Parallel Tensor Train Rounding using Gram SVD

Hussam Al Daas\textsuperscript{1}, \textbf{Grey Ballard}\textsuperscript{2}, and Lawton Manning\textsuperscript{2}

\textsuperscript{1} Rutherford Appleton Laboratories
\textsuperscript{2} Wake Forest University

SIAM Parallel Processing
February 26, 2022
Tensor Train (TT) can make very high dimensional problems tractable

Consider the parameter-dependent PDE:
\[-\text{div}(\sigma(x, y; \rho) \nabla(u(x, y; \rho))) = f(x, y) \quad \text{in } \Omega,\]
\[u(x, y; \rho) = 0 \quad \text{on } \partial \Omega,\]

where \(\sigma\) is defined as:
\[
\sigma(x, y; \rho) = \begin{cases} 
1 + \rho_i & \text{if } (x, y) \in D_i \\
1 & \text{elsewhere}
\end{cases}
\]

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 resolution [Dol13]
Tensor Train (TT) Notation

\[ \mathbf{X} \approx \{ \mathbf{T}_{\mathbf{x},n} \}, \quad \mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5} \]

\[ \mathbf{T}_{\mathbf{x},n} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n} \]

are TT cores

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

Goal: compute truncated SVDs of matricized TT-format tensors

- TT-ranks are ranks of unfoldings $X_{(1:n)}$ for $1 \leq n \leq N$
- if $X$ is $l_1 \times l_2 \times \cdots \times l_N$, then $X_{(1:n)}$ is $(l_1 \cdots l_n) \times (l_{n+1} \cdots l_N)$
- $X_{(1:n)}$ is generally a huge matrix, but it is highly structured
Low-rank matrix addition example

- Consider $A_1 B_1^T + A_2 B_2^T$, where each factor has $r$ columns.
- Can represent this in low-rank format $[A_1 \ A_2] [B_1 \ B_2]^T$ which now has rank $2r$.
- Goal is to compute low-rank approximation with rank $k < 2r$. 

\[
\begin{align*}
\begin{array}{c}
\begin{array}{c}
A_1
\end{array} \\
\begin{array}{c}
B_1^T
\end{array}
\end{array}
& + \\
\begin{array}{c}
\begin{array}{c}
A_2
\end{array} \\
\begin{array}{c}
B_2^T
\end{array}
\end{array}
& = \\
\begin{array}{c}
\begin{array}{c}
A
\end{array} \\
\begin{array}{c}
B^T
\end{array}
\end{array}
\end{align*}
\]
Matrix Rounding via QR Decompositions

QR-based algorithm for rounding rank-\(r\) matrix \(X = AB^T\):

```latex
\begin{align*}
\text{function } [U, V] &= \text{QR-ROUNDING}(A, B, k) \\
[Q_A, R_A] &= \text{QR}(A) \quad \triangleright\text{ (tall-skinny) QR decomposition} \\
[Q_B, R_B] &= \text{QR}(B) \quad \triangleright\text{ (tall-skinny) QR decomposition} \\
[\hat{U}_R, \hat{\Sigma}_R, \hat{V}_R] &= \text{TSVD}(R_A R_B^T, k) \quad \triangleright k\text{th truncated SVD} \\
U &= Q_A \hat{U}_R \\
V &= Q_B(\hat{V}_R \hat{\Sigma}_R) \\
\end{align*}
```

Here’s the algebra:

\[
AB^T = \underbrace{Q_A R_A}_{A} \underbrace{R_B^T Q_B^T}_{B^T} \approx \underbrace{Q_A \hat{U}_R \hat{\Sigma}_R \hat{V}_R^T}_{\text{approximation}} \underbrace{Q_B^T}_{B^T} \underbrace{R_A R_B^T}_{A}
\]

\[
U = Q_A \hat{U}_R \\
V = Q_B(\hat{V}_R \hat{\Sigma}_R) \\
\]
Gram SVD is a well-known method for computing singular values of rectangular matrices

- sing. values of $X$ are square roots of eigenvalues of $X^TX$
- if $X$ is tall and skinny then $X^TX$ is much smaller
- sacrifices some accuracy of singular values and vectors

If $X = AB^T$, then we can try direct application:

- $X^TX = B(A^TA)B^T$
- $XX^T = A(B^TB)A^T$

but we still have to orthogonalize one of the two factors in order to compute the eigendecomposition
Gram-based algorithm for rounding rank-$r$ matrix $X = AB^T$:

**function** $[U, V] = \text{GRAM-ROUNDING}(A, B, k)$

$G_A = A^T A$  \hspace{1cm} $\triangleright$ symmetric matrix multiplication
$G_B = B^T B$  \hspace{1cm} $\triangleright$ symmetric matrix multiplication

$[V_A, \Lambda_A] = \text{EIG}(G_A)$
$[V_B, \Lambda_B] = \text{EIG}(G_B)$

$[\hat{U}, \hat{\Sigma}, \hat{V}] = \text{TSVD}(\Lambda_A^{1/2} V_A^T V_B \Lambda_B^{1/2}, k)$  \hspace{1cm} $\triangleright$ $k$th truncated SVD

$U = A \left( V_A \Lambda_A^{-1/2} \hat{U} \right)$

$V = B \left( V_B \Lambda_B^{-1/2} \hat{V} \hat{\Sigma} \right)$

$AB^T \approx UV^T$
Here’s the algebra:

\[
\begin{align*}
\mathbf{A}\mathbf{B}^T &= \mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^T \mathbf{V}_B\mathbf{\Sigma}_B\mathbf{U}_B^T \\
&= \mathbf{A}\mathbf{V}_A\mathbf{\Lambda}_A^{-1/2} \mathbf{\Lambda}_A^{1/2}\mathbf{V}_A^T \mathbf{V}_B\mathbf{\Lambda}_B^{1/2} \mathbf{\Lambda}_B^{-1/2}\mathbf{V}_B^T \mathbf{B}^T \\
&= (\mathbf{A}\mathbf{V}_A\mathbf{\Lambda}_A^{-1/2})\mathbf{U}_A \mathbf{\Lambda}_A^{1/2}\mathbf{V}_A^T \mathbf{V}_B\mathbf{\Lambda}_B^{1/2} (\mathbf{B}\mathbf{V}_B\mathbf{\Lambda}_B^{-1/2})^T \\
&\approx (\mathbf{A}\mathbf{V}_A\mathbf{\Lambda}_A^{-1/2})\hat{\mathbf{U}} \hat{\mathbf{\Sigma}} \hat{\mathbf{V}}^T (\mathbf{B}\mathbf{V}_B\mathbf{\Lambda}_B^{-1/2})^T \\
&= \mathbf{A}(\mathbf{V}_A\mathbf{\Lambda}_A^{-1/2}\hat{\mathbf{U}}) (\hat{\mathbf{\Sigma}} \mathbf{\Lambda}_B^{-1/2}\mathbf{V}_B^T) \mathbf{B}^T
\end{align*}
\]
Gram-Rounding is at least twice as fast as QR-Rounding
- Gram-Rounding does half the flops of QR-Rounding
- Gram-Rounding dominated by matrix multiplication
- QR-Rounding dominated by computing QR and applying Q

QR-Rounding is more accurate than Gram-Rounding
- QR-Rounding computes singular values as small as $\sigma_1 \cdot \varepsilon$
- Gram-Rounding computes singular values only to $\sigma_1 \cdot \varepsilon^{1/2}$
Tensor Train (TT) Notation

\[ X \approx \{ \mathcal{I}_{x,n} \}, \quad X \in \mathbb{R}^{l_1 \times l_2 \times l_3 \times l_4 \times l_5} \]

\[ \mathcal{I}_{x,n} \in \mathbb{R}^{R_{n-1} \times l_n \times R_n} \]

\[ x_{ijklm} \approx \sum_{\alpha=1}^{R_1} \sum_{\beta=1}^{R_2} \sum_{\gamma=1}^{R_3} \sum_{\delta=1}^{R_4} \mathcal{I}_{x,1}(i, \alpha) \mathcal{I}_{x,2}(\alpha, j, \beta) \mathcal{I}_{x,3}(\beta, k, \gamma) \mathcal{I}_{x,4}(\gamma, l, \delta) \mathcal{I}_{x,5}(\delta, m) \]
Important core unfoldings

\( \mathcal{H}(\mathcal{F}_{x,n}) \in \mathbb{R}^{R_{n-1} \times I_n \times R_n} \) and \( \mathcal{V}(\mathcal{F}_{x,n}) \in \mathbb{R}^{R_{n-1} \times I_n \times R_n} \)

are horizontal and vertical unfoldings of \( n \)th core.
QR-Based TT Rounding Algorithm [Ose11]

```plaintext
function \{\mathcal{T}_{z,n}\} = TT\text{-}ROUNDING(\{\mathcal{T}_{x,n}\})

▷ Orthogonalization Phase
for n = N down to 2 do
  \[Y_n, R_n\] = QR(\mathcal{H}(\mathcal{T}_{x,n})^T)
  \[V(\mathcal{T}_{x,n-1}) = V(\mathcal{T}_{x,n-1}) \cdot R^T\]

▷ Truncation Phase
\[Z = X\]
for n = 1 to N - 1 do
  \[Y_n, R_n\] = QR(\mathcal{V}(\mathcal{T}_{z,n}))
  \[\hat{U}_R, \hat{\Sigma}, \hat{V}\] ≈ TSVD(\[R_n\])
  \[\mathcal{V}(\mathcal{T}_{z,n}) = APPLY\text{-}Q(Y_n, \hat{U}_R)\]
  \[\mathcal{H}(\mathcal{T}_{z,n+1})^T = APPLY\text{-}Q(Y_{n+1}, \hat{V}\hat{\Sigma})\]

We have parallelized this QR-based algorithm [DBB22] using the parallel TSQR algorithm [DGHL12]
```
TT-Rounding does truncated SVDs on $X_{(1)}$, $X_{(1:2)}$, $X_{(1:3)}$, etc., and here is the matrix expression of each unfolding [DBB22]:

$$X_{(1:n)} = (I_n \otimes Q_{(1:n-1)}) \cdot \mathcal{V}(T x, n) \cdot \mathcal{H}(T x, n+1) \cdot (I_{n+1} \otimes Z_{(1)})$$
Main Ideas of Gram-Based TT-Rounding

Same matrix expression of $n$th unfolding:

$$X_{(1:n)} = (I_n \otimes Q_{(1:n-1)}) \cdot \mathcal{V}(\mathcal{T}x,n) \cdot \mathcal{H}(\mathcal{T}x,n+1) \cdot (I_{n+1} \otimes Z_{(1)})$$

- just like matrix case, we need to compute $A^T A$ and $B^T B$
- two key differences:
  - $A$ and $B$ are both highly structured
  - we need Gram matrices for all $1 \leq n \leq N$
- we obtain efficiency by exploiting structure and by exploiting the computational overlap across modes
\[ X_{(1:n)} = (I_n \otimes Q_{(1:n-1)}) \cdot V(T_{x,n}) \cdot H(T_{x,n+1}) \cdot (I_{n+1} \otimes Z_{(1)}) \]
Tensor Network Diagram for Gram Matrix Computation

intermediate step, computing $G_3^L$ from $G_2^L$

tensor network for $G_3^L = A^T A$
Each core distributed across all $P$ processors
- Local $n$th core dimensions are $R_{n-1} \times \frac{l_n}{P} \times R_n$
- Key: vertical and horizontal unfoldings are 1D-distributed
We round synthetic tensors with input TT-ranks $R = 20$ (all modes) down to $R = 10$, scaling up the number of processors.

- results on Andes (ORNL), with 2 16-core AMD EPYC procs per node.

$$N = 50, 2K \times \cdots \times 2K \ (77 \text{ MB})$$

$$N = 16, 100M \times 50K \times \cdots \times 50K \times 1M \ (8 \text{ GB})$$
We round the 50-mode synthetic tensor with input TT-ranks $R = 20$ down to $R = 10$, scaling up processors with fixed local data.

- results on Andes (ORNL), with 2 16-core AMD EPYC procs per node

Dark signifies computation, light signifies communication.
Results for Cookies Problem

We solve the Cookies problem with Matlab implementation of TT-GMRES [Dol13] and TT-Rounding (sequential)

- we use 4 parameters (cookies) and vary spatial discretization

Dark signifies time in TT-Rounding

Gram has negligible effect on GMRES behavior
Summary

- TT format efficiently approximates high-dim. tensors
  - TT arithmetic causes rank growth, TT-Rounding is key
  - enables solving problems like parameter-dependent PDEs

- TT-Rounding means truncated SVDs of unfoldings $X_{(1:n)}$
  - QR-based approach most accurate but less efficient
  - Gram-based approach faster, can be accurate enough

- Gram-based computation exploits TT structure
  - efficient contraction of tensor network
  - par. distribution allows for comm.-efficient matrix multiplies
Thanks for your attention!

ballard@wfu.edu

MPI_ATTAC: Algorithms for Tensor Train Arithmetic and Computations
https://gitlab.com/aldaas/mpi_attac


TT-Rounding does SVDs on $X_{(1)}$, $X_{(1:2)}$, $X_{(1:3)}$, etc., so we seek similar matrix expressions of those unfoldings.

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

$$X_{(1:n)} = (I_n \otimes Q_{(1:n-1)}) \cdot V(Tx,n) \cdot H(Tx,n+1) \cdot (I_{n+1} \otimes Z_{(1)})$$

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

$$Q(i_1, \ldots, i_{n-1}, r_{n-1}) = T_{x,1}(i_1, :) \cdot T_{x,2}(:, i_2, :) \cdots T_{x,n-1}(:, i_{n-1}, r_{n-1})$$

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

$$Z(r_{n+1}, i_{n+2}, \ldots, i_N) = T_{x,n+2}(r_{n+1}, i_{n+2}, :) \cdot T_{x,n+3}(:, i_{n+3}, :) \cdots T_{x,N}(::, i_N).$$
TT tensor: \( I_n = 512K, R_n = 60 \rightarrow 30, N = 50 \)

70-80% of time spent in QR computations


Ivan Oseledets.  
Tensor-train decomposition.  

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