Improved Bounded Matrix Completion for Large-Scale Recommender Systems

Huang Fang[1], **Zhen Zhang**[1], **Yiqun Shao**[2], **Cho-Jui Hsieh**[1]
[1]Departments of Statistics and Computer Science
[2]Department of Mathematics
University of California, Davis
{hgfang, ezzhang, yqshao, chohsieh}@ucdavis.edu

Abstract

Matrix completion is a widely used technique for personalized recommender systems. In this paper, we focus on the idea of Bounded Matrix Completion (BMC) which imposes bounded constraints into the standard matrix completion problem. It has been shown that BMC works well for several real world datasets, and an efficient coordinate descent solver called BMA has been proposed in [R. Kannan and Park., 2012]. However, we observe that BMA can sometimes converge to non-stationary points, resulting in a relatively poor accuracy in those cases. To overcome this issue, we propose our new approach for solving BMC under the ADMM framework. The proposed algorithm is guaranteed to converge to stationary points. Experimental results on real world datasets show that our algorithm can reach a lower objective function value, obtain a higher prediction accuracy and have better scalability compared with existing bounded matrix completion approaches. Moreover, our method outperforms the state-of-art standard matrix factorization in terms of prediction error in many real datasets.

1 Introduction

Matrix factorization and matrix completion [Koren *et al.*, 2009a] are widely used for personalized recommender systems. Given partially observed ratings in the user-item matrix, the goal is to predict missing ratings by solving one of the following optimization problems:

$$\min_{W\in\mathbb{R}^{m\times k} H\in\mathbb{R}^{k\times n}} \frac{1}{2}\|\mathcal{P}_\Omega(Y-WH)\|_F^2+\frac{\lambda}{2}(\|W\|_F^2+\|H\|_F^2) \quad (1.1)$$

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\|\mathcal{P}_\Omega(Y-X)\|_F^2+\lambda\|X\|_*. \quad (1.2)$$

where (1.2) is the convex form, and the nonconvex form (1.1) is equivalent to the convex form if k is chosen large enough. In the above problems, $Y \in \mathbb{R}^{m\times n}$ is our observed rating matrix, Ω is the observed set, usually $|\Omega| \ll mn$. $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ are low rank matrices with rank $k \ll \min(m,n)$ and $\mathcal{P}_\Omega(\cdot)$ is the indicator function with the indicate set Ω. $\|\cdot\|_*$ denote the nuclear norm which is commonly used as a convex relaxation technique for the non-convex rank penalty.

Despite the excellent performance on many real world problems, traditional matrix factorization formulations do not enforce any constraint to the prediction values. However, in many real world problems, the ratings are bounded within a certain region. For example, the ratings of MovieLens and Netflix datasets are bounded in the range of $[0.5, 5]$, but the prediction of Eq (1.1) and (1.2) often output values out of this range. In those cases, it is reasonable to impose bounded constraints to the original optimization problems, leading to the following Bounded Matrix Completion (BMC) problem:

$$\min_{W\in\mathbb{R}^{m\times k} H\in\mathbb{R}^{k\times n}} \frac{1}{2}\|\mathcal{P}_\Omega(Y-WH)\|_F^2+\frac{\lambda}{2}(\|W\|_F^2+\|H\|_F^2)$$
$$\textbf{s.t.} \quad \mathrm{r}_{min} \le WH \le r_{max} \quad (1.3)$$

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\|\mathcal{P}_\Omega(Y-X)\|_F^2+\lambda\|X\|_* \;\textbf{s.t.}\; r_{min} \le X \le r_{max} \quad (1.4)$$

Note that we use $r_{min} \le X \le r_{max}$ to denote the element-wise constraints $r_{min} \le X_{ij} \le r_{max}$ for all i, j throughout the paper. This is a hard optimization problem because there are totally mn constraints, so a naive implementation will easily require $O(mn)$ memory, which is prohibitive for problems with more than 10,000 users/movies.

Recently, [R. Kannan and Park., 2012] proposed an efficient block coordinate descent algorithm for solving the non-convex form (1.3) of the BMC problem. The resulting algorithm, named BMA, often outperforms traditional matrix completion algorithms on real datasets with bounded ratings. However, the BMA algorithm does not always converge to stationary points. Instead, it can easily stuck in non-stationary points, as we will discuss in Section 2. This unstable convergence behavior often leads to a performance drop in practice.

In this paper, we propose a new algorithm for minimizing the BMC objective function. Our algorithm is based on the ADMM framework [Boyd *et al.*, 2011] for solving the convex form (1.4), so it is guaranteed to converge to the optimal solution. Moreover, with carefully-designed update rules, our algorithm does not encounter $O(mn)$ space complexity, and can scale to large datasets with 10 million ratings. Experimental results on real world datasets show that our algorithm can reach a better solution and is also much faster than existing bounded matrix completion approaches. More interestingly, our algorithm often outperforms state-of-the-art matrix factorization in terms of prediction accuracy.

The rest of the paper is outlined as follows. We present related work and give an example to show that coordinate descent does not work for bounded matrix completion in Section 2. Our main algorithm is proposed and analyzed in Section 3, and the experimental results on real datasets are presented in Section 4.

2 Related Work

There are many existing methods for solving (1.1) and (1.2), [Funk, 2006] first proposed matrix factorization for recommender systems and used SGD to solve the optimization problem. [Koren, 2008] proposed a baseline estimate and incorporate it with matrix factorization to promote predict accuracy. See also a useful survey in [Koren *et al.*, 2009b]. Later, several coordinate descent and SGD algorithms have been proposed to scale matrix factorization to very large datasets [Yu *et al.*, 2012; Yun *et al.*, 2014; Gemulla *et al.*, 2011]; unfortunately, none of the above algorithms can handle bounded constraints. There are several other recent trend of matrix completion focusing on handling side information [Rao *et al.*, 2015; Chiang *et al.*, 2015] and noisy observations [Hsieh *et al.*, 2015], and our algorithm can also be potentially used in those cases.

Recently, [R. Kannan and Park., 2012] proposed an algorithm called BMA to solve the bounded constraint matrix factorization problem (1.3), and they demonstrated that BMA can achieve lower Test RMSE compared with SGD (without bounded constraint). BMA is built on the idea of block coordinate descent. However, block coordinate descent assumes that the feasible area of W,H can be decomposed as Cartesian product, and the bounded constraints in (1.3) violates this assumption, so BMA lacks theoretical support and may not converge to stationary points.

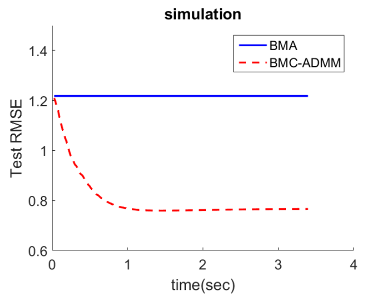


Figure 1: Simulation results show that BMA cannot converge to stationary points, while our algorithm, BMC-ADMM, converges to a stationary point with much lower test RMSE.

Here we give an example that BMA cannot converge to stationary points. Consider the problem with $Y = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $r_{min} = 0$ and $r_{max} = 1$, and the initial solution is $W = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, $H = \begin{pmatrix} 0 & 0 \end{pmatrix}$. When we fix W, H cannot change, and vice versa. But at this time W,H are not optimal and it is also not a stationary point. If we don't impose bounded constraints, when we fix W, H will change to $(0.5, 0.5)$, which eventually converges to optimal solution. So we can see that the bounded constraint makes it difficult to use block coordinate descent.

Here we use a small synthetic example to illustrate the potential convergence problem of BMA. We assume there are 1,000 rows and 1,000 columns with 49,153 ratings bounded between 0 and 5. We set the low rank $k = 1$ and initialize $W \in \mathbb{R}^{m\times 1}$ with standard Gaussian distribution variables and $H \in \mathbb{R}^{1\times n}$ with all zeros.

From the simulation result, BMA stucks at the initial solution and is not able to escape. However, our proposed method BMC-ADMM could converge to global optimal and get competing predict accuracy under the same initialization. Furthermore, BMA is much slower than standard matrix factorization, whose time complexity is $O(k^2mn)$ in each iteration and the space needed is $O(mn/\text{nblk} + |\Omega|)$, where nblk is the number of "blocks" used in BMA. Numerical experiments also show that the run time will increase significantly if the number of blocks is too large. The large time and space complexity imply that BMA cannot scale to large datasets very well.

3 Proposed Method

In this section, we describe our algorithm for the bounded matrix completion problem. We begin with the mathematical formulation under the ADMM framework [Boyd *et al.*, 2011], then describe our algorithm in detail. We also discuss the convergence property of our method and analyze the time and space complexity of our proposed algorithm.

3.1 Mathematical Formulation and Convergence Properties

To apply alternating direction method of multipliers (ADMM) framework, we first split the loss, regularization, and constraints in the original problem (1.4), leading to the following equivalent form:

$$\min_{X,Z,W} \frac{1}{2}\|\mathcal{P}_\Omega(Y - X)\|_F^2 + \lambda\|Z\|_* \tag{3.1}$$
$$\textbf{s.t. } X = Z,\ \ Z = W,\ \ r_{min} \le W \le r_{max}.$$

This looks like a standard ADMM problem and can be solved by alternately updating X, Z, W and the Lagrange multipliers corresponding to the constraints. Unfortunately, solving (3.1) by ADMM requires $O(mn)$ space complexity since X is a dense and full rank m by n matrix during the whole optimization procedure until converging to the optimal solution.

To resolve this issue, we propose to further decompose X into $\mathcal{P}_\Omega(X) + \mathcal{P}_{\hat{\Omega}}(X)$, where $\mathcal{P}_\Omega(X)$ is a sparse matrix and $\mathcal{P}_{\hat{\Omega}}(X)$ is stored as the product of two low rank matrices. Here we make a transformation and write the original X as $X_{new} + E$, where $X_{new} := \mathcal{P}_\Omega(X)$ is sparse and $E := \mathcal{P}_{\hat{\Omega}}(X)$. This variable transformation leads to the following equivalent problem:

$$\begin{aligned} \min_{X,Z,W}\ & \frac{1}{2}\|\mathcal{P}_\Omega(Y-X)\|_F^2 + \lambda\|Z\|_* \qquad (3.2)\\ \text{s.t. } & X+E=Z,\ \mathcal{P}_{\hat\Omega}(X)=0,\ \mathcal{P}_\Omega(E)=0\\ & Z=W,\ r_{min}\le W\le r_{max}. \end{aligned}$$

For simplicity, in (3.2) we use X to denote X_{new} described above, and $Z, E, W \in \mathbb{R}^{m\times n}$ are auxiliary matrices. The corresponding augmented Lagrangian is:

$$\begin{aligned} \underset{\mathcal{P}_{\hat\Omega}(X)=0,\mathcal{P}_\Omega(E)=0,r_{min}\le W\le r_{max}}{L(X,Z,W,U_1,U_2)} &= \frac{1}{2}\|\mathcal{P}_\Omega(Y-X)\|_F^2+\lambda\|Z\|_*\\ &+\langle U_1, X-Z+E\rangle+\langle U_2, Z-W\rangle\\ &+\frac{\rho_1}{2}\|X-Z+E\|_F^2+\frac{\rho_2}{2}\|Z-W\|_F^2. \end{aligned}$$

This is equivalent to the following scaled form:

$$\begin{aligned} \underset{\mathcal{P}_{\hat\Omega}(X)=0,\mathcal{P}_\Omega(E)=0,r_{min}\le W\le r_{max}}{L(X,Z,W,U_1',U_2')} &= \frac{1}{2}\|\mathcal{P}_\Omega(Y-X)\|_F^2+\lambda\|Z\|_*\\ +\frac{\rho_1}{2}\|X-Z+E+U_1'\|_F^2 &+ \frac{\rho_2}{2}\|Z-W+U_2'\|_F^2+\text{const}, \end{aligned}$$

where U_1, U_2 are Lagrangian multipliers, $U_1, U_2 \in \mathbb{R}^{m\times n}$, ρ_1, ρ_2 are penalty parameters, $U_1' = U_1/\rho_1, U_2' = U_2/\rho_2$. Note that the inequality constraint is not included in our augmented Lagrangian equation. The motivation of this is to facilitate the calculation of auxiliary matrix W, which is discussed and named as *augmented partial Lagrangian* in [Du *et al.*, 2014].

Since the objective function (3.2) is closed, proper and convex, it can be solved by finding a saddle point of the augmented Lagrangian $L(X,Z,W,U_1,U_2)$. Under the *Gauss-Seidel* framework, ADMM finds the saddle point using the following iterative procedure:

$$X^{k+1} = \underset{\mathcal{P}_{\hat\Omega}(X)=0}{\arg\min}\ \frac{1}{2}\|\mathcal{P}_\Omega(Y-X)\|_F^2+\frac{\rho_1}{2}\|X-Z^k+E^k+U_1'^k\|_F^2 \qquad (3.3)$$

$$\begin{aligned} Z^{k+1} = \underset{Z}{\arg\min}\ & \lambda\|Z\|_*+\frac{\rho_1}{2}\|X^{k+1}-Z+E+U_1'^k\|_F^2\\ &+\frac{\rho_2}{2}\|Z-W^k+U_2'^k\|_F^2 \qquad (3.4) \end{aligned}$$

$$E^{k+1} = \underset{\mathcal{P}_\Omega(E)=0}{\arg\min}\ \frac{\rho_1}{2}\|X^{k+1}-Z^{k+1}+E+U_1'^k\|_F^2 \qquad (3.5)$$

$$W^{k+1} = \underset{r_{min}\le W\le r_{max}}{\arg\min}\ \|Z^{k+1}-W+U_2'^k\|_F^2 \qquad (3.6)$$

$$U_1'^{k+1} = U_1'^k + X^{k+1} - Z^{k+1} + E^{k+1}$$

$$U_2'^{k+1} = U_2'^k + Z^{k+1} - W^{k+1}$$

Next we discuss how to solve each subproblem in detail.

- Eq (3.3) can be solved in closed form by the element-wise soft-thresholding operator: $X_{i,j}^{k+1} = \frac{1}{1+\rho_1}(Y_{i,j}+\rho_1 Z_{i,j}^k-\rho_1 E_{i,j}^k-\rho_1 U_{1i,j}'^k$ when $(i,j)\in\Omega$ and $X_{i,j}^{k+1}=0$ when $(i,j)\notin\Omega$.
- To solve (3.4), we first notice that this subproblem is equivalent to

$$\underset{X}{\arg\min}\ \lambda\|Z\|_* + \frac{\rho}{2}\|Z-A\|_F^2,$$

  where $A = \frac{\rho_1}{\rho_1+\rho_2}(E^k+X^{k+1}+U_1'^k)+\frac{\rho_2}{\rho_1+\rho_2}(W^k-U_2'^k)$ and $\rho=\rho_1+\rho_2$. Therefore, the original problem has a closed form solution by soft-thresholding singular values: $Z^{k+1} = U\text{soft}_{\lambda/\rho}(\Lambda)V^T$, where U,Λ,V is the SVD decomposition of A, $\text{soft}_{\lambda/\rho}(\Lambda) = \textbf{diag}[(\lambda_1-\lambda/\rho)_+,(\lambda_2-\lambda/\rho)_+,...,(\lambda_n-\lambda/\rho)_+]$. Computing SVD is costly, but here we only need to compute singular values that are larger than the threshold λ/ρ, which can be solved iteratively by PROPACK [Larsen, 1998] or power iterations [Halko *et al.*, 2011]. In this paper, we apply PROPACK to compute partial SVD decomposition efficiently. To use PROPACK, we have to specify the number of singular values greater than the threshold. We can either give a fixed number or dynamically predict this number in each iteration, which is presented in [Mazumder *et al.*, 2010; Hsieh and Olsen, 2014]. In this paper, we use fixed rank k to compute partial SVD decomposition for simplicity.
- For eq (3.5), the subproblem has a simple closed form solution by setting $E_{i,j}^{k+1}=0$ when $(i,j)\in\Omega$, and $E_{i,j}^{k+1} = Z_{i,j}^{k+1}-X_{i,j}^{k+1}-U_{1i,j}'^k$ for $(i,j)\notin\Omega$. Since $\mathcal{P}_{\hat\Omega}(X^{k+1})=0$ and $\mathcal{P}_{\hat\Omega}(U_1'^k)=0$, so $E=\mathcal{P}_{\hat\Omega}(Z^{k+1})$
- For eq (3.6), $W^{k+1}=\Pi_{\mathcal{C}}(Z^{k+1}+U_2'^k)$, where $\Pi_{\mathcal{C}}$ is the projection into $[r_{min}, r_{max}]$.

The resulting algorithm is summarized in Algorithm 1.

3.2 Complexity and Implementation Details

Time complexity

The most time consuming steps are (3.4) and (3.6). For (3.4), PROPACK uses iterative method to get truncated k SVD decomposition. For (3.6), the complexity is $O(kmn)$ for checking all the elements in W^{k+1} and project back to the bounded constraints. Numerical experiments (refer to section 4.4) show that (3.6) is the slowest step and dominates the run time.

Space complexity

Here we use the same trick as [Lin *et al.*, 2011] used to save memory usage, where Y, X, U_1 are sparse matrices($O(|\Omega|)$ complexity), Z is stored as the product of two low rank matrices($O(k(m+n))$ complexity), $Z = A_1A_2^T$, $A_1\in\mathbb{R}^{m\times k}$, $A_2\in\mathbb{R}^{n\times k}$. Since $E=\mathcal{P}_{\hat\Omega}(Z)$ and $\mathcal{P}_\Omega(E)=0$, we can express E by A_1 and A_2. U_2 is also represented as a sparse matrix where its number of non-zero elements equals to the number of elements of W that are on the boundary. For W, $W=\Pi_{\mathcal{C}}(Z+U_2')$, we need to store both A_1, A_2 and record the elements that are out of bound in order to represent W. Numerical experiments show that the number of elements out of bound is far less than $|\Omega|$, for example, on `movielens10m` training set, $|\Omega| = 8$ million, but mostly only $96,073$ elements are out of bound($nnz(U_2')$) during the computation.

Algorithm 1 BMC-ADMM for Bounded Matrix Completion

```
Input $Y, \lambda, k, r_{min}, r_{max} \rho_1 > 0, \rho_2 > 0, maxIter$
Initialize $X, W, Z$ with baseline initialization
Initialize $U_1, U_2$ with zeros matrices $\in \mathbb{R}^{m \times n}$
for i = 1,2,...,maxIter do
    # Solve Eq (3.3)
    $\mathcal{P}_\Omega(X^{i+1}) = \frac{1}{1+\rho_1}\mathcal{P}_\Omega(Y + \rho_1(Z^i - E^i - U_1^i))$
    $\mathcal{P}_{\hat{\Omega}}(X^{i+1}) = 0$
    # Solve Eq (3.4)
    $\rho = \rho_1 + \rho_2$
    $A = \frac{\rho_1}{\rho_1+\rho_2}(E^i + X^{i+1} + U_1^i) + \frac{\rho_2}{\rho_1+\rho_2}(W^i - U_2^i)$
    $(U, S, V) = Partial\_svd(A, k)$
    $D = diag(diag(S(1:k, 1:k) - \lambda/\rho)_+)$
    $Z^{i+1} = UDV^T$
    # Solve Eq (3.5)
    $\mathcal{P}_{\hat{\Omega}}(E^{i+1}) = \mathcal{P}_{\hat{\Omega}}(X^{i+1} - Z^{i+1} + U_1^i)$
    $\mathcal{P}_\Omega(E^{i+1}) = 0$
    # Solve Eq (3.6)
    $W^{i+1} = Z^{i+1} + U_2^i$
    $W^{i+1}(W^{i+1} > r_{max}) = r_{max}$
    $W^{i+1}(W^{i+1} < r_{min}) = r_{min}$
    # Update Scaled Lagrangian multipliers
    $U_1^{i+1} = U_1^i + X^{i+1} - Z^{i+1} + E^{i+1}$
    $U_2^{i+1} = U_2^i + Z^{i+1} - W^{i+1}$
    if stopping criterion is met then
        break
    end
end
Ouput rating matrix $W$
```

In summary, the total space complexity is $O((m+n)k + |\Omega|)$ which is the same as standard matrix factorization methods.

Initialization

Initialization for BMC is non-trivial since $W_0 H_0$ need to satisfy the bounded constraints. Here we discuss two initialization techniques.

The first one is random initialization: we generate two low rank matrices with random Gaussian variables W_0 and H_0, and then multiply them with a scalar α and adjust their first column and first row such that $\min(W_0 H_0) = r_{min}$ and $\max(W_0 H_0) = r_{max}$. The detailed procedure is in Algorithm 2.

In Algorithm 2, the construction of AdjustTerm and α is based on the following equations:

$$\begin{cases} \text{AdjustTerm} + \alpha v_{min} = r_{min} \\ \text{AdjustTerm} + \alpha v_{max} = r_{max}. \end{cases}$$

In this way, our initialized $W_0 H_0$ is bounded in $[r_{min}, r_{max}]$. This is an improved version of the random initialization proposed in [R. Kannan and Park., 2012] since the random initialization implemented in [R. Kannan and Park., 2012] can only generate non-negative W_0 and H_0 and guarantee that the initialized $W_0 H_0 \in [0, r_{max}]$.

Algorithm 2 Random initialization

```
Input $Y, k, r_{min}, r_{max}$
$m, n = size(Y)$
Initialized 2 random matrices $W_0 \in \mathbb{R}^{m \times k}$ and $H_0 \in \mathbb{R}^{k \times n}$
$v_{min}$ = min($W_0(:, 2:end) * H_0(2:end, :)$)
$v_{max}$ = max($W_0(:, 2:end) * H_0(2:end, :)$)
$\alpha = \frac{r_{max} - r_{min}}{v_{max} - v_{min}}$
AdjustTerm $= \frac{r_{min} v_{max} - r_{max} v_{min}}{v_{max} - v_{min}}$
$W_0 = \sqrt{\alpha} W_0, H_0 = \sqrt{\alpha} H_0$
$W_0(:, 1) =$ AdjustTerm, $H_0(1, :) = 1$
Ouput initialized matrices $W_0$ and $H_0$
```

Another initialization approach is called "baseline estimate". [Koren, 2008] presents that baseline estimate is a reasonable initialization for recommender systems, where the missing rating $\mu_{i,j} = \mu + g_i + h_j$, where μ is the global mean of ratings and g_i and h_j are corresponding bias terms for user i and movie j.

Note that these two initializations are also used by BMA, and we will use random initialization to compare the performance of BMA, CCD++ and our method in the following sections.

4 Experiments

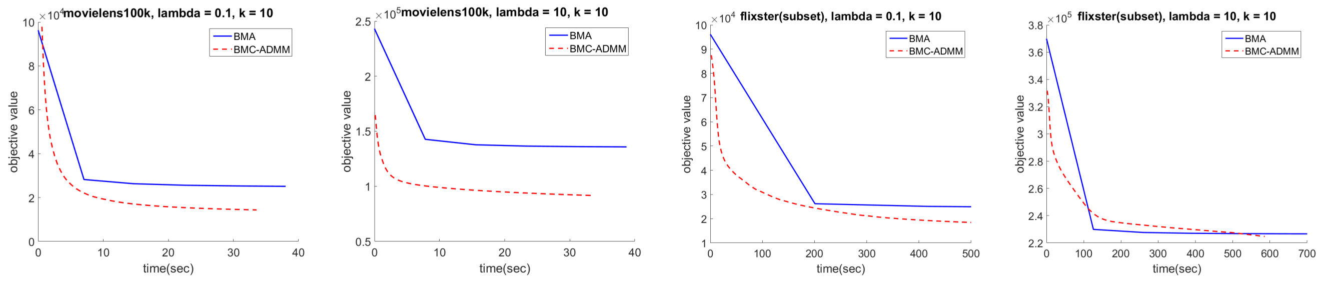
In this section, we compare ADMM with existing matrix factorization methods on real datasets, in terms of efficiency, quality of the solution, and prediction accuracy. We include the following algorithms/implementations in our comparisons:

- BMC-ADMM: Our proposed method for solving the bounded matrix completion problem (Algorithm 1).
- BMA [R. Kannan and Park., 2012]: The coordinate descent algorithm for solving the BMC problem proposed in [R. Kannan and Park., 2012].
- CCD++ [Yu *et al.*, 2012]: The coordinate descent solver for the traditional matrix completion problem (1.1).

We use five real world datasets to test the performance of the above matrix completion algorithms, and the detailed data statistics are listed in Table 1. We randomly split 80% as training data and 20% as testing data. For each algorithm and dataset, we choose the best regularization parameters from $\lambda \in \{0, 0.01, 0.1, 1, 10, 100\}$ based on validation set. All the following experiments are conducted on an Intel i5-4590 3.30 GHz CPU with 16GB RAM.

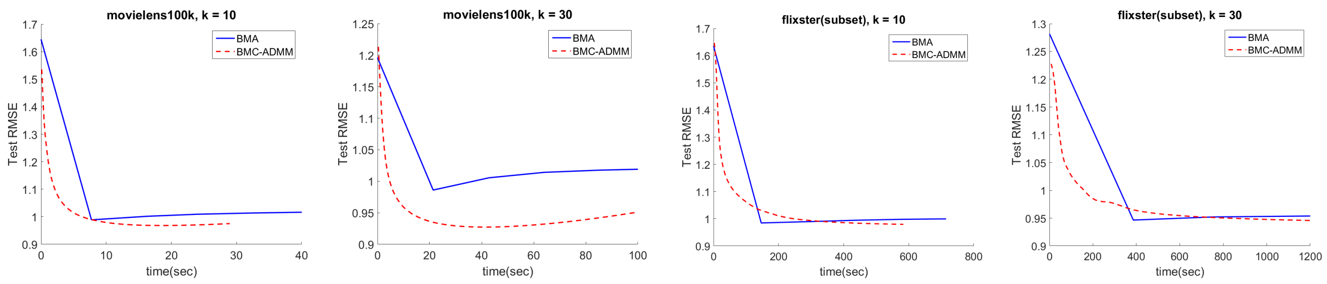
dataset	m	n	$\|\Omega\|$	range
movielens100k	671	9,066	100,004	[0.5, 5]
movielens10m	71,567	10,677	10,000,054	[0.5, 5]
Flixster_subset	14,761	4,903	81,495	[0.5, 5]
Flixster	147,612	48,794	8,196,077	[0.5, 5]
Jester	50,692	150	1,728,847	[-10, 10]

Table 1: Dataset Statistics



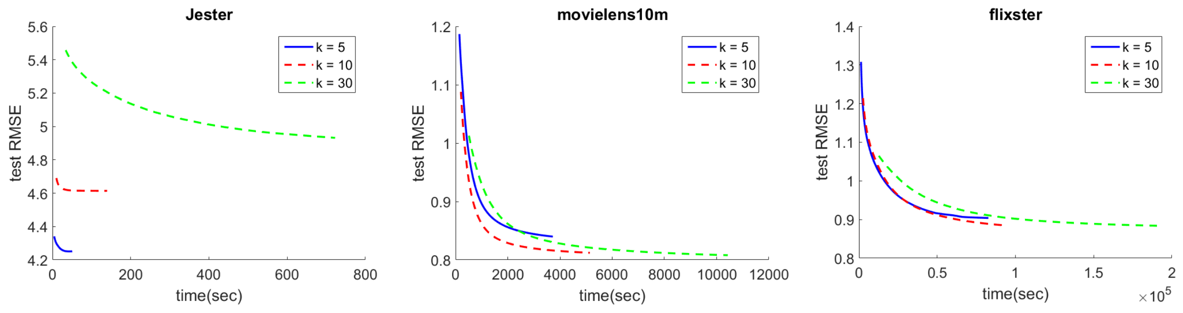
(a) movielens100k $\lambda = 0.1$ (b) movielens100k $\lambda = 10$ (c) flixster(subset) $\lambda = 0.1$ (d) flixster(subset) $\lambda = 10$

Figure 2: Convergence behaviour for different values of λ on movielens100k & Flixster(subset)



(a) movielens100k, $k = 10$ (b) movielens100k, $k = 30$ (c) flixster(subset), $k = 10$ (d) flixster(subset), $k = 30$

Figure 3: Test RMSE for different values of k (rank) on movielens100k & Flixster(subset)



(a) Jester (b) Movielens10m (c) Flixster

Figure 4: Test RMSE of BMC-ADMM for larger datasets

Note that BMA is slow and requires large memory when data is large unless we let nblk to be very large to save memory, but this will make the BMA even slower. Therefore, we use small datasets or a subset of large datasets to compare BMC-ADMM, BMA and CCD++, and use full large dataset to compare BMC-ADMM and CCD++.

4.1 Comparison of Bounded Matrix Completion (BMC) Algorithms

In the first set of experiments, we compare BMC-ADMM with the state-of-the-art bounded matrix completion algorithm, BMA [R. Kannan and Park., 2012]. Although BMA solves the non-convex form (1.3) while we solve the convex form (1.4), it is well known that these two objective functions are equivalent since $\|X\|_* = \min_{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{n\times k}: X=UV^T} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$ for sufficiently large k. Therefore, we first compare the objective function of BMC-ADMM with BMA in Figure 2.

We set $\rho_1 = \rho_2 = 1$ and try $\lambda = 0.1$ and 10, and compare the performance on all the datasets. In Figure 2, we observe that our algorithm BMC-ADMM can achieve lower objective function value than BMA, and the main reason is that BMA can stuck at non-stationary points, while our algorithm always converges to stationary points. In Figure 2, we also compare BMC-ADMM and BMA under different regularization parameter λ. Results show that BMC-ADMM can always find a solution with better objective function value.

dataset	k	Global Mean	BMC-ADMM	BMA	Standard MF (CCD++)
movielens100k	5	1.0617	**1.0073**	1.2545	1.104
movielens100k	10	1.0617	**0.9689**	0.9858	1.096
movielens100k	30	1.0617	**0.9177**	0.9970	1.087
movielens10m	5	1.06	0.8399	0.8887	**0.8205**
movielens10m	10	1.06	0.8122	0.87819	**0.8110**
movielens10m	30	1.06	**0.8080**	0.88488	0.8098
Flixster_subset	5	1.0555	1.0458	**0.9700**	1.2177
Flixster_subset	10	1.0555	1.0014	**0.9664**	1.2205
Flixster_subset	30	1.0555	**0.9287**	0.9592	1.2168
Flixster	5	1.0921	**0.9198**	- -	0.9247
Flixster	10	1.0921	**0.8854**	- -	0.9187
Flixster	30	1.0921	**0.8838**	- -	0.9165
Jester	5	5.2747	**4.2452**	4.4587	4.3268
Jester	10	5.2747	4.6222	4.5772	**4.4069**
Jester	30	5.2747	4.9631	4.501	**4.4262**

Table 2: Test RMSE Comparison

To compare the prediction performance of these two methods, we list the test RMSE for all the 5 datasets in Table 2 using their own best parameter chosen by validation sets. The results indicate that BMC-ADMM can achieve a better test RMSE than BMA in most of the datasets except Flixster subset. Unfortunately, due to the large time complexity and huge memory requirement, BMA is not able to scale to the full Flixster dataset. In conclusion, we observe that BMC-ADMM is faster, more accurate, and more scalable than BMA.

4.2 Comparison with Standard Matrix Completion

Next we compare our algorithm, BMC-ADMM, with matrix completion algorithm without bounded constraints on large datasets. We choose the coordinate descent solver for matrix completion developed in [Yu *et al.*, 2012] to compare with, and list the results on all the datasets in Table 2. To have a fair comparison, we tried different settings of rank $k = 5, 10, 30$ for all the algorithms, and select the best regularization parameter for each of them using the random validation set. The results in Table 2 indicates that BMC-ADMM outperforms other methods in most cases, and this means adding bounded constraints is really useful in practice if we want to achieve better prediction accuracy.

4.3 Scalability

To test the scalability of our BMC-ADMM, we run our algorithm on larger datasets, i.e. `movielens10m`, `flixster` and `Jester`. Result is presented in Figure 4. A larger rank k usually leads to a better test RMSE but slower convergence (see Figure 4 b,c), but sometimes it also leads to over-fitting, especially when the dataset has few users or movies (see Figure 4 a).

Experiments show that our algorithm is slower than standard matrix factorization method (without bounded constraint), but faster than BMA.

4.4 Time Spent on Each Part of BMC-ADMM

Table 3 present the detailed run time information of each step for our implementation. The update of W dominates our run time since its complexity is $O(kmn)$. Note that for Eq. (3.5), $E = \mathcal{P}_{\hat{\Omega}}(Z)$, so we don't need explicit computation for E once we get the update of Z. Therefore, there is no runtime for Eq. (3.5) in Table 3. The result presented here verified our computational complexity analysis, showing that computing the projection in (3.6) is the bottleneck of our algorithm.

dataset	k	Eq. (3.3)	Eq. (3.4)	Eq. (3.6)
movielens10m	5	0.67	1.47	**11.47**
movielens10m	10	1.04	2.01	**14.6**
Flixster	5	0.69	1.52	**109.42**
Flixster	10	1.20	2.18	**151.84**

Table 3: Average runtime(sec) of each step in each iteration

5 Discussions and Conclusions

In this paper, we consider the bounded matrix factorization problem. We point out the convergence problem of the existing algorithm for Bounded Matrix Factorization BMA and propose a novel algorithm based on the ADMM framework. Experimental results showed that our approach has better convergence properties than BMA and can achieve lower Test RMSE values in most cases. In terms of run time and space complexity, our algorithm also outperforms BMA but takes more time than standard matrix factorization (without bounded constraint). To further speedup our algorithm, we can use parallel computing to solve subproblem (3.6) since it is the bottleneck of our algorithm. Since this step is a simple element-wise projection, we expect a linear speedup by parallelizing it.

References

[Boyd *et al.*, 2011] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, pages 1–122, January 2011.

[Cabral *et al.*, 2013] Ricardo Cabral, Fernando De la Torre, Joo P. Costeira, and Alexandre Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, December 2013.

[Candes and Recht., 2009] E. J. Candes and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[Chiang *et al.*, 2015] Kai-Yang Chiang, Cho-Jui Hsieh, and Inderjit S Dhillon. Matrix completion with noisy side information. In *Advances in Neural Information Processing Systems*, pages 3447–3455, 2015.

[Du *et al.*, 2014] Simon S. Du, Y. Liu, B. Chen, and L. Li. Maxios: Large scale nonnegative matrix factorization for collaborative filtering. In *In Conference on Neural Information Processing Systems (NIPS) 2014, workshop on Distributed Machine Learning and Matrix Computations.*, 2014.

[Funk, 2006] Simon Funk. Stochastic gradient descent, http://sifter.org/∼ simon/journal/20061211.html, 2006.

[Gemulla *et al.*, 2011] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[Halko *et al.*, 2011] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[Hsieh and Olsen, 2014] Cho-Jui Hsieh and Peder A Olsen. Nuclear norm minimization via active subspace selection. In *ICML*, 2014.

[Hsieh *et al.*, 2015] Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit Dhillon. Pu learning for matrix completion. In *International Conference on Machine Learning*, pages 2445–2453, 2015.

[Koren *et al.*, 2009a] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, pages 42–49, 2009.

[Koren *et al.*, 2009b] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *In Proc. of the 14th ACM SIGKDD conference*, pages 426–434, 2008.

[Larsen, 1998] Rasmus Munk Larsen. Lanczos bidiagonalization with partial reorthogonalization, 1998.

[Lin *et al.*, 2011] Z. Lin, M. Chen, L. q. Wu, and Y. Ma. Linearized alternating direction method with adaptive penalty for low rank representation. In *Neural Information Processing Systems (NIPS)*, 2011.

[Mazumder *et al.*, 2010] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322, 2010.

[R. Kannan and Park., 2012] M. Ishteva R. Kannan and H. Park. Bounded matrix low rank approximation. In *ICDM 2012 IEEE 12th International Conference on*, pages 10–13, December 2012.

[Rao *et al.*, 2015] Nikhil Rao, Hsiang-Fu Yu, Pradeep Ravikumar, and Inderjit S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Neural Information Processing Systems (NIPS)*, December 2015.

[Yu *et al.*, 2012] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *IEEE International Conference of Data Mining*, 2012.

[Yun *et al.*, 2014] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, SVN Vishwanathan, and Inderjit Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *Proceedings of the VLDB Endowment*, 7(11):975–986, 2014.