Part 2: Communication Costs of Tensor Decompositions

Grey Ballard

CS 294/Math 270: Communication-Avoiding Algorithms
UC Berkeley

March 28, 2016

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Plan

- Introduction to tensor decompositions [KB09]
  - nomenclature and notation
  - popular decompositions: CP and Tucker

Plan

- Introduction to tensor decompositions [KB09]
  - nomenclature and notation
  - popular decompositions: CP and Tucker
- We're seeking comm-optimal sequential and parallel algorithms
  - few known lower bounds
  - few standard libraries or HPC implementations

Plan

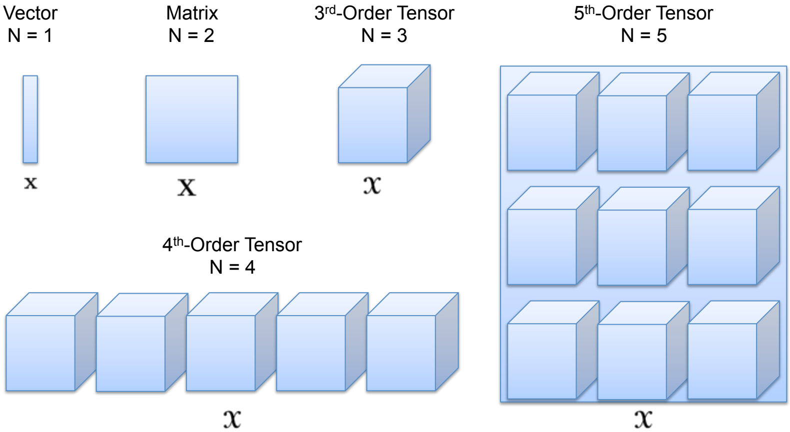
- Introduction to tensor decompositions [KB09]
  - nomenclature and notation
  - popular decompositions: CP and Tucker
- We're seeking comm-optimal sequential and parallel algorithms
  - few known lower bounds
  - few standard libraries or HPC implementations
- Many flavors of problems
  - dense or sparse
  - sequential or parallel
  - CP or Tucker (or alternatives like tensor train)
  - choices of mathematical algorithm

Plan

- Introduction to tensor decompositions [KB09]
  - nomenclature and notation
  - popular decompositions: CP and Tucker
- We're seeking comm-optimal sequential and parallel algorithms
  - few known lower bounds
  - few standard libraries or HPC implementations
- Many flavors of problems
  - dense or sparse
  - sequential or parallel
  - CP or Tucker (or alternatives like tensor train)
  - choices of mathematical algorithm
- We'll do two case studies of parallel algorithms
  - computing CP decomposition of sparse tensor [KU15]
  - computing Tucker decomposition of dense tensor [ABK15]

Outline

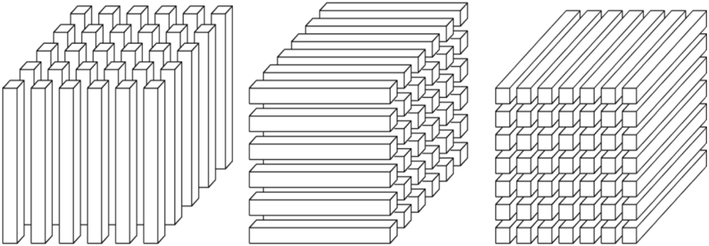
Tensors



An N^{th}-order tensor has N *modes*

Notation convention: vector **v**, matrix **M**, tensor $\mathcal{T}$

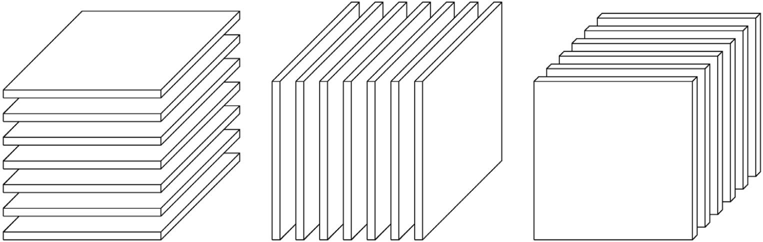
Fibers



Mode-1 Fibers Mode-2 Fibers Mode-3 Fibers

A tensor can be decomposed into the *fibers* of each mode
(fix all indices but one)

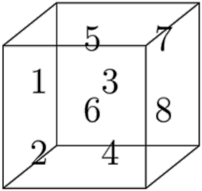
Slices



(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

A tensor can also be decomposed into the *slices* of each mode (fix one index)

$\mathcal{X} =$



$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

A tensor can be reshaped into matrices, called *unfoldings* or *matricizations*, for different modes (fibers form columns, slices form rows)

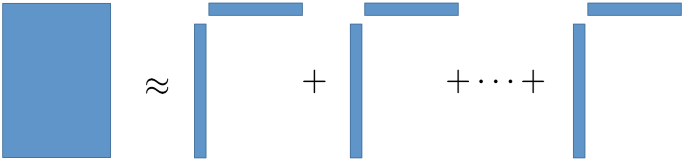
Outline

Low-rank approximations of tensors

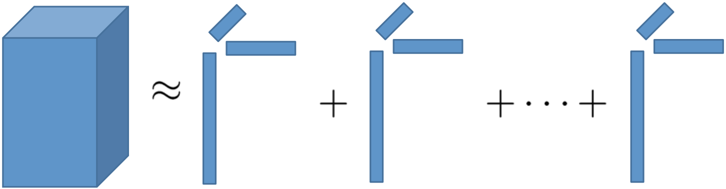
- Tensor “decompositions” are usually low-rank approximations
- They generalize matrix approximations from two viewpoints
  - sum of outer products (think PCA)
  - product of two rectangular matrices (think high-variance subspaces)
- Some applications seek true decompositions, but less common

Sum of outer products

Matrix:

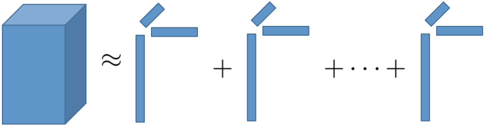


Tensor:



This is known as the CANDECOMP/PARAFAC (CP) decomposition

CP Notation



$$\mathfrak{T} \approx \mathbf{u}_1 \circ \mathbf{v}_1 \circ \mathbf{w}_1 + \cdots + \mathbf{u}_R \circ \mathbf{v}_R \circ \mathbf{w}_R, \qquad \mathfrak{T} \in \mathbb{R}^{I \times J \times K}$$

$$\mathfrak{T} \approx [\![\mathbf{U}, \mathbf{V}, \mathbf{W}]\!], \qquad \mathbf{U} \in \mathbb{R}^{I \times R}, \mathbf{V} \in \mathbb{R}^{J \times R}, \mathbf{W} \in \mathbb{R}^{K \times R} \text{ are } \textit{factor matrices}$$

$$t_{ijk} \approx \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr}, \qquad 1 \leq i \leq I, 1 \leq j \leq J, 1 \leq k \leq K$$

Notation convention: scalar dimension N, index n with $1 \leq n \leq N$

Applications of CP

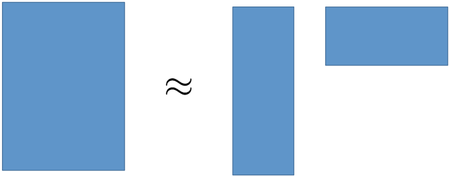
CP often used like PCA for multi-dimensional data

- interpretable components separated from noise

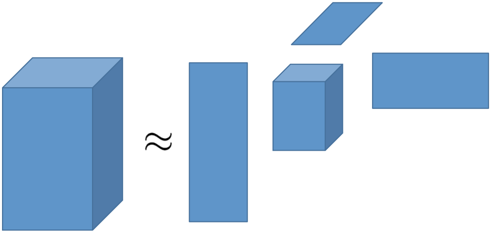
Sample applications

- chemometrics [AB03]
  - data is excitation wavelengths $\times$ emission wavelengths $\times$ time
  - components correspond to chemical species' signatures
- neuroscience [AABB$^+$07]
  - data is electrode $\times$ frequency $\times$ time
  - components help to describe origin of a seizure
- text analysis [BBB08]
  - data is term $\times$ author $\times$ time
  - components discover conversations

Matrix:

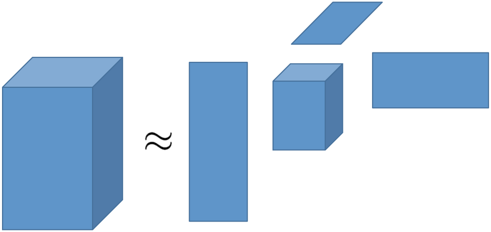


Tensor:



This is known as the Tucker decomposition

Tucker Notation



$\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}$ $\qquad$ $\mathcal{T} \in \mathbb{R}^{I \times J \times K}, \mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is *core tensor*

$\mathcal{T} \approx [\![\mathcal{G}; \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$, $\quad$ $\mathbf{U} \in \mathbb{R}^{I \times P}, \mathbf{V} \in \mathbb{R}^{J \times Q}, \mathbf{W} \in \mathbb{R}^{K \times R}$ are *factor matrices*

$$t_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} u_{ip} v_{jq} w_{kr}, \qquad 1 \le i \le I, 1 \le j \le J, 1 \le k \le K$$

Tensor-Times-Matrix (TTM)

Tensor version:

$$\mathcal{Y} = \mathcal{X} \times_2 \mathbf{M}$$

$$\mathcal{Y} \in \mathbb{R}^{I \times Q \times K} \qquad \mathcal{X} \in \mathbb{R}^{I \times J \times K} \qquad \mathbf{M} \in \mathbb{R}^{Q \times J}$$

Matrix version:

$$\mathbf{Y}_{(2)} = \mathbf{M}\mathbf{X}_{(2)}$$

$$\mathbf{Y}_{(2)} \in \mathbb{R}^{Q \times IK} \qquad \mathbf{X}_{(2)} \in \mathbb{R}^{J \times IK}$$

Element version:

$$y_{iqk} = \sum_{j=1}^{J} m_{qj} x_{ijk}$$

TTM is matrix multiplication with certain unfolding

Applications of Tucker

Tucker can be viewed as a richer form of CP, so it's also used like PCA

- a diagonal core tensor corresponds to a CP decomposition

Sample Application

- Computer vision: TensorFaces [VT02]
  - facial recognition system benefiting from varying lighting, expression, viewpoint

Tucker is typically more efficient than CP for compression

Sample Application

- Visual data compression [BRP15]
  - image, video, and 3D volume data

Ambiguities

There are several ambiguities that have to be handled carefully

CP scaling ambiguity

$$\mathfrak{T} \approx \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r \qquad \rightarrow \qquad \mathfrak{T} \approx \sum_{r=1}^{R} \lambda_r \cdot \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$$

where $\|\mathbf{u}_r\|_2 = \|\mathbf{v}_r\|_2 = \|\mathbf{w}_r\|_2 = 1$

Tucker basis ambiguity

$$\mathfrak{T} \approx \mathfrak{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}$$

where $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_P$, $\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I}_Q$, $\mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}_R$

Survey Paper

Notation can be a huge obstacle to working with tensors, standardization can help

I recommend following the conventions of the following paper:

Tensor Decompositions and Applications

Tammy Kolda and Brett Bader

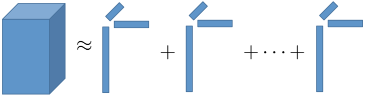
SIAM Review 2009

`http://epubs.siam.org/doi/abs/10.1137/07070111X`

Outline

Outline

CP Optimization Problem



For fixed rank R, we want to solve

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \left\| \mathcal{X} - \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r \right\|$$

which is a nonlinear, nonconvex optimization problem

- in the matrix case, the SVD gives us the optimal solution
- in the tensor case, uniqueness/convergence to optimum not guaranteed

Alternating Least Squares (ALS)

Fixing all but one factor matrix, we have a linear least squares problem:

$$\min_{\mathbf{V}} \left\| \mathcal{X} - \sum_{r=1}^{R} \hat{\mathbf{u}}_r \circ \mathbf{v}_r \circ \hat{\mathbf{w}}_r \right\|$$

or equivalently

$$\min_{\mathbf{V}} \left\| \mathbf{X}_{(2)} - \mathbf{V}(\hat{\mathbf{W}} \odot \hat{\mathbf{U}})^{\mathsf{T}} \right\|_F$$

where $\odot$ is the *Khatri-Rao* product, a column-wise Kronecker product

ALS works by alternating over factor matrices, updating one at a time by solving the corresponding linear least squares problem

CP-ALS

Repeat

1. Solve $\mathbf{U}(\mathbf{V}^\mathsf{T}\mathbf{V} * \mathbf{W}^\mathsf{T}\mathbf{W}) = \mathbf{X}_{(1)}(\mathbf{W} \odot \mathbf{V})$ for $\mathbf{U}$
2. Normalize columns of $\mathbf{U}$
3. Solve $\mathbf{V}(\mathbf{U}^\mathsf{T}\mathbf{U} * \mathbf{W}^\mathsf{T}\mathbf{W}) = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$ for $\mathbf{V}$
4. Normalize columns of $\mathbf{V}$
5. Solve $\mathbf{W}(\mathbf{U}^\mathsf{T}\mathbf{U} * \mathbf{V}^\mathsf{T}\mathbf{V}) = \mathbf{X}_{(3)}(\mathbf{V} \odot \mathbf{U})$ for $\mathbf{W}$
6. Normalize columns of $\mathbf{W}$ and store norms in λ

Linear least squares problems solved via normal equations using identity $(\mathbf{A} \odot \mathbf{B})^\mathsf{T}(\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^\mathsf{T}\mathbf{A} * \mathbf{B}^\mathsf{T}\mathbf{B}$, where $*$ is element-wise product

Outline

Matricized Tensor Times Khatri-Rao Product

CP-ALS spends most of its time in MTTKRP (dense or sparse)

- corresponds to setting up the right-hand-side of normal equations
- $\mathbf{M}^{(V)} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$, for example

Matricized Tensor Times Khatri-Rao Product

CP-ALS spends most of its time in MTTKRP (dense or sparse)

- corresponds to setting up the right-hand-side of normal equations
- $\mathbf{M}^{(V)} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$, for example

In the dense case, it usually makes sense to

1. form Khatri-Rao product explicitly
2. call dense matrix multiplication

Matricized Tensor Times Khatri-Rao Product

CP-ALS spends most of its time in MTTKRP (dense or sparse)

- corresponds to setting up the right-hand-side of normal equations
- $\mathbf{M}^{(V)} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$, for example

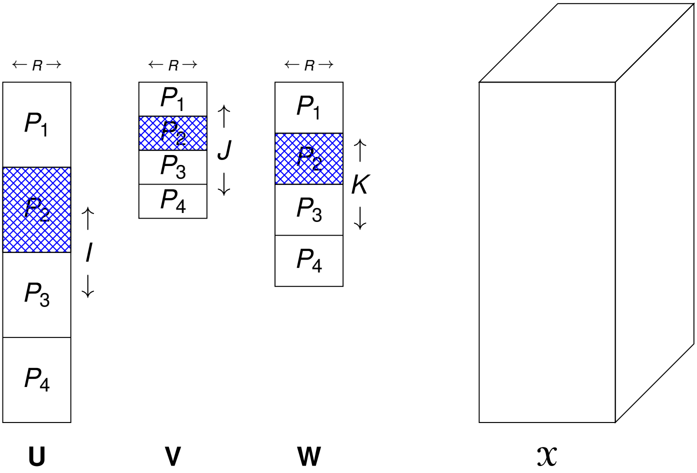
In the dense case, it usually makes sense to

1. form Khatri-Rao product explicitly
2. call dense matrix multiplication

In the sparse case, it usually make sense [BK07] to use

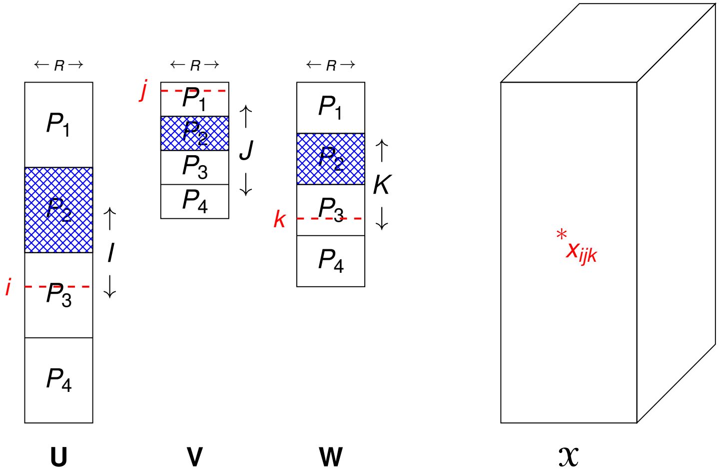
- element-wise formula $$m_{jr}^{(V)} = \sum_{i=1}^{I} \sum_{k=1}^{K} x_{ijk} u_{ir} w_{kr}$$
- row-wise formula $$m_{j,:}^{(V)} = \sum_{i=1}^{I} \sum_{k=1}^{K} x_{ijk} (u_{i,:} * w_{k,:})$$

Coarse-Grain Distribution for CP-ALS [KU15]



Rows of each factor matrices are distributed across processors

Coarse-Grain Distribution for CP-ALS [KU15]



Rows of each factor matrices are distributed across processors
Each tensor nonzero is copied to each process that will need it

Coarse-Grain Parallelization for MTTKRP [KU15]

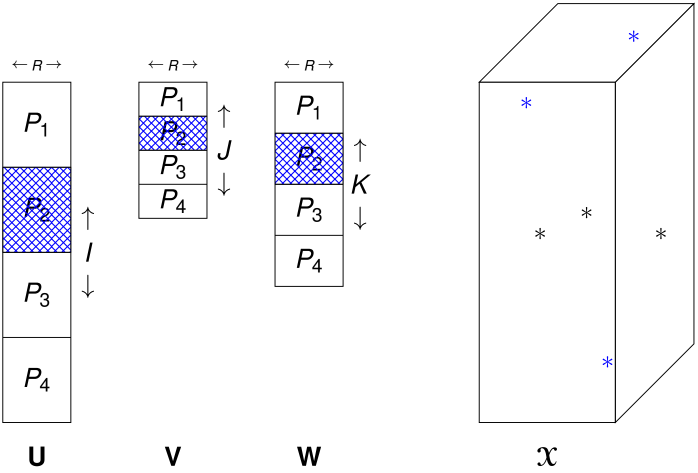
To update **V**, need to compute $\mathbf{M}^{(V)} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$

Let I_p, J_p, K_p be the subset of rows of $\mathbf{U}, \mathbf{V}, \mathbf{W}$ owned by processor p

Main loop:
for each $j \in J_p$
  for each nonzero x_{ijk} in slice j
    $m^{(V)}_{j,:} \leftarrow m^{(V)}_{j,:} + x_{ijk} \cdot (u_{i,:} * w_{k,:})$

In the inner loop, $u_{i,:}$ or $w_{k,:}$ require communication if $i \notin I_p$ or $k \notin K_p$

Fine-Grain Distribution of CP-ALS [KU15]



Rows of each factor matrices are distributed across processors
Tensor nonzeros are distributed across processors

Fine-Grain Parallelization for MTTKRP [KU15]

To update $\mathbf{V}$, need to compute $\mathbf{M}^{(V)} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$

Let $\mathcal{X}_p$ be the subset of nonzeros of $\mathcal{X}$ owned by processor p
Let I_p, J_p, K_p be the subset of rows of $\mathbf{U}, \mathbf{V}, \mathbf{W}$ owned by processor p

Main loop:
for each $x_{ijk} \in \mathcal{X}_p$

$$m_{j,:}^{(V)} \leftarrow m_{j,:}^{(V)} + x_{ijk} \cdot (u_{i,:} * w_{k,:})$$

In the inner loop, $u_{i,:}$ or $w_{k,:}$ require communication if $i \notin I_p$ or $k \notin K_p$
After the loop, $m_{j,:}^{(V)}$ for $j \notin J_p$ needs to be sent to owner processor

Minimizing Communication

- Algorithms defined for any distributions of factor matrices / tensor
- Distributions determine computational load balance and communication costs
- Finding optimal distribution for each algorithm is a hypergraph partitioning problem (subject to load balance constraint)
- Even if hypergraph is optimally partitioned, no guarantees that either algorithm is communication optimal

Coarse-Grain vs Fine-Grain

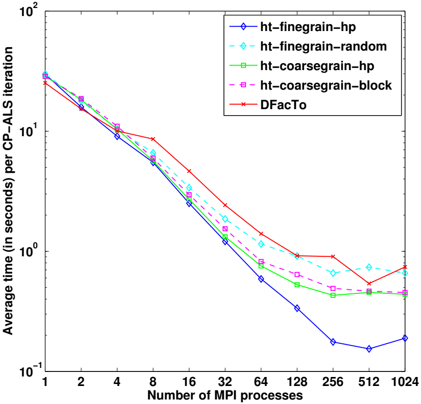
Coarse-Grain

- Owner computes: communicates only inputs within MTTKRP
- Requires replication of $\mathcal{X}$
- Generalizes row-wise algorithm for SpMV (for multiple vectors)

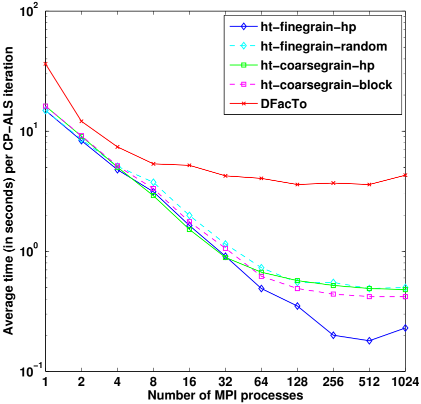
Fine-Grain

- Communicates inputs and outputs within MTTKRP
- No replication of $\mathcal{X}$
- Generalizes fine-grain algorithm for SpMV

Performance Comparison [KU15]



(a) Time on Netflix

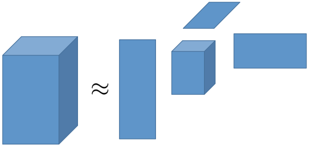


(b) Time on NELL-B

Outline

Outline

Tucker Optimization Problem



For fixed ranks P, Q, R, we want to solve

$$\min_{\hat{\mathcal{X}}} \left\| \mathcal{X} - \hat{\mathcal{X}} \right\|^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} (x_{ijk} - \hat{x}_{ijk})^2 \text{ subject to } \hat{\mathcal{X}} = [\![\mathcal{G}; \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$$

which turns out to be equivalent to

$$\max_{\mathbf{U},\mathbf{V},\mathbf{W}} \|\mathcal{G}\| \text{ subject to } \mathcal{G} = \mathcal{X} \times_1 \mathbf{U}^\mathsf{T} \times_2 \mathbf{V}^\mathsf{T} \times_3 \mathbf{W}^\mathsf{T}$$

which is a nonlinear, nonconvex optimization problem

Higher-Order Orthogonal Iteration (HOOI)

Fixing all but one factor matrix, we have a matrix problem:

$$\max_{\mathbf{V}} \left\| \mathcal{X} \times_1 \hat{\mathbf{U}}^\mathsf{T} \times_2 \mathbf{V}^\mathsf{T} \times_3 \hat{\mathbf{W}}^\mathsf{T} \right\|$$

or equivalently

$$\max_{\mathbf{V}} \left\| \mathbf{V}^\mathsf{T} \mathbf{Y}_{(2)} \right\|_F$$

where $\mathcal{Y} = \mathcal{X} \times_1 \hat{\mathbf{U}}^\mathsf{T} \times_3 \hat{\mathbf{W}}^\mathsf{T}$

HOOI works by alternating over factor matrices, updating one at a time by computing leading left singular vectors

Sequentially Truncated Higher-Order SVD

- HOOI is very sensitive to initialization
- Truncated Higher-Order SVD (T-HOSVD) typically used
- ST-HOSVD [VVM12] is more efficient than T-HOSVD, works by
  - initializing with identity matrices $\mathbf{U} = \mathbf{I}_I$, $\mathbf{V} = \mathbf{I}_J$, $\mathbf{W} = \mathbf{I}_K$
  - applying one iteration of HOOI
  - where ranks P, Q, R can be chosen based on error tolerance

ST-HOSVD Algorithm

1. $\mathbf{S}^{(1)} \leftarrow \mathbf{X}_{(1)}\mathbf{X}_{(1)}^{\mathsf{T}}$
2. $\mathbf{U} =$ leading eigenvectors of $\mathbf{S}^{(1)}$
3. $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}$
4. $\mathbf{S}^{(2)} \leftarrow \mathbf{Y}_{(2)}\mathbf{Y}_{(2)}^{\mathsf{T}}$
5. $\mathbf{V} =$ leading eigenvectors of $\mathbf{S}^{(2)}$
6. $\mathcal{Z} = \mathcal{Y} \times_2 \mathbf{V}$
7. $\mathbf{S}^{(3)} \leftarrow \mathbf{Z}_{(3)}\mathbf{Z}_{(3)}^{\mathsf{T}}$
8. $\mathbf{W} =$ leading eigenvectors of $\mathbf{S}^{(3)}$
9. $\mathcal{G} = \mathcal{Z} \times_3 \mathbf{W}$

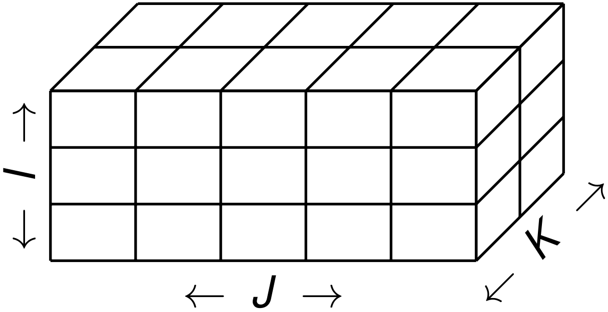
Left singular vectors of $\mathbf{A}$ computed as eigenvectors of $\mathbf{A}^{\mathsf{T}}\mathbf{A}$

Outline

Parallel Block Tensor Distribution

For N-mode tensor, use logical N-mode processor grid

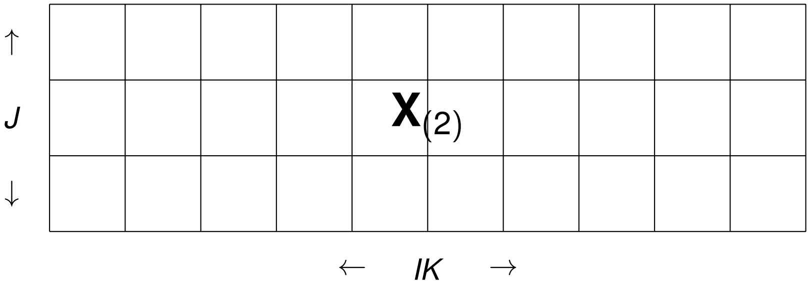
Proc. grid: $P_I \times P_J \times P_K = 3 \times 5 \times 2$



Local tensors have dimensions $\frac{I}{P_I} \times \frac{J}{P_J} \times \frac{K}{P_K}$

Key idea: each unfolded matrix is 2D block distributed

Proc. grid: $P_I \times P_J \times P_K = 3 \times 5 \times 2$



Logical mode-2 2D processor grid: $P_J \times P_I P_K$

Local unfolded matrices have dimensions $\frac{J}{P_J} \times \frac{IK}{P_I P_K}$

Kernel Matrix Computations

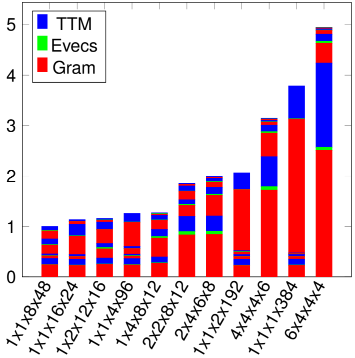
Key computations in ST-HOSVD are

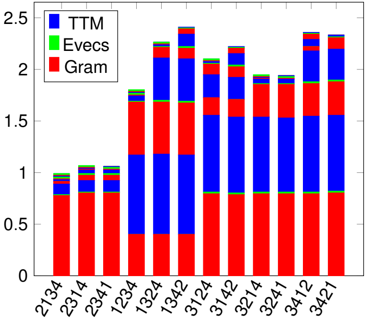
- Gram: computing $\mathbf{X}_{(2)}\mathbf{X}_{(2)}^{\mathsf{T}}$
- TTM: computing $\mathbf{Y}_{(2)} = \mathbf{V}^{\mathsf{T}}\mathbf{X}_{(2)}$

These are just matrix computations, done for each mode in sequence

- can determine lower bound/opt. alg. for individual computations
- how to minimize communication across all computations?

Parameter Tuning

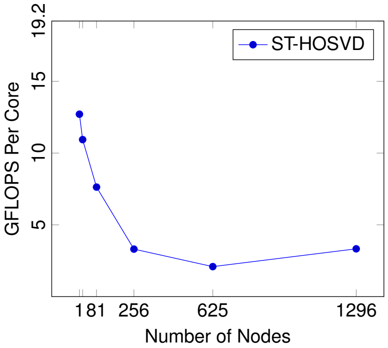




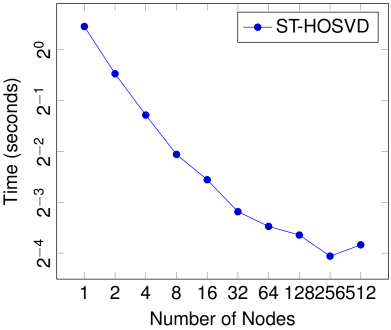
Varying processor grid for tensor of size $384 \times 384 \times 384 \times 384$ with reduced size of $96 \times 96 \times 96 \times 96$.

Varying mode order for tensor of size $25 \times 250 \times 250 \times 250$ with reduced size $10 \times 10 \times 100 \times 100$.

Parallel Scaling



Weak scaling for
$200k \times 200k \times 200k \times 200k$
tensor with reduced size
$20k \times 20k \times 20k \times 20k$,
using k^4 nodes for $1 \leq k \leq 6$.



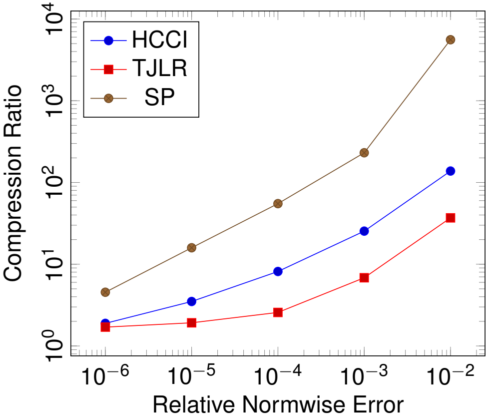
Strong scaling for
$200 \times 200 \times 200 \times 200$
tensor with reduced size
$20 \times 20 \times 20 \times 20$,
using 2^k nodes for $0 \leq k \leq 9$.

Application: Compression of Scientific Simulation Data

We applied ST-HOSVD to compress multidimensional data from numerical simulations of combustion, including the following data sets:

- **HCCI:**
  - Dimensions: $672 \times 672 \times 33 \times 627$
  - 672×672 spatial grid, 33 variables over 627 time steps
  - Total size: 70 GB
- **TJLR:**
  - Dimensions: $460 \times 700 \times 360 \times 35 \times 16$
  - $460 \times 700 \times 360$ spatial grid, 35 variables over 16 time steps
  - Total size: 520 GB
- **SP:**
  - Dimensions: $500 \times 500 \times 500 \times 11 \times 50$
  - $500 \times 500 \times 500$ spatial grid, 11 variables over 50 time steps
  - Total size: 550GB

Application: Compression of Scientific Simulation Data



Compression ratio: $\frac{IJK}{PQR+IP+JQ+KR}$ Relative Normwise Error: $\frac{\|\mathcal{X}-\hat{\mathcal{X}}\|}{\|\mathcal{X}\|}$

Outline

Numerical Questions

- CP-ALS solves least squares problems using normal equations
- ST-HOSVD computes singular vectors using the Gram matrix

- Are there applications that require better numerical stability?
- Can more numerically stable methods be implemented efficiently?

CA Questions

- What are the communication lower bounds for MTTKRP?
  - the computation can be expressed as nested loops
  - is there a tradeoff between computation and communicaton?

- What are the communication lower bounds for ST-HOSVD?
  - we've already improved the comm. costs of the published algorithm
  - can the parameter tuning problems be solved analytically?

For more details:

Scalable Sparse Tensor Decompositions in Distributed Memory Systems
Oguz Kaya and Bora Uçar
International Conference for High Performance Computing, Networking, Storage and Analysis 2015
`http://doi.acm.org/10.1145/2807591.2807624`

Parallel Tensor Compression for Large-Scale Scientific Data
Woody Austin, Grey Ballard, and Tamara G. Kolda
International Parallel and Distributed Processing Symposium 2016
`http://arxiv.org/abs/1510.06689`

References I

Evrim Acar, Canan Aykut-Bingol, Haluk Bingol, Rasmus Bro, and Bülent Yener.
Multiway analysis of epilepsy tensors.
Bioinformatics, 23(13):i10–i18, 2007.

C. M. Andersen and R. Bro.
Practical aspects of parafac modeling of fluorescence excitation-emission data.
Journal of Chemometrics, 17(4):200–215, 2003.

Woody Austin, Grey Ballard, and Tamara G. Kolda.
Parallel tensor compression for large-scale scientific data.
Technical Report 1510.06689, arXiv, 2015.
To appear in IPDPS.

References II

Brett W. Bader, Michael W. Berry, and Murray Browne.
Survey of Text Mining II: Clustering, Classification, and Retrieval, chapter Discussion Tracking in Enron Email Using PARAFAC, pages 147–163.
Springer London, London, 2008.

Brett W. Bader and Tamara G. Kolda.
Efficient MATLAB computations with sparse and factored tensors.
SIAM Journal on Scientific Computing, 30(1):205–231, December 2007.

Rafael Ballester-Ripoll and Renato Pajarola.
Lossy volume compression using tucker truncation and thresholding.
The Visual Computer, pages 1–14, 2015.

T. G. Kolda and B. W. Bader.
Tensor decompositions and applications.
SIAM Review, 51(3):455–500, September 2009.

References III

Oguz Kaya and Bora Uçar.
Scalable sparse tensor decompositions in distributed memory systems.
In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pages 77:1–77:11, New York, NY, USA, 2015. ACM.

M. Alex O. Vasilescu and Demetri Terzopoulos.
Computer Vision — ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I, chapter Multilinear Analysis of Image Ensembles: TensorFaces, pages 447–460.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen.
A new truncation strategy for the higher-order singular value decomposition.
SIAM Journal on Scientific Computing, 34(2):A1027–A1052, 2012.