NESTEROV-BASED ALTERNATING OPTIMIZATION FOR NONNEGATIVE TENSOR COMPLETION: ALGORITHM AND PARALLEL IMPLEMENTATION

Georgios Lourakis, Athanasios P. Liavas

School of Electrical and Computer Engineering, Technical University of Crete, Greece

Nonnegative Tensor Completion

- Let $\mathcal{X} \in \mathbb{R}_+^{I \times J \times K}$ be an incomplete tensor, and $\Omega \subseteq \{1 \dots I\} \times \{1 \dots J\} \times \{1 \dots K\}$ be the set of indices of its known entries [1].
- Also, let $\mathcal{M} \in \mathbb{R}^{I \times J \times K}$ with

$$\mathcal{M}(i,j,k) = \begin{cases} 1, & \text{if } (i,j,k) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

- We consider the Nonnegative Tensor Completion (NTC) problem

$$\min_{\mathbf{A} \geq \mathbf{0}, \mathbf{B} \geq \mathbf{0}, \mathbf{C} \geq \mathbf{0}} f_\Omega(\mathbf{A}, \mathbf{B}, \mathbf{C}) + \frac{\lambda}{2}\|\mathbf{A}\|_F^2 + \frac{\lambda}{2}\|\mathbf{B}\|_F^2 + \frac{\lambda}{2}\|\mathbf{C}\|_F^2, \tag{2}$$

where $\mathbf{A} = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_R] \in \mathbb{R}_+^{I \times R}$, $\mathbf{B} = [\mathbf{b}_1 \ \cdots \ \mathbf{b}_R] \in \mathbb{R}_+^{J \times R}$, $\mathbf{C} = [\mathbf{c}_1 \ \cdots \ \mathbf{c}_R] \in \mathbb{R}_+^{K \times R}$, and

$$f_\Omega(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \frac{1}{2}\|\mathcal{M} \circledast (\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!])\|_F^2, \tag{3}$$

with

$$[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{4}$$

Alternating optimization framework

- We can derive matrix-based equivalent expressions of f_Ω as

$$\begin{aligned} f_\Omega(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= \frac{1}{2}\left\|\mathbf{M_A} \circledast \left(\mathbf{X_A} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\right)\right\|_F^2 = \frac{1}{2}\left\|\mathbf{M_B} \circledast \left(\mathbf{X_B} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T\right)\right\|_F^2 \\ &= \frac{1}{2}\left\|\mathbf{M_C} \circledast \left(\mathbf{X_C} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T\right)\right\|_F^2, \end{aligned} \tag{5}$$

where $\mathbf{M_A}$, $\mathbf{M_B}$, $\mathbf{M_C}$, and $\mathbf{X_A}$, $\mathbf{X_B}$, $\mathbf{X_C}$ are the matrix unfoldings of $\mathcal{M}$ and $\mathcal{X}$, respectively.
- Solving (2) for $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ is a non-convex problem.
- Alternating optimization (AO):
  - Initialize $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0, l = 0$.
  1. $\mathbf{A}_{l+1} = \underset{\mathbf{A} \geq \mathbf{0}}{\operatorname{argmin}} f_\Omega(\mathbf{A}) := \frac{1}{2}\left\|\mathbf{M_A} \circledast \left(\mathbf{X_A} - \mathbf{A}(\mathbf{C}_l \odot \mathbf{B}_l)^T\right)\right\|_F^2 + \frac{\lambda}{2}\|\mathbf{A}\|_F^2$.
  2. $\mathbf{B}_{l+1} = \underset{\mathbf{B} \geq \mathbf{0}}{\operatorname{argmin}} f_\Omega(\mathbf{B}) := \frac{1}{2}\left\|\mathbf{M_B} \circledast \left(\mathbf{X_B} - \mathbf{B}(\mathbf{C}_l \odot \mathbf{A}_{l+1})^T\right)\right\|_F^2 + \frac{\lambda}{2}\|\mathbf{B}\|_F^2$.
  3. $\mathbf{C}_{l+1} = \underset{\mathbf{C} \geq \mathbf{0}}{\operatorname{argmin}} f_\Omega(\mathbf{C}) := \frac{1}{2}\left\|\mathbf{M_C} \circledast \left(\mathbf{X_C} - \mathbf{C}(\mathbf{B}_{l+1} \odot \mathbf{A}_{l+1})^T\right)\right\|_F^2 + \frac{\lambda}{2}\|\mathbf{C}\|_F^2$.
  - Iterate 1, 2, 3 until convergence.

Nonnegative Matrix Completion

- Let $\mathbf{X} \in \mathbb{R}_+^{m \times n}$ be an incomplete matrix, and $\Omega \subseteq \{1 \dots m\} \times \{1 \dots n\}$ be the set of indices of its known entries.
- Also, let $\mathbf{A} \in \mathbb{R}_+^{m \times r}$, $\mathbf{B} \in \mathbb{R}_+^{n \times r}$, and $\mathbf{M} \in \mathbb{R}^{m \times n}$ with

$$\mathbf{M}(i,j) = \begin{cases} 1, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

- We consider the Nonnegative Matrix Completion (NMC) problem

$$\min_{\mathbf{A} \geq \mathbf{0}} f_\Omega(\mathbf{A}) := \frac{1}{2}\left\|\mathbf{M} \circledast \left(\mathbf{X} - \mathbf{A}\mathbf{B}^T\right)\right\|_F^2 + \frac{\lambda}{2}\|\mathbf{A}\|_F^2. \tag{7}$$

- The gradient and the Hessian of f_Ω, at point $\mathbf{A}$, are given by

$$\nabla f_\Omega(\mathbf{A}) = -\left(\mathbf{M} \circledast \mathbf{X} - \mathbf{M} \circledast \mathbf{A}\mathbf{B}^T\right)\mathbf{B} + \lambda\mathbf{A}, \tag{8}$$

and

$$\nabla^2 f_\Omega(\mathbf{A}) = \left(\mathbf{B}^T \otimes \mathbf{I}\right) \mathtt{diag}^2(\mathtt{vec}(\mathbf{M}))(\mathbf{B} \otimes \mathbf{I}) + \lambda\mathbf{I}. \tag{9}$$

Nesterov-Type Algorithm for NMC

Algorithm 1: $\mathbf{A}_{\text{opt}} = \text{N\_NMC}(\mathbf{X}, \mathbf{M}, \mathbf{B}, \mathbf{A}_*)$

Input: $\mathbf{X}, \mathbf{M} \in \mathbb{R}_+^{m \times n}$, $\mathbf{B} \in \mathbb{R}_+^{n \times r}$, $\mathbf{A}_* \in \mathbb{R}_+^{m \times r}$, λ, μ, L

$\mathbf{W} = -(\mathbf{M} \circledast \mathbf{X})\mathbf{B}$

$q = \frac{\mu + \lambda}{L + \lambda}$

$\mathbf{A}_0 = \mathbf{Y}_0 = \mathbf{A}_*$

$\alpha_0 = 1$, $l = 0$

while (1) **do**

  $\nabla f_\Omega(\mathbf{Y}_l) = \mathbf{W} + (\mathbf{M} \circledast \mathbf{Y}_l\mathbf{B}^T)\mathbf{B} + \lambda\mathbf{Y}_l$

  if *(term_cond is TRUE)* **then**

    break

  else

    $\mathbf{A}_{l+1} = \left(\mathbf{Y}_l - \frac{1}{L+\lambda}\nabla f_\Omega(\mathbf{Y}_l)\right)_+$

    $\alpha_{l+1}^2 = (1 - \alpha_{l+1})\alpha_l^2 + q\alpha_{l+1}$

    $\beta_{l+1} = \frac{\alpha_l(1-\alpha_l)}{\alpha_l^2 + \alpha_{l+1}}$

    $\mathbf{Y}_{l+1} = \mathbf{A}_{l+1} + \beta_{l+1}(\mathbf{A}_{l+1} - \mathbf{A}_l)$

    $l = l + 1$

return $\mathbf{A}_l$.

Nesterov Based AO NTC

Algorithm 2: Nesterov-based AO NTC

Input: $\mathcal{X}$, Ω, $\mathbf{A}_0 > \mathbf{0}$, $\mathbf{B}_0 > \mathbf{0}$, $\mathbf{C}_0 > \mathbf{0}$.

$l = 0$

while (1) **do**

  $\mathbf{A}_{l+1} = \text{N\_NMC}(\mathbf{X_A}, \mathbf{M_A}, (\mathbf{C}_l \odot \mathbf{B}_l), \mathbf{A}_l)$

  $\mathbf{B}_{l+1} = \text{N\_NMC}(\mathbf{X_B}, \mathbf{M_B}, (\mathbf{C}_l \odot \mathbf{A}_{l+1}), \mathbf{B}_l)$

  $\mathbf{C}_{l+1} = \text{N\_NMC}(\mathbf{X_C}, \mathbf{M_C}, (\mathbf{A}_{l+1} \odot \mathbf{B}_{l+1}), \mathbf{C}_l)$

  if (term_cond is TRUE) **then** break; **endif**

  $l = l + 1$

return $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{C}_l$.

Computation of $\mathbf{W_A}$ and $\mathbf{Z_A}$

$$\mathbf{W_A} = (\mathbf{M_A} \circledast \mathbf{X_A})(\mathbf{C} \odot \mathbf{B}), \qquad \mathbf{Z_A} = (\mathbf{M_A} \circledast \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T)(\mathbf{C} \odot \mathbf{B}). \tag{10}$$

- The i-th row of $\mathbf{W_A}$, for $i = 1, \dots, I$, is computed as

$$\mathbf{W_A}(i,:) = \left(\mathbf{M_A}(i,:) \circledast \mathbf{X_A}(i,:)\right)(\mathbf{C} \odot \mathbf{B}). \tag{11}$$

- The computation involves the multiplication of a $(1 \times JK)$ row vector and a $(JK \times R)$ matrix.
- In order to reduce the computational complexity, we must exploit the sparsity of $\mathcal{X}$.
- Let nnz_i be the number of known entries in the i-th horizontal slice of $\mathcal{X}$. Also, let these known entries have indices $(i, j_q, k_q) \in \Omega$, for $q = 1, \dots, nnz_i$.
- The computation of the i-th row of $\mathbf{W_A}$ reduces to

$$\mathbf{W_A}(i,:) = \sum_{q=1}^{nnz_i} \mathcal{X}(i, j_q, k_q)\,\mathbf{C}(k_q,:) \circledast \mathbf{B}(j_q,:). \tag{12}$$

Efficient computation of $\mathbf{Z_A}$ can be achieved following similar arguments.

Distributed Memory Implementation

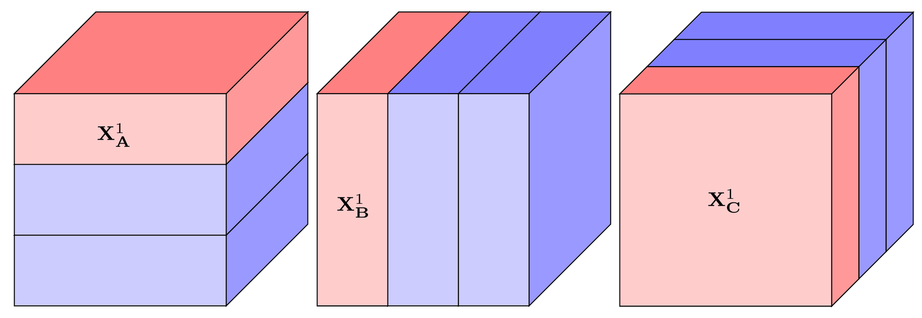

Figure: Tensor partition.

- We assume p processing elements [2].
- We partition the matricization $\mathbf{X_A}$ into p block rows as $\mathbf{X_A} = \left[\left(\mathbf{X_A^1}\right)^T \cdots \left(\mathbf{X_A^p}\right)^T\right]^T$, with $\mathbf{X_A^n} \in \mathbb{R}^{\frac{I}{p} \times JK}$. We partition similarly $\mathbf{X_B}$ and $\mathbf{X_C}$.
- The n-th block row of $\mathbf{X_A}$, $\mathbf{X_B}$, $\mathbf{X_C}$ have been allocated to the n-th processing element, for $n = 1, \dots, p$.
- We partition $\mathbf{A}_l$ into p block rows as $\mathbf{A}_l = \left[\left(\mathbf{A}_l^1\right)^T \cdots \left(\mathbf{A}_l^p\right)^T\right]^T$, with $\mathbf{A}_l^n \in \mathbb{R}^{\frac{I}{p} \times R}$, for $n = 1, \dots, p$.
- The n-th processor knows the whole $\mathbf{A}_l$, but updates the n-th block row of $\mathbf{A}_l$, $\mathbf{A}_l^n$, for $n = 1, \dots, p$.

Factor Update Implementation

The update of $\mathbf{A}_l$ is achieved via the updates of $\mathbf{A}_l^n$, for $n = 1, \dots, p$:

- The n-th processing element uses its local data $\mathbf{X_A^n}$, as well as the whole matrices $\mathbf{B}_l$ and $\mathbf{C}_l$, and computes the n-th block row of matrix $\mathbf{A}_{l+1}$, $\mathbf{A}_{l+1}^n$, via the Nesterov Matrix Completion algorithm.
- Each processing element broadcasts its output to all other processing elements; this operation can be implemented via an `Allgather` operation.

At this point, all processors know $\mathbf{A}_{l+1}$ and are ready for the update of $\mathbf{B}_l$ (and, then, of $\mathbf{C}_l$).

Numerical Experiments

- Results obtained from a Message Passing Interface (MPI) implementation of the AO NTC.
- The program is executed on a DELL PowerEdge R820 system with SandyBridge - Intel(R) Xeon(R) CPU E5 – 4650v2 (in total, 16 nodes with 40 cores each at 2.4 Gz) and 512 GB RAM per node.
- The matrix operations are implemented using routines of the C++ library Eigen [3].
- The performance metric we compute is the speedup attained using p processors.

Real Data

- The MovieLens 10M dataset [4], which contains time-stamped ratings of movies.
- Binning the time into seven-day-wide bins, results in a tensor of size $71567 \times 65133 \times 171$.
- The number of samples is 8M (99.99% sparsity).
- We first perform a random permutation on our data to resolve load imbalance issues.
- We measure the completion accuracy by measuring the mean squared error of 2M known ratings with our predictions.The mean squared error we achieved is 0.0033

(For the n-th known rating,with indices (i_n, j_n, k_n), we compute our prediction after rounding the quantity $\sum_{r=1}^R \mathbf{A}(i_n,:) \circledast \mathbf{B}(j_n,:) \circledast \mathbf{C}(k_n,:)$ to the closest integer.)

Speed Up

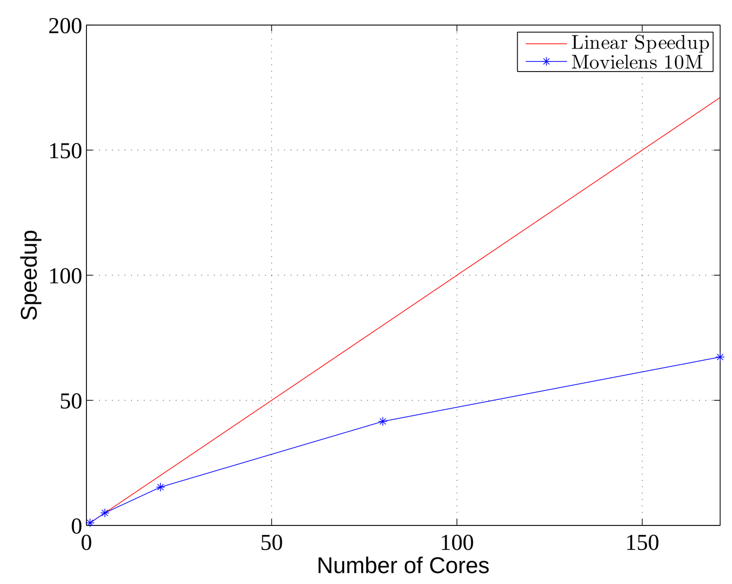

Figure: Speedup achieved for the MovieLens 10M dataset of size $71567 \times 65133 \times 171$ with p cores, for $p = 1, 5, 20, 171$.

Synthetic Data

- Synthetic data of the same size and sparsity level.
- True latent factors with i.i.d elements, uniformly distributed in $[0, 1]$.

Speed Up

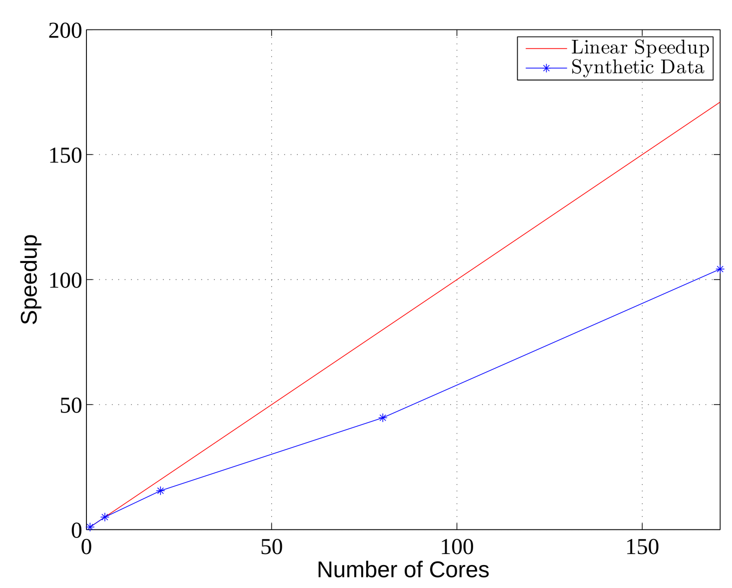

Figure: Speedup achieved for a $71567 \times 65133 \times 171$ tensor with p cores, for $p = 1, 5, 20, 171$.

References

1. L. Karlsson, D. Kressner and A. Uschmajew, "Parallel algorithms for tensor completion in the CP format," *Parallel Computing*, 2015.
2. K. Shin U. Kang, "Distributed Methods for High-dimensional and Large-scale Tensor Factorization," *IEEE International Conference on Data Mining*, pp. 989–994, 2014.
3. G. Guennebaud and B. Jacob *et al.*, "Eigen v3," *http://eigen.tuxfamily.org*, 2010.
4. F. Maxwell Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 1–19, June, 2015.