Nonnegative Tensor Completion

- Let \( X \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) be an incomplete tensor, and \( \Omega \subseteq \{1 \ldots i \} \times \{1 \ldots j \} \times \{1 \ldots k \} \) be the set of indices of its known entries [1].
- Also, let \( \mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) with \( \mathcal{A}(i,j,k) = 1 \) if \((i,j,k) \in \Omega; 0 \) otherwise.
- We consider the Nonnegative Tensor Completion (NTC) problem

\[
\min_{A: \mathcal{B}\subset \mathcal{C} \subset \mathcal{D}} f(A; B; C; D) = \frac{1}{2} \left\| A \right\|^2 + \frac{1}{2} \left\| B \right\|^2 + \frac{1}{2} \left\| C \right\|^2,
\]

where \( A = [a_{ij}] \in \mathbb{R}^{n_1 \times n_2 \times n_3} \), \( B = [b_{ij}] \in \mathbb{R}^{n_1 \times n_2 \times n_3} \), \( C = [c_{ij}] \in \mathbb{R}^{n_1 \times n_2 \times n_3} \), and

\[
f(A; B; C) = \frac{1}{2} \left\| \mathcal{A} \otimes (X - [A; B; C]) \right\|^2.
\]

with \( [A; B; C] = \sum_{i=1}^{R} b_i \otimes a_i \).

Alternating Matrix Completion

- We can derive matrix-based equivalent expressions of \( f_1 \) as

\[
f_1(A; B; C; D) = \frac{1}{2} \left\| M_A - (X - A) \otimes (B \otimes C) \right\|^2 + \frac{1}{2} \left\| M_B - (X - B) \otimes (A \otimes C) \right\|^2,
\]

where \( M_A, M_B, M_C \) are the matrix unfoldings of \( \mathcal{A} \) and \( \mathcal{X} \) respectively.
- Solving (2) for \( B, C, D \) is a non-convex problem.
- Alternating optimization (AO):
  - Initialize \( B_0, C_0 \).
  - Alternating optimization of (AO):
    - Compute \( A_{k+1} = \arg_{A} \min f_1(A; B_k; C_k; D_k) \).
    - Compute \( B_{k+1} = \arg_{B} \min f_1(A_{k+1}; B; C_k; D_k) \).
    - Compute \( C_{k+1} = \arg_{C} \min f_1(A_{k+1}; B_k; C; D_k) \).
- Iterate 1, 2, 3 until convergence.

Nonnegative Matrix Completion

- Let \( X \in \mathbb{R}^{n_1 \times n_2} \) be an incomplete matrix, and \( \Omega \subseteq \{1 \ldots m_1 \} \times \{1 \ldots n_2 \} \) be the set of indices of its known entries.
- Also, let \( X \in \mathbb{R}^{m_1 \times m_2} \) and \( M \in \mathbb{R}^{m_1 \times m_2} \) with

\[
M_{(i,j)} = \begin{cases} 1 & \text{if} \ (i,j) \in \Omega, \\ 0 & \text{otherwise,} \end{cases}
\]

- We consider the Nonnegative Matrix Completion (NMC) problem

\[
\min_{A: \Omega \neq \emptyset} f(A) = \frac{1}{2} \left\| X - \mathcal{R} \right\|^2 + \frac{1}{2} \left\| A - \mathcal{R} \right\|^2,
\]

where \( \mathcal{R} = \{ r_{12} \} \).
- The gradient and the Hessian of \( f_2 \) at point \( A \) are given by

\[
\nabla f_2(A) = -\left( M - \mathcal{R} \right) \mathcal{R} + \lambda A,
\]

and

\[
\nabla^2 f_2(A) = -\left( \mathcal{R}^\top \mathcal{R} + \lambda I \right) \mathcal{R} + \lambda I.
\]

Nesterov-Type Algorithm for NMC

Algorithm 1: Nesterov-type Alternating Optimization Algorithm

Input: \( X \in \mathbb{R}^{n_1 \times n_2} \), \( M \in \mathbb{R}^{n_1 \times n_2} \).

1. Initialize \( A_0 = X \).
2. while \( \| \nabla f_2(A_k) \| > \epsilon \)
   - Update \( A_{k+1} = \nabla f_2(A_k) + \lambda (A_k - A_{k-1}) \).
   - Return \( A_{k+1} \).

Nesterov Based AD NTC

Algorithm 2: Nesterov-type Alternating Optimization Algorithm

Input: \( X \in \mathbb{R}^{n_1 \times n_2 \times n_3} \).

1. Initialize \( A_0 \).
2. while \( \| \nabla f_2(A_k) \| > \epsilon \)
   - Update \( A_{k+1} = \nabla f_2(A_k) + \lambda (A_k - A_{k-1}) \).
   - Return \( A_{k+1} \).

Computation of \( W_k \) and \( Z_k \)

\[
W_k = (M_A \otimes X)(C \otimes B), \quad Z_k = (M_A \otimes (A \otimes C)) \mathcal{R}(C \otimes B).
\]

- The \( i \)-th row of \( W_k \), for \( i = 1 \ldots, n_1 \), is computed as

\[
W_k(i) = (M_A(i) \otimes X(i : \cdot, \cdot))(C \otimes B),
\]

- 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 \( X \).
- Let \( n_x \) be the number of known entries in the \( i \)-th horizontal slice of \( X \). Also, let known entries have \( (i, j, k) \in \Omega \), for \( 1 \leq n_x \leq n_3 \).
- The computation of the \( i \)-th row of \( W_k \) reduces to

\[
W_k(i) = \sum_{\Theta(i, \cdot)} X(i, j, k) C(k \cdot) \otimes B(j : \cdot),
\]

Efficient computation of \( Z_k \) can be achieved following similar arguments.

Distributed Memory Implementation

- We assume \( p \) processing elements [2].
- We partition the matrixization \( X_A \) into \( p \) block rows as

\[
X_A = (X_A)^T = \left( X_A^{(1)} \right)^T \ldots \left( X_A^{(p)} \right)^T,
\]

with \( X_A^{(i)} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \)

- The \( n \)-th block row of \( X_A \), \( X_C \), are allocated to the \( n \)-th processing element, for \( n = 1 \ldots, p \).
- We partition \( A \) into \( p \) block rows as

\[
A = (A)^T = \left( A^{(1)} \right)^T \ldots \left( A^{(p)} \right)^T,
\]

with \( A^{(i)} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) for \( n = 1 \ldots, p \).

Factor Update Implementation

The update of \( A_i \) is achieved via the updates of \( A^{(i)} \) for \( n = 1 \ldots, p \).

- The \( n \)-th processing element uses its local data \( X^{(i)} \), as well as the whole matrices \( B_i \) and \( C_i \), and
- Computes the \( n \)-th block row of matrix \( A_i \), \( A^{(i)} \), 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 \( A_i \) and are ready for the update of \( B_i \) (and, then, of \( C_i \)).

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 (20 nodes, 40 cores each at 2.4 GHz) 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 achieved 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 \( 71567 \times 65313 \times 171 \).
- 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 \( 20 \) known ratings with our predictions.
- The mean squared error we achieved is 0.0033

References