# Distributed Nuclear Norm Minimization Algorithm for Low-Rank Matrix Completion and Its Application to Low-Rank Tensor Completion


1st Katsumi Konishi
*Department of Information Science*
*Hosei University*
Tokyo, Japan
konishi@hosei.ac.jp

2nd Ryohei Sasaki
*School of Computer Science*
*Tokyo University of Technology*
Tokyo, Japan
sasakirh@stf.teu.ac.jp

3rd Toshihiro Furukawa
*Faculty of Engineering*
*Tokyo University of Science*
Tokyo, Japan


## I. Introduction

This paper deals with a low-rank matrix completion, which is a problem of estimating missing entries in a given low-rank matrix and has various applications in the field of signal processing such as collaborative filtering,and signal recovery problems.While the low-rank matrix completion is formulated as a matrix rank minimization problem, it is NP hard in general, and hence it is relaxed to a nuclear norm minimization problem [1], which is a problem of minimizing a sum of singular values of the matrix. Several algorithms have been proposed to solve nuclear norm minimization problems using the SVD based matrix shrinkage approach [2], [3], and a lot of numerical examples show that matrix shrinkage approaches achieve good performance.

The SVD based algorithm can be speeded up efficiently by using GPU computing, however, due to the limitation of memory on GPU, the SVD cannot be carried out for very huge matrices. In order to implement a low-rank matrix completion for very huge and enormous matrices, this paper proposes a distributed matrix shrinkage algorithm. A large matrix is divided into smaller matrices, and each of them is recovered by the matrix shrinkage approach using the synchronization of their singular vectors. This paper also deals with a low-rank tensor completion, which can be formulated as a nuclear norm minimization of its mode-$k$ expansion, and applies the proposed distributed algorithm. Numerical examples show that the proposed distributed algorithm works well.

## II. Preliminaries

### A. Low-Rank Matrix Completion

Most low-rank matrix completion algorithms have been proposed based on the following nuclear norm minimization problem,

$$\min \|X\|_* \text{ s.t. } X_{ij} = \bar{X}_{ij} \ \forall \ (i,j) \in \Omega, \tag{1}$$


This work was supported by JSPS KAKENHI Grant Number JP19H02163 and JP19K12050.


where $\|\cdot\|_*$ denotes the nuclear norm of a matrix, $X \in \boldsymbol{R}^{M\times N}$ is a design variable, $M \geq N$, $X_{ij}$ denotes the $(i,j)$ entry of $X$, $\bar{X}_{ij}$ is a given constant, and $\Omega$ denotes a index set of known entries of $X$. To obtain a solution of (1) several algorithms take a matrix shrinkage approach, and this paper focuses on the fixed point iterative algorithm [2] shown in Algorithm 1, where $S_\nu$ denotes the soft matrix shrinkage operator defined by $S_\nu(X) = U diag((\sigma_1-\nu)_+,\ldots,(\sigma_N-\nu)_+)V^T$, where $diag(a_1,a_2,\ldots,a_N)$ denotes a diagonal matrix whose diagonal elements are $a_1,\ a_2,\ \ldots,\ a_N$, $U$ and $V$ respectively denote matrices of left-singular vectors and right-singular vectors of $X$, $\sigma_i$ denotes the $i$th singular value, and $(a)_+ = \max(a,0)$.

### B. Low-Rank Tensor Completion

This paper deals with the Tucker rank of tensors and a low-rank $d$th-order tensor $\mathcal{P} \in \boldsymbol{R}^{M1\times\cdots\times M_d}$ which can be decomposed as $\mathcal{P} = \sum_{i_1=1}^{r_1}\cdots\sum_{i_d=1}^{r_d} \mathcal{G}_{i_1,\ldots,i_d} \boldsymbol{a}_{i_1}^{(1)} \otimes \cdots \otimes \boldsymbol{a}_{i_d}^{(d)}$ with a core tensor $\mathcal{G} \in \boldsymbol{R}^{r_1\times\cdots\times r_d}$ and vectors $\boldsymbol{a}_{i_k}^{(k)} \in \boldsymbol{R}^{M_k}$, $i_k \in \{1,\ldots,r_k\}$, for each $k \in \{1,\ldots,d\}$, where $\otimes$ denotes the Kronecker product. Let $\mathcal{P}^{(k)} \in \boldsymbol{R}^{M_k\times\Pi_{m\neq k}M_m}$ denote the mode-$k$ expansion of $\mathcal{P}$. Then we have that $\mathbf{rank}\mathcal{P}^{(k)} = r_k$, which implies that we can recover a low-rank tensor by solving the nuclear norm minimization problem of its mode-$k$ expansion as follows,

$$\min \sum_{k=1}^{d} \|\mathcal{P}^{(k)}\|_* \text{ s.t. } \mathcal{P}_{i_1,\ldots,i_d} = \bar{\mathcal{P}}_{i_1,\ldots,i_d} \ \forall (i_1,\ldots,i_d) \in \Omega_{\mathcal{P}}, \tag{2}$$

where $\Omega_{\mathcal{P}}$ denotes a index set of known entries of $\mathcal{P}$.

## III. Main Results

Suppose that $X \in \boldsymbol{R}^{M\times N}$ $(M \geq N)$ can be divided into $L$ matrices as $X = [X_1\ X_2\ \ldots\ X_L]$, where $X_i \in \boldsymbol{R}^{m\times N}$ and $M = mL$. Denoting a matrix of left-singular vectors and singular values of $X_i$ by $U_i = [\boldsymbol{u}_1^i\ \boldsymbol{u}_2^i\ \ldots\ \boldsymbol{u}_M^i]$ and $\sigma_1^i \geq \sigma_2^i \geq \ldots \geq \sigma_M^i$, respectively, let the SVD of $X_i$ be given by $X_i = U_i diag(\sigma_1^i,\sigma_2^i,\ldots,\sigma_M^i)V_i^T$. If $X$ and $X_i$'s have the same rank $r$ for all $i$, the space spanned

**Algorithm 1** Fixed point iterative algorithm.

**Require:** $X^0, \nu, \bar{X}$

$X \leftarrow X^0$

**repeat**

  $X_{ij} \leftarrow \bar{X}_{ij}$ for all $(i,j) \in \Omega$; $\quad X \leftarrow S_\nu(X)$

**until** converge

**Ensure:** $X$

**Algorithm 2** Distributed matrix shrinkage iterative algorithm.

**Require:** $X^0, \nu, \alpha, L, T, \bar{X}, \Omega$

$X \leftarrow X^0$

**repeat**

  $[X_1\ X_2\ \ldots\ X_L] \leftarrow X$

  **for** $i = 1$ to $L$ **do**

    **for** $loop = 1$ to $T$ **do**

      $[U_i,\ \sigma_1^i,\ \ldots,\ \sigma_N^i,\ V_i] \leftarrow \mathrm{SVD}(X_i)$

      $r \leftarrow \max_i(\arg\max_l \sigma_l^i$ subject to $\sigma_l^i \geq \alpha\sigma_1^i)$

      $W_i \leftarrow \frac{1}{L}[\sum_{j=1}^L U_j^r U_j^{rT} U_i^r \ \sum_{j=1}^L \tilde{U}_j^r \tilde{U}_j^{rT} \tilde{U}_i^r]$

      $X_i \leftarrow W_i diag([(\sigma_1^i - \nu)_+\ \ldots\ (\sigma_N^i - \nu)_+])V_i^T$

    **end for**

  **end for**

  $X \leftarrow [X_1\ X_2\ \ldots\ X_L]; \quad X_{i,j} \leftarrow \bar{X}_{i,j}\ \forall (i,j) \in \Omega$

**until** converge

**Ensure:** $X$

by $\boldsymbol{u}_1^i,\ \boldsymbol{u}_2^i,\ \ldots\ \boldsymbol{u}_r^i$ is equal to the space spanned by $\boldsymbol{u}_1^j,\ \boldsymbol{u}_2^j,\ \ldots\ \boldsymbol{u}_r^j$. This implies that $\boldsymbol{u}_1^i,\ \boldsymbol{u}_2^i,\ \ldots\ \boldsymbol{u}_r^i$ are obtained again by projecting them onto the space spanned by $\boldsymbol{u}_1^j,\ \boldsymbol{u}_2^j,\ \ldots\ \boldsymbol{u}_r^j$. Therefore, if $\mathbf{rank}X = \mathbf{rank}X_1 = \ldots = \mathbf{rank}X_L$, it holds that $\boldsymbol{u}_p^i = \sum_{q=1}^r (\boldsymbol{u}_p^{i\,T} \boldsymbol{u}_q^j)\boldsymbol{u}_q^j$ for any $i$ and $j$. This equation can be rewritten using $U_i$ and $U_j$ as $U_i^r = U_j^r U_j^{rT} U_i^r$. Similarly, focusing on the null space spanned by $\boldsymbol{u}_{r+1}^i\ \boldsymbol{u}_{r+2}^i\ \ldots\ \boldsymbol{u}_M^i$, we have that $\tilde{U}_i^r = \tilde{U}_j^r \tilde{U}_j^{rT} \tilde{U}_i^r$, where $\tilde{U}_i^r = [\boldsymbol{u}_{r+1}^i\ \boldsymbol{u}_{r+2}^i\ \ldots\ \boldsymbol{u}_M^i]$. Summing up these equations from $j = 1$ to $L$, we have that $U_i^r = \frac{1}{L}\sum_{j=1}^L U_j^r U_j^{rT} U_i^r$ and that $\tilde{U}_i^r = \frac{1}{L}\sum_{j=1}^L \tilde{U}_j^r \tilde{U}_j^{rT} \tilde{U}_i^r$. Based on these equations, this paper provides the following problem,

$$\begin{aligned} \min \sum_{i=1}^L \|X_i\|_* \quad \text{s.t. } & X_{i,j} = \bar{X}_{i,j}\ \forall (i,j) \in \Omega \\ & U_i^r = \tfrac{1}{L}\textstyle\sum_{j=1}^L U_j^r U_j^{rT} U_i^r\ \forall i = 1, \ldots, L, \\ & \tilde{U}_i^r = \tfrac{1}{L}\textstyle\sum_{j=1}^L \tilde{U}_j^r \tilde{U}_j^{rT} \tilde{U}_i^r\ \forall i = 1, \ldots, L, \end{aligned}$$

and a distributed matrix shrinkage iterative algorithm for the above problem is proposed as shown in Algorithm 2. Using this algorithm, this paper proposes Algorithm 3 for a low-rank tensor completion based on (2).

## IV. Numerical Examples

This section gives numerical examples applying Algorithm 3 to the 3rd-order tensor completion. All numerical examples were run in MATLAB 2022b with Parallel Computing Toolbox on a single GPU of an NVIDIA GeForce RTX 3090 with 24GB memory. A low-rank $1,000 \times 2,000 \times 1,000$ tensor $\mathcal{P}^{true}$ is given by a core tensor $\mathcal{G} \in \boldsymbol{R}^{3\times5\times10}$,

**Algorithm 3** Low-rank tensor completion algorithm.

**Require:** $\mathcal{P}^0, \nu, \alpha, L, T, \bar{\mathcal{P}}, \Omega_\mathcal{P}$

$\mathcal{P} \leftarrow \mathcal{P}^0$

**repeat**

  **for** $k = 1$ to $d$ **do**

    $X \leftarrow$ the mode-$k$ expansion of $\mathcal{P}$

    do lines 3-11 in Algorithm 2

  **end for**

  $\mathcal{P}_{i_1,\ldots,i_d} \leftarrow \bar{\mathcal{P}}_{i_1,\ldots,i_d}\ \forall (i_1, \ldots, i_d) \in \Omega_\mathcal{P}$

**until** converge

**Ensure:** $X$

TABLE I
Relative error and computing time.

| $L$ | relative error | computing time [s] |
|---|---|---|
| 2 | $4.755 \times 10^{-4}$ | 161.1 |
| 4 | $8.492 \times 10^{-4}$ | 168.4 |
| 8 | $1.080 \times 10^{-3}$ | 186.9 |
| 16 | $1.568 \times 10^{-3}$ | 223.6 |

$\boldsymbol{a}_{i_k}^{(1)} \in \boldsymbol{R}^{1000}$, $\boldsymbol{a}_{i_k}^{(2)} \in \boldsymbol{R}^{2000}$ and $\boldsymbol{a}_{i_k}^{(3)} \in \boldsymbol{R}^{1000}$ generated using i.i.d. Gaussian entries, and the tensor is regularized such that its Frobenius norm is equal to 1. The index set $\Omega_\mathcal{P}$ is generated using Bernoulli $\{0, 1\}$ random variables where $5\%$ of elements in $\mathcal{P}$ are known. For each experiment, we use the parameters $\nu = 2 \times 10^{-5}/L$, $\alpha = 1 \times 10^{-3}$ and $T = 20$, and the initial value $\mathcal{P}_0$ is provided by substituting $\mathcal{P}_{i_1,i_2,i_3}^{true}$ with 0 for $(i_1, i_2, i_3) \notin \Omega$. To reduce computing cost, the randomized SVD [4] was applied. Algorithm 3 is applied with $L \in \{2, 4, 8, 16\}$ for 20 loops, and Table I shows the results of relative errors $\|\mathcal{P} - \mathcal{P}^{true}\|_F / \|\mathcal{P}^{true}\|_F$ and computing time. Due to the limitation of memory on GPU, the algorithm with $L = 1$ cannot be carried out. We can see that the distributed algorithm work well and that the recovery accuracy does not get worse so much for larger $L$.

## V. Conclusion

This paper dealt with low-rank matrix and tensor completion problems and proposed a distributed matrix shrinkage iterative algorithm. The proposed algorithm divides a large matrix into smaller matrices and applies the matrix shrinkage approach to them using the synchronization of their left singular vectors. Numerical examples show that the proposed algorithm works well for the low-rank tensor completion.

## References


[1] E. J. Candes, B. Recht, "Exact matrix completion via convex optimization," Foundations of Computational Mathematics, vol. 9(6), pp. 717–772, 2009

[2] S. Ma, D. Goldfarb, L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," Mathematical Programming, vol. 128 (1), pp. 321–254, 2011

[3] K. Konishi, K. Uruma, T. Takahashi, and T. Furukawa, "Iterative partial matrix shrinkage algorithm for matrix rank minimization," Signal Processing, vol. 100, pp. 124–131, 2014.

[4] N. Halko, P. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," SIAM Review, vol. 53, Issue 2, pp. 217–288, 2011