Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466

Applications and Applied Mathematics:
An International Journal
(AAM)

Vol. 14, Issue 2 (December 2019), pp. 1145 – 1156

An ADMM-Factorization Algorithm for Low Rank Matrix Completion

[1]Rahman Taleghani and [2]Maziar Salahi

Department of Applied Mathematics
Faculty of Mathematical Sciences
University of Guilan
Rasht, Iran
[1]rahman.taleghani@gmail.com; [2]salahim@guilan.ac.ir

Received: January 3, 2019; Accepted: May 5, 2019

Abstract

In this paper, we propose an Alternating Direction Method of Multipliers (ADMM) based algorithm that is taking advantage of factorization for the fixed rank matrix completion problem. The convergence of the proposed algorithm to the KKT point is discussed. Finally, on several classes of test problems, its efficiency is compared with several efficient algorithms from the literature.

Keywords: Matrix completion; Factorization; Alternating direction method of multipliers; Convex optimization

MSC 2010 No.: 15A83, 15A18, 90C25

1. Introduction

The problem of recovering a low rank matrix from a sampling of its entries has attracted significant attention in various applications such as collaborative prediction (see Rennie and Srebro (2005)), model reduction (see Liu and Vandenberghe (2009)), data mining and pattern recognitions (see

Eldèn (2007)), multi-class learning (see Argyriou et al. (2008), Obozinski et al. (2010)) and etc. As a motivating example, consider traffic issue that is an irritating problem. City traffic measurements can be organized into a matrix where rows are sources, columns are destinations and each entry holds the volume of traffic that passed from a particular source to a particular destination. There are several reasons to collect cities' traffic volume data, for example, to identify the congestion area or to estimate the pollution through a city. At the same time, it is not possible to have an information about all destinations, that is some entries of such matrix is unknown. Therefore, to assess such problem, one should be able to estimate these unknown values (see Ruchansky (2016)).

The general form of low rank matrix completion problem that finds the lowest rank matrix from its known entries, is as follows:

$$\begin{aligned} &\min_{X\in\mathbb{R}^{m\times n}} rank(X) \\ &s.t \;\; X_{ij} = M_{ij} \quad \forall (i,j) \in \Omega, \end{aligned} \tag{1}$$

where M is a matrix that its known entries are in the index set $\Omega \subset \{(i,j) : 1 \leq i \leq m, 1 \leq j \leq n\}$. Problem (1) is generally NP-hard (see Vandenberghe and Boyd (1996)). In Candès and Recht (2009), the authors proved that under the assumption of randomly sampled entries, to recover matrix with high probability, the matrix $M \in \mathbb{R}^{m\times n}$ of rank r should have at least $|\Omega| > Cn^{1.2}rlog(n)$ sampled entries when the rank is replaced with nuclear norm:

$$\begin{aligned} &\min_{X\in\mathbb{R}^{m\times n}} ||X||_* \\ &s.t \;\; X_{ij} = M_{ij} \quad \forall (i,j) \in \Omega, \end{aligned} \tag{2}$$

where $||.||_*$ denotes the sum of singular values of X. In Candès and Tao (2010), the author presented optimality results for the minimum number of observed entries for which (2) results to exact recovery. In Keshavan et al. (2010), the authors used singular values on Grassman manifold optimization, their algorithm minimizes the error over the observed entries while the rank of data matrix is small. Soft-thresholding operation on the singular values of a certain matrix at each iterations is used (see Cai et al. (2010)). An accelerated proximal gradient algorithm proposed in Toh and Yun (2010) which terminates in $O(1/\sqrt{\epsilon})$ iterations with an ϵ-optimal solution. Zhang and Cheng (2010) proposed a non-linear constrained quadratic programming problem and designed a new algorithm based on projection method and Landweber iteration. A homotopy approach together with an approximate singular value decomposition (SVD) is used in Ma et al. (2011). Another algorithm that used manifold structure in its line-search updates is Ngo and Saad (2012). However, using SVD strategy is expensive when the size of data matrix increases. Thus, it is desirable to use other approaches. One of the popular heuristic approach to solve (1) is matrix factorization. Based on the simple factorization, instead of solving (1), several algorithms are designed to solve the non-convex problem, where usually alternating minimization scheme is used. For example, Wen et al. (2012) proposed an approach to solve the following problem

$$\begin{aligned} &\min_{X,Y,Z} \tfrac{1}{2}||XY - Z||_F^2 \\ &s.t \;\; Z_{ij} = M_{ij} \quad \forall (i,j) \in \Omega, \end{aligned} \tag{3}$$

where $||.||_F$ is the Frobenius norm, $X \in \mathbb{R}^{m\times s}$, $Y \in \mathbb{R}^{s\times n}$, $Z \in \mathbb{R}^{m\times n}$ and s is an integer which is the rank estimate. They performed a low rank matrix factorization based on successive over-relaxation iteration. Vandereycken (2013) proposed a new algorithm that minimizes the least

squares distance on the sampling set over the Riemannian manifold of fixed rank matrices. Another algorithm that used alternating direction approach is Xiao et al. (2013). In Wang and Li (2015), the authors proposed mean value algorithm for the Toeplitz matrix completion. Wang et al. (2015) proposed a two-step proximal gradient algorithm to solve nuclear norm regularized least squares. An alternating steepest descent algorithm and a scaled variant of it are introduced in Tanner and Wei (2016) for the fixed-rank matrix completion problem. Most recently, Huang and Wolkowicz (2018) combined nuclear norm minimization and facial reduction theory for solving matrix completion problem that has high accuracy.

In this paper, matrix factorization is combined with the alternating direction method of multipliers (ADMM) approach to solve (3). The rest of this paper is organized as follows. In Section 2, we present the ADMM-Factorization algorithm and then its convergence to the KKT point is discussed in Section 3. Finally, in Section 4, the numerical experiments on several classes of test problems are carried out, comparing the new algorithm with several known and efficient algorithms from the literature.

2. ADMM-Factorization algorithm

Recently, ADMM algorithms have been successfully used for solving several optimization problems by breaking them into smaller sub-problems (for example, see Steidl and Teuber (2010), Ng et al. (2010), Salahi and Taati (2017), and Taleghani and Salahi (2018)). Consider the augmented Lagrangian associated to (3) as follows:

$$L_\rho(X,Y,Z,\Lambda)=\frac{1}{2}||XY-Z||_F^2+\Lambda\cdot(P_\Omega(Z)-P_\Omega(M))+\frac{\rho}{2}||P_\Omega(Z)-P_\Omega(M)||_F^2,$$

where Λ is the Lagrange multipliers, ρ is the penalty parameter and $P_\Omega(A)$ is a projection that its known entries are in Ω and its unknown entries are zeros, that is

$$(P_\Omega(A))_{ij}=\begin{cases}A_{ij}, & \text{if } (i,j)\in\Omega,\\ 0, & \text{otherwise,}\end{cases}$$

and

$$A\cdot B=\sum_{i,j}A_{ij}B_{ij}.$$

Since model (3) is a non-convex optimization problem, we use alternating strategy for minimizing each variable. Thus, the ADMM steps for (3) can be outlined as follows:

$$X_{k+1}\leftarrow argmin_X L_\rho(X,Y_k,Z_k,\Lambda_k), \tag{4}$$
$$Y_{k+1}\leftarrow argmin_Y L_\rho(X_{k+1},Y,Z_k,\Lambda_k), \tag{5}$$
$$Z_{k+1}\leftarrow argmin_Z L_\rho(X_{k+1},Y_{k+1},Z,\Lambda_k), \tag{6}$$
$$\Lambda_{k+1}\leftarrow \Lambda_k+\gamma\rho(P_\Omega(Z_{k+1})-P_\Omega(M)), \tag{7}$$

where in (7), $\gamma \in (0, 1.618)$ is the step length. Moreover, due to the convexity the above steps can be written as follows:

$$X_{k+1} = (Z_k Y_k^T)(Y_k Y_k^T)^{-1}, \tag{8}$$

$$Y_{k+1} = (X_{k+1}^T X_{k+1})^{-1}(X_{k+1}^T Z_k), \tag{9}$$

$$Z_{k+1} = P_{\Omega^c}(X_{k+1}Y_{k+1}) + \frac{1}{1+\rho}(P_\Omega(X_{k+1}Y_{k+1}) - P_\Omega(\Lambda_k) + \rho P_\Omega(M)), \tag{10}$$

$$P_\Omega(\Lambda_{k+1}) = P_\Omega(\Lambda_k) + \gamma\rho(P_\Omega(Z_{k+1}) - P_\Omega(M)). \tag{11}$$

Now, the ADMM-Factorization algorithm for solving (3) can be outlined as follows.

ADMM-Factorization algorithm for solving (3)

Input: Matrices Y_0, Z_0 and Λ_0, $tol > 0$, appropriate penalty parameter $\rho > 0$, $maxiter$ and $k = 0$.
For $k = 1, \cdots, maxiter$ **do**
Perform (8) to (11),
If stopping criterion is satisfied, **then exit** with $(X_{k+1}, Y_{k+1}, Z_{k+1}, \Lambda_{k+1})$.
end if
end for

It is worth noting that the main computational costs of ADMM-Factorization algorithm is two least squares problems (8) and (9) that each of them requires $2mnr + 2nr^2 + O(r^3)$ and $2mnr + 2mr^2 + O(r^3)$ operations, respectively. The computational costs of (10) and (11) are also $2|\Omega^c|r + 2|\Omega|(2r + 1)$ and $2|\Omega|r + |\Omega|$ operations, respectively. Therefore, the total computational costs of ADMM-Factorization algorithm is $6mnr + 2(m + n)r^2 + |\Omega|(4r + 3)$ operations.

3. Convergence results

In this section, we discuss the convergence of ADMM-Factorization algorithm to a KKT point as in Xu et al. (2012). The KKT conditions for (3) are as follows:

$$\begin{aligned} &(XY - Z)Y^T = 0, \;\; X^T(XY - Z) = 0, \;\; P_{\Omega^c}(Z) - P_{\Omega^c}(XY) = 0, \\ &P_\Omega(Z) - P_\Omega(XY) = -P_\Omega(\Lambda), \;\; P_\Omega(Z) - P_\Omega(M) = 0. \end{aligned} \tag{12}$$

Lemma 3.1.

Suppose that $\{(\Lambda_k)\}$ is bounded and

$$\sum_{k=0}^{\infty} ||\Lambda_{k+1} - \Lambda_k||_F^2 < \infty. \tag{13}$$

Let $W_k = (X_k, Y_k, Z_k)$, then $||W_{k+1} - W_k||_F^2 \to 0$ as $k \to \infty$.

Proof:

The augmented Lagrangian function of (3) can be written as

$$L_\rho(X,Y,Z,\Lambda) = \frac{1}{2}||XY - Z||_F^2 + \frac{\rho}{2}||P_\Omega(M) - P_\Omega(Z) + \frac{\Lambda}{\rho}||_F^2 - \frac{1}{2}||\Lambda||_F^2. \tag{14}$$

Obviously L_ρ is strongly convex with respect to each variable X, Y, Z, Λ. For example, when Y, Z, Λ are fixed, we have $L_\rho(X_{k+1}) - L_\rho(X_k) \geq \partial_X L_\rho(X_{k+1}) \cdot \Delta X + ||\Delta X||_F^2$. If X_{k+1} is the minimizer of $L_\rho(X)$ at k-th iteration, then $\partial_X L_\rho(X_{k+1}) \cdot \Delta X \geq 0$. So we have $L_\rho(X_{k+1}) - L_\rho(X_k) \geq ||\Delta X||_F^2$, or

$$L_\rho(X_{k+1}) - L_\rho(X_k) \geq ||X_{k+1} - X_k||_F^2. \tag{15}$$

Similarly, we have

$$L_\rho(Y_{k+1}) - L_\rho(Y_k) \geq ||Y_{k+1} - Y_k||_F^2, \tag{16}$$

$$L_\rho(Z_{k+1}) - L_\rho(Z_k) \geq \rho||Z_{k+1} - Z_k||_F^2. \tag{17}$$

Now, let $c := min\{\rho, 1\}$. Then, by (15), (16) and (17) we have

$$\begin{aligned} &L_\rho(W_k, \Lambda_k) - L_\rho(W_{k+1}, \Lambda_{k+1}) \\ &\qquad = L_\rho(W_k, \Lambda_k) - L_\rho(W_{k+1},\ \Lambda_k) + L_\rho(W_{k+1}, \Lambda_k) - L_\rho(W_{k+1}, \Lambda_{k+1}) \\ &\qquad \geq c||W_k - W_{k+1}||_F^2 - \frac{1}{\gamma\rho}||\Lambda_k - \Lambda_{k+1}||_F^2 \\ &\qquad \geq c||W_k - W_{k+1}||_F^2 - \frac{1}{c\gamma}||\Lambda_k - \Lambda_{k+1}||_F^2. \end{aligned}$$

Since L_ρ is bounded, we have $\sum_{k=0}^{\infty} c||W_k - W_{k+1}||_F^2 - \sum_{k=0}^{\infty} \frac{1}{c\gamma}||\Lambda_k - \Lambda_{k+1}||_F^2 < \infty$. Now, using (13) we have $\sum_{k=0}^{\infty} c||W_k - W_{k+1}||_F^2 < \infty$, therefore $W_{k+1} - W_k \to 0$. ∎

Theorem 3.2.

Let $(X^*, Y^*, Z^*, \Lambda^*)$ be an accumulation point of $(X_k, Y_k, Z_k, \Lambda_k)$ generated by the ADMM-Factorization algorithm. Then, by the boundedness of $\{(\Lambda_k)\}$ and $\sum_{k=0}^{\infty} ||\Lambda_{k+1} - \Lambda_k||_F^2 \leq \infty$, such accumulation point satisfies the KKT conditions (12).

Proof:

From (8), (9) and (11) we have

$$\begin{aligned} (X_{k+1} - X_k)Y_k Y_k^T &= -(X_k Y_k - Z_k)Y_k^T, \\ X_{k+1}^T X_{k+1}(Y_{k+1} - Y_k) &= -X_{k+1}^T(X_{k+1}Y_k - Z_k), \\ \Lambda_{k+1} - \Lambda_k &= \gamma\rho(P_\Omega(Z_{k+1}) - P_\Omega(M)). \end{aligned}$$

Moreover, Lemma 3.1 implies that $(X_k Y_k - Z_k)Y_k^T \to 0, X_{k+1}^T(X_{k+1}Y_k - Z_k) \to 0$ and $P_\Omega(Z_{k+1}) - P_\Omega(M) \to 0$. Since (X^*, Y^*, Z^*) is an accumulation point of $\{(X_k, Y_k, Z_k)\}$, then there exists a subsequence $\{(X_{n_k}, Y_{n_k}, Z_{n_k})\}$ converging to (X^*, Y^*, Z^*). In addition, the boundedness of $\{\Lambda_k\}$ implies that there is a subsequence $\{\Lambda_{n_k}\}$ converging to Λ^*. Also, (10) means that

$P_{\Omega^c}(Z_{k+1}) - P_{\Omega^c}(X_{k+1}Y_{k+1}) = 0$ and $P_{\Omega}(Z_{k+1}) - P_{\Omega}(X_{k+1}Y_{k+1}) + P_{\Omega}(\Lambda_k) \to 0$. Therefore, the KKT conditions hold at the limit point. ∎

4. Numerical experiments

In this section, we compare the performance of ADMM-Factorization algorithm with LMaFit (see Wen et al. (2012)), FR (see Huang and Wolkowicz (2018)), OptSpace (see Keshavan et al. (2010)), ScGrassMC (see Ngo and Saad (2012)), ScaledASD and ASD (see Tanner and Wei (2016)) on several classes of test problems. The implementation is done in MATLAB R2018a on a Laptop with 2.5GHz CPU and 8GB of memory. The stopping criteria is considered as following:

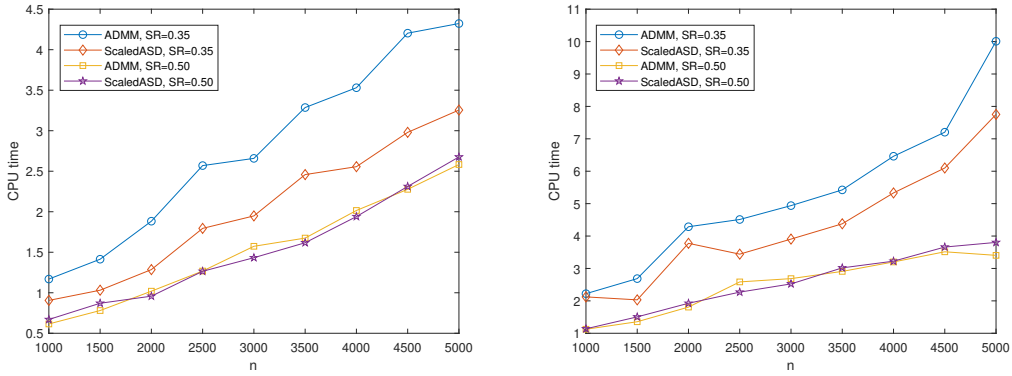
$$Relative\ error = \frac{||M - XY||_F}{max(1, ||M||_F)} \leq tol.$$

In all experiments, ρ and Λ_0 are set to 10^8 and one, respectively. Moreover, the initial points Y_0 is identity matrix and Z_0 is considered $P_{\Omega}(M)$.

Table 1. Comparison of the relative errors when SR=0.56

[m, n, r]	[300, 2000, 3]		[500, 2000,3]		[700,2000,3]	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**0.64**	9.2385e-11	**0.88**	9.6116e-11	1.10	6.6715e-11
FR	14.76	1.6201e-11	15.20	2.8972e-11	17.03	1.2119e-11
ScaledASD	0.65	8.7190e-11	0.94	3.8023e-11	**1.05**	2.1632e-11
ASD	1.00	6.5016e-11	1.26	5.3880e-11	2.47	7.5217e-11
LMaFit	1.50	9.2385e-11	1.91	6.7629e-11	2.72	6.6515e-11
ScGrassMC	1.01	3.5698e-11	1.43	2.5705e-11	1.79	9.3699e-11
OptSpace	27.53	1.1826e-11	30.75	7.0879e-11	29.38	6.8844e-11
[m, n, r]	[300, 4000, 3]		[500, 4000,3]		[700,4000,3]	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**1.17**	6.8615e-11	**1.64**	7.0324e-11	2.04	6.3010e-11
FR	115.69	2.5772e-11	123.74	3.9322e-11	120.01	4.8375e-11
ScaledASD	1.19	8.7081e-11	1.76	5.8733e-11	**1.99**	3.2104e-11
ASD	1.82	8.0497e-11	2.74	7.1827e-11	4.49	8.2264e-11
LMaFit	2.83	6.8615e-11	3.60	8.7583e-11	5.16	6.3015e-11
ScGrassMC	1.88	7.0200e-11	2.82	2.9860e-11	3.62	1.6557e-11
OptSpace	107.91	9.0650e-11	96.99	8.8429e-11	113.57	8.3175e-11
[m, n, r]	[300, 2000, 4]		[500, 2000,4]		[700,2000,4]	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**0.63**	9.3778e-11	**0.87**	9.2609e-11	1.08	6.3832e-11
FR	15.62	2.0426e-11	17.97	4.4225e-11	19.04	6.6462e-11
ScaledASD	0.68	5.4367e-11	0.92	9.8879e-11	**1.05**	5.4243e-11
ASD	0.76	4.6217e-11	1.35	6.2202e-11	1.55	9.2135e-11
LMaFit	1.57	9.3778e-11	1.98	5.9898e-11	2.73	6.3931e-11
ScGrassMC	1.07	4.3719e-11	1.61	4.1699e-11	2.01	3.0729e-11
OptSpace	33.21	1.0913e-11	41.90	1.0379e-11	35.26	1.0207e-11
[m, n, r]	[300, 4000, 4]		[500, 4000, 4]		[700, 4000, 4]	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**1.17**	8.5583e-11	**1.62**	7.1459e-11	**2.04**	8.1371e-11
FR	118.37	7.5755e-11	120.69	3.7979e-11	122.87	3.3297e-11
ScaledASD	1.37	4.5705e-11	1.73	9.8508e-11	2.12	5.0948e-11
ASD	1.47	5.3090e-11	3.75	7.3760e-11	3.06	5.5103e-11
LMaFit	2.98	8.5484e-11	3.61	8.3521e-11	5.21	8.1371e-11
ScGrassMC	2.08	9.7090e-11	3.61	8.3521e-11	3.99	3.1474e-11
OptSpace	125.21	7.4970e-11	113.91	8.5583e-11	127.87	6.7809e-11

- **First class of test problems:** This class of test problems is generated based on Lemma from Huang and Wolkowicz (2018). First we generate two random matrices $M_L \in \mathbb{R}^{m\times r}$ and $M_R \in$



(a) $m = 500$ and rank $= 5$ (b) $m = 500$ and rank $= 15$

Figure 1. Comparison of the CPU times of ADMM-Factorization algorithm and ScaledASD for different SR when rank is fixed and $tol = 10^{-10}$ (first class of test problems)

$\mathbb{R}^{r\times n}$ with normally distributed random entries and then set $M = M_L M_R$. Finally, a subset Ω (observed entries) of M indices are selected randomly (for example p entries of M) and the observed matrix is $M_\Omega := P_\Omega(M)$. The ratio $\frac{p}{mn}$ is the sampling ratio and is denoted by SR.
In Table 1, we compare the relative errors of ADMM-Factorization algorithm and the other algorithms when $SR = 0.56$. The results show that ADMM-Factorization algorithm and ScaledASD outperform the other algorithms and in most cases ADMM-Factorization algorithm has better CPU times than the ScaledASD. In Figure 1, we compare the two top algorithms for different dimensions when m and rank are fixed and SR is 0.35 and 0.50. The results show that ADMM-Factorization algorithm has comparable CPU times with ScaledASD when we increase SR.

- **Second class of test problems:** In this class of test problems, we use noisy matrices (see Vandereycken (2013)). In fact a random perturbations matrix is added to the data matrix M, i.e.,

$$M_\epsilon := M + \epsilon \frac{||M||_F}{||P_\Omega(N)||_F} P_\Omega(N),$$

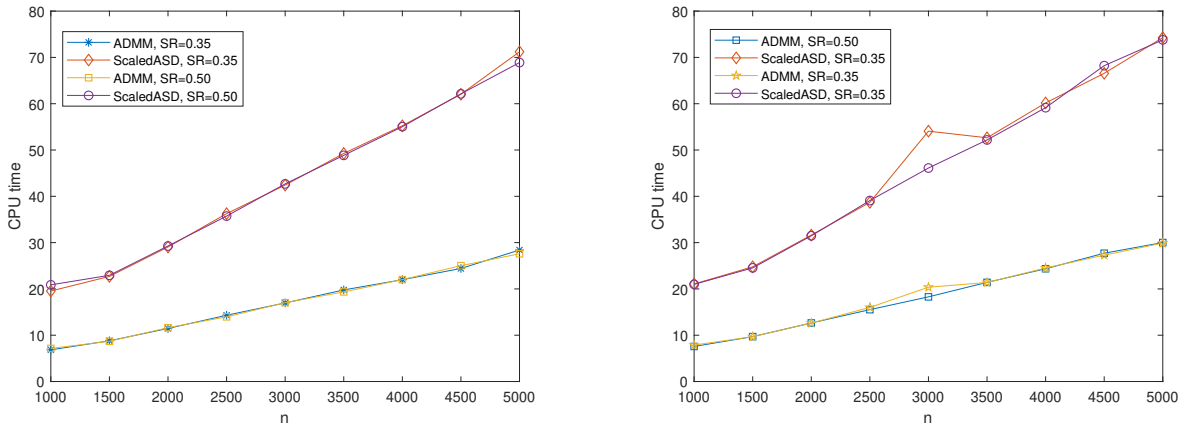
where ϵ is noise level and $N \in (0, 0.001)$ is a random matrix. So the relative errors are of order of ϵ, i.e.,

$$\frac{||M_\epsilon - M||_F}{||M||_F} \simeq \epsilon.$$

Here, we set SR equal to 0.25 and rank equal to 2 and 3. The results are reported in Table 2. As we see, the CPU times of ADMM-Factorization algorithm are significantly better than the other algorithms, specially in high dimensions. Also '-' in this table means that the algorithm needs more than 500 seconds, so it quits. In Figure 2, the performance of ADMM-Factorization and ScaledASD is compared when $\epsilon = 10^{-7}$, maxiter=500 and rank is 5 and 15. The results show that ADMM-Factorization algorithm has significantly better CPU times than ScaledASD.

Table 2. Comparison of the relative errors when both tolerance and noise level are 10^{-7} and maxiter is 200

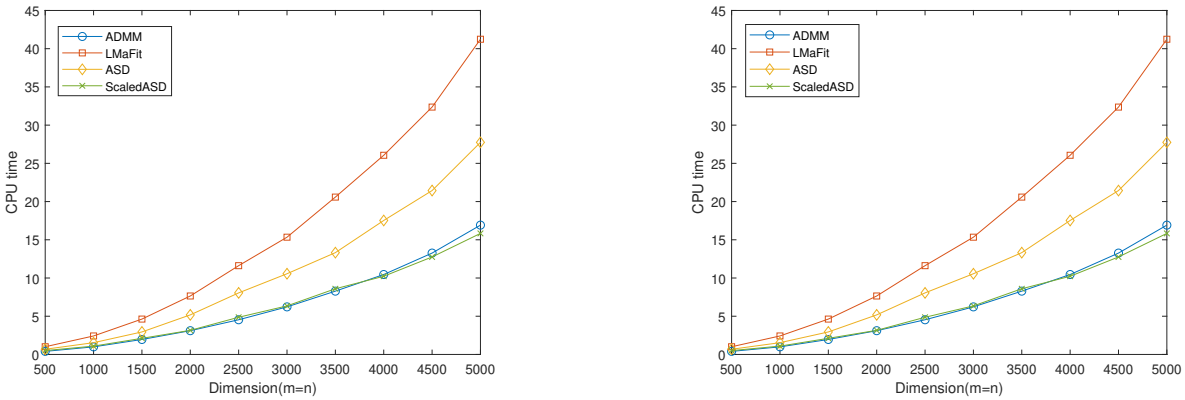
[m, n, r]	[1000, 1000, 2]		[2000, 2000, 2]		[3000,3000, 3]	
SR	0.25		0.25		0.25	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**4.18**	1.5684e-7	**16.13**	1.5425e-7	**34.09**	1.5664e-7
FR	6.18	4.5214e-7	34.71	9.7983e-7	106.62	5.5655e-7
ScaledASD	11.40	1.5654e-7	43.22	1.5654e-7	95.93	7.3021e-7
ASD	10.18	1.5654e-7	39.96	1.5657e-7	86.64	6.7799e-7
LMaFit	10.60	1.5684e-7	39.15	2.5254e-7	86.37	1.5664e-7
ScGrassMC	19.79	1.5684e-7	72.77	1.4551e-7	163.09	1.5664e-7
OptSpace	67.91	9.9551e-6	291.26	1.8208e-6	–	–
[m, n, r]	[1000, 1000, 3]		[2000, 2000, 3]		[3000, 3000, 3]	
SR	0.25		0.25		0.25	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**4.20**	1.6014e-7	**16.07**	1.5713e-7	**29.37**	1.5777e-7
FR	6.33	1.1540e-1	37.18	1.3555e-7	108.63	2.9171e-7
ScaledASD	11.44	1.6014e-7	45.13	4.3258e-7	97.94	1.5777e-7
ASD	10.30	1.6268e-7	40.37	3.7946e-6	89.45	2.3324e-7
LMaFit	10.59	1.6014e-7	40.64	9.1259e-8	88.36	2.5628e-7
ScGrassMC	19.42	1.6014e-7	76.05	5.7531e-7	172.35	5.2563e-7
OptSpace	6.33	1.3370e-1	359.96	1.0269e-3	–	–
[m, n, r]	[4000, 4000, 2]		[4000, 4000, 3]		[5000, 5000, 3]	
SR	0.25		0.25		0.25	
	Time	Rel.err	Time	Rel.err	Time	Rel.err
ADMM	**60.72**	1.5654e-7	**60.52**	1.5714e-7	**94.53**	1.5707e-7
FR	286.87	1.5239e-7	295.22	9.2478e-7	–	–
ScaledASD	168.29	2.3548e-7	169.57	5.6548e-7	256.25	8.3215e-7
ASD	152.82	8.3658e-7	152.63	2.0839e-7	234.03	1.0444e-3
LMaFit	150.84	9.3548e-7	152.39	9.4587e-7	234.44	2.3547e-7
ScGrassMC	303.8	7.3475e-7	307.58	5.4875e-7	466.82	2.5708e-7
OptSpace	–	–	–	–	–	–



(a) $m = 500$ and rank=5 (b) $m = 500$ and rank=15

Figure 2. Comparison of the CPU times of ADMM-Factorization algorithm and ScaledASD for different SR when rank is fixed and tol=10^{-7} (second class of test problems)

- **Third class of test problems:** In this class of test problems, we generate M, an $n \times n$ positive semidefinite matrix of rank r, by sampling an $n \times r$ factor M_F with normally distributed random entries and setting $M = M_F M_F^T$ (see Candès and Recht (2009)).



(a) rank=15 (b) rank=20

Figure 3. Comparison of the CPU times of algorithms for different dimensions when SR is 0.60 and tol=10^{-10} (third class of test problems)

In Figure 3, we compare the CPU times of top four algorithms. As we see, in all cases ADMM-Factorization algorithm has comparable CPU times with ScaledASD.

- **Fourth class of test problems:** In this class of test problems, we use two types of test images: goldhill, man (http://www.utdallas.edu/ cxc123730/mh_ bcs_ spl.html) and the faces (https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html), to compare the top four algorithms. The size of goldhill and man images are 512×512 and the size of faces are 112×92. The missing pixels positions are randomly distributed and more than 50 percent of each images are used as the observations. We used estimated rank 40, 30 and 10 for goldhill, man and faces, respectively. Also, in all experiments we stopped the algorithms after 500 iterations. The output images and the corresponding time and relative errors are also given in Figure 4 and Table 3, respectively. Here also ADMM-Factorization has better CPU times than the other three algorithms while having comparable relative errors.

Table 3. Comparison of different algorithms for image reconstruction

Figures	ADMM		LMaFit		ASD		ScaledASD	
	Time	Rel.err	Time	Rel.err	Time	Rel.err	Time	Rel.err
goldhill	**4.85**	8.4373e-2	12.37	8.4373e-2	12.08	1.2595e-1	14.14	8.4367e-2
man	**4.51**	1.1835e-1	11.20	1.1835e-1	10.91	1.1852e-1	12.75	1.1835e-1
face 1	**0.15**	9.9306e-2	0.25	9.9306e-2	0.19	9.9155e-2	0.29	9.9981e-2
face 2	**0.15**	7.4322e-2	0.25	7.4322e-2	0.19	7.5560e-2	0.29	7.4349e-2

- **Fifth class of test problems:** In this class of test problems, we use MovieLens latest dataset (https://grouplens.org/datasets/movielens) that contains 943 users,1682 movies and 100000 ratings (ml100k-u1.test) (see Harper and Konstan (2016)). Sampling ratio in this experiment is 0.06 and the stopping criteria is

$$Residual\ error = \frac{||P_\Omega(M) - P_\Omega(XY)||_F}{||P_\Omega(M)||_F} \leq tol.$$

Here, also we compare the top four algorithms (ADMM-Factorization, LMaFit, ASD and ScaledASD). In Figure 5, their CPU times are reported when the tolerance and maxiter are 10^{-3}

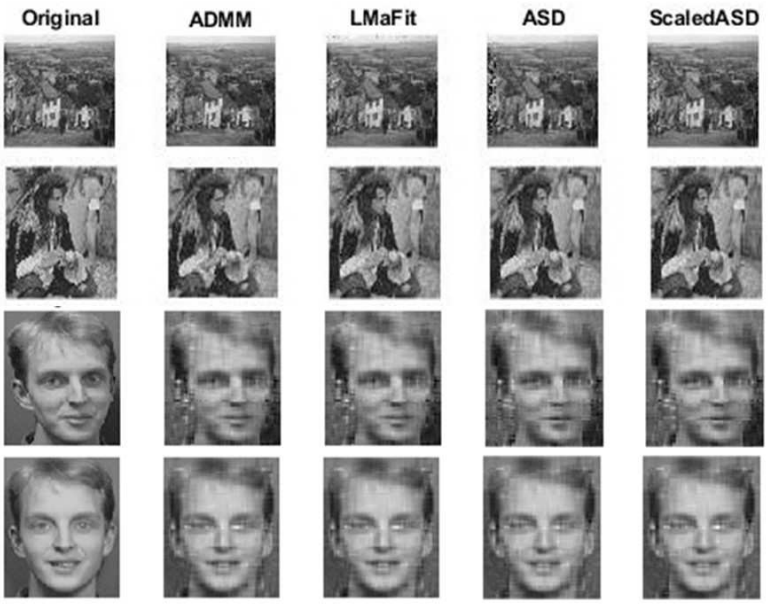


Figure 4. Original and recovered goldhill, man, face 1 and face 2 images using ADMM-Factorization algorithm, LMaFit, ASD and ScaledASD

and 500, respectively. We should note that LMaFit's relative errors for different rank are greater than 10^{-1} and thus we did not include it in the figure.

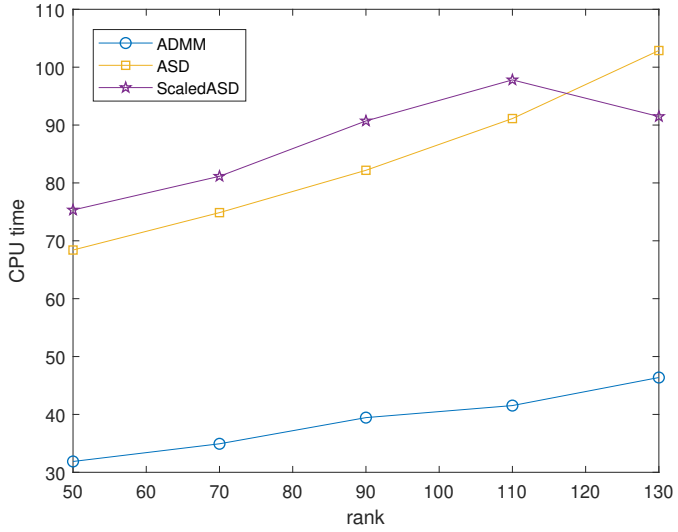


Figure 5. Comparison of the CPU times of algorithms for real data set

5. Conclusions

In this paper, we proposed an ADMM-Factorization algorithm for matrix completion problem. Despite its simplicity, numerical results show that its CPU times performance in five classes of test

problems in most cases is better than the other state-of-the-art matrix completion algorithms in the literature while having the same relative errors.

Acknowledgments:

The authors would like to thank reviewers for their comments and suggestions and University of Guilan for partially supporting this research.

REFERENCES

Argyriou, A., Evgeniou, T. and Pontil, M. (2008). Convex multi-task feature learning, Machine Learning, Vol. 73, No. 3, pp. 243–272.

Cai, J., Candes, E. J. and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion, SIAM Journal on Optimization, Vol. 20, No. 4, pp. 1956–1982.

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization, Foundations of Computational Mathematics, Vol. 9, No. 6, pp. 717–772.

Candès, E. J. and Tao, T. (2010). Exact matrix completion via convex optimization, IEEE Transactions on Information Theory, Vol. 56, No. 5, pp. 2053–2080.

Eldèn, L. (2007). *Matrix methods in data mining and pattern recognition*, SIAM.

Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context, ACM Transactions on Interactive Intelligent Systems, Vol. 5, No. 4, pp. 19.

Huang, S. and Wolkowicz, H. (2018). Low-rank matrix completion using nuclear norm minimization and facial reduction, Journal of Global Optimization, Vol. 72, No. 1, pp. 5–26.

Keshavan, R. H., Montanari, A. and Oh, S. (2010). Matrix completion from a few entries, IEEE Transactions on Information Theory, Vol. 56, No. 6, pp. 2980–2998.

Liu, Z. and Vandenberghe, L. (2009). Interior-point method for nuclear norm approximation with application to system identification, SIAM Journal on Matrix Analysis and Applications, Vol. 31, No. 3, pp. 1235–1256.

Ma, S., Goldfarb, D. and Chen, L. (2011). Fixed point and Bregman iterative methods for matrix rank minimization, Mathematical Programming, Vol. 128, No. 1-2, pp. 321–353.

Ng, M. K., Weiss, P. and Yuan, X. (2010). Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods, SIAM Journal on Scientific Computing, Vol. 32, No. 5, pp. 2710–2736.

Ngo, T. and Saad, Y. (2012). Scaled gradients on Grassmann manifolds for matrix completion, Advances in Neural Information Processing Systems, pp. 1412–1420.

Obozinski, G., Taskar, B. and Jordan, M. I. (2010). Joint covariate selection and joint subspace selection for multiple classification problems, Statistics and Computing, Vol. 20, No. 2, pp. 231–252.

Rennie, J. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative

prediction, Proceedings of the 22nd International Conference on Machine Learning, pp. 713–719.

Ruchansky, N. (2016). *Matrix Completion with Structure*, Ph.D. thesis, Boston University.

Salahi, M. and Taati, A. (2017). Alternating direction method of multipliers for the extended trust region subproblem, Iranian Journal of Numerical Analysis and Optimization, Vol. 7, No. 1, pp. 107–117.

Steidl, G. and Teuber, T. (2010). Removing multiplicative noise by Douglas-Rachford splitting methods, Journal of Mathematical Imaging and Vision, Vol. 36, No. 2, pp. 168–184.

Tanner, J. and Wei, K. (2016). Low rank matrix completion by alternating steepest descent methods, Applied and Computational Harmonic Analysis, Vol. 40, No. 2, pp. 417–429.

Taleghani, R. and Salahi, M. (2018). Low rank matrix completion problem by alternating direction method of multipliers, Advanced Modeling and Optimization, Vol. 20, No. 2, pp. 419–428.

Toh, K. and Yun, S. (2010). An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, Pacific Journal of Optimization, Vol. 6, No. 15, pp. 615-640.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming, SIAM Review, Vol. 38, No. 1, pp. 49–95.

Vandereycken, B. (2013). Low-rank matrix completion by Riemannian optimization, SIAM Journal on Optimization, Vol. 23, No. 2, pp. 1214–1236.

Wang, C. and Li, C. (2015). A mean value algorithm for Toeplitz matrix completion, Applied Mathematics Letters, Vol. 41, pp. 35–40.

Wang, Q., Cao, W. and Jin, Z. (2015). Two-Step Proximal Gradient Algorithm for Low-Rank Matrix Completion, Statistics, Optimization & Information Computing, Vol. 4, No. 2, pp. 174–182.

Wen, Z., Yin, W. and Zhang, Y. (2012). Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, Mathematical Programming Computation, Vol. 4, No. 4, pp. 333–361.

Xiao, Y., Wu, S. and Li, D. (2013). Splitting and linearizing augmented Lagrangian algorithm for subspace recovery from corrupted observations, Advances in Computational Mathematics, Vol. 38, No. 4, pp. 837–858.

Xu, Y., Yin, W., Wen, Z. and Zhang, Y. (2012). An alternating direction algorithm for matrix completion with nonnegative factors, Frontiers of Mathematics in China, Vol. 7, No. 2, pp. 365–384.

Zhang, H. and Cheng, L. Z. (2010). Projected Landweber iteration for matrix completion, Journal of Computational and Applied Mathematics, Vol. 235, No. 3, pp. 593–601.