ROBUST ONLINE MATRIX COMPLETION ON GRAPHS

Symeon Chouvardas, Mohammed Amin Abdullah, Lucas Claude, Moez Draief

Mathematical and Algorithmic Sciences Lab,
Huawei France R&D,
Paris, France.

ABSTRACT

We study online robust matrix completion on graphs. At each iteration a vector with some entries missing is revealed and our goal is to reconstruct it by identifying the underlying low-dimensional subspace from which the vectors are drawn. We assume there is an underlying graph structure to the data, that is, the components of each vector correspond to nodes of a certain known and fixed graph, and their values are related accordingly. We propose an algorithm that exploits the graph to reconstruct the incomplete data, in the scenario where there is outlier noise. The theoretical properties of the algorithms are studied and numerical experiments using both synthetic and real world datasets verify the improved performance of the proposed technique compared to other state of the art algorithms.

Index Terms— Matrix Completion, Graphs, Robust Subspace Tracking.

1. INTRODUCTION

Modern technologies have produced a vast array of data-generating devices (e.g., smart phones, cameras, sensors) and processes (e.g., web searches, surveys, social media interaction). Frequently, it is known or conjectured that there is an underlying *structure* to the data, and further, that this structure reflects some simpler underlying process. To make this more concrete, consider the notion of *sparsity* in a movie-rating database such as used by Netflix. If the rows of the ratings matrix represent movies and the columns people, then it is reasonable to suppose, and indeed has been found to be the case ([1, 2]), that the ratings matrix is of low rank. This would reflect that ratings vectors really "live" in a small subspace of the ambient space they are generated in. Consequently, understanding this subspace better allows the exploitation of the data, for example, through more tightly focused marketing. On the other hand, the matrix will be incomplete, since any given user will rate only a very small subset of all the movies available. Thus, one would desire *completion* of the matrix: a low-rank matrix with sufficiently many observed entries can be *exactly* reconstructed, and in the last decade, this has been a very active area of research in the signal processing and machine learning communities. However, low-dimensional subspaces are not the only form of structure; indeed, there can also be an underlying *graphical* structure that represents connections/relations between entities. For example, one may generate a graph where movies are nodes and and two movies are linked if they both star a famous actor. Such a structure is sometimes easy to discover, and it immediately begs the question of how (if at all) it can be used to help fill in the missing entries of the matrix.

1.1. RELATED WORK

The problem of matrix completion is a well studied one and several solutions have been proposed during the past years, see for example [3, 4, 5]. The online setup has its roots in the so–called subspace tracking problem, e.g., [6], in which the columns of a matrix are revealed sequentially one per iteration step and the goal is the identification of the underlying subspace. Extensions of these works, which deal with the presence of missing entries and/or outliers have been studied in [7, 8, 9, 10, 11, 12, 13, 14]. The batch version of the matrix completion on graphs problem was originally presented in [15] and extended to its robust version, which deals with the presence of outliers, in [16].

1.2. OUR CONTRIBUTION

In this work, we extend the idea presented in [15] and we propose a robust online algorithm for matrix completion exploiting graph information. Here, we propose an online solution, i.e., the columns of the matrix appear and are processed sequentially, one per iteration step. To that direction, at each iteration step we define a proper cost function and we minimize it to produce the updated estimates. Furthermore, we study the case where there is outlier noise, which corrupts a small subset of the observed vector. We propose a robust solution, which estimates the outlier noise and cleans the data before updating the quantities of interest. To the best of our knowledge, this is the first work dealing with the online matrix completion on graphs problem. Including graph information makes the problem more challenging both in deriving the algorithm as well as analyzing it.

For lack of space, we omit proofs and refer the reader to [17], which includes expanded coverage of the results of this paper.

Notation: Lowercase and uppercase boldfaced letters stand for vectors and matrices respectively. $\|\boldsymbol{A}\|$ is the operator norm and $\|\boldsymbol{A}\|_F$ the Frobenius norm of matrix $\boldsymbol{A}$. $\|\boldsymbol{x}\|$, $\|\boldsymbol{x}\|_1$ denote the Euclidean and the ℓ_1 norms of vector $\boldsymbol{x}$, respectively. The symbol $\otimes$ stands for the Kronecker product. Finally, $\boldsymbol{I}_{mr}$ is the $mr \times mr$ identity matrix and $\boldsymbol{O}_{a\times b}$ is the zero matrix of dimension $a \times b$.

2. MATRIX COMPLETION ON GRAPHS

The original task of matrix completion (MC), e.g., [3, 18], is the recovery of a data matrix from a sample of its entries. Formally, given a matrix $\boldsymbol{X}$ of dimension $m \times n$ we have access to $k \ll m \cdot n$ entries and the goal is the prediction of the rest unobserved ones. It has been shown that with low rank and some mild assumptions regarding the positions of the observed entries, this can be achieved [4, 3]. The problem can be summarized as follows: Compute a minimum rank matrix $\boldsymbol{A}$ which is equal to the observation matrix $\boldsymbol{X}$ in the set of

observed entries Ω; i.e.,

$$\min_{\boldsymbol{A}} \operatorname{rank}(\boldsymbol{A})$$
$$s.t.\ A_{ij} = X_{ij},\ \forall i,j \in \Omega$$

where $X_{ij},\ A_{ij}$ is the i,j–th entry of $\boldsymbol{X}$ and $\boldsymbol{A}$ respectively. The above is NP-hard [3]. However, it has been shown [4] that it can be relaxed and solved efficiently via convex optimization. The relaxation can be written as follows:

$$\min_{\boldsymbol{A}} \|\boldsymbol{A}\|_* \tag{1}$$
$$s.t.\ A_{ij} = X_{ij},\ \forall i,j \in \Omega, \tag{2}$$

where $\|\boldsymbol{A}\|_*$ denotes the nuclear norm of the matrix $\boldsymbol{A}$ with definition: $\|\boldsymbol{A}\|_* = \sum_{k=1}^{\min(m,n)} \sigma_k(\boldsymbol{A})$, with $\sigma_k(\cdot)$ being the k–th larger singular value. This model can be generalized so that to take into account the presence of noise. In that case the equality constraint can be relaxed and the optimization problem becomes:

$$\min_{\boldsymbol{A}}\ \lambda_1\|\boldsymbol{A}\|_* + \frac{1}{2}\|P_\Omega(\boldsymbol{A}-\boldsymbol{X})\|_F^2, \tag{3}$$

where P_Ω is an operator which sets the entries of its matrix argument not in Ω to zero, and keeps the rest unchanged and $\lambda_1 > 0$ is a regularization term.

Low rank implies the linear dependence of rows/columns of $\boldsymbol{X}$. However, this dependence is unstructured. In many situations, the rows and/or columns of matrix $\boldsymbol{X}$ possess additional structure that can be incorporated into the completion problem in the form of a regularization. In this paper, we assume that the rows of $\boldsymbol{X}$ are given on vertices of graphs. More formally, let us be given an undirected graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{W})$ on the rows with vertices $\boldsymbol{V} = \{1,\dots,m\}$, edges $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ and non-negative weights on the edges represented by the symmetric $m \times m$ matrix $\boldsymbol{W}$. If there is an edge between i,j, then $W_{ij} = W_{ji} = 0$, and we shall assume the graph has no parallel edges or loops. The latter means that the diagonal elements of $\boldsymbol{W}$ are zero.

The weights capture strength of association between the row elements, and we represent them via the *Laplacian*. This is a positive semidefinite (PSD) matrix $\boldsymbol{L}$ defined as $\boldsymbol{D} - \boldsymbol{W}$ where $\boldsymbol{D}$ is the diagonal matrix such that $D_{ii} = \sum_{j=1}^m W_{ij}$.

The problem of matrix completion over graphs can be formulated as follows, [15]:

$$\min_{\boldsymbol{A}}\ \lambda_1\|\boldsymbol{A}\|_* + \frac{1}{2}\|P_\Omega(\boldsymbol{A}-\boldsymbol{X})\|_F^2 + \lambda_2 \operatorname{tr}\left(\boldsymbol{A}^T\boldsymbol{L}\boldsymbol{A}\right), \tag{4}$$

where $\operatorname{tr}\left(\boldsymbol{A}^T\boldsymbol{L}\boldsymbol{A}\right)$ is a graph smoothing regularization constraint and $\lambda_2 > 0$ is the regularization parameter associated with it. In fact it holds that

$$\sum_{i,j} W_{ij}\|\boldsymbol{a}_i - \boldsymbol{a}_j\|^2 = \operatorname{tr}\left(\boldsymbol{A}^T\boldsymbol{L}\boldsymbol{A}\right),$$

with $\boldsymbol{a}_i$ being the i–th row of the matrix $\boldsymbol{A}$. In words we demand that the rows corresponding to neighboring nodes to be "close" (in some sense) to each other. This problem, which was originally been proposed in [15] has been generalized in [16] to tackle scenarios where outliers are present.

Before we turn our focus to the online problem, we present some useful properties of the nuclear norm. The nuclear norm of a matrix $\boldsymbol{M}$ of rank r can be written as [19]

$$\|\boldsymbol{M}\|_* = \min_{\boldsymbol{U}\in\mathbb{R}^{m\times r},\boldsymbol{R}\in\mathbb{R}^{r\times n}} \{\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{R}\|_F^2\} \quad s.t.\, \boldsymbol{M} = \boldsymbol{U}\boldsymbol{R}. \tag{5}$$

Note that the number of columns of the matrix $\boldsymbol{U}$, denoted by r, is also a variable. The problem of estimating r goes beyond the scope of this paper ***and from now on we will consider that r will be equal to the rank of $\boldsymbol{X}$ and will be known***. This assumption was also made in other papers (e.g., [8, 7]) dealing with online matrix completion. Taking this into account and substituting (5) into (4) leads us to:

$$\min_{\boldsymbol{U},\boldsymbol{R}:\,\boldsymbol{U}\boldsymbol{R}\in\mathbb{R}^{m\times n}} \lambda_1\left(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{R}\|_F^2\right) + \frac{1}{2}\|P_\Omega(\boldsymbol{U}\boldsymbol{R}-\boldsymbol{X})\|_F^2 + \lambda_2 \operatorname{tr}\left(\boldsymbol{R}^T\boldsymbol{U}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{R}\right). \tag{6}$$

2.1. ONLINE MATRIX COMPLETION ON GRAPHS

The above deals with the batch problem, i.e., the one in which all the measurements are available a priori and are used in the computations as a whole. However, in many applications, having access to all the data may be impractical and/or infeasible. More specifically, in big data applications, the data might not be able to be stored and the algorithm needs to retrieve them from slow memory devices or to access them over networks. Moreover, in batch operation the unknown subspace has to be re-computed from scratch whenever a new datum becomes available. Our goal here is to present an online solution, but before we do so, we discuss robustification.

A drawback of the matrix completion techniques, which rely on the Frobenious norm minimization is that they are sensitive to heavy tailed noise. In the batch scenario, Robust PCA (RPCA) originally proposed in [5] overcomes this limitation. In particular, the model generating the matrix comprising missing entries is the following:

$$\boldsymbol{X} = \boldsymbol{M} + \boldsymbol{S}, \tag{7}$$

where $\boldsymbol{M}$ is a low rank matrix and $\boldsymbol{S}$ is a sparse matrix, the entries of which have arbitrarily large amplitude; the latter matrix models the outlier noise. The optimization problem for the matrix completion takes the following form:

$$\min_{\boldsymbol{M},\boldsymbol{S}} \|\boldsymbol{M}\|_* + \lambda_s\|\boldsymbol{S}\|_1,$$
$$\text{s.t. } \boldsymbol{M} + \boldsymbol{S} = \boldsymbol{X},$$

where $\|\cdot\|_1$ promotes sparsity and has the following definition $\|\boldsymbol{S}\|_1 = \sum_{i,j}|S_{i,j}|$, i.e., the sum of absolute values of the entries of S.

The aforementioned problem has been extended to the online scenario, e.g., [8, 7]. This will be our starting point for deriving the online robust MC algorithm on graphs. At each step, i.e., t, a single column of the matrix $\boldsymbol{X}$, denoted $\boldsymbol{x}_t \in \mathbb{R}^m$ (which also has missing entries) becomes available. It is generated as

$$\boldsymbol{x}_t = P_{\Omega_t}\left(\boldsymbol{U}\boldsymbol{r}_t + \boldsymbol{s}_t + \boldsymbol{v}_t\right),$$

where $\boldsymbol{U}$ is an $m \times r$ matrix, $\boldsymbol{r}_t \in \mathbb{R}^r$, $\boldsymbol{v}_t \in \mathbb{R}^m$ is the noise process, $\boldsymbol{s}_t$ is the outlier vector and Ω_t is the set of observed entries at time t. For this, we assume that the ℓ_0 (pseudo) norm, which counts the number of non-zero coefficients, is bounded and smaller than m, i.e., $\|\boldsymbol{s}_t\|_0 \le m' < m$.[1] The problem we want to solve becomes:

$$\min_{\boldsymbol{U},\{\boldsymbol{r}_\tau\},\{\boldsymbol{s}_\tau\}} \frac{\lambda_1}{2}\|\boldsymbol{U}\|_F^2 + \sum_{\tau=1}^{t}\Big(\frac{1}{2}\|P_{\Omega_\tau}(\boldsymbol{x}_\tau - \boldsymbol{U}\boldsymbol{r}_\tau - \boldsymbol{s}_\tau)\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{r}_\tau\|_2^2 + \frac{\lambda_2}{2}(\boldsymbol{r}_\tau^T\boldsymbol{U}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{r}_\tau) + \frac{\lambda_3}{2}\|\boldsymbol{s}_\tau\|_1\Big), \tag{8}$$

[1] In practice if $m' = O(\log m)$ then we can recover the sparse vector.

Algorithm 1: Online Robust Matrix Completion on Graphs

Input: λ_1, λ_2, $\boldsymbol{L}$
Output: Computed Subspaces $\boldsymbol{U}_t$ and vectors $\boldsymbol{r}_t,\ \boldsymbol{s}_t$
Initialize: $\boldsymbol{U}_0$
for $t = 1, 2, \ldots$ **do**
  Compute $\boldsymbol{s}_t$ by solving the lasso (11) using $\boldsymbol{U}_{t-1}$ and $\boldsymbol{\Omega}_t$
  Compute $\boldsymbol{r}_t$ by applying equation (10)
  Update $\boldsymbol{R}_t$ and $\boldsymbol{Q}_t$ using $\boldsymbol{x}_t$, $\boldsymbol{r}_t$ and $\boldsymbol{s}_t$
  Compute $\boldsymbol{U}_t$ using(12)

where $\lambda_3 > 0$.

For given $\boldsymbol{\Omega}, \boldsymbol{U}$, $\boldsymbol{x}$ and $\boldsymbol{s}$, the expression

$$\|\boldsymbol{\Omega}(\boldsymbol{x} - \boldsymbol{U}\boldsymbol{r} - \boldsymbol{s})\|_2^2 + \lambda_1\|\boldsymbol{r}\|_2^2 + \lambda_2(\boldsymbol{r}^T\boldsymbol{U}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{r}) + \lambda_3\|\boldsymbol{s}\|_1 \quad (9)$$

is minimized when

$$\boldsymbol{r} = \boldsymbol{B}(\boldsymbol{x} - \boldsymbol{s}), \quad (10)$$

where $\boldsymbol{B} = \boldsymbol{A}^{-1}\boldsymbol{U}^T\boldsymbol{\Omega}$ and $\boldsymbol{A} = \lambda_1\boldsymbol{I}_r + \boldsymbol{U}^T(\boldsymbol{\Omega} + \lambda_2\boldsymbol{L})\boldsymbol{U}$. Treating $\boldsymbol{r}$ as a function of $\boldsymbol{s}$ and plugging it back into (9), the joint minimization of $\boldsymbol{r}$ and $\boldsymbol{s}$ for given $\boldsymbol{U}$ is formulated as

$$\min_{\boldsymbol{s}} \|\boldsymbol{\Omega}(\boldsymbol{I}_m - \boldsymbol{U}\boldsymbol{B})(\boldsymbol{x} - \boldsymbol{s})\|_2^2 + \|\sqrt{\lambda_1}\boldsymbol{B}(\boldsymbol{x} - \boldsymbol{s})\|_2^2 + \|\sqrt{\lambda_2}\boldsymbol{L}^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{B}(\boldsymbol{x} - \boldsymbol{s})\|_2^2 + \lambda_3\|\boldsymbol{s}\|_1,$$

where we have used that $\boldsymbol{L}$ is PSD. This, in turn can be formulated as the following lasso estimator:

$$\min_{\boldsymbol{s}} \|\boldsymbol{C}(\boldsymbol{x} - \boldsymbol{s})\|_2^2 + \lambda_3\|\boldsymbol{s}\|_1, \quad (11)$$

where $\boldsymbol{C}$ is the $(m + r + m) \times m$ matrix such that $\boldsymbol{C} = \left[(\boldsymbol{\Omega}(\boldsymbol{I}_m - \boldsymbol{U}\boldsymbol{B}))^T, \sqrt{\lambda_1}\boldsymbol{B}^T, \sqrt{\lambda_2}\left(\boldsymbol{L}^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{B}\right)^T\right]^T$. This is a convex optimization problem and therefore efficiently solvable. We use the above to compute $\boldsymbol{r}_t$ and $\boldsymbol{s}_t$ using $\boldsymbol{\Omega}_t$, $\boldsymbol{x}_t$ and $\boldsymbol{U}_{t-1}$. Similar to the above algorithm, we use these computed values to compute $\boldsymbol{U}_t$. Taking partial derivative of (8) with respect to $\boldsymbol{U}$ and setting it to zero, we get $\boldsymbol{Q}_t = \lambda_1\boldsymbol{U} + \lambda_2\boldsymbol{L}\boldsymbol{U}\boldsymbol{R}_t + \sum_{\tau=1}^t \boldsymbol{\Omega}_\tau\boldsymbol{U}\boldsymbol{r}_\tau\boldsymbol{r}_\tau^T$ where $\boldsymbol{Q}_t = \sum_{\tau=1}^t \boldsymbol{\Omega}_\tau(\boldsymbol{x}_\tau - \boldsymbol{s}_\tau)\boldsymbol{r}_\tau^T$ and, as before, $\boldsymbol{R}_t = \sum_{\tau=1}^t \boldsymbol{r}_\tau\boldsymbol{r}_\tau^T$.

We vectorize and solve, thereby getting

$$\boldsymbol{u} = \left(\sum_{\tau=1}^t \boldsymbol{r}_\tau\boldsymbol{r}_\tau^T \otimes \boldsymbol{\Omega}_{\boldsymbol{\tau}} + \lambda_1\boldsymbol{I}_{mr} + \boldsymbol{R}_t \otimes \boldsymbol{L}\right)^{-1} \boldsymbol{q}_t, \quad (12)$$

where $\boldsymbol{q}_t = \mathrm{vec}\{\boldsymbol{Q}_t\}$. The core steps of the algorithm are summarized in Algorithm 1.

3. CONVERGENCE

Defining $g_t(\boldsymbol{U}, \boldsymbol{r}, \boldsymbol{s}) := \left(\frac{1}{2}\|P_{\Omega_t}(\boldsymbol{x}_t - \boldsymbol{U}\boldsymbol{r} - \boldsymbol{s})\|_2^2 + \frac{\lambda_1}{2}\|\boldsymbol{r}\|_2^2 + \frac{\lambda_2}{2}(\boldsymbol{r}^T\boldsymbol{U}^T\boldsymbol{L}\boldsymbol{U}\boldsymbol{r}) + \lambda_3\|\boldsymbol{s}\|_1\right)$, Algorithm 1 effectively aims to minimize the following[2]: $C_t(\boldsymbol{U}) = \frac{1}{t}\sum_{\tau=1}^t \min_{\boldsymbol{r},\boldsymbol{s}} g_\tau(\boldsymbol{U}, \boldsymbol{r}, \boldsymbol{s}) + \frac{\lambda_1}{2t}\|\boldsymbol{U}\|_F^2$.

[2]We normalize with t so as to prevent the existence of unbounded values. It can be readily seen that the solution at each time step doesn't depend on the normalization.

As time increases, minimization of $C_t(\boldsymbol{U})$ becomes computationally demanding since it involves solving t least squares and t ℓ_1 minimization problems for the estimation of $\boldsymbol{r}$ and $\boldsymbol{s}$ respectively. For this reason, the algorithm actually minimizes the following approximation of the above cost function:

$$\widehat{C}_t(\boldsymbol{U}) = \frac{1}{t}\sum_{\tau=1}^t g_\tau(\boldsymbol{U}, \boldsymbol{r}_\tau, \boldsymbol{s}_\tau) + \frac{\lambda_1}{2t}\|\boldsymbol{U}\|_F^2, \quad (13)$$

where $\{\boldsymbol{r}_t, \boldsymbol{s}_t\} = \arg\min_{\boldsymbol{r},\boldsymbol{s}} g_t(\boldsymbol{U}_{t-1}, \boldsymbol{r}, \boldsymbol{s})$.

Theorem 1 *Under certain assumptions, which are detailed and justified in [17], Algorithm 1 converges to a stationary point of the objective function, i.e.,* $\lim_{t\to\infty} \nabla C_t(\boldsymbol{U}_t) = \boldsymbol{O}_{r\times m}$.

Roughly, this theorem states that asymptotically the estimated subspace of the approximate cost function, which overestimates the original one, will converge to stationary point of the latter. As our experiments demonstrate this provides a good approximation.

4. EXPERIMENTS

4.1. SYNTHETIC NETFLIX DATASET

4.1.1. Generating data and noise

We generate a synthetic Netflix dataset, similarly as in [15]; the matrix that springs from this dataset obeys both low–rank and graph structure properties. Rows represent users and the columns represent movies; the corresponding entries denote the rating. We consider a number $m_c = 10$ of communities, forming a partition for the rows. The underlying graph is constructed as follows: two individuals are adjacent in the graph if and only if they belong to the same community. Similarly we assume that we have $n_c = 20$ communities for the columns. The data matrix is then constructed by assigning a random value from $\{1, \ldots, 5\}$ to each couple (movies community, users community). It can be readily seen that this ideal matrix is of rank $r = \min(m_c, n_c)$. However, to make the experimental setup more realistic we add noise and we also permute all the columns.

Noise is generated as follows. Assuming that an user is likely to have a different opinion on a movie than the rest of his community, we define $N_{prob} \in [0, 1]$ the probability of a rating to be affected by the noise, and $N_{level} \in \{1 \ldots 5\}$ the maximum level of noise. Then, for each entry X_{ij} of the data matrix, we pick the parameter a according to a Bernoulli $\mathcal{B}(1, N_{prob})$ distribution and the parameter b according to the uniform $\mathcal{U}(\{-N_{level}, -N_{level}+1, \ldots, N_{level}-1, N_{level}\})$ distribution. The entry of the corresponding corrupted matrix is then defined as:

$$\tilde{X}_{ij} = \max(\min(X_{ij} + ab, 5), 1)$$

One can easily verify that this definition preserves the fact that the occurring noisy entry will take a value, which belongs to the $\{1, \ldots, 5\}$ set.

4.1.2. Error measurement

We run the online algorithm and compute for each time step the euclidean distance between the predicted vector, $\hat{\boldsymbol{x}}_i = \boldsymbol{U}_i\boldsymbol{r}_i$, and the true one, divided by the norm of the latter. Afterwards, we compute the mean over time and the resulting metric is given by: $err(t) = 20\log_{10}\left(\frac{1}{t}\sum_{i=1}^t \frac{\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2}{\|\boldsymbol{x}_i\|_2}\right)$.

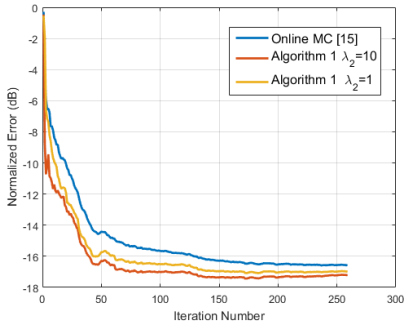

Fig. 1. Errors for Netflix dataset

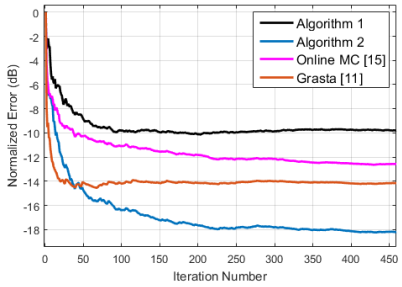

Fig. 2. Comparison of the standard and robust methods in the continuous values experiment.

4.1.3. Results

In the following we study the realistic case of 20% missing entries in the observations. We assume the sampling of the entries is uniform, which may not be true in practice. However the case of non-uniform sampling goes beyond the scope of this paper.

We compare our algorithm with that presented in [7], which is suitable for robust online matrix completion, albeit no graph information is included. In both algorithms the regularization parameter related to the sparse outlier noise is set equal to zero, since in this experiment we do not assume outliers. The rest parameters are chosen via cross validation so that all the algorithms exhibit the best trade–off between convergence speed and steady state error floor. Contenting ourselves with a small level of noise ($N_{prob} = 0.3$ and $N_{level} = 1$), we obtain the results presented in Figure 1. It can be readily observed that the Laplacian regularization improves the performance, as expected. In fact, Algorithm 1 converges faster to a lower steady state error floor, compared to the algorithm in [7]. Moreover, setting $\lambda_2 = 10$ exhibits a slightly improved performance compared to the $\lambda_2 = 1$ case.

4.2. CONTINUOUS VALUES DATASET: THE ROBUST CASE

In the previous experiment, the entries of the data matrix are integers taking values between 1 and 5. Such experiments do not permit us to evaluate if the robust algorithm deals well with very large (outlier) values. Therefore, we turn our focus now on another dataset generated in a similar way as in the previous experiment, albeit the entries now are allowed to take continuous values. To that end, they are drawn from a zero–mean normal distribution with variance equal to 1. We add i.i.d. Gaussian noise, with standard deviation equal to $\sigma = 0.2$. Furthermore, we add an "outlier" sparse matrix, the non-zero entries of which have a high magnitude compared to the data matrix. The sparse matrix is generated randomly and 1% of its entries are non-zero. These non-zeros entries are constructed so that their magnitude is at least 10 times the maximum value of the data matrix. Doing so, we have significant outliers. We compare the following algorithms: Algorithm 1 with $\lambda_3 = \infty$[3], labelled "Algorithm 1"; the same with $0 < \lambda_3 < \infty$, labelled "Algorithm 2"; a grassmannian manifold based algorithm suitable for online robust matrix completion, [9]; and the algorithm of [7]. Again, the parameters are chosen via cross validation. Figure 2 presents the evolution of the error at each time step. It can be readily seen that, Algorithm 1 converges to a high error floor, since the presence of outliers is not taken into account. Furthermore, the proposed algorithm outperforms the other robust based schemes, since it exploits the underlying graph structure.

[3]This effectively reduces the algorithm to one which takes no account of sparisty, since it will set the vector $\boldsymbol{s}$ to zero

4.3. REAL NETWORK DATA

Let us now evaluate our proposed algorithm using data collected from a real network. In particular, we use the dataset captured in 2006, [20] on GEANT, the high bandwidth pan-European research and education backbone. The network comprises 22 nodes and 36 links. We consider that at each time step, the load from a subset of the links becomes available to us, whereas the load for the rest of them is unknown. Our goal is to estimate the load for these links. To that direction, we employ Algorithm 1 with different values of the Laplacian related regularization parameter λ_2, as well as the online matrix completion algorithm of [7]. In all the algorithms, we fix λ_1 to be equal to 0.1, since this particular choice leads to a fast convergence speed and a low steady state error floor at the same time. The results are shown in Fig. 3. The online matrix completion algorithm is able to provide a good estimate of the missing entries due to the low–rank property of the link load traffic matrix. Indeed, the network traffic pattern is highly correlated both temporally and spatially (i.e., across different links). This amounts to claiming that the data exhibit a low rank structure. Nevertheless, the results can be enhanced significantly if we exploit the network graph topology, via the Laplacian smoothing.

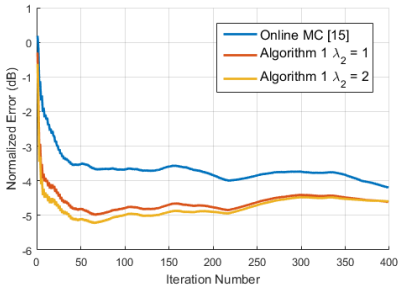

Fig. 3. Comparison of the proposed algorithm using the GEANT database

5. REFERENCES

[1] Robert M Bell and Yehuda Koren, "Lessons from the netflix prize challenge," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 75–79, 2007.

[2] James Bennett and Stan Lanning, "The netflix prize," in *Proceedings of KDD cup and workshop*, 2007, vol. 2007, p. 35.

[3] Emmanuel J Candès and Benjamin Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[4] Emmanuel J Candès and Terence Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[5] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 11, 2011.

[6] Bin Yang, "Projection approximation subspace tracking," *IEEE Transactions on Signal Processing*, vol. 43, no. 1, pp. 95–107, Jan 1995.

[7] Morteza Mardani, Gonzalo Mateos, and Georgios Giannakis, "Dynamic anomalography: Tracking network anomalies via sparsity and low rank," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 50–66, 2013.

[8] Jiashi Feng, Huan Xu, and Shuicheng Yan, "Online robust PCA via stochastic optimization," in *Advances in Neural Information Processing Systems*, 2013, pp. 404–412.

[9] Jun He, Laura Balzano, and John Lui, "Online robust subspace tracking from partial information," *arXiv preprint arXiv:1109.3827*, 2011.

[10] Laura Balzano, Robert Nowak, and Benjamin Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE, 2010, pp. 704–711.

[11] Clemens Hage and Martin Kleinsteuber, "Robust pca and subspace tracking from incomplete observations using\ ell _0-surrogates," *Computational Statistics*, vol. 29, no. 3-4, pp. 467–487, 2014.

[12] Yuejie Chi, Yonina C Eldar, and Robert Calderbank, "Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 5947–5959, 2013.

[13] Han Guo, Chenlu Qiu, and Namrata Vaswani, "An online algorithm for separating sparse and low-dimensional signal sequences from their sum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4284–4297, 2014.

[14] Symeon Chouvardas, Yannis Kopsinis, and Sergios Theodoridis, "Robust subspace tracking with missing entries: The set-theoretic approach," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5060–5070, 2015.

[15] Vassilis Kalofolias, Xavier Bresson, Michael Bronstein, and Pierre Vandergheynst, "Matrix completion on graphs," *preprint, arXiv:1408.1717*, 2014.

[16] Nauman Shahid, Nathanael Perraudin, Vassilis Kalofolias, and Pierre Vandergheynst, "Fast robust PCA on graphs," *preprint, arXiv:1507.08173*, 2015.

[17] Symeon Chouvardas, Mohammed Amin Abdullah, Lucas Claude, and Moez Draeif, "Robust online matrix completion on graphs," *preprint, arXiv:1605.04192*, 2016.

[18] Sergios Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press, 2015.

[19] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[20] Steve Uhlig, Bruno Quoitin, Jean Lepropre, and Simon Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Rev*, vol. 36, no. 1, 2006.