README for the Matlab Package of Several LRTC algorithms

Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye

I. About This Package

This Matlab package implements several low rank tensor completion algorithms proposed in our paper [2], including simple LRTC (SiLRTC), fast LRTC (FaLRTC), high accuracy LRTC (HaLRTC), and two extended algorithms of SiLRTC and FaLRTC namely SiLRTCnr and FaLRTCnr respectively[1]. The algorithms in this package are more efficient than the original LRTC algorithm proposed in our ICCV paper [1]. Hence, this package is an improved version of the LRTC algorithm proposed in our ICCV paper [1]. One can run the "example.m" file to have a quick glance of all algorithms.

II. Problem

This Matlab package aims to solve the following optimization problem:

$$\begin{aligned} \min_{\mathcal{X}} : \ & \|\mathcal{X}\|_* := \sum_{i=1}^{n} \alpha_i \|\mathcal{X}_{(i)}\|_* \\ s.t. : \ & \mathcal{X}_\Omega = \mathcal{T}_\Omega, \end{aligned} \tag{1}$$

where $\mathcal{X}$ is an n-mode (dimensional) tensor and $\alpha_i > 0$. See the paper [2] for more details.

III. Algorithms

Before we explain all algorithms implemented in this package, let us introduce their common inputs and outputs. First, all of them return

X	the tensor estimated by this algorithm
$errList$	the list of differences between two consecutive iterations

.

Second the inputs shared by all algorithms are:

T	the input tensor with missing entries; the missing entries can be filled by any value
$Omega(\Omega)$	a binary tensor with the same size as T: 0 means missing and 1 means observed
$alpha(\alpha)$	the coefficient vector α which defines the tensor trace norm in Eq. (1)
$maxIter$	the maximal iteration number
$epsilon$	the tolerance of the difference between two consecutive iterations
$X0$	the initial value of $\mathcal{X}$

.

A. SiLRTC

This algorithm is a simplified version of the LRTC algorithm proposed in our ICCV paper [1]. It relaxes the original problem Eq. (1) into the following problem:

$$\begin{aligned} \min_{\mathcal{X}, M_i} : \ & \sum_{i=1}^{n} \alpha_i \|M_i\|_* + \frac{\beta_i}{2} \|\mathcal{X}_{(i)} - M_i\|_F^2 \\ s.t. : \ & \mathcal{X}_\Omega = \mathcal{T}_\Omega. \end{aligned} \tag{2}$$

One can see that when β_i goes to positive infinity, the solution of Eq. (2) will converge to that of Eq. (1). Note that this algorithm is a simplified version of the LRTC algorithm proposed in [1] by removing a redundant variable. The Matlab

Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye are with Arizona State University, Tempe, AZ, 85287.
E-mail: {Ji.Liu, pmusials, Peter.Wonka, and Jieping.Ye}@asu.edu

[1] These two algorithms are not included in our paper, but we think that it is worth implementing them in our package.

function implementing **Algorithm 1** in [2] is defined as follows:

$$
\begin{aligned}
[X, errList] = & SiLRTC(\\
& T, \\
& Omega, \\
& alpha, \\
& beta, \qquad \text{\% the relaxation vector } \beta = [\beta_1, ..., \beta_n] \text{ defined in Eq. (2)} \\
& maxIter, \\
& epsilon, \\
& X0);
\end{aligned}
$$

B. SILRTCnr (SiLRTC without Relaxation)

This algorithm basically solves the same formulation in Eq. (2) as the SiLRTC algorithm, except increasing β iteratively by $\beta^{k+1} = \beta^k / factor$ where $factor$ is a constant in the range $(0, 1]$ and k indicates the k^{th} iteration. We define the Matlab function in the following:

$$
\begin{aligned}
[X, errList] = & SiLRTCnr(\\
& T, \\
& Omega, \\
& alpha, \\
& factor, \qquad \text{\% the increasing rate of } \beta \text{ defined above} \\
& maxIter, \\
& epsilon, \\
& X0);
\end{aligned}
$$

C. FaLRTC

This algorithm relaxes the dual variables and solves the following problem:

$$
\begin{aligned}
\min_{\mathcal{X}} : & \sum_{i=1}^{n} \max_{\|\mathcal{Y}_{i(i)}\| \le 1} \alpha_i \langle \mathcal{X}, \mathcal{Y}_i \rangle - \frac{\mu_i}{2} \|\mathcal{Y}\|_F^2 \\
s.t. : & \mathcal{X}_\Omega = \mathcal{Y}_\Omega.
\end{aligned}
\tag{3}
$$

where $\mu_i > 0$. One can verify that if $\mu := [\mu_1, ..., \mu_2] = 0$, the problem is identical to the original problem in Eq. (1). This following function implements **Algorithm 2** in [2] except iteratively updating μ in the way introduced in the end of Section 5:

$$
\begin{aligned}
[X, errList] = & FaLRTC(\\
& T, \\
& Omega, \\
& alpha, \\
& mu, \qquad \text{\% the relaxation vector } \mu \text{ defined in Eq. (3)} \\
& L0, \qquad \text{\% the initial step size parameter, a positive number small enough, i.e., stepsize = 1 / L} \\
& C, \qquad \text{\% the decreasing rate in the range (0.5, 1)} \\
& maxIter, \\
& epsilon, \\
& X0);
\end{aligned}
$$

D. FaLRTCnr (FaLRTC without Relaxation)

Basically, the FaLRTCnr algorithm solves Eq. (3) as well. The motivation of the FaLRTCnr algorithm is actually similar to the SiLRTCnr algorithm, i.e., changing the relaxation parameter iteratively such that it is closer and closer to the original problem in Eq. (1). This algorithm updates $\mu_{k+1} = A/k^{factor}$ where $factor$ is a positive constant given by the user and A is a constant determined by the input tensor. We take $factor = 2$ in our example file. The function in our package is defined as follows:

$[X, errList] = FaLRTCnr($
$T,$
$Omega,$
$alpha,$
$factor,$ % the parameter controlling the decreasing speed of the relaxation paramter μ in Eq. (1)
$L0,$ % the initial step size parameter, a positive number small enough, i.e., stepsize = 1 / L
$C,$ % the decreasing rate in the range (0.5, 1)
$maxIter,$
$epsilon,$
$X0);$

E. HaLRTC

The algorithm solves the original problem in Eq. (1). The function is defined as follows:

$[X, errList] = FaLRTC($
$T,$
$Omega,$
$alpha,$
$rho,$ % a number small enough, empirically $rho = 10^{-7}$
$maxIter,$
$epsilon,$
$X0);$

REFERENCES

[1] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *ICCV*, pages 2114–2121, 2009.
[2] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.