Binary Matrix Completion on Graphs: Application to Collaborative Filtering

Divyanshu Talwar, Aanchal Mongia, Emilie Chouzenoux, Angshul Majumdar

Binary Matrix Completion on Graphs: Application to Collaborative Filtering

Divyanshu Talwar[1], Aanchal Mongia[1], Emilie Chouzenoux[2], and Angshul Majumdar[1]

divyanshu15028@iiitd.ac.in, aanchalm@iiitd.ac.in, emilie.chouzenoux@centralesupelec.fr, and angshul@iiitd.ac.in

[1]Indraprastha Institute of Information Technology, Delhi, India

[2]CVN, CentraleSupélec, Inria Saclay, Gif-sur-Yvette, France

Abstract – This work addresses the problem of completing a partially observed matrix where the entries are either ones or zeroes. This is typically called one-bit matrix completion or binary matrix completion. In this problem, the association among the rows and among the columns can be modeled through graph Laplacians. Since the Laplacians cannot be computed from the incomplete matrix, they must be simultaneously estimated while completing the matrix. We model the problem as graph regularized binary matrix completion where the graphs need to be learnt from the data. We proposed an algorithm based on an alternating minimization scheme, taking advantage of an efficient proximity-based inner solver. The algorithm is applied to the problem of collaborative filtering. Experiments on benchmark datasets with state-of-the-art techniques in collaborative filtering show that the proposed method improves over the rest by a considerable margin.

Index Terms — matrix completion, graph signal processing, collaborative filtering, recommender system

1. Introduction

Consider the problem of completing a partially observed matrix where the matrix is known to be of low rank. This is called matrix completion. In the general problem, the entries in the matrix can range from minus infinity to plus infinity. Our interest lies in the specific case where the entries are binary; this has been called one-bit matrix completion [1] or binary matrix completion [2]. One bit matrix completion is an extreme case of quantized matrix completion [3].

The problem of matrix completion arises in many areas of signal processing and machine learning. There are two broad approaches to solve it. Traditionally, the low-rank matrix was factored into a thin and a fat matrix. Those matrices were then recovered by matrix factorization [4, 5] or its deep version [6]. Another approach is via nuclear norm minimization [7, 8] where the matrix is directly recovered by promoting a low-rank solution. Since rank minimization is known to be NP-hard, its convex surrogate (nuclear norm) is minimized instead.

Matrix completion finds applications in several signal processing problems such as seismic data interpolation [9], array signal processing [10, 11], etc. Applied machine learning problems such as collaborative filtering [12], clustering [13], classification [14], etc. have also been modeled as matrix completion. In most applied machine learning problems, the entries are binary. For example, in collaborative filtering, the matrix gathers information about whether a user likes a product or not. In such scenarios, binary matrix completion is a more appropriate choice.

In typical signal processing applications such as [9-11], there is no relationship among the rows or among the columns. In contrast, in machine learning applications like [12-14], the relationship is known. For example, classification, the labels are known [14]. In drug-target interaction, the structures of the drug molecules and tissue structure are known [15, 16]. In collaborative filtering, the similarity between the users and the items can also be pre-computed and represented by a graph [17, 18].

The aforesaid studies [15-18] solve the general matrix completion problem on graphs where the entries in the matrix are unconstrained (i.e. they are real numbers). However, for the said problems, the entries are actually binary, representing interactions between users and items [12, 17, 18], or drugs and targets [15, 16. Thus, ideally one would like to solve them via binary matrix completion on graphs. To the best of our knowledge, it has not been attempted before. In this work, we propose to address the problem of binary matrix completion on graphs. We will make a particular focus on solving the collaborative filtering problem.

Let us point out that a major difference between prior matrix completion techniques on graphs [17, 18] and our proposal. In the aforesaid studies, the graph is assumed to be known; 'by known', we mean that the graph has been computed once from the partially observed data matrix and is fixed input for the said studies. In contrast, in this work, we will learn the graph jointly with performing the binary matrix completion task. Our method can then be viewed as a marriage of graph learning [19] with matrix completion. Note that the work [19] proposes formulations for learning graphs from data; it is nothing to do with matrix completion. The intuitive understanding behind our proposal is given at the beginning of section 3.

2. Background

The problem of collaborative filtering can be expressed in as a matrix completion problem,

$$Y = R(X) + N \tag{1}$$

Hereabove, the matrix X is the rating matrix; we assume that users are along the rows and items along the columns. R

is the so-called restriction operator which passes the value of the available ratings from X to Y. Y is the partially filled matrix of observed ratings. The noise in the system, represented by N, is usually assumed to be i.i.d. normally distributed. The problem is to retrieve X given Y and R. In standard matrix completion, the entries of X are assumed to be real. However, in practical collaborative filtering, X is a binary matrix representing user's choice on items; the user either likes (1) or does not like (0). Therefore, this problem should be treated as a binary matrix completion problem [1].

In the latent factor model [20], it is assumed that the user's choice is determined by certain hidden/latent factors. The items possess these factors to a certain extent. If there is a match between the user's propensity towards the factors and the intensity with which these factors are present in the item, the user 'likes' the item (represented by 1); on the other hand, if there is a mismatch in the expectations of the user and what the item offers, the user 'dislikes' the item (represented by 0). This phenomenon is modeled as an inner product between the user's and item's latent factors. When all the users and items are considered, one can express the rating matrix in the following form:

$$X = UV \tag{2}$$

where U denotes the users' latent factor matrix (with users on its rows) and V denotes the items' latent factor matrix (with items on its columns). Matrix U is tall while V is fat, thus modeling the low-rank nature of X, adjusted by the setting of the number of latent factors, corresponding to the number of columns for U, or rows for V matrices. For collaborative filtering, the matrix factorization formulation is embedded in (1) giving rise to:

$$Y = R(UV) + N \tag{3}$$

The solution to the matrix factorization problem (3) can be formulated [9]:

$$\min_{U,V} \|Y - R(UV)\|_F^2 + \lambda\left(\|U\|_F^2 + \|V\|_F^2\right) \tag{4}$$

The first term is the data fidelity term arising out of the Gaussian nature of noise. The ridge type regularizations, weighted by $\lambda>0$, applied on both latent factor matrices prevent over-fitting.

A more direct approach to solve the original problem (1) is to directly solve for the rating matrix X. This is achieved by minimizing a least-squares term penalized by a nuclear norm [21]:

$$\min_{X} \|Y - R(X)\|_F^2 + \lambda\|X\|_* \tag{5}$$

The nuclear norm, with weight $\lambda>0$ is a convex surrogate of the rank penalty [12]. Formulation (5) is convex, but it presents a limited performance in applications to collaborative filtering [9].

In neighborhood-based models of collaborative filtering, the matrix is completed by linear interpolation. The similarity between the user's [22] or item's [23] is used as interpolation weights. The similarity is computed from the data, i.e. for computing the similarity between the users, the ratings of the users on different items are used. Studies like [18] proposed combining the similarity scores of [22, 23] into the matrix factorization framework (3). They encode the similarity information in graph Laplacians, by including those into penalty terms:

$$\min_{U,V} \|Y - R(UV)\|_F^2 + \lambda \left(Tr\left(U^T L_U U\right) + Tr\left(V L_V V^T\right)\right) \quad (6)$$

Here λ is a positive regularization parameter, L_U and L_V are the graph Laplacians for the user and item latent factor matrices respectively. In [17] a similar idea was proposed for the completion problem formulation (5), leading to the resolution of:

$$\min_{X} \|Y - R(X)\|_F^2 + \mu \|X\|_* + \lambda \left(Tr\left(X L_U X^T\right) + Tr\left(X^T L_V X\right)\right) \quad (7)$$

depending on two positive penalty weights λ and μ. Note that the aforementioned works assumed the knowledge of the involved graph Laplacians matrices.

This section on relevant background only refers to studies that are pertinent for understanding our proposal. For a thorough review of matrix completion techniques in collaborative filtering, one can peruse [24].

3. Proposed Binary Matrix Completion on Graphs

The graph regularization incorporated in both (6) and (7) has been shown to improve the results by a large margin [17,18]. However, there are two shortcomings of these studies. First, they treat the matrix X to be with real-valued entries, whereas those are actually binary in applications such as collaborative filtering [1]. Second, the similarities (and hence the Laplacians) between the users and items are computed once from the incomplete data. Ideally, they should have been estimated iteratively while completing the matrix. This second point solicits further explanation. Consider three users whose actual ratings are as follows:

U1: [1 0 0 1 0 1 1 0]
U2: [1 1 1 1 1 0 0 0]
U3: [0 0 0 1 0 0 1 0]

These ratings are not fully observed, say the partially observed ratings are:

U1: [1 x x 1 x x 1 x]
U2: [1 x x 1 x x x 0]
U3: [0 x x x x 0 x 0]

Going by the previous studies [15, 16] the similarity between U1 and U3 computed from the partially observed ratings will be low (Hamming distance: 3) and the similarity between U1 and U2 will be higher (Hamming distance: 1). However, when the similarities are computed from the filled rating vectors, one can see that the similarity between U1 and U3 is higher (Hamming distance: 2) compared to the similarity between U1 and U2 (Hamming distance: 5). This toy example shows the importance of computing the similarities (thereby the graph Laplacians) from the completed matrix, rather than from the partially observed one. Since the completed matrix is obviously not available, the best possible action is to learn the graph jointly with the resolution of the matrix completion task, following an iterative and alternating scheme.

Such iterative schemes have been used in prior matrix completion studies, for example in [25] prior information was assumed to be encoded into matrix completion via known subspaces of the matrix. In such a scenario, the problem turned out to be a weighted matrix completion. In reality, such subspaces cannot be exactly known, the authors argue that they can only be partially known. To address this issue the paper showed how the subspaces can be iteratively estimated from the data and used for matrix completion. Our work is of the same essence; had the prior information about user and item graphs be exactly known, our problem would be graph regularized matrix completion. Such is not the case; hence we have to resort to estimating the graphs from the data iteratively and use it for matrix completion.

3.1. Formulation

For the sake of clarity, we repeat the equation for graph regularized matrix completion (7)

$$\min_X \|Y - R(X)\|_F^2 + \mu\|X\|_* + \lambda\left(Tr\left(XL_U X^T\right) + Tr\left(X^T L_V X\right)\right)$$

For binary matrix completion, the values in X can only be 0 or 1. This is expressed as:

$$\min_X \|Y - R(X)\|_F^2 + \mu\|X\|_* + \lambda\left(Tr\left(XL_U X^T\right) + Tr\left(X^T L_V X\right)\right), \text{ s.t. } X \in \{0,1\} \quad (8)$$

Solving (8) with the said discrete constraints is difficult. Therefore, we propose a convex relaxation of the problem (8), that allows entries of X to lie between 0 and 1. The problem then reads:

$$\min_X \|Y - R(X)\|_F^2 + \mu\|X\|_* + \lambda\left(Tr\left(XL_U X^T\right) + Tr\left(X^T L_V X\right)\right), \text{ s.t. } X \in [0,1] \quad (9)$$

The formulation (9) assumes the graph Laplacians L_U and L_V to be known and fixed, but, as we have argued, this is not the case in practice and we need to learn them from the data X. Following [19], we formulate the following joint problem for simultaneous matrix completion and graph learning:

$$\min_{X,W_U,W_V} \|Y - R(X)\|_F^2 + \mu \|X\|_* + \lambda \left(\|W_U \circ Z_U(X)\|_{1,1} + \|W_V \circ Z_V(X)\|_{1,1} \right)$$
$$+\lambda\sigma^2 \sum_{i,j} W_U(i,j)\left(\log(W_U(i,j)) - 1\right) + \lambda\sigma^2 \sum_{i,j} W_V(i,j)\left(\log(W_V(i,j)) - 1\right) \text{ s.t. } X \in [0,1] \quad (10)$$

Let us introduce some useful notations. For every pair of row indexes *(i,j)* of *X*, we denote $(x_i^{\rightarrow}, x_j^{\rightarrow})$ the i-th and j-th rows, and we define the entry *(i,j)* of $Z_U(X)$ as $Z_U(X)(i,j) = \left\| x_i^{\rightarrow} - x_j^{\rightarrow} \right\|_2^2$. Similarly, for every pair of column indexes *(i,j)* of *X*, we denote $(x_i^{\downarrow}, x_j^{\downarrow})$ the i-th and j-th columns of this matrix and set $Z_V(X)(i,j) = \left\| x_i^{\downarrow} - x_j^{\downarrow} \right\|_2^2$. Thus, as emphasized in [19], we have that $\|W_U \circ Z_U\|_{1,1} = Trace\left(XL_U X^T\right)$ and $\|W_V \circ Z_V\|_{1,1} = Trace\left(X^T L_V X\right)$, by using the standard Laplacian operators definition $L_U = \Delta_U - W_U$ where Δ_U is a diagonal matrix with i-th diagonal term equals to $\Delta_U(i,i) = \sum_j W_U(i,j)$, and $L_V = \Delta_V - W_V$ with $\Delta_V(i,i) = \sum_j W_V(j,i)$. Thus, the first line of (10) is the same as that of (9). The terms in the second line can be viewed as entropy-like regularization on both graph weights, which allow learning the graph structures in a regularized manner. Due to the coupling between variables *X* and *(W_U, W_V)*, Problem (10) is nonconvex, and we will proceed in an alternative manner for its resolution. In every iteration, the variables W_U and W_V, related to the graph Laplacian are first updated, by minimizing the loss function in (10) with respect to those variables. The updates are given in [19],

$$W_U(i,j) = \exp\left(-\frac{\left\| x_i^{\rightarrow} - x_i^{\rightarrow} \right\|_2^2}{\sigma^2} \right)$$
$$W_V(i,j) = \exp\left(-\frac{\left\| x_i^{\downarrow} - x_i^{\downarrow} \right\|_2^2}{\sigma^2} \right) \quad (11)$$

Then, we perform the minimization of the function with respect to variable *X*, which amounts to solving the convex minimization problem (9). To this aim, we propose to make use of the efficient parallel proximal algorithm, called PPXA, proposed in [26]. In this approach, 5 terms, X_1, X_2, X_3, X_4, X_5, with the same size as *X*, are produced, associated to the resolution of 5 sub-problems, corresponding to the computation of the proximity operators for the 5 terms involved in the loss function in (9). For a given iteration *k*, this leads to:

$$\hat{X}_1^k = \arg\min_X \tfrac{\theta}{2}\left\|Y - R(X)\right\|_F^2 + \tfrac{1}{2}\left\|X_1^{k-1} - X\right\|_F^2$$
$$\hat{X}_2^k = \arg\min_X \mu\theta\left\|X\right\|_* + \tfrac{1}{2}\left\|X_2^{k-1} - X\right\|_F^2$$
$$\hat{X}_3^k = \min\left(\max\left(X_3^{k-1}, 0\right), 1\right)$$
$$\hat{X}_4^k = \arg\min_X Tr\left(XL_U X^T\right) + \tfrac{1}{2}\left\|X_4^{k-1} - X\right\|_F^2$$
$$\hat{X}_5^k = \arg\min_X Tr\left(X^T L_V X\right) + \tfrac{1}{2}\left\|X_5^{k-1} - X\right\|_F^2$$

with θ=5 (i.e. the number of sub-problems solved in parallel). The variable $\hat{X}_1^k$ requires solving a least-squares problem, which we solve using conjugate gradient. The variable $\hat{X}_2^k$ is a nuclear norm minimization that is obtained by singular value shrinkage [27]. $\hat{X}_3^k$ is solved by simple max and min thresholding. $\hat{X}_4^k$ and $\hat{X}_5^k$ are solved by the Sylvester equation. Once the 5 sub-problems are solved (in parallel), the next iterate is computed as the average of the variables,

$$X^k = \tfrac{1}{\theta}\left(\hat{X}_1^k + \hat{X}_2^k + \hat{X}_3^k + \hat{X}_4^k + \hat{X}_5^k\right) \quad (12)$$

Each of the proxy variables are finally updated as follows,

$$X_i^k = X_i^{k-1} + 2\hat{X}^k - \hat{X}^{k-1} - \hat{X}_i^k \quad (13)$$

The above PPXA iterations are guaranteed to converge to the solution of the convex problem (10). In practice, we alternate the graph weight update (11) with only one iteration of this method, initialized with the previous X value (associated with the past graph weight matrices). This leads to Algorithm 1, where we denoted R the matrix such that $RX = R(X)$. Moreover, the maximum and minimum operations should be performed elementwise. An empirical threshold of 0.5 is used, in the final output X, to reach a binary matrix. We have not used the exact binarization constraint in this work; following prior works [29, 30] we have followed an iterative hard thresholding approach. The exact solution to the binarization constraints would require a recently developed class of techniques called mathematical programming with equilibrium constraints [31]. Even though our solution is mathematically optimal, as we will see in the results, it gives very good results in practice.

Algorithm 1: Binary Matrix Completion with Graph Learning

Initialize: $X_1^0, X_2^0, X_3^0, X_4^0, X_5^0, X^0 = Y$

For $k = 1$ to maxiter

$$W_U{}^k(i,j) = \exp\left(-\frac{\left\|x_i^{k-1\rightarrow} - x_i^{k-1\rightarrow}\right\|_2^2}{\sigma^2}\right), \text{and update } L_U^k,$$

$$W_V{}^k(i,j) = \exp\left(-\frac{\left\|x_i^{k-1\downarrow} - x_i^{k-1\downarrow}\right\|_2^2}{\sigma^2}\right), \text{and update } L_V^k,$$

$$\hat{X}_1^k = \mathrm{LSQR}\left(\left(\theta \mathrm{R}^T \mathrm{R} + I\right), \left(X_1^{k-1} + \theta \mathrm{R}^T Y\right)\right),$$

$$\hat{X}_2^k = USV^T, \text{with } U\Sigma V^T = SVD\left(X_2^{k-1}\right),\ S = signum(\Sigma)\ \max\left(0, |\Sigma| - \tfrac{\lambda\theta}{2}\right),$$

$$\hat{X}_3^k = \min\left(\max\left(X_3^{k-1}, 0\right), 1\right),$$

$$\hat{X}_4^k = \left(2\theta\mu L_V{}^k + I\right)^{\dagger} X_4^{k-1},$$

$$\hat{X}_5^k = X_5^{k-1}\left(2\theta\mu L_U{}^k + I\right)^{\dagger}$$

$$X^k = \tfrac{1}{\theta}\left(\sum_{n=1}^{5} \hat{X}_n^k\right)$$

For $m = 1$ to 5

$$X_m^k = X_m^{k-1} + \left(2X^k - X^{k-1} - \hat{X}_m^k\right)$$

end For

end For

One can notice that our algorithm involves solving one least squares problem, one singular value decomposition, one sorting and multiplication with two Sylvesters equations. The complexity of solving the least square problem via conjugate gradient is O(n) since the iterations are run for a fixed number of iterations (usually 20) only. The complexity of SVD is $O(n^3)$. For the sorting operation, the worst case complexity is $O(n^2)$; but can be done faster in $O(n^{3/2}\log n)$ [32]. Using Bartel-Stewart algorithm the cost of solving the sylvester's equation is $O(n^3)$. One can therefore see that the dominating cost of our proposed algorithm is of $O(n^3)$.

4. Experimental Validation

4.1. Simulation Experiments

In the first set of experiments, we simulate a synthetic matrix completion problem. A low-rank matrix of size 100 x 100 of rank 'r' is created by the product of two random matrices of sizes 100 x r and r x 100. The elements of this matrix are thresholded to simulate a low-rank binary matrix. A random mask sampling *X*% of the values. Even though

the objective is to complete the partially observed matrix, our specific goal is to show that our estimated Graph Laplacian (iteratively computed) is close to the true Laplacian (computed from the fully observed matrix).

For each configuration (of sampling proportion and rank) we generate 100 different matrices and masks and employ our algorithm to complete the matrix. The normalized mean squared error (NMSE) between the actual Laplacian (computed from the fully observed matrix) and estimated Laplacian is reported, defined as:

$$NMSE = \frac{\left\| L_{actual} - L_{estimated} \right\|_F^2}{\left\| L_{actual} \right\|_F^2}$$

The means and the standard deviations are reported in Table 1.

Table 1. Mean and Standard Deviations for Different Ranks and Sampling Proportions

Sampling Proportion	Rank = 5 (mean ±std)	Rank = 10 (mean ±std)	Rank = 15 (mean ±std)	Rank = 20 (mean ±std)
20%	0.28 ±0.16	0.33 ±0.19	0.35 ±0.22	0.53 ±0.29
40%	0.19 ±0.13	0.23 ±0.15	0.29 ±0.16	0.41 ±0.21
60%	0.13 ±0.10	0.17 ±0.11	0.23 ±0.13	0.28 ±0.15
80%	0.08 ±0.04	0.11 ±0.05	0.16 ±0.09	0.21 ±0.10

These results are as expected. When the estimated matrix is close to the actual ground truth, the graph Laplacian is estimated correctly, while when the estimated matrix and the ground truth differ, the true graph Laplacian and the estimated one also differ. According to the theoretical recovery guarantees in low-rank matrix completion [28], the number of observed samples (m) required should be

$$m \geq C \cdot n^{1.2} \cdot r \cdot \log n$$

where n is the number of elements in the matrix and r is the rank. Another way to interpret this minimum sampling requirement is to say that when the number of samples is large and the rank is small, the quality of the recovery will be good, but when the number of samples is small and the rank is large, the recovery will deteriorate. This is what is happening in our case. When the estimated matrix is properly recovered, the Laplacian is close to the ground truth and vice versa.

The concept of the requirement of iteratively updating the graph Laplacians has been discussed at the beginning of section 3. Here we empirically validate it. We show the plot NMSE between the ground truth graph Laplacian and the estimated graph Laplacians with iterations. Results are shown for two cases, rank = 5, sampling proportion = 80% and

rank = 20, sampling proportion = 20%. We have taken two extreme cases – the best and the worst scenarios. In both cases, we see from the following plot that the error reduces with iterations.

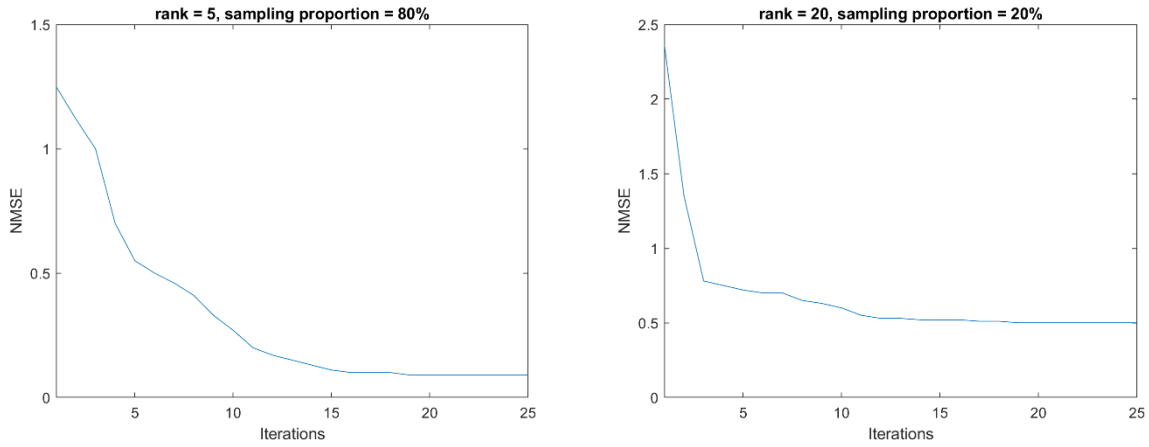


Fig. 1. Error vs Iterations.

4.2. Collaborative Filtering

We carry our evaluation on movie recommendations. Experiments are carried out on three popular datasets Movielens 100K, Movielens 1M, Movielens, and 10M. All of them are freely available from https://grouplens.org/datasets/movielens/.

1. movie-100K: 100,000 ratings for 1682 movies by 943 users;
2. movie-1M: 1 million ratings for 3900 movies by 6040 users;
3. movie-10M: 10 million ratings for 10681 movies by 71567 users.

For these datasets, the splits between training and test sets are already pre-defined. The protocol is to carry out 5 fold cross-validation on these sets. The first two datasets (100K and 1M) can be handled on a personal computer, but the 10M dataset is too large. Therefore, we use a divide and conquer strategy, as it was described in [28] in the context of matrix completion. Our implementations were run on an Intel i7 processor with 16 GB RAM running a 64 bit Windows 10. We have compared our work with binary matrix completion (BMC) [1] and matrix completion graph (MCG) [17]. We have also compared against two recent techniques, namely neural graph collaborative filtering (NGCF) [33] and Markov random field (MRF) [34].

For evaluation metrics we have used area under the curve (AUC), area under the precision recall curve (AUPR), Normalized Discounted Cumulative Gain (NDCG) and Hit Rate. These are the latest metrics used for evaluating collaborative filtering algorithms, for definitions please refer to [35].

Our work needs specification of the three parameters μ, λ and σ. For all the datasets the same set of parameters were found to yield uniformly good results. The values used were μ=.1, λ=.1 and σ=10. For the benchmark studies, the parametric values were taken from the corresponding papers.

Table 2. Results on Movie-100K

Algorithm	AUC	AUPR	NDCG@10	Hit Rate
BMC	.6827 ± .0025	.1597 ± .0013	.6436 ± .1575	.7086 ± .0030
MCG	.7198 ± .0034	.7405 ± .0045	.7984 ± .1032	.6226 ±.0042
NGCF	.7333 ± .0109	.7057 ± .0097	.8796 ± .1048	.7098 ± .0109
MRF	.7629 ± .0135	.7708 ± .0159	.9291 ± .1083	.7137 ± .0145
Proposed	**.7616 ± .0042**	**.7815 ± .0048**	**1.000 ± .0000**	**.7117 ± .0037**

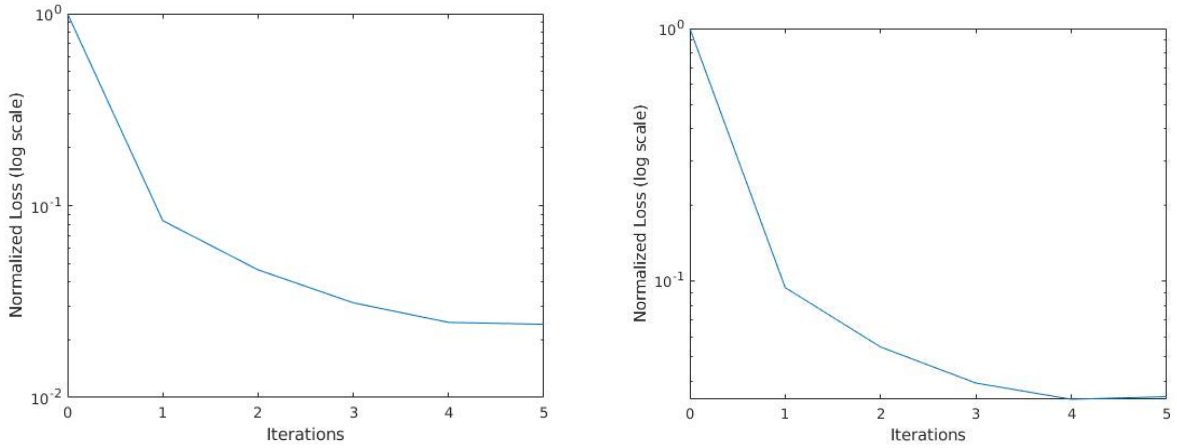
Table 3. Results on Movie-1M

Algorithm	AUC	AUPR	NDCG@10	Hit Rate
BMC	.7193 ± .0007	.1275 ± .0001	.8077 ± .1801	**.7478 ± .0012**
MCG	.6993 ± .0018	.7369 ± .0023	.7762 ± .2058	.6784 ± .0009
NGCF	.7342 ± .0126	.7008 ± .0094	.8891 ± .1304	.7298 ± .0097
MRF	.7587 ± .0151	.7887 ± .0101	.9525 ± .1510	.7396 ± .0123
Proposed	**.7711 ± .0016**	**.7984 ± .0019**	**1.000 ± .0000**	.7320 ± .0012

Table 4. Results on Movie-10M

Algorithm	AUC	AUPR	NDCG@10	Hit Rate
BMC	.7039 ± .0010	.1385 ± .0004	.6247 ± .2608	**.7412 ± .0010**
MCG	.6617 ± .0011	.7166 ± .0019	.9149 ± .1452	.6740 ± .0017
NGCF	.7313 ± .0087	.7015 ± .0086	.9593 ± .2432	.7336 ± .0092
MRF	.7446 ± .0109	.7598 ± .0163	.9804 ± .3397	.7415 ± .0131
Proposed	**.7568 ± .0039**	**.7961 ± .0030**	**.9983 ± .0025**	.7382 ± .0024

The results are shown in Tables 2, 3 and 4. We report the average mean and standard deviation for 5-fold cross-validation. From these results, we find that the matrix completion-based techniques are more robust than others. BMC, MCG and our proposed algorithm have far smaller standard deviations compared to NGCF and MRF. For all three datasets, our proposed technique yields the best results in terms of AUC, AUPR and NDCG. In terms of Hit Rate, we are doing slightly worse than BMC. We show the empirical convergence plot of our algorithm, depicting the evolution of the loss function in (10) along iterations. The results are shown on semi-log scale. One can notice that our algorithm reaches stability in very few iterations. The plots are shown for 100K and 1M. It is not possible to show convergence plots for the largest dataset, since it is solved in a piecemeal fashion using the divide-and-conquer approach [28].



(a) (b)

Fig. 2 – Convergence plot on semi-log scale. 1a. Movielens-100K. 1b. Movielens-1M

4.3. Ablation Study

In the final set of experiments we empirically look at the effects of different penalties in the optimization problem. There are three main penalties in our proposed optimization problem – a) Nuclear norm, b) User graph update, and c) Item graph update. Based on these penalties we can have four realistic scenarios.

1. Keeping user and item graph terms but omitting the nuclear norm term.
2. Assuming the item and user graphs to be constant and not updating them.
3. Assuming no item information is available, therefore omitting the item graph terms.
4. Assuming no user information is available, therefore omitting the user graph terms.

The results of these four different scenarios are shown for all the three datasets in the following Table 5. We only show the results in terms of AUC.

Table 5. AUC results for ablation studies

Dataset	w/o nuclear norm	w/o graph update	w/o item graph	w/o user graph	with all penalties
Movie-100K	.7213 ± .0158	.7410 ± .0098	.7303 ± .0095	.7311 ± .0090	.7616 ± .0042
Movie-1M	.7286 ± .0115	.7451 ± .0072	.7325 ± .0066	.7317 ± .0075	.7711 ± .0016
Movie-10M	.7199 ± .0161	.7357 ± .0083	.7286 ± .0088	.7290 ± .0085	.7568 ± .0039

We observe that omitting the nuclear norm (column 2) has a pronounced effect on the result; the AUC falls considerably if this is omitted. This is expected since the entire concept of low-rank matrix completion hinges on this particular penalty. Without the graph update (column 3) the algorithm assumes that the user and item graphs are fixed. These graphs are computed from partially observed entries; hence as argued, are not very accurate. This is reflected

in the drop in AUC. Completely ignoring the item or user graphs (columns 4 and 5) also deteriorate the results; in fact, the deterioration is worse than not updating the graphs. This is expected since we are completely ignoring the prior information about items and users; this was not the case in column 3.

5. Conclusion

This work proposes binary matrix completion on graphs. The difference with prior graph regularized matrix completion studies is that we can learn the graph from the data, while prior studies assumed the graph to be fixed and given (computed from partially observed matrix). By iteratively learning the graph while completing the matrix improves the quality of the graph Laplacian and overall matrix completion. Experiments on collaborative filtering showed that the proposed approach yields better results than binary matrix completion (without graphs) and matrix completion with fixed graphs. Our algorithm yields better overall results than state-of-the-art algorithms in collaborative filtering.

In this work our goal was to recover a binary matrix, however our binarization constraint was only approaximate. In the future we would like to improve upon this. We will look into well known approaches like algebraic techniques for semidefinite optimization [36, 37] as well as into new approaches being developed in recent times [38, 39].

An interesting practical aspect of recommendations today is for video streaming services. The algorithms that are developed for the static scenario (addressed in this work) do not always translate to data streams. However, in recent times with the advent of smart TVs and OTT platforms recommendations for video streams is of increasing importance. We would like to explore how the IoT aspects can be incorporated into recommendation systems in the future. Some literature on related area already exists [40, 41]; we would like to build upon them.

Acknowledgment

This work is funded by the Indo-French project CEFIPRA DST-CNRS-2016-02. Angshul Majumdar is partially supported by Infosys Center for Artificial Intelligence at IIIT Delhi.

References

[1] M. A. Davenport, Y. Plan, E. van den Berg and M. Wootters, "1-Bit matrix completion," Information and Inference: A Journal of the IMA, vol. 3, no. 3, pp. 189-223, 2014.

[2] C. Liu and H. Shan, "Binary Matrix Completion with Nonconvex Regularizers," IEEE Access, vol. 7, pp. 65415-65426, 2019.

[3] S. A. Bhaskar, "Quantized matrix completion for low rank matrices," IEEE ICASSP, pp. 3741-3745, 2015.

[4] R. Schachtner, G. Pöppel and E. W. Lang, "Towards unique solutions of non-negative matrix factorization problems by a determinant criterion," Digital Signal Processing, vol. 21, no. 4, pp. 528-534, 2011.

[5] J. Fan and J. Cheng, "Matrix completion by deep matrix factorization," Neural Networks, vol. 98, pp. 34-41, 2018.

[6] R. Sun and Z. Luo, "Guaranteed Matrix Completion via Non-Convex Factorization," IEEE Transactions on Information Theory, vol. 62, no. 11, pp. 6535-6579, 2016.

[7] B. Recht, "A Simpler Approach to Matrix Completion," Journal of Machine Learning Research, vol. 12, no. 12, pp. 3413-3430, 2011.

[8] S. Majidian, M. M. Mohades and M. H. Kahaei, "Matrix completion with weighted constraint for haplotype estimation," Digital Signal Processing, vol. 108, no. 102880, 2021.

[9] R. Kumar, C. Da Silva, O. Akalin, A. Y. Aravkin, H. Mansour, B. Recht and F. J. Herrmann, "Efficient matrix completion for seismic data reconstruction," Geophysics, vol. 80, no. 5, pp. V97-V114, 2015.

[10] K. Liu and Y. D. Zhang, "Coprime array-based DOA estimation in unknown nonuniform noise environment," Digital Signal Processing, vol. 79, pp. 66-74, 2018.

[11] S. A. Hamza and M. G. Amin, " Sparse array design for maximizing the signal-to-interference-plus-noise-ratio by matrix completion," Digital Signal Processing, vol. 105, no. 102678, 2020.

[12] J. Abernethy, F. Bach, T. Evgeniou, and J. P. Vert, "A New Approach to Collaborative Filtering: Operator Estimation with Spectral Regularization," Journal of Machine Learning Research, vol. 10, no. 3, 2009.

[13] J. Fan and W. S. Chow, "Sparse subspace clustering for data with missing entries and high-rank matrix completion," Neural Networks, vol. 93, pp. 36-44, 2017.

[14] B. Liu, Y. Li and Z. Xu, "Manifold regularized matrix completion for multi-label learning with ADMM," Neural Networks, vol. 101, pp. 57-67, 2018.

[15] A. Mongia and A. Majumdar, “Drug-target interaction prediction using Multi Graph Regularized Nuclear Norm Minimization,” PLOS ONE, vol. 15, no. 1, p.e0226484, 2020.

[16] P. Ezzat, M. Zhao, M. Wu, X. Li and C. Kwoh, "Drug-Target Interaction Prediction with Graph Regularized Matrix Factorization," IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 14, no. 3, pp. 646-656, 2017.

[17] A. Mongia and A. Majumdar, "Matrix completion on multiple graphs: Application in collaborative filtering," Signal Processing, vol. 165, pp. 144-148, 2019.

[18] Q. Gu, J. Zhou and C. Ding, “Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs,” SIAM SDM, pp. 199-210, 2010.

[19] V. Kalafolias, “How to learn a graph from smooth signals”, Proceedings of Machine Learning Research, vol. 51, pp. 920-929, 2016.

[20] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," IJCAI, 1999.

[21] O. Shamir and S. Shalev-Shwartz, “Collaborative filtering with the trace norm,” COLT, pp. 661-678, 2011.

[22] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," UAI, pp. 43-52, 1998.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," WWW, pp. 285- 295, 2001.

[24] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang and Y. Li, "A survey of matrix completion methods for recommendation systems," Big Data Mining and Analytics, vol. 1, no. 4, pp. 308-323, 2018.

[25] A. Eftekhari, D. Yang and M. B. Wakin, "Weighted Matrix Completion and Recovery With Prior Subspace Information," in IEEE Transactions on Information Theory, vol. 64, no. 6, pp. 4044-4071, 2018.

[26] P.L. Combettes and J.C. Pesquet, “A proximal decomposition method for solving convex variational inverse problems”, Inverse Problems, vol. 24, no. 6, p. 065014, 2008.

[27] A. Majumdar and R. K. Ward, “Some empirical advances in matrix completion,” Signal Processing, vol. 91, no. 5, pp. 1334-1338, 2011.

[28] L. Mackey, A. Talwalkar and M. I. Jordan, “Distributed matrix completion and robust factorization,” Journal of Machine Learning Research, vol. 16, no.1, pp. 913-960, 2015.

[29] A. Beck and Y. C. Eldar, “Sparsity constrained nonlinear optimization: Optimality conditions and algorithms,” SIAM Journal on Optimization, vol. 23, no. 3, pp. 1480–1509, 2013.

[30] X. Yuan and T. Zhang, “Truncated power method for sparse eigenvalue problems,” Journal of Machine Learning Research, vol. 14, no. 1, pp. 899–925, 2013.

[31] G. Yuan and B. Ghanem, “An exact penalty method for binary optimization based on MPEC formulation,” AAAI Conference on Artificial Intelligence, pp. 2867–2875, 2017.

[32] S. Kavitha, V. Vijay and A. B. Saketh, “Matrix Sort-A Parallelizable Sorting Algorithm,” International Journal of Computer Applications, 975, p.8887, 2016.

[33] X. Wang, X. He, M. Wang, F. Feng and T. S. Chua, “Neural graph collaborative filtering,” ACM SIGIR, pp. 165-174, 2019.

[34] H. Steck, “Markov Random Fields for Collaborative Filtering,” NIPS, pp. 5473-5484, 2019.

[35] S. Zhang, L. Yao, L. V. Tran, A. Zhang and Y. Tay, “Quaternion collaborative filtering for recommendation”, arXiv preprint arXiv:1906.02594, 2019.

[36] J. Dattorro, “Convex optimization & Euclidean distance geometry,” Lulu. Com, 2010.

[37] P. Parrilo, “Algebraic techniques and semidefinite optimization,” Lecture 18, MIT 6.972, 2006.

[38] X. Li, C. Sun and Y. Ye, “Simple and Fast Algorithm for Binary Integer and Online Linear Programming,” NeurIPS, 2020.

[39] S. Elloumi, A. Lambert and A. Lazare, “Solving unconstrained 0-1 polynomial programs through quadratic convex reformulation,” Journal of Global Optimization,vol. 80, no. 2, pp.231-248, 2021.

[40] X. Liu, X. B. Zhai, W. Lu and C. Wu, "QoS-Guarantee Resource Allocation for Multibeam Satellite Industrial Internet of Things With NOMA," in IEEE Transactions on Industrial Informatics, vol. 17, no. 3, pp. 2052-2061, 2021.

[41] X. Liu and X. Zhang, "Rate and Energy Efficiency Improvements for 5G-Based IoT With Simultaneous Transfer," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 5971-5980, Aug. 2019