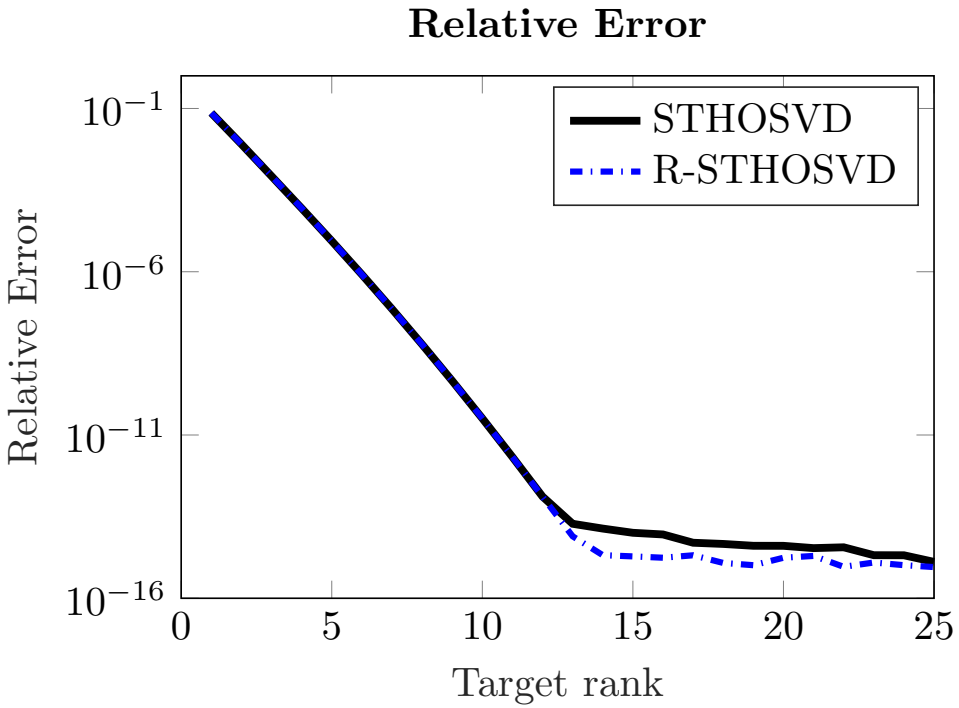
Results: Relative Error and Runtime

Test tensor: $\mathcal{X} \in \mathbb{R}^{25\times25\times25\times25\times25}$

- $\mathcal{X}_{ijklm} = \frac{1}{i+j+k+l+m}$
- $\mathcal{X}$ has $9,765,625$ nonzeros

Inputs

- oversampling parameter $p = 5$
- processing order $\rho = [1, 2, 3, 4, 5]$



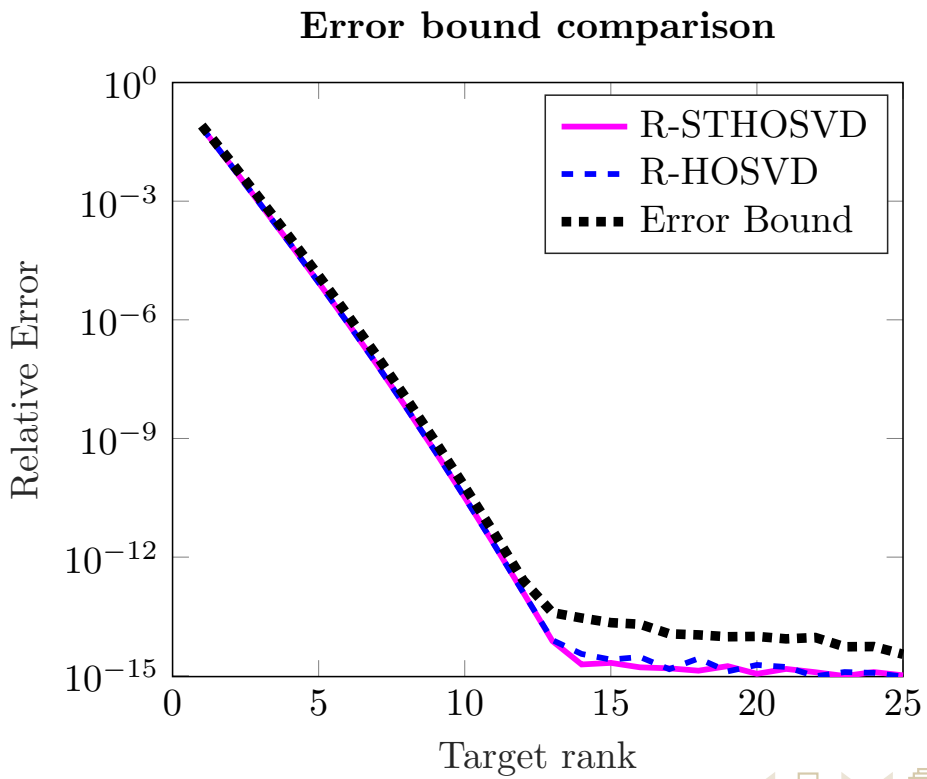
Runtime in seconds when $r = (5, 5, 5, 5, 5)$

n	STHOSVD	R-STHOSVD
25	0.6455	0.2875
35	3.1974	1.2090
45	15.0629	3.5587
50	21.9745	5.5606

Results: Comparison of Error

Recall R-STHOSVD Error Bound:

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F \leq \left(\left(1 + \frac{r}{p-1}\right) \sum_{j=1}^{d} \sum_{i=r+1}^{n} \sigma_i^2(X_{(j)}) \right)^{1/2}$$



Takeaways

Main idea:

Incorporating randomized matrix techniques into tensor decompositions reduces the cost, and provably increases the expected error by only a small factor.

Algorithms:

- Randomized HOSVD: processes modes independently
- Randomized STHOSVD: processes modes in given order, new core and approximation after processing each mode

Outline

1. Randomization in NLA
   - Background on randomized matrix techniques
2. Tensors
   - Intro to tensors, tensor operations
   - Tucker decompositions
3. Randomized algorithms for Tucker decompositions
   - Algorithms and analysis
4. **Improvements on randomized algorithms for Tucker decompositions**
   (Work with Grey Ballard and Zitong Li)
   - Kronecker-structured random matrices
   - Parallel implementation
5. Structure-preserving Tucker decomposition and algorithms
   - Variation on the Tucker decomposition with strong benefits

Tucker Format

Recall general approach (HOSVD):

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 formed via TTM's

Tucker Format

Recall general approach (HOSVD):

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 formed via TTM's

Our approach:

- Use a modified randomized algorithm 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

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 $r \leq \operatorname{rank} X$
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 $X \approx QQ^\top X$

Inputs: matrix $X \in \mathbb{R}^{m \times n}$
target rank $r \leq \operatorname{rank} X$
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 Φ_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

Halko, Martinsson, Tropp, SIAM Review, 2011

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:
   $$\mathcal{Y} = \mathcal{X} \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}$$
   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
   a. 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:
   $$\mathcal{Y} = \mathcal{X} \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}$$
   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
   a. 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:
   $$\mathcal{Y} = \mathcal{X} \times_1 \Phi_1^{(j)} \times \cdots \times_{j-1} \Phi_{j-1}^{(j)} \times_{j+1} \Phi_{j+1}^{(j)} \times \cdots \times_d \Phi_d^{(j)}$$
   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**
   a. Deterministic HOSVD on $\mathcal{G}$, 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 $Y = X_{(j)}\Omega \quad \rightarrow \quad$ 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 $Y = X_{(j)}\Omega \quad \rightarrow \quad$ 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 $Y = X_{(j)}\Omega \quad \rightarrow \quad$ 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 $Y = X_{(j)}\Omega \quad \rightarrow \quad$ one multi-TTM with skinny matrices

Two options for our approach:

1. Use an independent product of Φ_j's per mode

2. Reuse same Kronecker factors Φ_j in Ω_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 $Y = X_{(j)}\Omega \quad \rightarrow \quad$ 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 $Y = X_{(j)}\Omega \quad \rightarrow \quad$ one multi-TTM with skinny matrices

Two options for our approach:

1. Use an independent product of Φ_j's per mode
   - Generating and storing more random matrices
   - "rKron"
2. Reuse same Kronecker factors Φ_j in Ω_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 random matrices: $\Phi = DH$
  - D diagonal Rademacher
  - H randomly sampled columns from Hadamard matrix

Error bound

Except with probability at most $\frac{d}{\beta^2}$,

$$\|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2 \leq d\left(1 + \alpha\left(1 + \sqrt{\frac{\beta(r^2 + r)}{(r+p)^{d-1}}}\right)\right)\|\mathcal{X} - \widehat{\mathcal{X}}_{\text{opt}}\|_F^2$$

Theoretical Bound

Error bound

Except with probability at most $\frac{d}{\beta^2}$,

$$\|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2 \leq d\left(1 + \alpha\left(1 + \sqrt{\frac{\beta(r^2 + r)}{(r+p)^{d-1}}}\right)\right)\|\mathcal{X} - \widehat{\mathcal{X}}_{\text{opt}}\|_F^2$$

Notes:

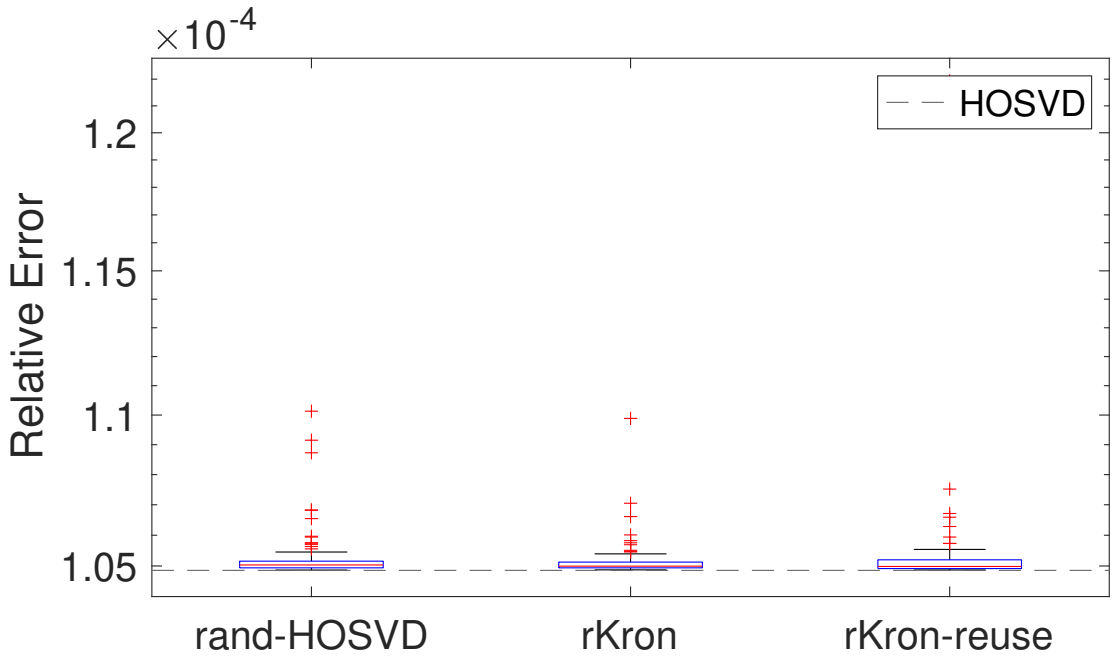
- Uses SRHT random matrices that can be represented as a Kronecker product themselves
- Allows for independent product of random matrices per mode, or reuse of same product of random matrices
- Pessimistic compared to accuracy shown in numerical results

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

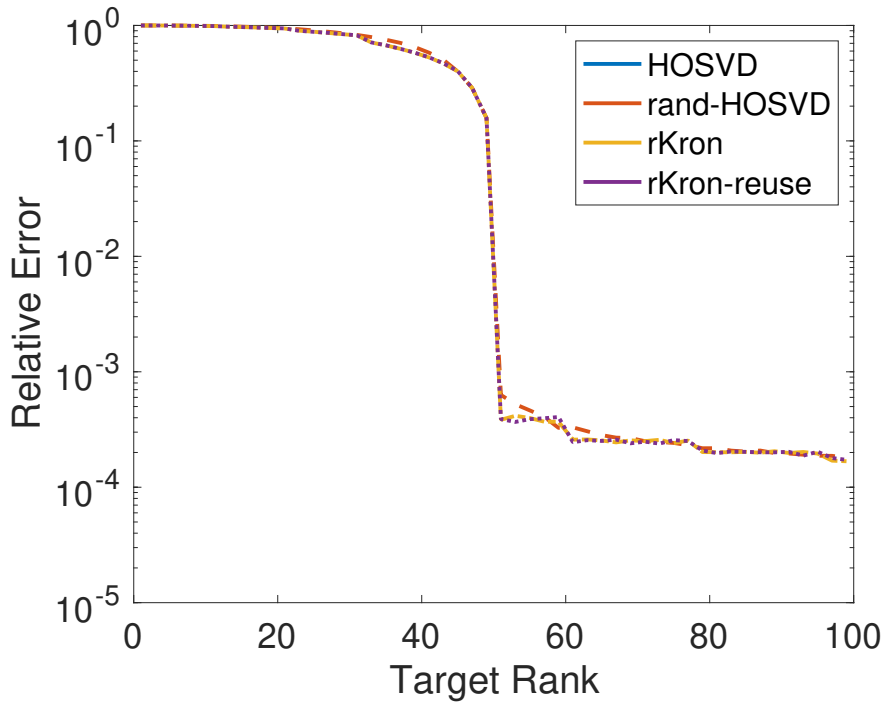
Relative Error over 100 trials



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

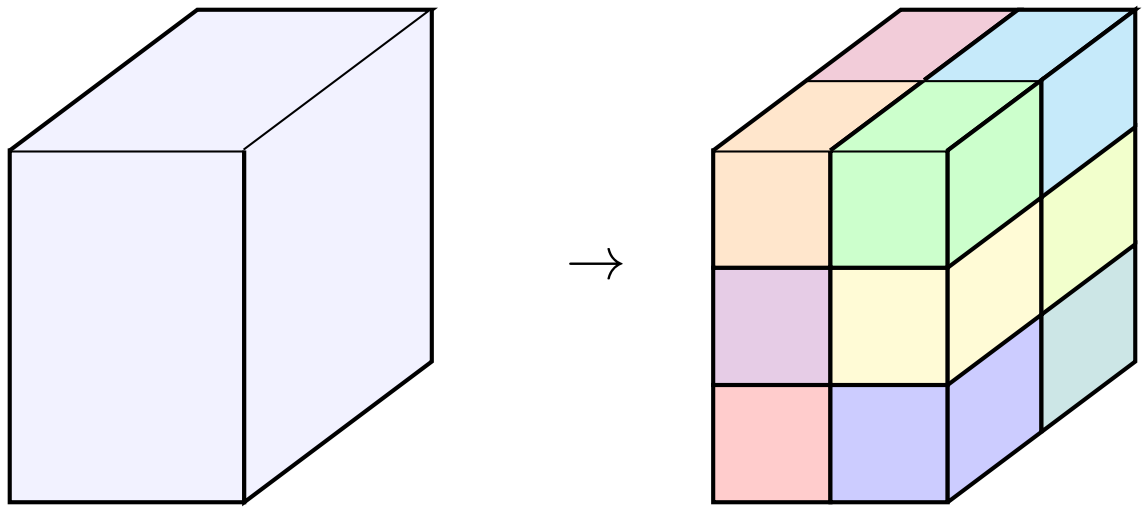
Relative Error with increasing rank



Idea for parallelization

- Store tensor on processor grid
- Parallelize key operations so communication between processors is small as well as cost of computation
- Key operation for our algorithm: multi-TTM

Example: $3 \times 2 \times 2$ processor grid



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 $X_{(j)} \approx A_j A_j^\top X_{(j)}$

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
   a. Deterministic HOSVD on $\mathcal{G}$, combine factor matrices with A_j's

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

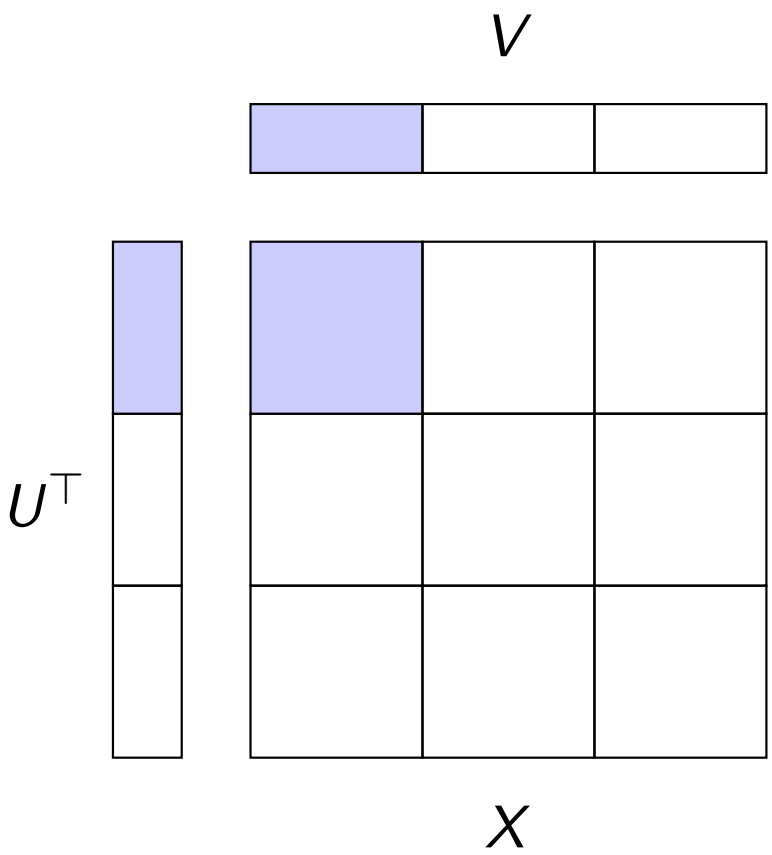
Two approaches based on communication: in sequence and all-at-once

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

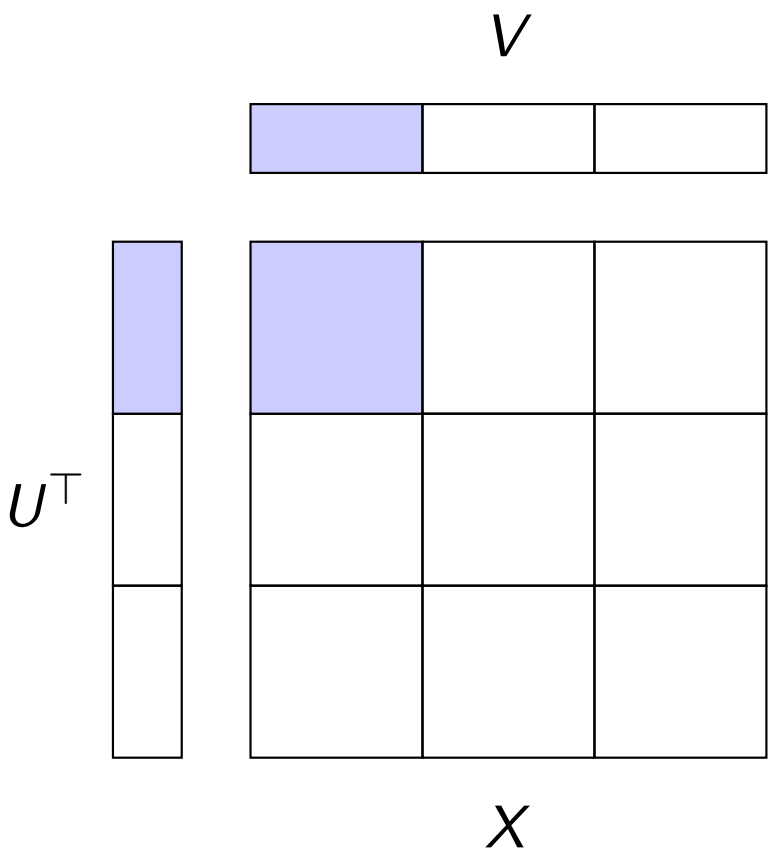
[4] Ballard, Klinvex, Kolda, ACM TOMS, 2020

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

Comparison: multi-TTM

In-sequence:

- fewer flops, more communication
- better choice when matrices are fat
- In randomized HOSVD algorithm, use for core multi-TTM $\mathcal{G} = \mathcal{X} \times_1 A_1^\top \times \cdots \times_d A_d^\top$
- factor matrices A_j have more $(r + p)$ columns

All-at-once:

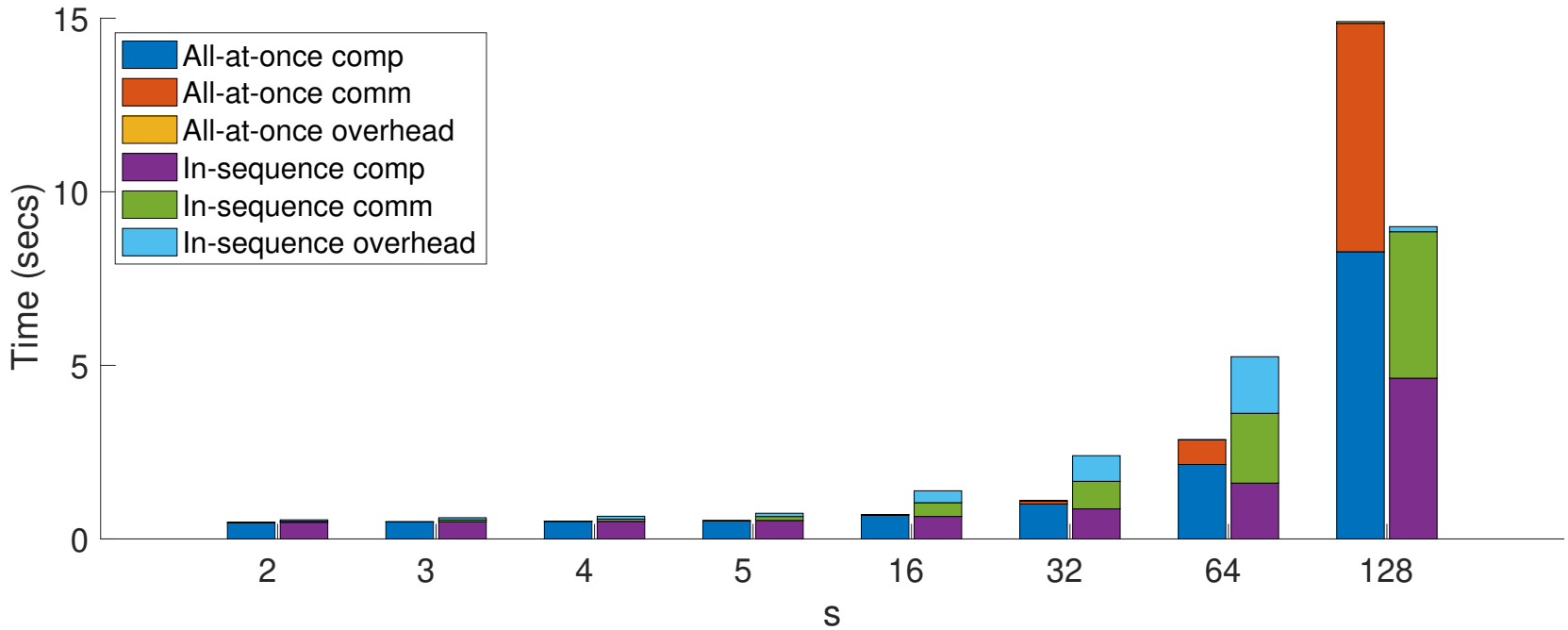
- slightly more flops, generally less communication
- better choice when matrices are skinny
- In randomized HOSVD algorithm, use to compute sketch $\mathcal{Y} = \mathcal{X} \times_2 \Phi_2^\top \times \cdots \times_d \Phi_d^\top$
- random matrices are very skinny (s columns)

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:

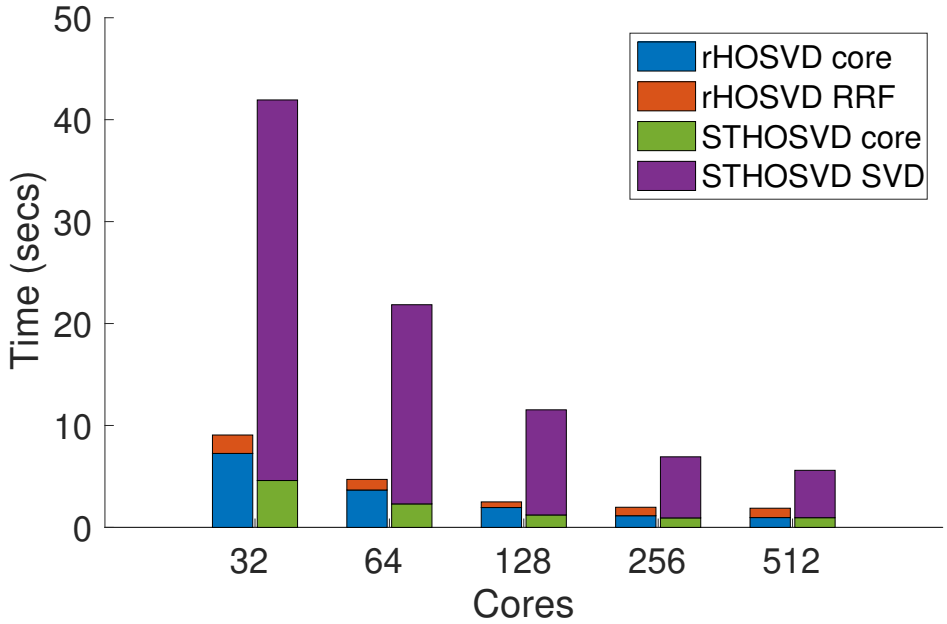


Numerical Results: Parallel Runtime

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:



Takeaways

Main idea:

Exploiting Kronecker structure and parallelizing key operations of typical randomized HOSVD further reduces the cost of computing Tucker decompositions, and allows for use on distributed systems.

Summary of approach:

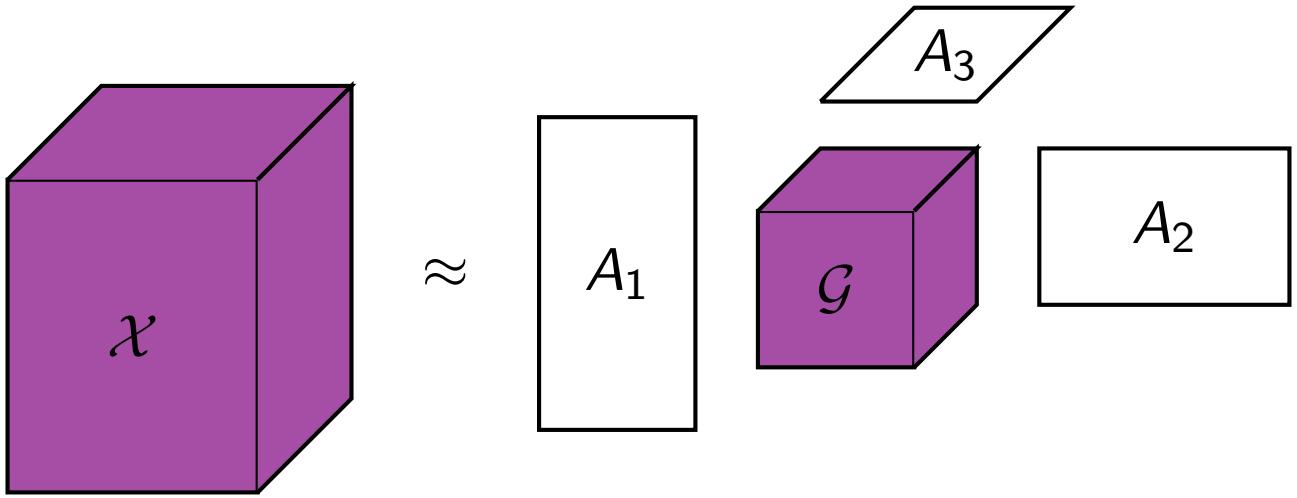
- Use a Kronecker product of random matrices to exploit structure and employ multi-TTM instead of large matrix multiply
- Different versions: re-using or constructing independent Kronecker products
- New method for computing a multi-TTM in parallel
  - An all-at-once approach that can communicate less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

Outline

1. Randomization in NLA
   - Background on randomized matrix techniques
2. Tensors
   - Intro to tensors, tensor operations
   - Tucker decompositions
3. Randomized algorithms for Tucker decompositions
   - Algorithms and analysis
4. Improvements on randomized algorithms for Tucker decompositions
   - Kronecker-structured random matrices
   - Parallel implementation
5. **Structure-preserving Tucker decomposition and algorithms** (Work with Arvind Saibaba and Misha Kilmer)
   - Variation on the Tucker decomposition with strong benefits

Structure Preserving Decomposition

We propose the decomposition



where $\mathcal{G}$ preserves the structure of $\mathcal{X}$ (e.g., sparsity, positivity) and the columns of A_j are close to orthonormal

and two algorithms:

- Structure-Preserving Higher Order SVD (SP-HOSVD)
- Structure-Preserving Sequentially Truncated Higher Order SVD (SP-STHOSVD)

Idea (for matrices)

For a matrix X, randomized range finder gives

$$X \approx QQ^\top X$$

For other decompositions,

$$X \approx QQ^\top X$$

factor matrix dense even if X is sparse

[1] Gu, Ming, Eisenstat, SIAM Journal on Scientific Computing, 1996

Idea (for matrices)

For a matrix X, randomized range finder gives

$$X \approx QQ^\top X$$

For other decompositions,

$$X \approx QQ^\top X$$

factor matrix dense even if X is sparse

Instead, use subset selection (like strong rank-revealing QR[1]) on $Q^\top$ to obtain selection operator P

$$X \approx Q(P^\top Q)^{-1}P^\top X$$

factor matrix contains entries of X

[1] Gu, Ming, Eisenstat, SIAM Journal on Scientific Computing, 1996

SP-STHOSVD Algorithm

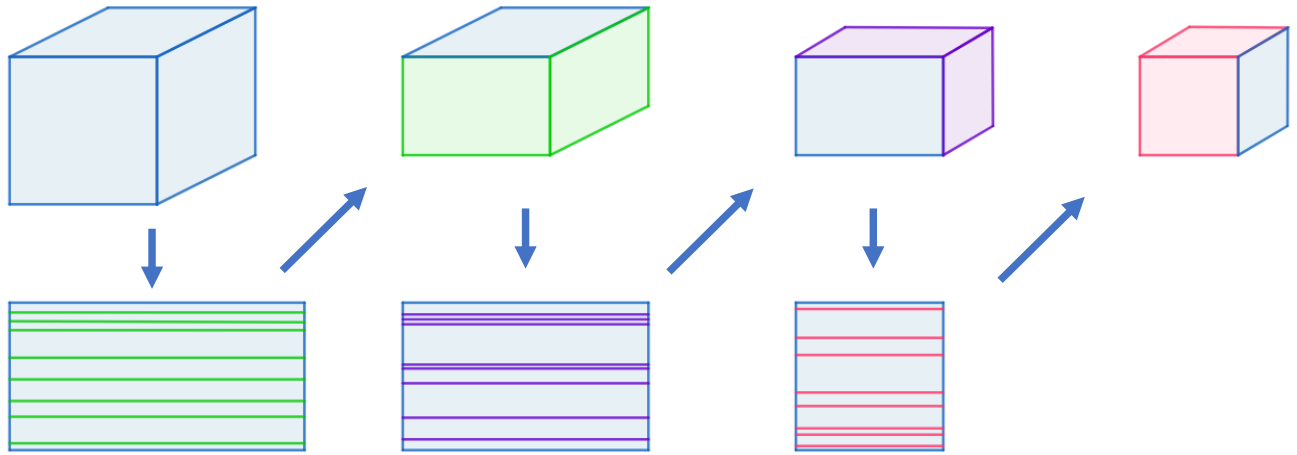
Main Steps

For current mode,

1. **Randomized range finder of unfolding**
2. Extract rows using sRRQR (strong rank revealing QR)
3. Update core tensor

Details

- Form product $Y = G_{(j)}\Omega_j$, and compute thin QR $Y = QR$
- Use sRRQR on $Q_j^\top$ to choose P_j
- Factor matrix is $A_j = Q_j(P_j^\top Q_j)^{-1}$
- New core tensor is $P_j^\top G_{(j)}$



SP-STHOSVD Algorithm

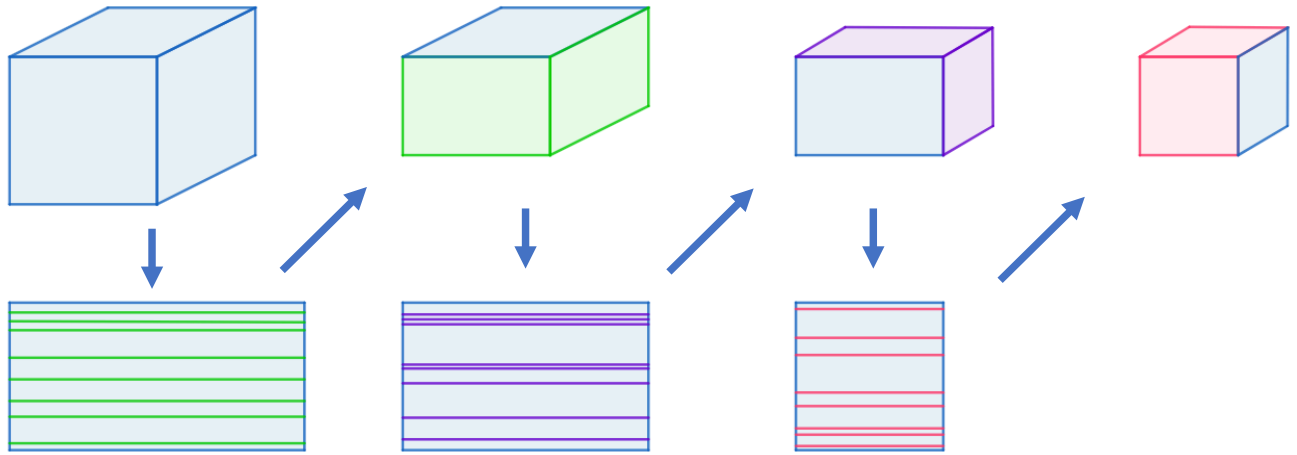
Main Steps

For current mode,

1. Randomized range finder of unfolding
2. **Extract rows using sRRQR (strong rank revealing QR)**
3. Update core tensor

Details

- Form product $Y = G_{(j)}\Omega_j$, and compute thin QR $Y = QR$
- Use sRRQR on $Q_j^\top$ to choose P_j
- Factor matrix is $A_j = Q_j(P_j^\top Q_j)^{-1}$
- New core tensor is $P_j^\top G_{(j)}$



SP-STHOSVD Algorithm

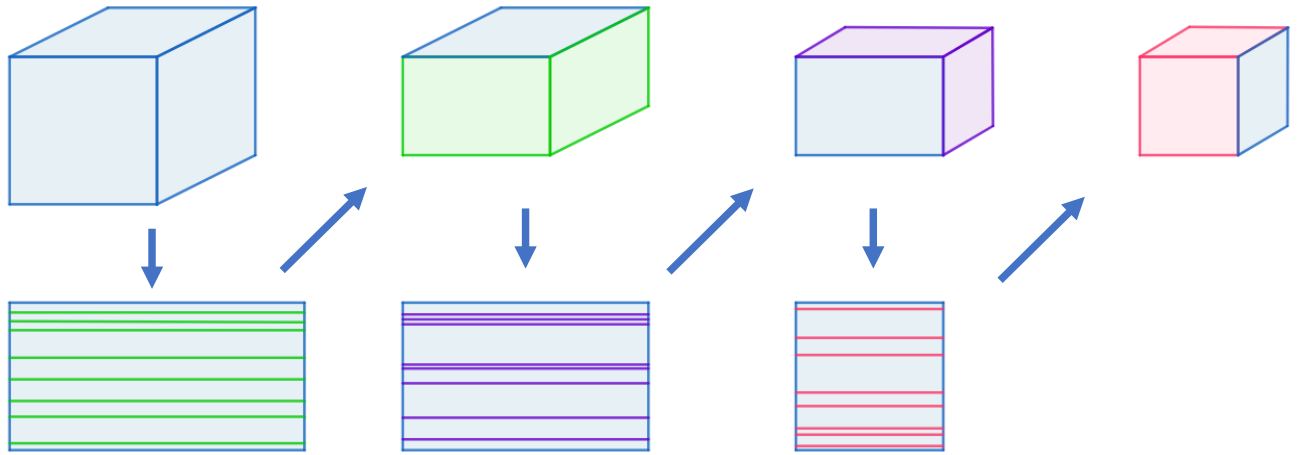
Main Steps

For current mode,

1. Randomized range finder of unfolding
2. Extract rows using sRRQR (strong rank revealing QR)
3. **Update core tensor**

Details

- Form product $Y = G_{(j)}\Omega_j$, and compute thin QR $Y = QR$
- Use sRRQR on $Q_j^\top$ to choose P_j
- Factor matrix is $A_j = Q_j(P_j^\top Q_j)^{-1}$
- New core tensor is $P_j^\top G_{(j)}$



Computational Cost

For d-mode tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times \cdots \times n}$
target rank $(r, r, \ldots, r)$
processing order ρ
and $\ell = r + p$

Cost of SP-STHOSVD

$$\mathcal{O}\left(dn\ell^2 + \sum_{j=1}^{d} \text{nnz}(\mathcal{G}^{(\rho_j)})\ell\right)$$

Computational Cost

For d-mode tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times \cdots \times n}$
target rank $(r, r, \ldots, r)$
processing order ρ
and $\ell = r + p$

Cost of SP-STHOSVD

$$\mathcal{O}\left(dn\ell^2 + \sum_{j=1}^{d} \text{nnz}(\mathcal{G}^{(\rho_j)})\ell \right)$$

Dominant costs per mode

- Using sRRQR and thin QR factorization
- Forming product of unfolded tensor and random matrix

Computational Cost

For d-mode tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times \cdots \times n}$
target rank $(r, r, \ldots, r)$
processing order ρ
and $\ell = r + p$

Cost of SP-STHOSVD

$$\mathcal{O}\left(dn\ell^2 + \sum_{j=1}^{d} \text{nnz}(\mathcal{G}^{(\rho_j)})\ell \right)$$

Dominant costs per mode

- Using sRRQR and thin QR factorization
- Forming product of unfolded tensor and random matrix

Error Analysis

Assumptions:

- $\widehat{\mathcal{X}} = [\mathcal{G}; A_1, \ldots, A_d]$ the SP-STHOSVD of $\mathcal{X} \in \mathbb{R}^{n \times n \times \cdots \times n}$
- Target rank $r = (r, r, \ldots, r)$
- Oversampling parameter $p \geq 2$ such that $r + p < n$
- Processing order $\rho = [1, 2, \ldots, d]$
- $\widehat{\mathcal{X}}_{\text{opt}}$ the optimal rank-$(r, r, \ldots, r)$ approximation to $\mathcal{X}$
- $g(n, r) = \sqrt{1 + 4r(n - r)}$, $f_p(r) = \sqrt{1 + \frac{r}{p-1}}$, and $\ell = r + p$

Theorem (M, Saibaba, Kilmer)

$$\mathbb{E}_{\{\Omega_k\}_{k=1}^d} \|\mathcal{X} - \widehat{\mathcal{X}}\|_F \leq f_p(r) \left(\sum_{j=1}^{d} (g(n, \ell))^j \right) \|\mathcal{X} - \widehat{\mathcal{X}}_{opt}\|_F$$

Results: Large Sparse Datasets

NELL-2: a portion of the Never Ending Language Learning knowledge base from the "Read the Web" project at Carnegie Mellon University

Enron: word counts in emails released during an investigation by the Federal Energy Regulatory Commission

Original Tensor	Order	Size	Nonzeros
NELL-2	3	$12092 \times 9184 \times 28818$	$76,879,419$
Enron	4	$6066 \times 5699 \times 244268 \times 1176$	$54,202,099$

Condensed Tensor	Order	Size	Nonzeros
NELL-2	3	$807 \times 613 \times 1922$	$19,841$
Enron	3	$405 \times 380 \times 9771$	$6,131$

FROSTT (Formidable Repository of Open Sparse Tensors and Tools), frostt.io

Results: Relative Error and Runtime

NELL-2:

Target Rank	Relative Error		Runtime	
	SP-ST	R-ST	SP-ST	R-ST
30	0.2968	0.1319	0.5690	17.0642
90	0.1950	0.0699	1.5889	23.3303
150	0.1431	0.0478	2.1310	33.6867
210	0.1181	0.0367	3.0832	45.5227

Enron:

Target Rank	Relative Error		Runtime	
	SP-ST	R-ST	SP-ST	R-ST
20	0.6015	0.2081	0.4086	31.5615
70	0.3548	0.0870	1.3276	36.6431
120	0.1503	0.0458	2.8175	39.7169
170	0.0756	0.0239	6.2158	45.8429

SP-ST: Structure-preserving STHOSVD, R-ST: Randomized STHOSVD

Takeaways

Main idea:

Preserving structure in the core tensor allows for computations with extremely large, sparse data that is infeasible for standard (even randomized) algorithms for tensor decompositions.

Summary:

- Combined randomized range finder and subset selection to extract entries from the tensor itself to form the core so any structure is preserved
- Particularly useful for large, sparse data

Overall Conclusions

Main idea:

Using randomization for tensor decompositions improves efficiency without losing much accuracy, particularly when additional structure is exploited

Shown through three main projects:

- Randomized algorithms/analysis for Tucker decompositions[1]
- Improvements using Kronecker-structured random matrices and parallel implementation (preprint coming soon!)
- Structure-preserving Tucker decomposition and algorithms/analysis[1]

[1] Minster, Saibaba, Kilmer, SIMODS, 2020