AMO - Advanced Modeling and Optimization, Volume 20, Number 2, 2018

Low rank matrix completion by alternating direction method of multipliers

Rahman Taleghani, Maziar Salahi

Department of Applied Mathematics, Faculty of Mathematical Sciences

University of Guilan, Rasht, Iran

salahim@guilan.ac.ir

Abstract: In matrix completion problem a subset of a matrix entries is known and the goal is to find the full matrix. Several efficient algorithms are developed among them are nuclear norm minimization or low-rank factorization. In this paper, we introduce an Alternating Direction Method of Multipliers (ADMM) based algorithm for the fixed rank matrix completion problem. On several test problems, the efficiency of ADMM algorithm is compared with a known algorithm from the literature.

***Key words*:** Matrix completion, Alternating direction method of multipliers, Non-convex optimization.

AMS Subject Classification: 15A83, 90C26.

1. Introduction

The problem of recovering a low rank matrix from a sampling of its entries arises in various applications such as model reduction [18], data mining and pattern recognitions [9], machine learning [1,2], collaborative filtering [22] and multi-class learning [3, 20]. The mathematical formulation of the matrix completion (MC) problem, that aims to find the lowest rank matrix from its known entries, is as follows:

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

where $rank\,(Z)$ denote the rank of Z and Ω is the set of observed entries of the unknown matrix. Problem (1) is generally NP-hard [8, 25]. It is well known that, if a matrix has rank r, then it has exactly r nonzero singular values. In [10, 6], the authors show that, the solution of problem (1) under certain conditions can be found by solving the following convex optimization problem:

$$\begin{aligned} &\min \| Z \|_* \\ &s.t \ \ Z_{ij} = M_{ij}, \ \forall (i,j) \in \Omega, \end{aligned} \tag{2}$$

where $\|.\|_*$ is the sum of the singular values and called the nuclear norm or trace norm. Candes and Tao proved that under certain incoherence assumptions on the singular vectors of the matrix, recovery is possible by solving a

convenient convex program as soon as the number of entries is on the order of the information theoretic limit (see also [7]). Different algorithms are proposed in the literature to find the solution of (1), for example see [15, 17, 5]. All these algorithms use singular value decomposition (SVD) that is expensive when the size of the matrix increase. Thus, other algorithms based on simple factorization are widely used for model (1) instead of SVD, for example in [26, 14, 24] the authors proposed to decrease the residual between data matrix and recovered matrix instead of model (2). The authors in [26] proposed to solve

$$\begin{aligned} &\min \frac{1}{2}\| XY - Z \|_F^2 \\ &s.t \quad Z_{ij} = M_{ij}, \quad (i,j) \in \Omega. \end{aligned}$$

They assume a low rank factorization model and non-SVD approach based on non-linear successive over-relaxation (SOR) and dynamically estimate and adjust the value of rank during iterations. In this paper, instead of model (1), we propose to solve the following problem:

$$\begin{aligned} &\min \frac{1}{2}\| XUY - Z \|_F^2 \\ &s.t \quad Z_{ij} = M_{ij}, \quad (i,j) \in \Omega, \end{aligned} \tag{3}$$

where M is a matrix with known entries with indices in set Ω, $X \in R^{m\times s}$, $U \in R^{s\times s}$, $Y \in R^{s\times n}$ and $Z \in R^{m\times n}$ that s is integer and our estimate rank. Let the rank of data matrix M be r and

$$P_\Omega(M) = \begin{cases} M_{ij} & (i,j) \in \Omega, \\ 0 & otherwise. \end{cases}$$

Model (3) is a non-convex quadratic optimization problem because of XUY term in the objective function. The three matrix XUY instead of XY in objective function allows us to control the rank of XUY with dimension of U, because U is a full rank square matrix and $s \le min\{m, n\}$. The rank estimate s should ideally be equal to the rank of the data matrix M because the correct rank is generally unknown.

To solve model (3), we present an Alternating Direction Method of Multipliers (ADMM) algorithm that has been widely used for solving several convex and non-convex optimization problems by breaking them into smaller sub-problems. It has also been applied successfully to a number of statistical problems [4], image restoration and denoising [11, 23, 19] and support vector machines [12]. The rest of this paper is organized as follows. In Section 2 we present the ADMM algorithm. Comparison of ADMM algorithm with Opts [16] is given in Section 3 on several test problems.

2. ADMM algorithm

The KKT conditions for (3) are as follow:

$$\frac{\partial L}{\partial X} = (XUY - Z)(UY)^T = 0, \tag{4.a}$$

$$\frac{\partial L}{\partial Y} = (XU)^T(XUY - Z) = 0, \tag{4.b}$$

$$\frac{\partial L}{\partial U} = X^T(XUY - Z)Y^T = 0, \tag{4.c}$$

$$\frac{\partial L}{\partial Z} = (Z - XUY) + \Lambda = 0, \tag{4.d}$$

$$\frac{\partial L}{\partial \Lambda} = P_\Omega(Z - M) = 0. \tag{4.e}$$

Now consider the augmented Lagrangian associated to (3) as follows:

$$L_\rho(X, U, Y, Z, \Lambda) = \frac{1}{2}\| XUY - Z \|_F^2 + \Lambda \bullet P_\Omega(Z - M) + \frac{\rho}{2}\| P_\Omega(Z - M) \|_F^2, \tag{5}$$

where $\Lambda \in R^{m\times n}$ is the Lagrange multiplier and $\rho > 0$ is the penalty parameter. The inner product of two matrices $A^{m\times n}$ and $B^{m\times n}$ is defined as

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

The steps of ADMM can be outlined as follow:

$$X_{k+1} \leftarrow \arg\min_X L_\rho(X, U_k, Y_k, Z_k, \Lambda_k), \tag{6.a}$$

$$Y_{k+1} \leftarrow \arg\min_Y L_\rho(X_{k+1}, U_k, Y, Z_k, \Lambda_k), \tag{6.b}$$

$$U_{k+1} \leftarrow \arg\min_U L_\rho(X_{k+1}, U, Y_{k+1}, Z_k, \Lambda_k), \tag{6.c}$$

$$Z_{k+1} \leftarrow \arg\min_Z L_\rho(X_{k+1}, U_{k+1}, Y_{k+1}, Z, \Lambda_k), \tag{6.d}$$

$$\Lambda_{k+1} = \Lambda_k + \gamma\rho P_\Omega(Z_{k+1} - M). \tag{6.e}$$

As we see, in (6.a) since Y, U, Z and Λ are fixed matrices, it is a convex quadratic optimization problem. The same hold for (6.b), (6.c) and (6.d). Specifically, the above steps are the following due to the convexity:

$$X_{k+1} = (Z_k(U_kY_k)^T)((U_kY_k)(U_kY_k)^T)^{-1}, \tag{7.a}$$

$$Y_{k+1} = ((X_{k+1}U_k)^T(X_{k+1}U_k))^{-1}((X_{k+1}U_k)^T Z_k), \tag{7.b}$$

$$U_{k+1} = (X_{k+1}^T X_{k+1})^\dagger X_{k+1}^T Z_k Y_{k+1}^T (Y_{k+1}Y_{k+1}^T)^\dagger, \tag{7.c}$$

$$Z_{k+1} = X_{k+1}U_{k+1}Y_{k+1} + P_\Omega(M - X_{k+1}U_{k+1}Y_{k+1}) - \frac{1}{\rho}\Lambda_k, \tag{7.d}$$

$$\Lambda_{k+1} = \Lambda_k + \gamma\rho P_\Omega(Z_{k+1} - M), \tag{7.e}$$

where in (7.c) $A^{\dagger}$ is the Moore-Penrose pseudo inverse of A and in (7.e), γ is the step size. Now, the ADMM algorithm for solving (3) can be outlined as follows.

ADMM algorithm for solving (3)

Input matrix U_0, Y_0, Z_0 and Λ. Choose appropriate penalty parameter

$\rho > 0$ and step size $\gamma > 0$. Set $tol > 0$, $\max iter > 0$ and $k = 0$.

For $k = 1, ..., \max iter$ **do**

Perform (7.a) to (7.e) .

If a stopping criterion is satisfied then exit with $(X_{k+1}, U_{k+1}, Y_{k+1}, Z_{k+1}, \Lambda_{k+1})$ as an output.

end if

end for

3 Numerical results

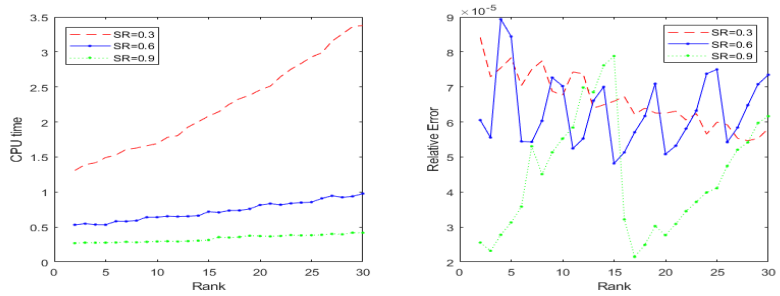
In this section, we compare the ADMM algorithm with OptSpace [16] on several test problems. The implementation is done in MATLAB 2018a on a 2.5GHz CPU using a laptop with 8GB RAM. We set Y_0, U_0 equal to the identity matrix and Z_0 to $P_{\Omega}(M)$. Penalty parameter ρ and Lagrange multipliers Λ are set to 10^8 and zero, respectively. We also set $tol = 10^{-4}$ and $max\ iter = 100$. The stopping criteria in our numerical results are considered as following:

$$rel.err = \frac{\| M - Z \|_F}{\max(1, \| M \|_F)} \leq tol,$$

$$res.err = \frac{\| P_{\Omega}(M - XUY) \|_F}{\max(1, \| P_{\Omega}(M) \|_F)} \leq tol.$$

We report the computational results for three types of data matrix M with rank r.

- **First class of examples**: We create two random matrices $M_L \in R^{m \times r}$ and $M_R \in R^{r \times n}$ with "$randn$" command of MATLAB and then set $M = M_L M_R$. After that, a subset Ω (observed entries) of M indices are selected randomly (for example p entries of M). The ratio $p/(mn)$ between the numbers of observed entries and the numbers of M entries is denoted by "SR" which is the sampling ratio [26].



(a) (b)

Fig. 1 (a) and (b) shows the sensitivity of ADMM for $SR = \{0.3, 0.6, 0.9\}$ when $m = n = 1000$ and $r = \{2, 3, \ldots, 30\}$.

First we study the sensitivity of ADMM algorithm to different SR values when $m = n = 1000$ and $r = \{10, 11, ..., 30\}$. The average CPU time and rank for each SR values are demonstrated in Figure 1. As we see, when we fix $SR = 0.3, 0.6, 0.9$, in different rank values the CPU time (Figure 1.a) and the relatives errors (Figure 1.b) are decreasing. Now, we compare ADMM algorithm and OptSpace for $m = n = 1000$ and different values of ranks and SR. Results are reported in Table 1.

Table 1: Numerical results of ADMM algorithm and OptSpace with different ranks and SR for $m = n = 1000$.

Problems		**ADMM**		**OptSpace**	
SR	**rank**	Time	rel.err	Time	rel.err
0.30	**10**	1.86	0.0001	22.49	0.0001
0.45	**10**	0.74	0.0001	18.94	0.0001
0.75	**10**	0.31	0.0001	21.02	0.0001
0.30	**20**	2.84	0.0001	122.93	0.0001
0.45	**20**	0.98	0.0001	95.03	0.0001
0.75	**20**	0.38	0.0001	97.46	0.0001
0.30	**30**	4.09	0.0001	396.65	0.0001
0.45	**30**	1.28	0.0001	338.40	0.0001
0.75	**30**	0.47	0.0001	297.93	0.0001

As we see, ADMM is much faster while having the same relative error. Next, we compare ADMM algorithm and OptSpace when $SR = 0.25$, for different ranks and dimensions as reported in Table 2.

Table 2: Numerical results of ADMM algorithm, OptSpace with fixed SR and different ranks and dimensions.

Problems		**ADMM**		**OptSpace**	
r	**n**	Time	rel.err	Time	rel.err
3	**100**	0.09	0.0035	0.21	0.0001
5	**100**	0.15	0.0091	0.55	0.4784
10	**100**	0.18	0.0199	0.39	0.9727
20	**100**	0.25	0.0156	0.47	1.0000
30	**100**	0.39	0.0823	0.56	1.0000
3	**500**	0.47	0.0001	1.17	0.0000
5	**500**	0.59	0.0001	2.66	0.0001
10	**500**	0.77	0.0001	7.35	0.0001
20	**500**	1.63	0.0001	53.91	0.0001
30	**500**	1.95	0.0018	16.67	0.9346
50	**500**	1.49	0.3150	35.65	0.9247

From Table 2, we see that the ADMM algorithm has comparable accuracy with OptSpace while faster and for some instances the time difference become significant. In Table 3 we compare the relative residuals of ADMM algorithm and OptSpace. As we see, the relative residuals of ADMM algorithm and OptSpace are approximately the same but ADMM's CPU times are significantly smaller.

Table 3: Comparison the relative residuals between ADMM algorithm, OptSpace when $m = n$.

Problems			**ADMM**		**OptSpace**	
n	**r**	**SR**	res.err	Time	res.err	Time
1000	**10**	**0.10**	0.0003	8.88	0.0002	30.00
1000	**10**	**0.15**	0.0001	5.27	0.0001	19.72
1000	**10**	**0.30**	0.0001	1.85	0.0001	19.65
1000	**20**	**0.09**	0.0043	4.83	0.0003	235.34
2000	**10**	**0.10**	0.0007	16.63	0.0002	51.28
2000	**10**	**0.20**	0.0001	10.06	0.0001	53.46
2000	**10**	**0.35**	0.0001	4.68	0.0001	65.05
2000	**10**	**0.55**	0.0001	2.56	0.0001	63.19
3000	**10**	**0.15**	0.0001	28.40	0.0001	113.64
3000	**10**	**0.20**	0.0001	19.94	0.0001	99.37
3000	**20**	**0.15**	0.0001	37.29	0.0001	399.40

- **Second class of examples**: First, we create two random matrices X and Y with "*rand* " command of MATLAB. Then, the noise matrix N is randomly generated using $N(0, 0.01^2)$ and finally we set the data matrix M equal to $XY + N$ [13].

Table 4: The relative errors comparison between ADMM algorithm and OptSpace for the second experience of test problem when $m = n$.

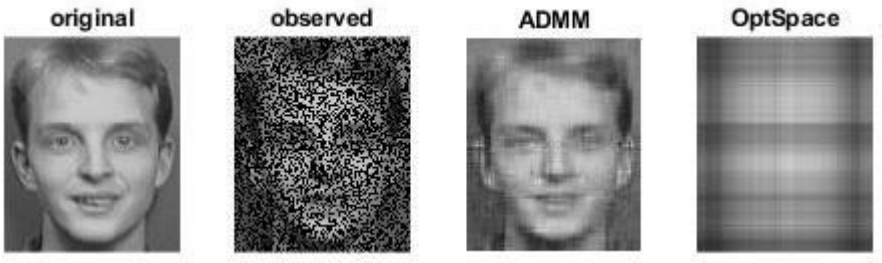
Problems			**ADMM**		**OptSpace**	
n	**r**	**SR**	rel.err	Time	rel.err	Time
500	**10**	**0.15**	0.0001	3.02	0.0942	17.29
500	**10**	**0.25**	0.0001	1.98	0.0979	19.10
500	**10**	**0.45**	0.0001	0.35	0.0963	24.32
1000	**10**	**0.10**	0.0071	10.45	0.0962	50.60
1000	**10**	**0.20**	0.0001	3.10	0.0953	66.85
1000	**10**	**0.30**	0.0001	1.81	0.0947	86.48
1500	**10**	**0.15**	0.0001	9.40	0.0954	138.70
1500	**12**	**0.25**	0.0001	3.82	0.0905	181.18
1500	**12**	**0.35**	0.0001	3.33	0.0824	210.21
1500	**15**	**0.25**	0.0001	8.27	0.0703	176.38
1500	**15**	**0.45**	0.0001	2.63	0.0743	239.75
1500	**20**	**0.45**	0.0001	3.57	0.0716	235.41
2000	**10**	**0.25**	0.0001	5.39	0.0955	321.51
2000	**10**	**0.45**	0.0001	2.21	0.0692	444.08

In Table 4, we increase the dimensions of test problem when $m = n$. As we see, the relative errors and CPU times of ADMM algorithm are substantially smaller than OptSpace. Additionally, in Table 5, we compare the relative residuals of ADMM algorithm and OptSpace. The results show that ADMM algorithm performance for large instances is much better than OptSpace.

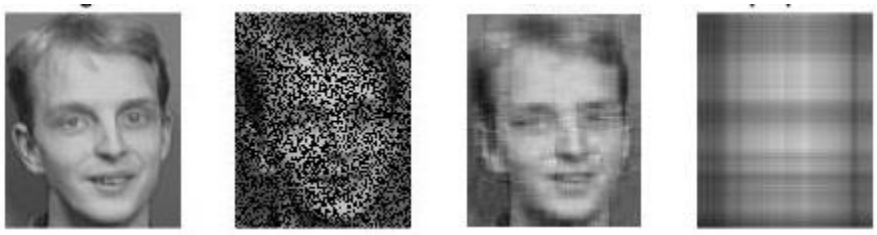
Table 5: The relative residuals comparison between ADMM algorithm and OptSpace for the second class of test problems when $m = n$.

Problems			ADMM		OptSpace	
n	**r**	**SR**	res.err	Time	res.err	Time
700	**10**	**0.25**	0.0011	1.87	0.0945	34.33
700	**10**	**0.45**	0.0011	1.97	0.0963	51.36
700	**10**	**0.75**	0.0012	1.99	0.0981	56.65
1000	**10**	**0.25**	0.0011	3.45	0.0962	79.16
1000	**10**	**0.40**	0.0011	3.48	0.0949	97.76
1000	**10**	**0.70**	0.0011	3.50	0.0964	144.98
1500	**10**	**0.35**	0.0011	7.36	0.0952	211.91
1500	**10**	**0.50**	0.0011	7.28	0.0962	260.41
1500	**10**	**0.75**	0.0011	7.43	0.0957	264.58
2000	**10**	**0.50**	0.0011	12.72	0.0974	469.64
2000	**10**	**0.75**	0.0011	13.44	0.0010	880.07

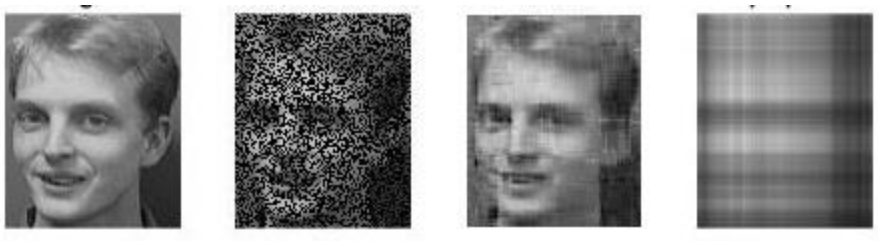
- **Third class of examples**: Here we use ORL face data set [21]. The task here is to fill in missing pixel values of an image. The missing pixel position are randomly distributed. In each figure, the ADMM algorithm results, OptSpace results, observed pixel of photo and the original picture are demonstrated. In this experiment, we pick the sampling ratio (SR) to be approximately %50. The 112×92 original image is shown in Figure 2 and its recovering time and relative errors are reported in Table 6 where we set rank r equal to 10.



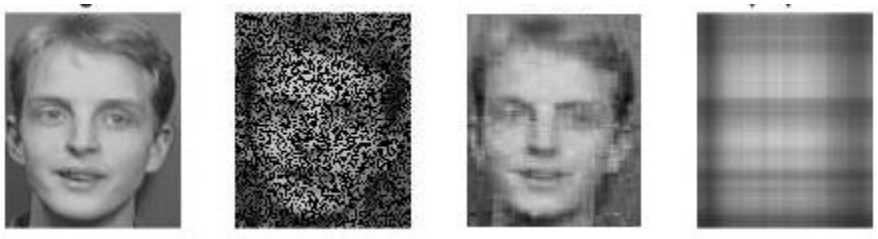
(a)



(b)



(c)



(d)

Fig. 2 Original and recovered images using ADMM and OptSpace.

Table. 6 The relative errors comparison between ADMM algorithm and OptSpace for recovered images.

Problems	**ADMM**		**OptSpace**	
Fig. 2	rel.err	time	rel.err	time
(a)	0.0728	0.36	0.2082	0.95
(b)	0.0705	0.33	0.2138	0.99
(c)	0.0854	0.34	0.2181	1.08
(d)	0.0764	0.39	0.1995	0.99

4. Conclusions

In this paper, we proposed an Alternating Direction Method of Multipliers algorithm for matrix completion problem using three term factorization. Despite its simplicity, numerical results show that it performs significantly better than OptSpace from time point of view while having the same level of accuracy.

References:

1. Y. Amit, M. Fink, N. Srebro, and S. Ullman, Uncovering shared structures in multiclass classification, in Proceedings of the 24th international conference on machine learning, ACM, p. 17–24, 2007.

2. A. Argyriou, T. Evgeniou, and M. Pontil, Multi-task feature learning, Advances in Neural Information Processing Systems, 19, p. 41–48, 2007.

3. A. Argyriou, T. Evgeniou, and M. Pontil, Convex multi-task feature learning, Journal of Machine Learning, 73, p. 243–272, 2008.

4. J. M. Bioucas-Dias and M. A. T. Figueiredo, Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing, Foundations and Trends in Machine Learning, 14, p. 1–4, 2010.

5. J.F. Cai, E.J. Candes, and Z.W. Shen, A singular value thresholding algorithm for matrix completion, SIAM Journal on Control and Optimization, 20, p. 1956–1982, 2010.

6. E. J. Candes and B. Recht, Exact matrix completion via convex optimization, Foundations of Computational Mathematics, 9, p. 717–772, 2009.

7. E. J. Candes and T. Tao, The power of convex relaxation: near-optimal matrix completion, IEEE Transactions on Information Theory, 56, p. 2053–2080, 2010.

8. J. David, Algorithms for analysis and design of robust controllers, PhD thesis, University Leuven, Belgium, 1994.

9. L Eldén, Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms), Society for Industrial and Applied Mathematics, Philadelphia, 2007.

10. M. Fazel, Matrix rank minimization with applications, PhD thesis, Stanford University, 2002.

11. M. A. T. Figueiredo, and J. M. Bioucas-Dias, Restoration of poissonian images using alternating direction optimization, IEEE Transactions on Image Processing, 19, p. 3133–3145, 2010.

12. P. A. Forero, A. Cano, and G. B. Giannakis, Consensus-based distributed support vector machines, Journal of Machine Learning Research, 11, p. 1663–1707, 2010.

13. D. Hajinezhad, and Q. Shi, Alternating direction method of multipliers for a class of nonconvex bilinear optimization: convergence analysis and applications, Journal of Global Optimization, 70, p. 261–288, 2018.

14. P. Haldar and D. Hernando, Rank-constrained solutions to linear matrix equations using power factorization, IEEE Signal Processing Letters, 16, p. 584–587, 2009.

15. P. Jain, R. Meka, and I. Dhillon, Guaranteed rank minimization via singular value projection, part of the Neural Information Processing Systems Conference , NIPS, p. 937–945, 2010.

16. R. H. Keshavan, A. Montanari, and S. Oh, Matrix completion from a few entries, IEEE Transactions on Information Theory, 56 , p. 2980–2998, 2010.

17. R.H. Keshavan, A. Montanari, and S. Oh, Matrix completion from noisy entries, Journal of Machine Learning, 99, p. 2057–2078, 2010.

18. Z. Liu and L. Vandenberghe, Interior-point method for nuclear norm approximation with application to system identification, SIAM Journal on Matrix Analysis and Applications, 31, p. 1235–1256, 2009.

19. M. Ng, P. Weiss, and X. Yuang, Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods, ICM Research Report, 32, p. 2710-2736, 2009.

20. G. Obozinski, B. Taskar, and M. I. Jordan, Joint covariate selection and joint subspace selection for multiple classification problems, Statistics and Computing, 20, p. 231–252, 2010.

21. ORL Database of Faces, Cambridge University Computer Laboratory. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

22. J. D. M. Rennie and N. Srebro, Fast maximum margin matrix factorization for collaborative prediction, In Proceedings of the International Conference of Machine Learning, p. 713–719, 2005.

23. G. Steidl and T. Teuber, Removing multiplicative noise by Douglas-Rachford splitting methods, Journal of Mathematical Imaging and Vision, 36, p. 168–184, 2010.

24. J. Tanner and K. Wei., Low rank matrix completion by alternating steepest descent methods, Applied and Computational Harmonic Analysis, 40, p. 417–429, 2016.

25. L. Vandenberghe and S. Boyd, Semidefinite programming, SIAM Review, 38, p. 49-95, 1996.

26. Z. Wen, W. Yin, and Y. Zhang, Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm, Mathematical Programming Computation, 4, p. 333–361, 2012.