Parallel Numerical Algorithms

Chapter 7 – Differential Equations
Section 7.5 – Tensor Analysis

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

CS 554 / CSE 512

Outline

Tensors

A *tensor* $\boldsymbol{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ has

- *Order* d (i.e. d *modes* / *indices*)
- *Dimensions* n_1-by-$\cdots$-by-n_d
- *Elements* $t_{i_1 \ldots i_d} = t_{\boldsymbol{i}}$ where $\boldsymbol{i} \in \bigotimes_{i=1}^{d} \{1, \ldots, n_i\}$

Order d tensors represent d-dimensional arrays

- $(d \geq 3)$-dimensional arrays are prevalent in scientific computing
  - Regular grids, collections of matrices, multilinear operators
  - Experimental data, visual/graphic data
- Tensors analysis is the expression and study of numerical methods using tensor representations

Reshaping Tensors

When using tensors, it is often necessary to transition between high-order and low-order representations of the same object

- Recall for a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ its *unfolding* is given by

$$\boldsymbol{v} = \text{vec}\left(\boldsymbol{A}\right), \Rightarrow \boldsymbol{v} \in \mathbb{R}^{mn}, v_{i+jm} = a_{ij}$$

- A tensor $\boldsymbol{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ can be fully unfolded the same way

$$\boldsymbol{v} = \text{vec}\left(\boldsymbol{T}\right), \Rightarrow \boldsymbol{v} \in \mathbb{R}^{n_1 \cdots n_d}, v_{i_1 + i_2 n_1 + i_3 n_1 n_2 + \ldots} = t_{i_1 i_2 i_3 \ldots}$$

- Often we also want to *fold* tensors into higher-order ones
- Generally, we can *reshape* (fold or unfold) any tensor

$$\boldsymbol{U} = o_{n_1 \times \cdots \times n_d}(\boldsymbol{V}) \quad \Rightarrow \quad \boldsymbol{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}, \quad \text{vec}\left(\boldsymbol{U}\right) = \text{vec}\left(\boldsymbol{V}\right)$$

Tensor Transposition

For tensors of order ≥ 3, there is more than one way to transpose modes

- A *tensor transposition* is defined by a permutation $\boldsymbol{p}$ containing elements $\{1, \ldots, d\}$

$$\boldsymbol{Y} = \boldsymbol{X}^{\langle \boldsymbol{p} \rangle} \quad \Rightarrow \quad y_{i_{p_1}, \ldots, i_{p_d}} = x_{i_1, \ldots, i_d}$$

- In this notation, a transposition of matrix $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}^T = \boldsymbol{A}^{\langle [2,1] \rangle}$$

- Tensor transposition is a convenient primitive for manipulating multidimensional arrays and mapping tensor computations to linear algebra
- In tensor derivations, indices are often carried through to avoid transpositions

Tensor Symmetry

We say a tensor is *symmetric* if $\forall j, k \in \{1, \ldots, d\}$

$\boldsymbol{t}_{i_1 \ldots i_j \ldots i_k \ldots i_d} = \boldsymbol{t}_{i_1 \ldots i_k \ldots i_j \ldots i_d}$ or equivalently $\boldsymbol{T} = \boldsymbol{T}^{\langle [1, \ldots, j, \ldots, k, \ldots d] \rangle}$

A tensor is *antisymmetric* (skew-symmetric) if $\forall j, k \in \{1, \ldots, d\}$

$$\boldsymbol{t}_{i_1 \ldots i_j \ldots i_k \ldots i_d} = (-1)\boldsymbol{t}_{i_1 \ldots i_k \ldots i_j \ldots i_d}$$

A tensor is *partially-symmetric* if such index interchanges are restricted to be within subsets of $\{1, \ldots, d\}$, e.g.

$$\boldsymbol{t}_{ijkl} = \boldsymbol{t}_{jikl} = \boldsymbol{t}_{jilk} = \boldsymbol{t}_{ijlk}$$

Tensor Products and Kronecker Products

Tensor products can be defined with respect to maps $f : V_f \to W_f$ and $g : V_g \to W_g$

$$h = f \times g \quad \Rightarrow \quad g : (V_f \times V_g) \to (W_f \times W_g), \quad h(x, y) = f(x)g(y)$$

Tensors can be used to represent multilinear maps and have a corresponding definition for a tensor product

$$\boldsymbol{T} = \boldsymbol{X} \times \boldsymbol{Y} \quad \Rightarrow \quad t_{i_1,\ldots,i_m,j_1,\ldots,j_n} = x_{i_1,\ldots,i_m} y_{j_1,\ldots,j_n}$$

The *Kronecker product* between two matrices $\boldsymbol{A} \in \mathbb{R}^{m_1 \times m_2}$, $\boldsymbol{B} \in \mathbb{R}^{n_1 \times n_2}$

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} \quad \Rightarrow \quad c_{i_2+i_1 m_2, j_2+j_1 n_2} = a_{i_1 j_1} b_{i_2 j_2}$$

corresponds to transposing and reshaping the tensor product

$$\boldsymbol{A} \otimes \boldsymbol{B} = o_{m_1 n_1, m_2 n_2}((\boldsymbol{A} \times \boldsymbol{B})^{\langle [3,1,4,2] \rangle})$$

Tensor Partial Sum

- $\boldsymbol{Y}$ of order $d-r$ is a *partial sum* of $\boldsymbol{X}$ of order d if for some $\boldsymbol{q}$ containing r elements of $\{1,\ldots,d\}$

$$\boldsymbol{Y} = \sum_{\boldsymbol{q}}(\boldsymbol{X}) \quad \Rightarrow \quad \bar{\boldsymbol{X}} = \boldsymbol{X}^{\langle[q_1,\ldots,q_r,\ldots]\rangle},$$

$$y_{i_1\ldots i_{d-r}} = \sum_{j_1}\cdots\sum_{j_r}\bar{x}_{j_1,\ldots j_r,i_1,\ldots i_{d-r}}$$

- Partial summations provide a powerful primitive operation when coupled with transposition and reshape

Tensor Trace

- $\boldsymbol{Z}$ of order $d-2r$ is a *trace* of $\boldsymbol{X}$ of order $d \geq r$ if for some $\boldsymbol{p}, \boldsymbol{q}$ each containing a different set of r elements of $\{1, \ldots, d\}$

$$\boldsymbol{Y} = \operatorname*{trace}_{\boldsymbol{p},\boldsymbol{q}}(\boldsymbol{X}) \quad \Rightarrow \quad \bar{\boldsymbol{X}} = \boldsymbol{X}^{\langle [p_1,\ldots,p_r q_1,\ldots,q_r,\ldots] \rangle},$$

$$y_{i_1 \ldots i_{d-2r}} = \sum_{j_1} \cdots \sum_{j_r} \bar{x}_{j_1,\ldots j_r, j_1,\ldots j_r, i_1,\ldots i_{d-2r}}$$

- The trace of a matrix $\boldsymbol{A}$ in this notation is

$$\operatorname{trace}(\boldsymbol{A}) = \operatorname*{trace}_{[0],[1]}(\boldsymbol{A}) = \sum_i a_{ii}$$

Tensor Contraction

Tensor contraction is a transpose of a trace of a tensor product

$$\boldsymbol{C} = \left[\operatorname*{trace}_{\boldsymbol{p},\boldsymbol{q}}(\boldsymbol{A} \times \boldsymbol{B})\right]^{\langle \boldsymbol{r} \rangle} \quad \text{for some } \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}$$

- Examples in linear algebra include: vector inner and outer products, matrix–vector product, matrix–matrix product
- The *contracted* modes of $\boldsymbol{A}$ appear in $\boldsymbol{p}$ and of $\boldsymbol{B}$ in $\boldsymbol{q}$, while *uncontracted* modes appear in $\boldsymbol{r}$
- Matrix multiplication would be given by $\boldsymbol{p} = [2]$, $\boldsymbol{q} = [3]$, $\boldsymbol{r} = [1, 4]$

Tensor Times Matrix

Tensor times matrix (TTM) is one of the most common tensor contractions involving tensors of order ≥ 3

- Given an order 3 tensor $\boldsymbol{T}$ and matrix $\boldsymbol{V}$, TTM computes order 3 tensor $\boldsymbol{W}$, generalizes naturally to higher-order $\boldsymbol{T}$
- TTM can contract one of three modes of $\boldsymbol{T}$

$$\boldsymbol{W} = \left[\underset{[3],[4]}{\mathrm{trace}}(\boldsymbol{T} \times \boldsymbol{V})\right]^{\langle[1,2,5]\rangle} \quad \text{or} \quad \boldsymbol{W} = \left[\underset{[2],[4]}{\mathrm{trace}}(\boldsymbol{T} \times \boldsymbol{V})\right]^{\langle[1,3,5]\rangle}$$

$$\text{or} \quad \boldsymbol{W} = \left[\underset{[1],[4]}{\mathrm{trace}}(\boldsymbol{T} \times \boldsymbol{V})\right]^{\langle[2,3,5]\rangle}$$

- In the first case, we have

$$w_{ijk} = \sum_{l} t_{ijl} v_{lk}$$

Tensor Contraction Diagrams

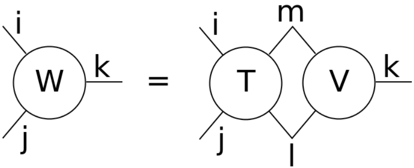
Consider the tensor contraction

$$w_{ijk} = \sum_{lm} t_{ijlm} v_{mkl}$$

which we can also write in tensor notation

$$\boldsymbol{W} = \left[\operatorname*{trace}_{[3,4],[7,5]} (\boldsymbol{T} \times \boldsymbol{V}) \right]^{\langle [1,2,6] \rangle}$$

or in the following diagrammatic form



CP decomposition

The SVD corresponds to a sum of outer products of the form

$$\boldsymbol{A} = \sum_{k=1}^{r} \sigma_k \boldsymbol{u_k} \boldsymbol{v_k}^T$$

so it is natural to seek to approximate a tensor as

$$\boldsymbol{T} = \sum_{k=1}^{r} \sigma_k \boldsymbol{u}_k^{(1)} \times \cdots \times \boldsymbol{u}_k^{(d)}$$

where each $\boldsymbol{u}_k^{(i)}$ is orthogonal to any other $\boldsymbol{u}_{k'}^{(i)}$, yielding the *canonical polyadic (CP) decomposition* of $\boldsymbol{T}$

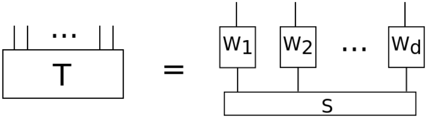
- r is referred to as the *canonical rank* of the tensor

Computing the CP decomposition

- Computing the canonical rank is *NP hard*
- Approximation by CP decomposition is *ill-posed*
- Regularization (imposing bounds on the norm of the factor matrices) make the optimization problem feasible
- *Alternating least squares (ALS)* commonly used for computation
  - Optimizes for one factor matrix at a time
  - Least squares problem for each matrix
- Alternatives include coordinate and gradient descent methods, much like in numerical optimization for matrix completion

Tucker Decomposition

The *Tucker decomposition* introduces an order d *core tensor* into the CP decomposition



$$t_{i_1 \dots i_d} = \sum_{k_1 \cdots k_d} s_{k_1 \dots k_d} w^{(1)}_{i_1 k_1} \cdots w^{(d)}_{i_d k_d}$$

where the columns of each $\boldsymbol{W}^{(i)}$ are orthonormal

- Unlike CP decomposition (given by 'diagonal' tensor S), each index appears in no more than two tensors
- Tucker decomposition is not *low-order* since the order of $\boldsymbol{T}$ matches that of $\boldsymbol{S}$

Computing the Tucker Decomposition

The *SVD (HOSVD)* can refer to the Tucker decomposition or the following basic method for its computation

- Compute the left singular vectors $\boldsymbol{W}^{(i)}$ of all d single-mode unfoldings of $\boldsymbol{T}$

$$\boldsymbol{T}_{(i)} = o_{n_i \times n_1 \cdots n_{i-1} n_{i+1} \cdots n_d}(\boldsymbol{T}^{\langle [i,1,\ldots,i-1,i+1,\ldots d] \rangle})$$

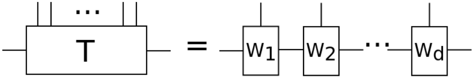
- Compute the core tensor by projecting $\boldsymbol{A}$ onto these singular vectors along each mode

$$\boldsymbol{S} = \left[\underset{[1,\ldots d],[d+2,d+4,\ldots,2d]}{\text{trace}} (\boldsymbol{T} \times \boldsymbol{W}^{(1)} \times \cdots \times \boldsymbol{W}^{(d)}) \right]^{\langle [d+1,d+3,\ldots,2d-1] \rangle}$$

- The HOSVD works well when the core tensor $\boldsymbol{S}$ can be shown to have decaying entries

Tensor Train Decomposition

The *tensor train* decomposition has the following diagrammatic representation



- The tensor train is a chain of contracted order 3 tensors with the two ends having order 2
- Elements of $\boldsymbol{T}$ are given by matrix product chain

$$t_{i_1 \dots i_d} = \boldsymbol{u}^{(i_1)} \boldsymbol{U}^{(i_2)} \cdots \boldsymbol{U}^{(i_{d-1})} \boldsymbol{u}^{(i_d)}$$

- Has been used for decades in physics, known as the *matrix product states (MPS)* representation

Tensor Train Properties

- The *tensor train (TT) ranks* are given by the dimensions of the auxiliary modes in the factorization
- The *tensor train (TT) rank* is the maximum number of columns in any matrix $\boldsymbol{U}^{(i_j)}$
- The tensor train rank is a matrix rank, i.e. it corresponds to a low-rank decomposition of a matrix (given by contracting two parts of the tensor train)
- Summation and products of tensor trains can be readily computed

Quantized Tensor Train

- The *quantized tensor train (QTT)* corresponds to the application of TT to a tensor that is reshaped to be higher-order (e.g. each resulting mode dimension is constant)
- For some classes of matrices QTT analytic decompositions are known
- Toeplitz matrices have constant TT rank
- 3D Poisson operator has constant TT rank, more generally for FEM methods with simple mass and stiffness matrices

Computing the Tensor Train Decomposition

- Product of matrix with constant QTT rank and vector of dimension n has cost $\Theta(n \log n)$
- Given general vector, can compute TT decomposition by hierarchical SVD
- Faster algorithms leveraging low-rank of SVD are possible
- Can interpolate order d tensor with dimensions equal to n and TT rank r with cost $O(dnr^2)$
- Can efficiently obtain *cross approximation*, i.e. a lower-rank approximation to an existing TT approximation
- For order d tensor with n-dimensions and TT rank r, cross approximation cost is $O(dnr^3)$

Strassen's Algorithm

Strassen's algorithm $\begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}$

$$\begin{aligned}
\boldsymbol{M}_1 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{22}) \\
\boldsymbol{M}_2 &= (\boldsymbol{A}_{21} + \boldsymbol{A}_{22}) \cdot \boldsymbol{B}_{11} \\
\boldsymbol{M}_3 &= \boldsymbol{A}_{11} \cdot (\boldsymbol{B}_{12} - \boldsymbol{B}_{22}) \\
\boldsymbol{M}_4 &= \boldsymbol{A}_{22} \cdot (\boldsymbol{B}_{21} - \boldsymbol{B}_{11}) \\
\boldsymbol{M}_5 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{12}) \cdot \boldsymbol{B}_{22} \\
\boldsymbol{M}_6 &= (\boldsymbol{A}_{21} - \boldsymbol{A}_{11}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{12}) \\
\boldsymbol{M}_7 &= (\boldsymbol{A}_{12} - \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{21} + \boldsymbol{B}_{22})
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{C}_{11} &= \boldsymbol{M}_1 + \boldsymbol{M}_4 - \boldsymbol{M}_5 + \boldsymbol{M}_7 \\
\boldsymbol{C}_{21} &= \boldsymbol{M}_2 + \boldsymbol{M}_4 \\
\boldsymbol{C}_{12} &= \boldsymbol{M}_3 + \boldsymbol{M}_5 \\
\boldsymbol{C}_{22} &= \boldsymbol{M}_1 - \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_6
\end{aligned}$$

Minimize products $\Rightarrow$ minimize number of recursive calls

$$T(n) = 7T(n/2) + O(n^2) = O(7^{\log_2 n}) = O(n^{\log_2 7})$$

For convolution, DFT matrix reduces from naive $O(n^2)$ products to $O(n)$, both of these are bilinear algorithms

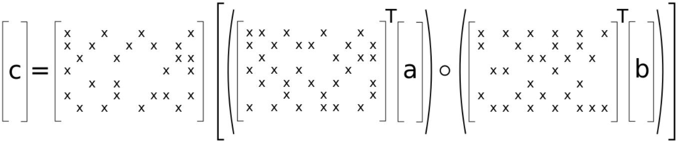
Bilinear Algorithms

Definition (Bilinear algorithms (V. Pan, 1984))

A bilinear algorithm $\Lambda = (\boldsymbol{F}^{(A)}, \boldsymbol{F}^{(B)}, \boldsymbol{F}^{(C)})$ computes

$$\boldsymbol{c} = \boldsymbol{F}^{(C)}[(\boldsymbol{F}^{(A)\mathsf{T}}\boldsymbol{a}) \odot (\boldsymbol{F}^{(B)\mathsf{T}}\boldsymbol{b})],$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are inputs and $\odot$ is the Hadamard (pointwise) product.



Bilinear Algorithms as Tensor Factorizations

A bilinear algorithm corresponds to a CP tensor decomposition

$$c_i = \sum_{r=1}^{r} f_{ir}^{(C)} \Big(\sum_j f_{jr}^{(A)} \boldsymbol{a}_j \Big) \Big(\sum_k f_{kr}^{(B)} b_k \Big)$$

$$= \sum_j \sum_k \Big(\sum_{r=1}^{r} f_{ir}^{(C)} f_{jr}^{(A)} f_{kr}^{(B)} \Big) a_j b_k$$

$$= \sum_j \sum_k t_{ijk} a_j b_k \quad \text{where} \quad t_{ijk} = \sum_{r=1}^{r} f_{ir}^{(C)} f_{jr}^{(A)} f_{kr}^{(B)}$$

For multiplication of $n \times n$ matrices,

- $\boldsymbol{T}$ is $n^2 \times n^2 \times n^2$
- Classical algorithm has rank $r = n^3$
- Strassen's algorithm has rank $r \approx n^{\log_2(7)}$

References

- Golub, Gene H., and Charles F. Van Loan. Matrix computations. Vol. 3. *JHU Press*, 2012.
- Kolda, Tamara G., and Brett W. Bader. Tensor decompositions and applications. *SIAM review* 51.3 (2009): 455-500.
- Khoromskij, Boris N. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems* 110.1 (2012): 1-19.
- Grasedyck, Lars, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen* 36.1 (2013): 53-78.

References

- Kazeev, Vladimir A., and Boris N. Khoromskij. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM Journal on Matrix Analysis and Applications* 33.3 (2012): 742-758.
- Khoromskaia, Venera, Boris Khoromskij, and Reinhold Schneider. QTT representation of the Hartree and exchange operators in electronic structure calculations. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.* 11.3 (2011): 327-341.
- Khoromskij, Boris N., and Ivan V. Oseledets. QTT approximation of elliptic solution operators in higher dimensions. *Russian Journal of Numerical Analysis and Mathematical Modelling* 26.3 (2011): 303-322.

References

- De Silva, Vin, and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008): 1084-1127.
- Pan, Victor. How can we speed up matrix multiplication? *SIAM review* 26.3 (1984): 393-415.
- Karlsson, Lars, Daniel Kressner, and André Uschmajew. Parallel algorithms for tensor completion in the CP format. *Parallel Computing* 57 (2016): 222-234.