Randomized Algorithms for Rounding the Tensor Train Format

Grey Ballard

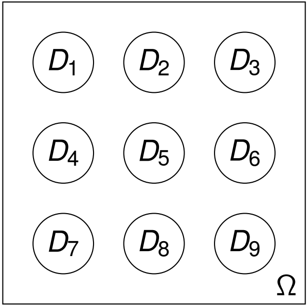
joint with Hussam Al Daas, Paul Cazeaux, Eric Hallman, Agnieszka Miedlar, Mirjeta Pasha, Tim Reid, and Arvind Saibaba

SIAM Applied Linear Algebra

May 21, 2021

WAKE FOREST UNIVERSITY

Tensor Train (TT) can make very high dimensional problems tractable



Consider the parameter-dependent PDE:

$$\begin{aligned} -\mathrm{div}(\sigma(x,y;\boldsymbol{\rho})\nabla(u(x,y;\boldsymbol{\rho}))) &= f(x,y) && \text{in } \Omega, \\ u(x,y;\boldsymbol{\rho}) &= 0 && \text{on } \partial\Omega, \end{aligned}$$

where σ is defined as:

$$\sigma(x,y;\boldsymbol{\rho}) = \begin{cases} 1+\rho_i & \text{if } (x,y) \in D_i \\ 1 & \text{elsewhere} \end{cases}$$

known as *cookies* problem [Tob12]

- Solving for all parameter values simultaneously, u is 11-D
- With mild assumptions, solution u has low TT ranks
- TT-based iterative linear solver exploits low-rank structure
  - can solve problem for high spatial and parameter resolution

TT Rounding

Given a tensor in TT format, often need to compress the ranks

- algebraic operations on TT formats over-extend ranks
- recompression (rank truncation) subject to error threshold
  - or subject to target ranks
- analogous to floating point rounding

TT Rounding

Given a tensor in TT format, often need to compress the ranks

- algebraic operations on TT formats over-extend ranks
- recompression (rank truncation) subject to error threshold
  - or subject to target ranks
- analogous to floating point rounding

Low-rank matrix addition example

- consider $\mathbf{A}_1\mathbf{B}_1^T + \mathbf{A}_2\mathbf{B}_2^T$, where each factor has r columns
- can represent this in low-rank format $\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix}^T$ which now has rank $2r$
- goal is to compute low-rank approximation with rank $k < 2r$

Randomized Low-Rank Matrix Approximation [HMT11]

Randomized algorithm for low-rank approximation of matrix $\mathbf{X}$:

function $[\mathbf{U}, \mathbf{V}] = \textsc{Rand-Low-Rank}(\mathbf{X})$

$\mathbf{Y} = \mathbf{X}\Omega$ ▷ Ω is random matrix with k columns

$[\mathbf{U}, \sim] = \mathrm{QR}(\mathbf{Y})$ ▷ (tall-skinny) QR decomposition

$\mathbf{V}^T = \mathbf{U}^T\mathbf{X}$ ▷ $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$

Randomized Low-Rank Matrix Approximation [HMT11]

Randomized algorithm for low-rank approximation of matrix $\mathbf{X}$:

function $[\mathbf{U},\mathbf{V}] = \textsc{Rand-Low-Rank}(\mathbf{X})$
- $\mathbf{Y} = \mathbf{X}\boldsymbol{\Omega}$ $\quad\triangleright$ $\boldsymbol{\Omega}$ is random matrix with k columns
- $[\mathbf{U},\sim] = \mathrm{QR}(\mathbf{Y})$ $\quad\triangleright$ (tall-skinny) QR decomposition
- $\mathbf{V}^T = \mathbf{U}^T\mathbf{X}$ $\quad\triangleright$ $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$

Same algorithm tailored for rank-r matrix $\mathbf{X} = \mathbf{A}\mathbf{B}^T$ $(r > k)$:

function $[\mathbf{U},\mathbf{V}] = \textsc{Rand-Rounding}(\mathbf{A},\mathbf{B})$
- $\mathbf{Y} = \mathbf{A}(\mathbf{B}^T\boldsymbol{\Omega})$ $\quad\triangleright$ $\boldsymbol{\Omega}$ is random matrix with k columns
- $[\mathbf{U},\sim] = \mathrm{QR}(\mathbf{Y})$ $\quad\triangleright$ (tall-skinny) QR decomposition
- $\mathbf{V}^T = (\mathbf{U}^T\mathbf{A})\mathbf{B}^T$ $\quad\triangleright$ $\mathbf{A}\mathbf{B}^T \approx \mathbf{U}\mathbf{V}^T$

Matrix Rounding

Randomized algorithm for rank-r matrix $\mathbf{X} = \mathbf{A}\mathbf{B}^T$:

function $[\mathbf{U}, \mathbf{V}] = \textsc{Rand-Rounding}(\mathbf{A}, \mathbf{B})$

$\mathbf{Y} = \mathbf{A}(\mathbf{B}^T\boldsymbol{\Omega})$ ▷ $\boldsymbol{\Omega}$ is random matrix with k columns

$[\mathbf{U}, \sim] = \text{QR}(\mathbf{Y})$ ▷ (tall-skinny) QR decomposition

$\mathbf{V}^T = (\mathbf{U}^T\mathbf{A})\mathbf{B}^T$ ▷ $\mathbf{A}\mathbf{B}^T \approx \mathbf{U}\mathbf{V}^T$

Deterministic algorithm (from Hussam's talk):

function $[\mathbf{U}, \mathbf{V}] = \textsc{Det-Rounding}(\mathbf{A}, \mathbf{B})$

$[\mathbf{Q}_A, \mathbf{R}_A] = \text{QR}(\mathbf{A})$ ▷ (tall-skinny) QR decomposition

$[\mathbf{Q}_B, \mathbf{R}_B] = \text{QR}(\mathbf{B})$ ▷ (tall-skinny) QR decomposition

$[\hat{\mathbf{U}}_R, \hat{\Sigma}_R, \hat{\mathbf{V}}_R] = \text{tSVD}(\mathbf{R}_A\mathbf{R}_B^T, k)$ ▷ kth truncated SVD

$\mathbf{U} = \mathbf{Q}_A\hat{\mathbf{U}}_R$

$\mathbf{V} = \mathbf{Q}_B(\hat{\mathbf{V}}_R\hat{\Sigma}_R)$ ▷ $\mathbf{A}\mathbf{B}^T \approx \mathbf{U}\mathbf{V}^T$

Benefits of randomization for matrix rounding

1. if $\mathbf{A}$ is $m \times r$, $\mathbf{B}$ is $n \times r$, and they are rounded to rank k, reduces computation from $O((m+n)r^2)$ to $O((m+n)rk)$

2. shifts some computational burden from QR to matrix multiplication, which often has higher performance

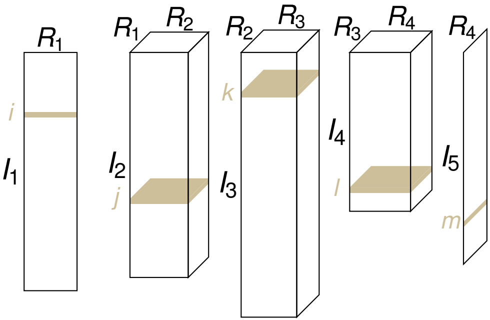
3. opens possibility of choosing Ω for faster multiplication

4. enables cheaper rounding of sums of s low-rank matrices:

$$\mathbf{A}_1\mathbf{B}_1^T + \mathbf{A}_2\mathbf{B}_2^T + \cdots + \mathbf{A}_s\mathbf{B}_s^T$$

   - sketch of the sum is the sum of the sketches
   - cost of randomized algorithm is linear rather than quadratic in s

Tensor Train (TT) Notation



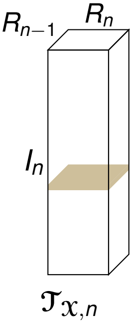
$\mathfrak{X} \approx \{\mathfrak{T}_{X,n}\}, \mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$

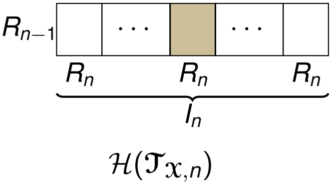
$\mathfrak{T}_{X,n} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are *TT cores*

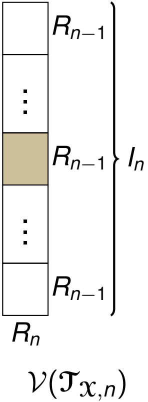
$$x_{ijklm} \approx \sum_{\alpha=1}^{R_1} \sum_{\beta=1}^{R_2} \sum_{\gamma=1}^{R_3} \sum_{\delta=1}^{R_4} \mathfrak{T}_{X,1}(i,\alpha)\mathfrak{T}_{X,2}(\alpha,j,\beta)\mathfrak{T}_{X,3}(\beta,k,\gamma)\mathfrak{T}_{X,4}(\gamma,l,\delta)\mathfrak{T}_{X,5}(\delta,m)$$

Important core unfoldings

$\mathcal{H}(\mathfrak{T}_{\mathcal{X},n}) \in \mathbb{R}^{R_{n-1} \times I_n R_n}$ and $\mathcal{V}(\mathfrak{T}_{\mathcal{X},n}) \in \mathbb{R}^{R_{n-1} I_n \times R_n}$
are horizontal and vertical unfoldings of nth core







Deterministic TT Rounding Algorithm [Ose11]

function $\{\mathcal{T}_{\mathcal{Z},n}\} = \text{TT-ROUNDING}(\{\mathcal{T}_{\mathcal{X},n}\})$

$\triangleright$ Orthogonalization Phase

for $n = N$ down to 2 **do**

$[\mathbf{Y}_n, \mathbf{R}_n] = \text{QR}(\mathcal{H}(\mathcal{T}_{\mathcal{X},n})^T)$ $\triangleright$ (tall-skinny) QR factorization

$\mathcal{V}(\mathcal{T}_{\mathcal{X},n-1}) = \mathcal{V}(\mathcal{T}_{\mathcal{X},n-1}) \cdot \mathbf{R}^{\mathsf{T}}$ $\triangleright$ Apply **R** to previous core

$\triangleright$ Truncation Phase

$\mathcal{Z} = \mathcal{X}$

for $n = 1$ to $N-1$ **do**

$[\mathbf{Y}_n, \mathbf{R}_n] = \text{QR}(\mathcal{V}(\mathcal{T}_{\mathcal{Z},n}))$ $\triangleright$ (tall-skinny) QR factorization

$[\hat{\mathbf{U}}_R, \hat{\Sigma}, \hat{\mathbf{V}}] \approx \text{TSVD}(\mathbf{R}_n)$ $\triangleright$ Truncated SVD of **R**

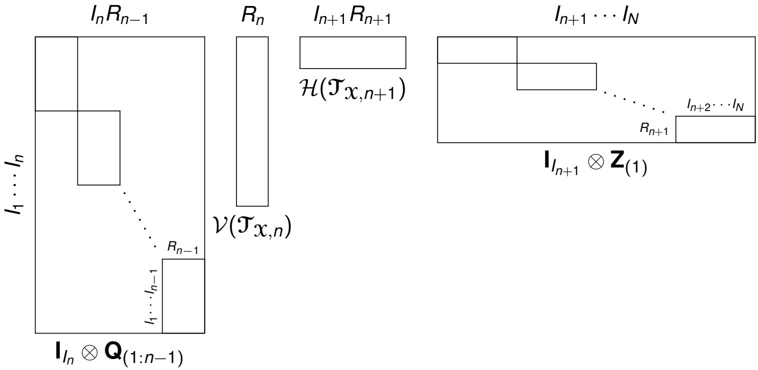
$\mathcal{V}(\mathcal{T}_{\mathcal{Z},n}) = \text{APPLY-Q}(\mathbf{Y}_n, \hat{\mathbf{U}}_R)$ $\triangleright$ Form explicit $\hat{\mathbf{U}}$

$\mathcal{H}(\mathcal{T}_{\mathcal{Z},n+1})^T = \text{APPLY-Q}(\mathbf{Y}_{n+1}, \hat{\mathbf{V}}\hat{\Sigma})$ $\triangleright$ Apply $\hat{\Sigma}\hat{\mathbf{V}}^{\mathsf{T}}$ to next core

More details on TT rounding...

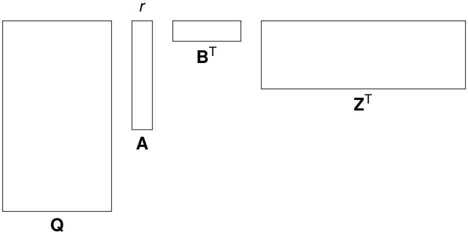
TT rounding does truncated SVDs on $\mathbf{X}_{(1)}$, $\mathbf{X}_{(1:2)}$, $\mathbf{X}_{(1:3)}$, etc., and we have matrix expressions of those unfoldings [ADBB20]

$$\mathbf{X}_{(1:n)} = (\mathbf{I}_{I_n} \otimes \mathbf{Q}_{(1:n-1)}) \cdot \mathcal{V}(\mathfrak{T}_{\mathcal{X},n}) \cdot \mathcal{H}(\mathfrak{T}_{\mathcal{X},n+1}) \cdot (\mathbf{I}_{I_{n+1}} \otimes \mathbf{Z}_{(1)})$$



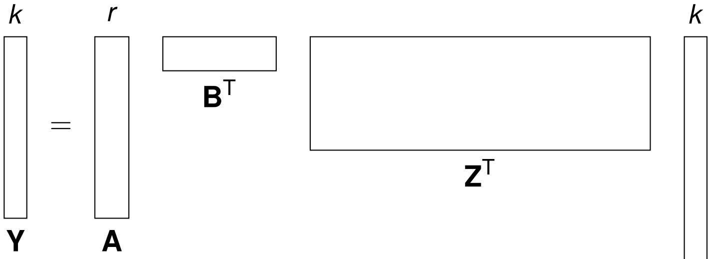
Opportunities for randomized low-rank approximation

Suppose you want to compute the (randomized) truncated SVD of a rank-r matrix $\mathbf{QAB}^\mathsf{T}\mathbf{Z}^\mathsf{T}$, where $\mathbf{A}$ and $\mathbf{B}$ are tall and skinny with r columns, and

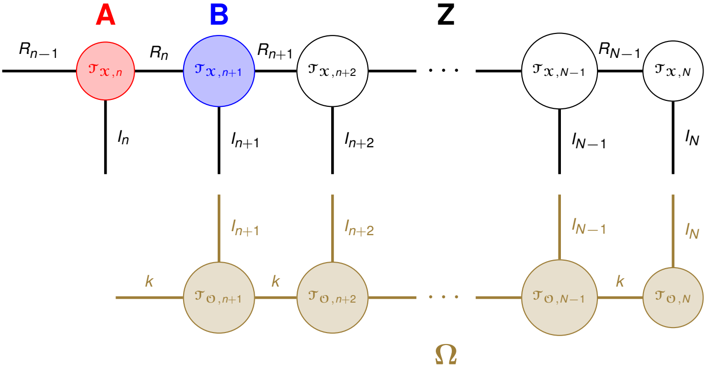


1. $\mathbf{Q}$, $\mathbf{B}$, and $\mathbf{Z}$ are all column orthonormal
   - use RAND-LOW-RANK($\mathbf{A}$)
2. $\mathbf{Q}$ is column orthonormal but $\mathbf{B}$ and $\mathbf{Z}$ are not
   - use RAND-ROUNDING($\mathbf{A}, \mathbf{ZB}$)
3. none of $\mathbf{Q}$, $\mathbf{B}$, and $\mathbf{Z}$ is column orthonormal
   - use GEN-NYSTRÖM($\mathbf{QAB}^\mathsf{T}\mathbf{Z}^\mathsf{T}$) [Nak20]

Most important computation: $\mathbf{Y} = \mathbf{A}\mathbf{B}^T\mathbf{Z}^T\Omega$



- $\mathbf{Z}$ is structured, represents trailing TT cores
- we choose structured Ω to exploit TT format



Tensor network diagram: vertices represent tensors, edges represent modes, connected edges represent contractions

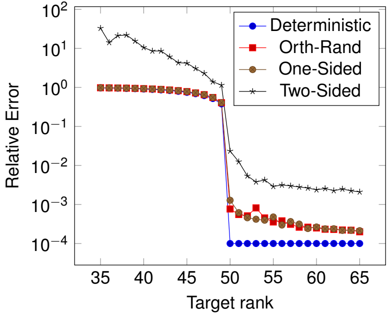
Summary of Algorithms

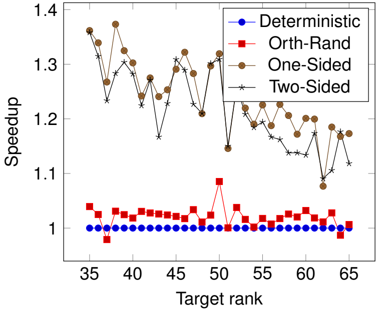
- Deterministic
  - uses orthogonalization phase and truncation phase
- Orthogonalize-then-Randomize (Orth-Rand)
  - uses same orthogonalization phase and randomizes truncation phase with Gaussian projection
  - can use adaptive range-finder algorithm
- Randomize-then-Orthogonalize (One-Sided)
  - avoids TT orthogonalization, uses TT-structured random projection
  - can exploit linearity in sums of TT tensors
- Generalized Nyström (Two-Sided)
  - avoids orthogonalization phase and uses TT-structured random projection on left and right

We round a synthetic TT tensor:

$$\mathcal{Y} = \mathcal{X} + 10^{-4} \cdot \mathcal{N},$$

where $\|\mathcal{X}\| = \|\mathcal{N}\| = 1$ and each has 10 modes of dimension 1000 with all TT ranks equal to 50, using target ranks between 35 and 65

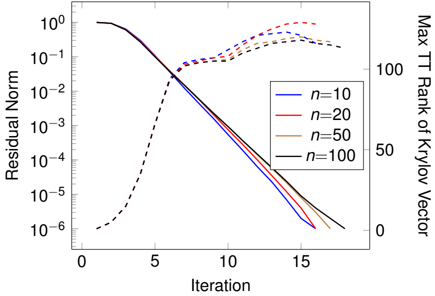


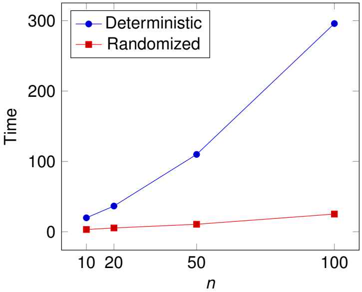


Experimental results for cookies problem

We solve the cookies problem with 4 cookies using a tensor with dimension $1781 \times n \times n \times n \times n$ using TT-GMRES

- bottleneck is rounding sum of TT tensors
- deterministic algorithm forms the sum and rounds via orthogonalization
- randomized One-Sided algorithm exploits the sum of sketches, gets same answers





Summary

- TT rounding is key operation for TT arithmetic
  - efficient deterministic algorithms exploit low-rank structure
- Randomized algorithms can reduce arithmetic cost and maintain sufficient accuracy
  - benefits depend on ratio of two low ranks
- Randomized approaches yield more benefits for higher-level problems, like rounding sums of TT tensors
  - bottleneck within Krylov solver that exploits TT structure

Thanks for your attention!

ballard@wfu.edu

References I

Hussam Al Daas, Grey Ballard, and Peter Benner.
Parallel algorithms for tensor train arithmetic.
Technical Report 2011.06532, arXiv, 2020.

N. Halko, P. Martinsson, and J. Tropp.
Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Review, 53(2):217–288, 2011.

Yuji Nakatsukasa.
Fast and stable randomized low-rank matrix approximation.
Technical Report 2009.11392, arXiv, 2020.

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

References II

Christine Tobler.

Low-rank Tensor Methods for Linear Systems and Eigenvalue Problems.

PhD thesis, ETH Zurich, 2012.

Motivation: what if you have to solve many PDEs?

A single PDE simulation can already create a ton of data...
what if we have design/uncertain parameters?

Suppose you have 10 parameters, each with 10 possible values

- now you have to run your simulation 10^{10} times...
- and store all this data...

If the resulting data could be compressed, why not compute the compressed representation from the start?

More details on TT rounding...

TT rounding does SVDs on $\mathbf{X}_{(1)}$, $\mathbf{X}_{(1:2)}$, $\mathbf{X}_{(1:3)}$, etc.,
so we seek similar matrix expressions of those unfoldings

The unfolding of $\mathcal{X}$ that maps the first n tensor dimensions to rows can be expressed as a product of four matrices:

$$\mathbf{X}_{(1:n)} = (\mathbf{I}_{I_n} \otimes \mathbf{Q}_{(1:n-1)}) \cdot \mathcal{V}(\mathcal{T}_{\mathcal{X},n}) \cdot \mathcal{H}(\mathcal{T}_{\mathcal{X},n+1}) \cdot (\mathbf{I}_{I_{n+1}} \otimes \mathbf{Z}_{(1)})$$

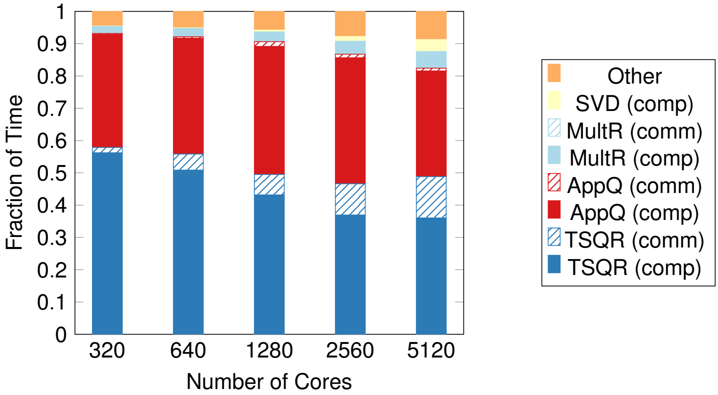
where $\mathcal{Q}$ is $I_1 \times \cdots \times I_{n-1} \times R_{n-1}$ with

$$\mathcal{Q}(i_1, \ldots, i_{n-1}, r_{n-1}) = \mathcal{T}_{\mathcal{X},1}(i_1, :) \cdot \mathcal{T}_{\mathcal{X},2}(:, i_2, :) \cdots \mathcal{T}_{\mathcal{X},n-1}(:, i_{n-1}, r_{n-1}),$$

and $\mathcal{Z}$ is $R_{n+1} \times I_{n+2} \times \cdots \times I_N$ with

$$\mathcal{Z}(r_{n+1}, i_{n+2}, \ldots, i_N) = \mathcal{T}_{\mathcal{X},n+2}(r_{n+1}, i_{n+2}, :) \cdot \mathcal{T}_{\mathcal{X},n+3}(:, i_{n+3}, :) \cdots \mathcal{T}_{\mathcal{X},N}(:, i_N).$$

Time breakdown of (parallel) TT rounding



- TT tensor: $I_n = 512K$, $R_n = 60 \rightarrow 30$, $N = 50$
- 70-80% of time spent in QR computations