23rd International Symposium on Mathematical Theory of Networks and Systems
Hong Kong University of Science and Technology, Hong Kong, July 16-20, 2018

Graph learning for regularized low-rank matrix completion

Shuyu Dong[1], P.-A. Absil[2] and K. A. Gallivan[3]

Abstract— **Low rank matrix completion is the problem of recovering the missing entries of a large data matrix by using the low-rankness assumption. Much attention has been put recently to exploiting correlations between the column/row entities, through side information or data adaptive models, to improve the matrix completion quality. In this paper, we propose a novel *graph learning* algorithm and apply it to the learning of a graph adjacency matrix from a given, incomplete data matrix, in a way such that the weighted graph edges encode pairwise similarities between the rows/columns of the data matrix. Subsequently we present a graph-regularized low-rank matrix completion method. Experiments on synthetic and real datasets show that this regularized matrix completion approach achieves significant improvement for the matrix completion task.**

I. Introduction

Low rank matrix completion, often formulated as a rank minimization or a low-rank matrix factorization problem, arises in applications such as recommender systems [1]–[3]. Recent works (*e.g.* [4]) also considered low-rank methods for forecasting and imputation of high-dimensional time series such as traffic occupancy and electricity consumption data. For some applications, in addition to the low-rank assumption, regularization methods exploiting other properties of the data are needed since data often come with structure. Graphs are often useful for modeling the structure of discrete input spaces and, in particular, for capturing the correlations or pairwise similarities between the columns and/or rows of a matrix. Regularization approaches based on graph information (*e.g.* [3]–[5]) consist of introducing a graph-based Dirichlet energy function

$$\|\boldsymbol{X}\|_{\boldsymbol{L}}^2 \triangleq \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) \tag{1}$$

or $\|\boldsymbol{X}^T\|_{\boldsymbol{L}'}^2 = \mathrm{Tr}(\boldsymbol{X}\boldsymbol{L}'\boldsymbol{X}^T)$, where $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is the sought low-rank complete matrix and $\boldsymbol{L} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{L}' \in \mathbb{R}^{n\times n}$ are given graph Laplacian matrices that carry the graph information of the discrete space of row, respectively column, indices of $\boldsymbol{X}$. This energy function (1) is involved in several existing and closely related methods for matrix completion. Kalofolias et al. [5] proposed the following rank minimization problem using the matrix nuclear norm ($\|\cdot\|_*$)

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \gamma\|\boldsymbol{X}\|_* + \|\mathcal{P}_\Omega(\boldsymbol{X}-\boldsymbol{M})\|_F^2 + \frac{\gamma_r}{2}\|\boldsymbol{X}\|_{\boldsymbol{L}}^2 + \frac{\gamma_c}{2}\|\boldsymbol{X}^T\|_{\boldsymbol{L}'}^2, \tag{2}$$

where $\boldsymbol{M}$ is the incomplete data matrix, Ω is the index subset of the observed entries and P_Ω is the mask projector on observed entries. In [5], the Laplacian matrices $\boldsymbol{L}$ and $\boldsymbol{L}'$ (for the columns and rows respectively) are constructed by an ϵ-neighborhood graph model, for which pairwise distances between rows/columns are computed based on the known entries of the matrix, $P_\Omega(\boldsymbol{M})$. On the other hand, Rao et al. [3] considered adding $\|\boldsymbol{U}\|_{\boldsymbol{L}}^2$ and $\|\boldsymbol{H}^T\|_{\boldsymbol{L}'}^2$ to a (low-rank) matrix factorization problem with $\boldsymbol{U}$ and $\boldsymbol{H}$ such that $P_\Omega(\boldsymbol{U}\boldsymbol{H}^T) \approx P_\Omega(\boldsymbol{M})$. In [3], the construction of $\boldsymbol{L}$ and $\boldsymbol{L}'$ is essentially based on *side information* which, for datasets of user ratings (MovieLens and other social network ratings) is either directly available from the connectivity between the users of the social network or can be obtained by collecting user features such as age (numeric), gender (binary) and occupation. More precisely, in the latter case, two column/row indices are connected if their associated user features are sufficiently close. Within these approaches, the graph Laplacian matrix is involved in the following way: by taking $\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$ as a regularization term, we assume that the m row indices of $\boldsymbol{X}$ are modeled by the nodes of a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ and that the correlation or similarity between any pair (i, j) of row vectors of $\boldsymbol{X}$ is encoded by the edge ($W_{ij} \geq 0$) connecting them. As we will show in Section IV, given the graph Laplacian $\big(\boldsymbol{L} = \mathrm{Diag}(\boldsymbol{W}\boldsymbol{1}) - \boldsymbol{W}\big)$ associated with $\mathcal{G}$, minimizing $\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$ has the effect of promoting solutions such that their columns are *smooth* on the graph $\mathcal{G}$. In other words, the *distance* separating $(X_{it})_{t=1,..,n}$ and $(X_{jt})_{t=1,..,n}$ is expected to be small if the pair nodes (i, j) is strongly connected. Intuitively, in applications with the aforementioned users-ratings data as well as traffic occupancy data (Section VI-B), where the matrix to be recovered has a certain pattern of correlations between its columns/rows, graph-regularized low-rank solutions are expected to be more favorable than other non-regularized low-rank solutions of the same rank.

The question is how we can obtain an appropriate graph Laplacian from data. This question arises as an important problem itself. Specifically, we focus on how to obtain a useful graph Laplacian from an incomplete data matrix.

In this paper, we consider recovering a partially observed matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ by solving a problem of the following form, for $r \ll \min(m, n)$:

$$\underset{\boldsymbol{X}\in\mathcal{M}_r}{\mathrm{minimize}}\ \ell_\Omega(\boldsymbol{X}; \boldsymbol{M}) + \beta\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}), \tag{3}$$

*This work was supported by the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160, and by "Communauté française de Belgique - Actions de Recherche Concertées" (contract ARC 14/19-060).

[1]Shuyu Dong and [2]P.-A. Absil are with the Department of Mathematical Engineering, ICTEAM, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium {shuyu.dong,pa.absil}@uclouvain.be

[3]K. A. Gallivan is with the Department of Mathematics, 208 Love Building, 1017 Academic Way, Florida State University, Tallahassee FL 32306-4510, USA kgallivan@fsu.edu

where $\mathcal{M}_r = \{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \text{rank}(\boldsymbol{X}) = r\}$ is the set of $m \times n$ real-valued matrices of a fixed rank, $\Omega \subset [\![m]\!] \times [\![n]\!]$ is the index set of the observed entries, $\ell_\Omega(\boldsymbol{X}; \boldsymbol{M})$ is a data fidelity term restricted to Ω and $\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$ is the additional regularizer for $\boldsymbol{X}$ depending on a graph Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{m\times m}$, which needs to be inferred. As our main purpose is to *learn* an appropriate graph Laplacian matrix from the incomplete data and to investigate the subsequent graph-based regularization, the low-rank assumption can certainly be imposed in ways other than using $\mathcal{M}_r$.

We tackle the question of finding $\boldsymbol{L}$ for (3) by solving a *graph learning* problem in the context of matrix completion. We refer to graph learning [6]–[8] as the idea of learning a graph Laplacian matrix from data samples, which consists of minimizing $\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$ as a function of $\boldsymbol{L}$ for a given data matrix $\boldsymbol{X}$ and certain constraints on the Laplacian matrix variable. One obvious difficulty for graph learning in the context of matrix completion is the lack of fully accessible data, since the missing values of $\boldsymbol{M}$ themselves need to be recovered. To address this difficulty, we propose to learn a graph Laplacian matrix in an *off-line* manner (without alternating with the matrix completion iterations) by solving the following type of problems

$$\underset{\boldsymbol{L}\in\mathcal{L}}{\text{minimize}}\ \text{Tr}(\boldsymbol{\gamma_M}\boldsymbol{L}) + \mathcal{R}(\boldsymbol{L}), \tag{4}$$

where $\mathcal{L}$ is the set (7) of all graph Laplacian matrices, $\boldsymbol{M}$ is the incomplete matrix, $\mathcal{R}(\boldsymbol{L})$ is a regularization term to be specified in Section III-B and $\boldsymbol{\gamma_M} \in \mathbb{R}^{m\times m}$ is a *Gram* matrix that must be computed (see Sections IV,VI-B) depending on the choice of a kernel function.

Our main contributions are as follows. First, since $\boldsymbol{M}$ is only partially observed, building a Gram matrix $\boldsymbol{\gamma_M}$ from $\boldsymbol{M}$ is a nontrivial step to solving problem (4). In Section IV, we propose a computationally efficient method that constructs γ_M from a low-rank approximation to $\boldsymbol{M}$. This low-rank approximation does not require any extra computation as it corresponds to the initialization step of our proposed algorithm for matrix completion (3). In our problem setting (4), the trace function $\text{Tr}(\boldsymbol{\gamma_M}\boldsymbol{L})$ is seen as an extension of $\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$ in the sense that the covariance term $\boldsymbol{X}\boldsymbol{X}^T$ in $\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) = \text{Tr}(\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{L})$ is replaced by the more general Gram matrix $\boldsymbol{\gamma_M}$. We will show that this extension is essential for the successful application to real data in our experiments (see Section VI-B). Second, in contrast to previous formulations for graph learning, we present a *nonconvex* formulation for problem (4), whose non-convexity comes from a spherical constraint that preserves the scale of the graph edge weights. Then we propose a simple, efficient algorithm which shows comparable performance (see Section VI-A) for graph learning compared to the state-of-the-art [7].

By solving problem (4) as a preliminary step for problem (3), our approach differs with [5] in the sense that the Laplacian matrix is *not* directly constructed from pairwise distances using any specific graph model. Moreover, unlike in [3], our approach does *not* use any side information, since we are interested in applications where no side information is available, as it is the case for the real-data application we present (see Section VI-B) in this paper.

II. Notation

For an integer $m > 0$, the index set $\{1, .., m\}$ is denoted by $[\![m]\!]$. The matrix entries and column vectors of $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ are denoted by X_{ij} and $\boldsymbol{X}_{(j)}$ respectively. The *row* vectors of $\boldsymbol{X}$ are denoted by $\boldsymbol{X}_{(i)}^T$ or $(\boldsymbol{X}_{it})_{t=1,..,n}$. The diagonal matrix and the vector of diagonal entries of $\boldsymbol{X}$ are denoted by $\text{Diag}(\boldsymbol{X})$ and $\text{diag}(\boldsymbol{X})$ respectively. The unit vector in the i-th direction and the vector of all-ones and all-zeros are denoted by $\boldsymbol{e}_i, \boldsymbol{1}$ and $\boldsymbol{0}$ respectively. The Euclidean inner product of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ is denoted and defined by $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \boldsymbol{u}^T\boldsymbol{v}$. The Frobenius matrix norm of a matrix $\boldsymbol{X}$ is denoted and defined as $\|\boldsymbol{X}\|_F = \sqrt{\text{Tr}(\boldsymbol{X}^T\boldsymbol{X})}$. Throughout the paper, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ or simply $\mathcal{G}$ denotes an undirected graph with nonnegative edge weights. $\mathcal{V}$ and $\mathcal{E}$ denote the set of graph nodes and edges respectively and $\boldsymbol{W}$ is an associated weighted adjacency matrix. $\boldsymbol{W} \geq 0$ means element-wise non-negativity. We denote by $\mathbb{S}^{q-1}$ and $\mathbb{S}_+^{q-1}$ the unit sphere in $\mathbb{R}^q$ and its intersection with the nonnegative orthant of $\mathbb{R}^q$ respectively.

III. Graph learning: Preliminaries

A. The graph Laplacian matrix and Dirichlet semi-norm

For any undirected and positively weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with m nodes, the weighted graph adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{m\times m}$ satisfies

$$\boldsymbol{W} = \boldsymbol{W}^T, \boldsymbol{W} \geq 0 \text{ and } \text{diag}(\boldsymbol{W}) = \boldsymbol{0}, \tag{5}$$

where the zero diagonal entries of $\boldsymbol{W}$ signify the absence of self-loops in $\mathcal{G}$. We consider the combinatorial graph Laplacian matrix defined as

$$\boldsymbol{L} \triangleq \text{Diag}(\boldsymbol{W}\boldsymbol{1}) - \boldsymbol{W}. \tag{6}$$

Denote by $\mathcal{L}$ the set of all such graph Laplacian matrices. From (5-6), we have

$$\mathcal{L} \triangleq \{\boldsymbol{L} \in \mathbb{R}^{m\times m} : (\forall i \neq j)\ L_{ij} = L_{ji} \leq 0, \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}\}. \tag{7}$$

As all constraints on $\boldsymbol{L}$ are linear, $\mathcal{L}$ is a closed, convex subset of $\mathbb{R}^{m\times m}$. $\boldsymbol{L}$ is positive semi-definite because for any vector $\boldsymbol{x} \in \mathbb{R}^m$,

$$\mathcal{S}_{\boldsymbol{L}}(\boldsymbol{x}) \triangleq \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} W_{ij}(x_i - x_j)^2 \geq 0. \tag{8}$$

For this reason, the graph Laplacian matrix induces a Dirichlet semi-norm (*e.g.* [9]) for m-dimensional real vectors, defined as $\|\boldsymbol{x}\|_{\boldsymbol{L}} = \sqrt{\mathcal{S}_{\boldsymbol{L}}(\boldsymbol{x})}$.

Intuitively, this definition says that for a given graph structure $\mathcal{G}$ endowed with $\boldsymbol{L}$, $\|\boldsymbol{x}\|_{\boldsymbol{L}}$ tends to be small if $\boldsymbol{x}$ is *smooth* on $\mathcal{G}$, in the sense that the function values of $\boldsymbol{x}$ over the neighborhood of any node evolve slowly/smoothly. The Dirichlet energy function (1) is related to this semi-norm through

$$\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) = \sum_{t=1}^{n} \|\boldsymbol{X}_{(t)}\|_{\boldsymbol{L}}^2.$$

Thus the usage of this energy function in [3]–[5] is basically motivated by the idea of considering the matrix columns/rows as real functions defined on the column/row index set $\mathcal{V}$, with a properly defined graph structure $\boldsymbol{W}$ such that (1) acts as a "smoothness promoting" regularization that captures the structural information of the discrete domain of column/row indices, which is otherwise lost in the Frobenius matrix norm.

B. Graph learning: related work

The problem of learning a graph Laplacian matrix from data samples, introduced by [6], is formulated as follows

$$\underset{\boldsymbol{L}\in\mathcal{L}:\mathrm{Tr}(\boldsymbol{L})=\theta}{\text{minimize}} \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) + \beta\|\boldsymbol{L}\|_F^2, \tag{9}$$

where $\boldsymbol{X}$ is a given data matrix, the feasible set $\mathcal{L}$ is given by (7) and the parameter $\theta > 0$ maintains the trace of $\boldsymbol{L}$ at a fixed level. More precisely, [6] proposed to learn $\boldsymbol{X}$ and $\boldsymbol{L}$ in an alternating way by solving

$$\min_{\boldsymbol{X},\boldsymbol{L}\in\mathcal{L}:\mathrm{Tr}(\boldsymbol{L})=\theta} \|\boldsymbol{Y}-\boldsymbol{X}\|_F^2 + \alpha\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) + \beta'\|\boldsymbol{L}\|_F^2,$$

where $\boldsymbol{Y}$ is the matrix containing noisy data samples and columns of $\boldsymbol{X}$ are modeled as i.i.d. samples of the following sparse Gaussian Markov random fields (GMRF, *e.g.* [10])

$$\mathcal{X} \sim \mathcal{N}(0, \bar{\boldsymbol{L}}^\dagger), \tag{10}$$

where $\bar{\boldsymbol{L}}^\dagger$ is the Moore-Penrose pseudoinverse of the hidden matrix $\bar{\boldsymbol{L}}$. With the sparsity constraint (via $\mathrm{Tr}(\boldsymbol{L}) = \theta$ and the regularizer $\beta'\|\boldsymbol{L}\|_F^2$) on $\boldsymbol{L}$, problem (9) can be seen as a special instance of the sparse inverse covariance estimation problem (*e.g.* [11], [12]).

More recently, [7] proposed to reformulate problem (9) into optimization problems with the weighted adjacency matrix $\boldsymbol{W}$ by using the following property:

Property 1 ([7]). *Given $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ of m nodes and a data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$. Let $\boldsymbol{Z}$ be the Euclidean pairwise distance matrix of $\boldsymbol{X}$'s rows,* i.e., *$Z_{ij} \triangleq \|\boldsymbol{X}_{(i)}^T - \boldsymbol{X}_{(j)}^T\|_2^2$. Then $\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) = \frac{1}{2}\mathrm{Tr}(\boldsymbol{Z}\boldsymbol{W})$, where $\boldsymbol{L}$ is the graph Laplacian matrix associated with $\mathcal{G}$.*

As a result, problems for graph Laplacian learning like (9) can be transformed into

$$\underset{\substack{\boldsymbol{W}=\boldsymbol{W}^T,\boldsymbol{W}\geq 0\\ \mathrm{diag}(\boldsymbol{W})=\boldsymbol{0}}}{\text{minimize}} \frac{1}{2}\mathrm{Tr}(\boldsymbol{Z}\boldsymbol{W}) + \beta\left(\|\boldsymbol{W}\boldsymbol{1}\|_2^2 + \|\boldsymbol{W}\|_F^2\right), \tag{11}$$

which has a more convenient feasible set than the set (7) of graph Laplacian matrices. Moreover, as explained in [7], $\mathrm{Tr}(\boldsymbol{Z}\boldsymbol{W})$ is already a sparsity promoting function because of the positiveness of (Z_{ij}) and (W_{ij}), thus the equality constraint $\mathrm{Tr}(\boldsymbol{L}) = \sum_{i,j}|W_{ij}| = \theta$ need not be included. Alternatively, [7] also considered replacing the ℓ_2 norm-based regularizer $\|\boldsymbol{W}\boldsymbol{1}\|_2^2$ (acting effectively on vertex degrees $d_i = \sum_{j=1}^m W_{ij}$) by the log barrier function $-\sum_{i=1}^m \log(d_i)$ to have

$$\underset{\substack{\boldsymbol{W}=\boldsymbol{W}^T,\boldsymbol{W}\geq 0\\ \mathrm{diag}(\boldsymbol{W})=\boldsymbol{0}}}{\text{minimize}} \frac{1}{2}\mathrm{Tr}(\boldsymbol{Z}\boldsymbol{W}) - \gamma\boldsymbol{1}^T\log(\boldsymbol{W}\boldsymbol{1}) + \beta\|\boldsymbol{W}\|_F^2. \tag{12}$$

A good property of this formulation is that solutions to this problem of any sparsity level (determined by the parameters $\gamma, \beta \geq 0$) are always connected graphs because of the log barrier function that prevents any vertex from being isolated with degree zero.

IV. Graph learning from incomplete data

In this section, we consider learning a graph Laplacian matrix $\boldsymbol{L}$ from data samples with missing entries, which is a preliminary step for the regularized matrix completion problem (3).

As shown in Section (III-B), learning a graph Laplacian $\boldsymbol{L}$ from (fully accessible) data samples consists in minimizing the Dirichlet energy $\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$ as a function of $\boldsymbol{L}$ such that the columns of $\boldsymbol{X}$ are smooth real functions on the graph associated with $\boldsymbol{L}$. However, in the context of problem (3), we only have an incomplete data matrix $\boldsymbol{M}$ which has a large proportion of missing values at entries in Ω^c. To address this difficulty, we make use of an inexpensive low-rank approximation to $\boldsymbol{M}$ by defining $\widehat{\boldsymbol{M}} \triangleq \boldsymbol{U}_0\boldsymbol{S}_0\boldsymbol{V}_0^T$, where $\{\boldsymbol{U_0}, \boldsymbol{S_0}, \boldsymbol{V_0}\}$ are rank-r matrix factors of the *zero-filled matrix* of $\boldsymbol{M}$. More precisely, $\{\boldsymbol{U_0}, \boldsymbol{S_0}, \boldsymbol{V_0}\} = r\text{-SVD}(\boldsymbol{M}_0)$, where

$$\boldsymbol{M}_0 = \begin{cases} M_{ij} & \text{if } (i,j)\in\Omega \\ 0 & \text{otherwise.}\end{cases} \tag{13}$$

We will see (Section V) that $\widehat{\boldsymbol{M}}$ comes without any extra computation since this r-SVD estimate corresponds to the initialization step of our matrix completion algorithm, as it is for many other low-rank matrix completion methods. Moreover, to remedy the loss of information in $\widehat{\boldsymbol{M}}$ (of rank $r \ll \min(m,n)$), we propose to extend the definition of the objective function $\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) = \mathrm{Tr}(\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{L})$ by replacing the sample covariance term $\boldsymbol{X}\boldsymbol{X}^T$ with a more general *Gram* matrix, that is, a generalized kernel matrix of the row *features* of $\boldsymbol{X}$.

More precisely, the data at our disposal consists of $\widehat{\boldsymbol{M}}$'s row vectors $\widehat{\boldsymbol{M}}_{(1)}^T, .., \widehat{\boldsymbol{M}}_{(m)}^T \in \mathcal{X} \subset \mathbb{R}^n$. Since the Euclidean inner product is not necessarily adapted for the geometry underlying our data samples distributed in $\mathcal{X}$, we adopt the idea of feature mappings in kernel-based methods by considering a feature space $\mathcal{F}$ endowed with the Euclidean inner product and a feature map $\Phi : \mathcal{X} \to \mathcal{F}$, and then carrying out graph learning in $\mathcal{F}$. The feature mapping with Φ is realized implicitly, as in most kernel-based methods, by defining a generalized Gram matrix $\boldsymbol{\gamma} \in \mathbb{R}^{m\times m}$ such that

$$[\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}]_{ij} = k\big(\widehat{\boldsymbol{M}}_{(i)}^T, \widehat{\boldsymbol{M}}_{(j)}^T\big) = \langle\Phi(\widehat{\boldsymbol{M}}_{(i)}^T), \Phi(\widehat{\boldsymbol{M}}_{(j)}^T)\rangle, \tag{14}$$

where $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is a given (positive semi-definite) kernel function. In particular, the sample covariance estimate $\widehat{\boldsymbol{M}}\widehat{\boldsymbol{M}}^T$ corresponds to the trivial case where the kernel function reduces to the Euclidean inner product $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$.

With the generalization (14), we have the following *kernelized* formulation of problem (9)

$$\underset{\boldsymbol{L}\in\mathcal{L}:\mathrm{Tr}(\boldsymbol{L})=\theta}{\text{minimize}} \mathrm{Tr}(\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}\boldsymbol{L}) + \beta\|\boldsymbol{L}\|_F^2. \tag{15}$$

In a similar way as done for Property 1, we show that

Proposition 1. *Given a undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ of m nodes and a (positive semi-definite) Gram matrix $\boldsymbol{\gamma} \in \mathbb{R}^{m\times m}$. Let $\boldsymbol{\zeta} \in \mathbb{R}^{m\times m}$ be defined by*

$$\boldsymbol{\zeta} = \mathit{diag}(\boldsymbol{\gamma})\mathbf{1}^T + \mathbf{1}\mathit{diag}(\boldsymbol{\gamma})^T - 2\boldsymbol{\gamma}. \tag{16}$$

Then we have $\mathit{Tr}(\boldsymbol{\gamma}\boldsymbol{L}) = \frac{1}{2}\mathit{Tr}(\boldsymbol{\zeta}\boldsymbol{W})$, where $\boldsymbol{L}$ is the graph Laplacian associated with $\mathcal{G}$.

Proof. see Appendix VII-B. $\square$

This enables us to transform (15) into an optimization problem with the graph adjacency matrix $\boldsymbol{W}$:

$$\underset{\substack{\boldsymbol{W}=\boldsymbol{W}^T,\boldsymbol{W}\geq 0\\ \mathrm{diag}(\boldsymbol{W})=\mathbf{0}}}{\text{minimize}} \frac{1}{2}\mathrm{Tr}(\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}\boldsymbol{W}) + \beta\big(\|\boldsymbol{W}\mathbf{1}\|_2^2 + \|\boldsymbol{W}\|_F^2\big),$$

where $\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}} \in \mathbb{R}^{m\times m}$ is obtained from the constructed Gram matrix $\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}$ via (16). Similarly, the kernelized formulation of (12) is

$$\underset{\substack{\boldsymbol{W}=\boldsymbol{W}^T,\boldsymbol{W}\geq 0\\ \mathrm{diag}(\boldsymbol{W})=\mathbf{0}}}{\text{minimize}} \frac{1}{2}\mathrm{Tr}(\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}\boldsymbol{W}) - \gamma\mathbf{1}^T\log(\boldsymbol{W}\mathbf{1}) + \beta\|\boldsymbol{W}\|_F^2. \tag{17}$$

These problems can effectively be solved by using the primal-dual algorithm of [7]. Note that choosing a kernel function for computing $\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}$ and then $\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}$ is a separate topic from solving the problem. In the rest of this section, we treat $\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}$ (or $\boldsymbol{\zeta}$ in Section IV-A) as input data and focus on solving the $\boldsymbol{W}$-related graph learning problems in a *fixed-scale* manner.

A. Fixed-scale graph learning

In the formulation (17), the scale of $\boldsymbol{W}$ depends on the input data and the parameters. Instead, we propose to control it directly by a fixed-scale approach where the graph learning objective function is minimized over a spherical space $\{\boldsymbol{W} : \|\boldsymbol{W}\|_F = \theta\}$ instead of the nonnegative orthant. This drives us to consider in particular the following problem

$$\begin{aligned}&\underset{\boldsymbol{W}=\boldsymbol{W}^T}{\text{minimize}}\ \frac{1}{2}\mathrm{Tr}(\boldsymbol{\zeta}\boldsymbol{W}) - \gamma\mathbf{1}^T\log(\boldsymbol{W}\mathbf{1})\\ &\text{s.t. } \boldsymbol{W}\geq 0, \mathrm{diag}(\boldsymbol{W}) = \mathbf{0}, \|\boldsymbol{W}\|_F = \theta.\end{aligned} \tag{17b}$$

Next, we show that our main problem (17b) is equivalent to a constrained minimization problem on a hypersphere via *half-vectorization*: the feasible set of (17b) has the structure of the nonnegative orthant of a sphere, *i.e.*, the intersection of the unit sphere $\mathbb{S}^{q-1}$ with the nonnegative orthant of the vector space:

$$\mathbb{S}_+^{q-1} \triangleq \Big\{\boldsymbol{w}\in\mathbb{R}^q : \|\boldsymbol{w}\|_2 = 1, \boldsymbol{w}\geq\mathbf{0}\Big\}, \tag{18}$$

where $q = m(m-1)/2$. Indeed, any graph adjacency matrix satisfying (5) can be written

$$\boldsymbol{W} = \sum_{i,j\in[\![m]\!],i>j} W_{ij}\boldsymbol{E}_{(i,j)}, \tag{19}$$

where $W_{ij}\geq 0$ and $\boldsymbol{E}_{(i,j)} = \boldsymbol{e}_i\boldsymbol{e}_j^T + \boldsymbol{e}_j\boldsymbol{e}_i^T$. In other words, $(W_{ij})_{i,j\in[\![m]\!],i>j}$ constitute the nonnegative coordinates of $\boldsymbol{W}$ in the linear subspace spanned by $\big\{\boldsymbol{E}_{(i,j)} : i,j\in[\![m]\!], i>j\big\}$, which has dimension $q = m(m-1)/2$. Naturally, there exists a bijective *half-vectorization* map l (the same technique can be found in [6], [7]) that vectorizes the strict lower triangular part of $\boldsymbol{W}$ such that $w_{l(i,j)} = W_{ij}(= W_{ji})$. Hence l induces a bijection between the feasible set of (17b) and

$$\left\{\boldsymbol{w}\in\mathbb{R}^q : \|\boldsymbol{w}\|_2 = \frac{\theta}{\sqrt{2}}, \boldsymbol{w}\geq\mathbf{0}\right\}.$$

Therefore, we can identify the feasible set of (17b) with $\mathbb{S}_+^{q-1}$ (18) by setting the scale parameter θ to $\sqrt{2}$, without loss of generality. The underlying half-vectorized formulation is

$$\underset{\boldsymbol{w}\in\mathbb{S}_+^{q-1}}{\text{minimize}}\ f_\gamma(\boldsymbol{w}) \triangleq \langle\vec{\boldsymbol{\zeta}}, \boldsymbol{w}\rangle - \gamma\mathbf{1}^T\log(\mathcal{A}\boldsymbol{w}), \tag{17c}$$

where $\vec{\boldsymbol{\zeta}}\in\mathbb{R}^q$ denotes the half-vectorization of $\boldsymbol{\zeta}\in\mathbb{R}^{m\times m}$ and $\mathcal{A}$ denotes the linear application that maps the graph adjacency coordinates to the graph's vertex degree vector $\mathcal{A}:\mathbb{R}^q\to\mathbb{R}^m : \boldsymbol{w}\to\boldsymbol{d} = \boldsymbol{W}\mathbf{1}$.

B. Constrained optimization on the sphere

The feasible set $\mathbb{S}_+^{q-1}$ is a closed subset of the sphere, hence a closed and nonconvex set of $\mathbb{R}^q$. We propose to solve problem (17c) with a projected gradient algorithm. Define the Euclidean projection on $\mathbb{S}_+^{q-1}$ as the following operator

$$P_{\mathbb{S}_+^{q-1}}(\boldsymbol{x}) = \underset{\boldsymbol{w}\in\mathbb{S}_+^{q-1}}{\mathrm{argmin}}\|\boldsymbol{w}-\boldsymbol{x}\|_2, \forall\boldsymbol{x}\in\mathbb{R}^q. \tag{20}$$

The following result provides a closed-form solution for (20):

Proposition 2. *For all $\boldsymbol{x}\in\mathbb{R}^q$:*

$$P_{\mathbb{S}_+^{q-1}}(\boldsymbol{x}) = \begin{cases}\boldsymbol{x}_+/\|\boldsymbol{x}_+\|_2 & \text{if } \boldsymbol{x}\notin\mathbb{R}_-^q\\ \boldsymbol{e}_i,\ \text{with } i = \mathrm{argmax}\{x_i\} & \text{otherwise,}\end{cases} \tag{21}$$

where $\boldsymbol{x}_+ = (\max\{x_i, 0\})_{i=1,..,q}$ and $\mathbb{R}_-^q = \{\boldsymbol{x} : x_i\leq 0, \forall i\in[\![q]\!]\}$.

Proof. We follow the same proof as for (Lemma 1 of [13]) which solves $\min_{w\in\mathbb{S}_+^{q-1}}\langle\boldsymbol{w},\boldsymbol{b}\rangle$. It suffices to identify our problem as the case where $\boldsymbol{b} = -\boldsymbol{x}$. $\square$

While the Euclidean projection on convex sets is unique, this is not the case with the nonconvex set $\mathbb{S}_+^{q-1}$. Indeed, for $\boldsymbol{x}\in\mathbb{R}^q$, there can be several points $\boldsymbol{w}\in\mathbb{S}_+^{q-1}$ minimizing $\|\boldsymbol{w}-\boldsymbol{x}\|_2$: this happens in fact only if $\boldsymbol{x}\in\mathbb{R}_-^q$ and if $\boldsymbol{x}$ admits multiple maximal coefficients. In this last case, one solution is chosen arbitrarily. The closed-form expression (21) requires very lightweight computations thus provides a way to carrying out simple projected gradient-based iterations on $\mathbb{S}_+^{q-1}$. We propose to solve problem (17c) with the following algorithm:

Algorithm 1 (GL-SPH) Projected gradient over $\mathbb{S}_+^{q-1}$ with backtracking line search

```
Input: $\boldsymbol{\zeta}\in\mathbb{R}^{m\times m}; \gamma > 0; \sigma,\beta\in]0,1[, M, K\geq 1$.
Output: $\boldsymbol{w}\in\mathbb{S}_+^{q-1}$.
1: Initialization: $\boldsymbol{w} = \boldsymbol{w}^0$.
```

```
 2: for $k \in \{0, 1, .., K\}$ do
 3:   Stopping-criterion:
 4:   if $\|P_{\mathbb{S}_+^{q-1}}\big(\boldsymbol{w}^k - \nabla f_\gamma(\boldsymbol{w}^k)\big) - \boldsymbol{w}^k\|_2 \le \epsilon$ then
 5:     stop.
 6:   end if
 7:   Backtracking line search: $t \leftarrow 1$.
 8:   while True do
 9:     Set trial point: $\boldsymbol{w}_+ = P_{\mathbb{S}_+^{q-1}}\big(\boldsymbol{w}^k - t\nabla f_\gamma(\boldsymbol{w}^k)\big)$.
10:     if Line search criterion attained then
11:       break; (turn to line 15)
12:     end if
13:     Set $t \leftarrow \beta t$.
14:   end while
15:   PGD update: $\boldsymbol{w}^{k+1} = \boldsymbol{w}_+$.
16: end for
```

In line 10 of Algorithm 1, the line search criterion is

$$f_\gamma(\boldsymbol{w}_+) \le \max_{0\le j\le \min\{k,M-1\}} f_\gamma(\boldsymbol{w}^{k-j}) + \sigma\left\langle \nabla f_\gamma(\boldsymbol{w}^k), \boldsymbol{w}_+ - \boldsymbol{w}^k \right\rangle . \tag{22}$$

The backtracking line search (line 7-14) is simplified from nonmonotone gradient projection algorithms as proposed by [14]. The initialization step (line 1) corresponds to generating a random graph (w.r.t. an arbitrary graph type) and initializing $\boldsymbol{w}^0$ with the graph's edge weights. Similar to the algorithm of [7], Algorithm 1 has a per-iteration time complexity of $\mathcal{O}(m^2)$, where m is the size of the graph. Note that even without any extra structural priors (other than the sparsity of graph edges) for learning the graph, this basic algorithm has a much lower computational complexity than basic methods (*e.g.* [11], [12]) for sparse inverse covariance estimation, which normally amount to $\mathcal{O}(m^3)$.

Using proper choices for the parameter α (problem (17b)), empirical tests on synthetic data show that this algorithm converges to local minimum of the problem in a comparable amount of time as required by the state-of-the-art method [7]. Graphs learned by this algorithm are evaluated in Section VI-A.

V. Graph-regularized low-rank matrix completion

In this section, we illustrate the use of the learned graph Laplacian matrix $\boldsymbol{L}$ for the type of problems (3) as introduced in Section I. Specifically, considering the recent work of [15] for robust matrix completion (RMC) as a starting point, we target an instance of (3) with the following loss function

$$\|P_\Omega(\boldsymbol{X} - \boldsymbol{M})\|_1 \triangleq \sum_{(i,j)\in\Omega} |X_{ij} - M_{ij}|.$$

In fact, [15] proposed to solve the matrix completion problem by minimizing a smooth approximation to the ℓ_1-norm based loss (for a small δ):

$$F_\delta(\boldsymbol{X}) \triangleq \underbrace{\sum_{(i,j)\in\Omega} \sqrt{\delta^2 + (X_{ij} - M_{ij})^2}}_{\ell_{\boldsymbol{M},\Omega}(\boldsymbol{X})} + \alpha\|\boldsymbol{X}\|_{\Omega^c}^2$$
$$= \textstyle\sum_{(i,j)\in\Omega} \left(\sqrt{\delta^2 + (X_{ij} - M_{ij})^2} - \alpha X_{ij}^2\right) + \alpha\|\boldsymbol{X}\|_F^2$$

on the set of fixed-rank matrices

$$\mathcal{M}_r = \{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \text{rank}(\boldsymbol{X}) = r\}.$$

The reason for introducing the approximate function $\ell_{\boldsymbol{M},\Omega}(\boldsymbol{X})$ as the data fidelity term is two-fold: (i) to make it more robust to *outliers* in $P_\Omega(\boldsymbol{M})$ as is the ℓ_1-norm minimization and (ii) to have a smooth and differentiable objective function at the same time. The Riemannian conjugate gradient descent is used to minimize $F_\delta(\boldsymbol{X})$ over the matrix manifold $\mathcal{M}_r$. We refer to [15] for more details regarding this problem setting as well as the underlying optimization method.

By adding the graph Dirichlet energy function to F_δ, our main problem (3) writes

$$\begin{aligned} &\underset{\boldsymbol{X}}{\text{minimize}}\ F_{\delta,\boldsymbol{L}}(\boldsymbol{X}) \triangleq F_\delta(\boldsymbol{X}) + \beta\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}) \\ &\text{subject to}\quad \boldsymbol{X} \in \mathcal{M}_r, \end{aligned} \tag{23}$$

which can be solved by using the same Riemannian gradient-based algorithm of [15]. Note, however, the gradient of $F_{\delta,\boldsymbol{L}}$ needs to be computed differently at each iteration due to the additional regularization term $\text{Tr}(\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X})$.

At a feasible iterate $\boldsymbol{X} \in \mathcal{M}_r$, the Euclidean gradient of $F_{\delta,\boldsymbol{L}}$ is

$$\nabla F_{\delta,\boldsymbol{L}}(\boldsymbol{X}) = \boldsymbol{S} + 2\alpha(\boldsymbol{I} + \frac{\beta}{\alpha}\boldsymbol{L})\boldsymbol{X}, \tag{24}$$

where $\boldsymbol{S} = \nabla\ell_{\boldsymbol{M},\Omega}(\boldsymbol{X}) \in \mathbb{R}^{m\times n}$ has the following sparse form

$$S_{ij} = \begin{cases} \frac{X_{ij} - M_{ij}}{\sqrt{\delta^2 + (X_{ij} - M_{ij})^2}} - 2\alpha X_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

Subsequently, our algorithm for solving problem (23) is obtained by changing the Euclidean gradient function in (Algorithm 1 (RMC)) of [15] to $\nabla F_{\delta,\boldsymbol{L}}(\boldsymbol{X})$ (24). Note that in [15], the Riemannian gradient-based algorithm handles the matrix variable $\boldsymbol{X} \in \mathcal{M}_r$ implicitly, in the form of a tuple of rank-r factors $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \in \text{St}(r, m) \times \text{GL}(r) \times \text{St}(r, n)$ such that $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Here $\text{St}(r, m), \text{St}(r, n)$ are the (orthogonal) Stiefel manifold of $\mathbb{R}^{m\times r}$ and $\mathbb{R}^{n\times r}$ respectively and $\text{GL}(r)$ is the set of invertible $r \times r$ real-valued matrices. Practically, the replacement of $\nabla F_\delta(\boldsymbol{X})$ by $\nabla F_{\delta,\boldsymbol{L}}(\boldsymbol{X})$ (24-25) occurs in the computation of the Riemannian gradient (*i.e.* the orthogonal projection on the tangent space, see (6) in [15]), which consists of low-complexity matrix multiplications involving $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ (all rank-r matrices) and $\nabla F_{\delta,\boldsymbol{L}}(\boldsymbol{X})$. Hereafter, we refer to this adapted algorithm as Algorithm RMC-RegL.

The following scheme combines our proposed graph learning method (Section IV) with Algorithm RMC-RegL:

Algorithm 2 (GL+RMC-REGL) Graph learning and regularized matrix completion

```
Input: Subscripts $\Omega = \{(i_l, j_l) : l = 1, .., k\}$, observed matrix entries $P_\Omega(\boldsymbol{M})$ and rank value $r$. Parameters $\gamma, \alpha, \beta$.
Output: Matrix estimation $\widehat{\boldsymbol{X}}$ (and $\boldsymbol{W}, \boldsymbol{L}$).
1: Initialization: $\widehat{\boldsymbol{M}}$ or $(\boldsymbol{U}_0, \boldsymbol{S}_0, \boldsymbol{V}_0) \leftarrow r$-SVD$(\boldsymbol{M}_0)$.
2: Compute: $\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}$ and $\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}$.
3: Graph learning: $\boldsymbol{W}, \boldsymbol{L} \leftarrow$ GL-SPH$(\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}, \gamma)$. (Algorithm 1)
4: $\widehat{\boldsymbol{X}} \leftarrow$ RMC-REGL$(P_\Omega(\boldsymbol{M}), \Omega, r, \boldsymbol{L}, \alpha, \beta)$.
```

In line 2, the Gram matrix $\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}$ is computed via (14), depending on the choice of a kernel function. The computation of $\boldsymbol{\zeta}_{\widehat{\boldsymbol{M}}}$ is given by (16). In our experiments (Section VI-B) with traffic occupancy data in the form of high-dimensional time series, we use the Gaussian kernel function $k_\sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / 2\sigma^2}$ for the computation of $\boldsymbol{\gamma}_{\widehat{\boldsymbol{M}}}$.

VI. Numerical experiments

In the first part of this section, we evaluate the performance of our graph learning method on fully observed synthetic data and compare it with the state-of-the-art method [7]. In the second part, we apply the graph learning method on incomplete matrices from real data and compare matrix completion qualities of our graph-regularized matrix completion method with its baseline method [15].

A. Graph learning on synthetic data

The synthetic data is composed of *graph signals* generated via Gaussian Markov random fields (10) on several types of graphs: given a graph $\mathcal{G}_{\text{gt}} = (\mathcal{V}, \mathcal{E}_{\text{gt}}, \boldsymbol{W}_{\text{gt}})$ as *ground truth*, i.i.d. data samples are generated from the multivariate Gaussian model (10) associated with $\mathcal{G}_{\text{gt}}$. For this purpose, graphs of three different types are randomly generated using the GSP toolbox [16]. These graph types are (i) *Community* graphs: networks that have a clear partitioning pattern with closely connected *clusters* or subgraphs. Every node has typically much more connections with nodes of the same cluster than nodes of other clusters, (ii) *Sensor* networks: the nodes $\mathcal{V}$ are embedded in the 2-dimensional square $[0, 1]^2$ and uniformly distributed. The edge weights are then defined by $W_{ij} = Z_{ij} \mathbb{1}_{\{Z_{ij} \geq T\}}$, where $Z_{ij} = \exp\left(-\frac{\|\mathcal{V}_i - \mathcal{V}_j\|_2^2}{2\sigma^2}\right)$ and (iii) *Erdős-Rényi* graphs: a random graph model as proposed by [17].

The quality of the learned graphs is assessed by relative errors (see $\ell_{1,2}$ (edge/degree), Table I) of $\boldsymbol{W}$ in terms of ℓ_1 and ℓ_2 norm-based distances to the ground truth $\boldsymbol{W}_{\text{gt}}$. In addition, qualities of the edge/non-edge classification are evaluated by the *F-measure* (harmonic mean of precision and recall scores of the learned edges $\mathcal{E}$ w.r.t. $\mathcal{E}_{\text{gt}}$, Appendix VII-A). Particularly, the F-measure is computed from both hard $(\mathbb{1}\{\boldsymbol{W}^* > 0\})$ and soft $(\mathbb{1}\{\boldsymbol{W}^* > \epsilon\})$ thresholded results.

Our proposed method (Algorithm 1) is compared to the state-of-the-art [7] and baseline (k-NN with RBF-Gaussian weighted edges) methods. For all graph types, the number of nodes is set to $m = 300$ and number of samples used for graph learning is $n = 2000$. The scores are obtained by running 20 times the same experiment with optimal parameters for each method.

TABLE I: Evaluation of graph learning on synthetic data

	Metric	Baseline (kNN)	[7]	GL-SPH
Community	ℓ_2 (edge)	0.6490	0.4590	0.4861
	ℓ_1 (edge)	0.6655	0.3893	0.4067
	ℓ_2 (degree)	0.3145	0.0738	0.0953
	ℓ_1 (degree)	0.2668	0.0553	0.0766
	F-measure(0)	0.7710	0.1967	0.8860
	F-measure(ϵ)	0.7710	0.8654	0.8860
Sensor	ℓ_2 (edge)	0.5919	0.3524	0.3330
	ℓ_1 (edge)	0.7268	0.3729	0.3409
	ℓ_2 (degree)	0.3884	0.1386	0.1356
	ℓ_1 (degree)	0.3502	0.1254	0.1154
	F-measure(0)	0.8209	0.2894	0.7399
	F-measure(ϵ)	0.8209	0.5570	0.7399
Erdős-Rényi	ℓ_2 (edge)	1.1371	0.6863	0.7249
	ℓ_1 (edge)	1.2863	0.6850	0.6932
	ℓ_2 (degree)	1.6320	0.0951	0.1627
	ℓ_1 (degree)	0.6780	0.0805	0.1512
	F-measure(0)	0.3497	0.1563	0.7240
	F-measure(ϵ)	0.3497	0.6874	0.7240

Table I shows that the proposed method has performances close to those of the state-of-the-art method [7] in terms of relative errors (Appendix VII-A). This is expected since the objective function (17b) is not fundamentally different from the one proposed in [7]. Our method also performs very well in (hard-thresholded) edge classifications (F-measure(0)). For all graph types tested, both algorithms learned graphs that are closer to the ground truths than those constructed with the baseline (k-NN) method.

B. Graph learning and regularized matrix completion

We apply Algorithm 2 to a dataset obtained from the UCI repository [18]: the PeMS *Traffic* occupancy data. The *Traffic* dataset (link) is a matrix of size $963 \times 10,560$ containing traffic occupancy rates (between 0 and 1) recorded across time by $m = 963$ sensors placed along different car lanes of the San Francisco bay area freeways. The recordings are sampled every 10 minutes covering a period of 15 months. The column index set corresponds to the time domain and the row index set corresponds to geographical points (sensors), which are referred to as the spatial domain. We are interested in learning graphs in the spatial domain. Unlike the case with data from social networks or any other kind with useful meta-data, there is no obvious way to find any side information for the *Traffic* dataset that may help constructing a spatial-domain graph. This further enhances the need to learn a graph instead of constructing one for the spatial domain.

In the following experiments for matrix completion, the data matrix is only observed in the form of $P_\Omega(\boldsymbol{M})$, on a uniformly sampled index subset $\Omega \subset [\![m]\!] \times [\![n]\!]$ for a chosen sampling ratio $\frac{|\Omega|}{mn}$. We use Algorithm 2 to produce a recovered matrix $\widehat{\boldsymbol{X}}$ from the observed matrix entries $(\Omega, P_\Omega(\boldsymbol{M}))$ and evaluate the quality of $\widehat{\boldsymbol{X}}$ by (i) the Root Mean Squared Error (RMSE), (ii) the normalized RMSE

(NRMSE) [4] and (iii) the normalized deviation (ND) [4] on the whole index set $\left(\llbracket m \rrbracket \times \llbracket n \rrbracket\right)$ and/or the test set Ω^c. See Appendix VII-A for definitions of these metrics. In the graph learning step (line 3, Algorithm 2), different values of γ (problem (17b)) results in graphs of different sparsity levels (or edge densities), which can be measured by the average vertex degree $\left(\sum_{i,j=1}^{m} W_{ij}/m\right)$. Figure 1 shows how the learned graphs with different edge densities influence the regularized matrix completion step (RMC-REGL, Algorithm 2): each curve corresponds to the NRMSE scores of recovered matrices obtained by using different values of γ (problem (17b)) and a fixed value of β (problem (23)).

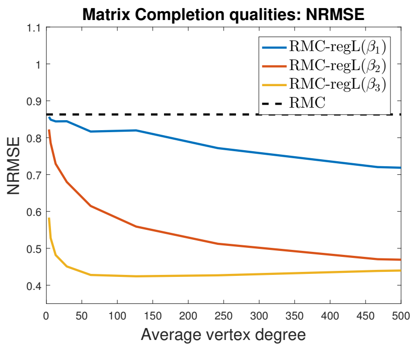

Fig. 1: Matrix completion scores (NRMSE) by Algorithm RMC-REGL compared to the baseline method RMC [15] (black dashed line). X-axis: edge densities of the learned graphs $\left(\text{for } \gamma = 5.10^{-5} \sim 5.10^{-3}\right)$.

TABLE II: Matrix completion scores on data with different sampling ratios.

	$\lvert\Omega\rvert/mn$	RMC [15]	GL+RMC-REGL
RMSE (all/test entries)	4%	0.0641 / 0.0653	**0.0278 / 0.0281**
	6%	0.0476 / 0.0489	**0.0254 / 0.0256**
	8%	0.0402 / 0.0415	**0.0247 / 0.0249**
	10%	0.0394 / 0.0410	**0.0243 / 0.0245**
	20%	0.0285 / 0.0298	**0.0236 / 0.0237**
	30%	0.0259 / 0.0269	**0.0235 / 0.0236**
	40%	0.0278 / 0.0305	**0.0234 / 0.0236**
NRMSE	4%	1.2371	**0.5311**
	6%	0.9250	**0.4848**
	8%	0.7847	**0.4716**
	10%	0.7753	**0.4639**
	20%	0.5643	**0.4490**
	30%	0.5086	**0.4477**
	40%	0.5767	**0.4468**
ND	4%	0.4694	**0.2356**
	6%	0.3146	**0.1992**
	8%	0.2543	**0.1865**
	10%	0.2334	**0.1776**
	20%	0.1783	**0.1617**
	30%	0.1639	**0.1588**
	40%	0.1662	**0.1578**

Using masked *Traffic* data with different sampling ratios as input data, we report matrix completion scores of Algorithm 2 after performing grid search for optimal values of the aforementioned parameters γ, α, β. Table II shows the scores in comparison with the baseline [15].

In addition to matrix completion scores, the quality of the recovered data as multivariate time series is also assessed. We first evaluate the Root Relative Squared Error (RRSE, see Appendix VII-A) for each of the m (univariate) time series $\widehat{\boldsymbol{X}}_{(i)}^T, i \in \llbracket m \rrbracket$ and then examine the overall imputation quality by computing the average and standard deviation (see Table III) of the RRSEs over all m nodes.

TABLE III: Mean/Std* of RRSEs over all univariate time series in $\widehat{\boldsymbol{X}}$.

	$\lvert\Omega\rvert/mn$	RMC [15]	GL+RMC-REGL
RRSE*	4%	0.9411 / 0.6376	**0.5773 / 0.1121**
	6%	0.6627 / 0.4437	**0.5278 / 0.1156**
	8%	0.5772 / 0.3439	**0.5131 / 0.1164**
	10%	0.5533 / 0.3467	**0.5043 / 0.1190**
	20%	0.5049 / 0.1946	**0.4920 / 0.1188**
	30%	0.4961 / 0.1463	**0.4901 / 0.1201**
	40%	0.4947 / 0.1786	**0.4886 / 0.1196**

The RRSE is a normalized quantity relative to the empirical standard deviation of the time series. Intuitively, any random Gaussian time series with the same mean and standard deviation will typically have a RRSE close to 1 (black dashed line in Figure 2).

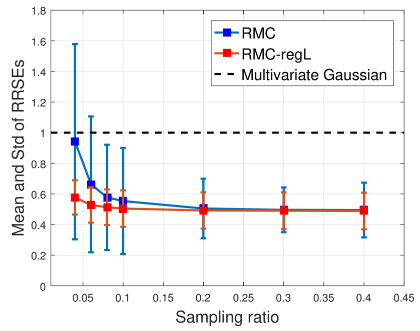

Fig. 2: Mean/Std of RRSEs at different sampling ratios: the average and standard deviation are computed over all rows of $\widehat{\boldsymbol{X}}$.

As shown in Figure 2, the time series (*i.e.* rows of $\widehat{\boldsymbol{X}}$) recovered by the graph-regularized method typically have better RRSE scores than those recovered by the baseline method. Moreover, the variance of these RRSE scores is reduced. At very low sampling ratios (*e.g.* 4%) in particular, the graph-regularized method is still able to recover all matrix rows with non-trivial (much lower than 1) qualities.

VII. CONCLUSIONS

We investigated a graph-regularized approach to low-rank matrix completion by proposing to learn graphs from an inexpensive low-rank approximation to the incomplete data matrix. Without any side information or specific graph-construction models, we formulated a fixed-scale graph learning problem as a constrained optimization problem over the unit-norm hypersphere and proposed an efficient projected gradient algorithm. Our framework for graph-regularized matrix completion (GL+RMC-REGL) showed superior performance for missing value imputation of high-dimensional time series compared to the matrix completion method without graph-regularization.

APPENDIX

A. Evaluation and metrics

The metrics used to evaluate a learned graph w.r.t. the ground truth ($\boldsymbol{w}_{\text{gt}}$) are: (1) *Relative error* of edge weights, for $\xi = 1, 2, \|\boldsymbol{w} - \boldsymbol{w}_{\text{gt}}\|_\xi / \|\boldsymbol{w}_{\text{gt}}\|_\xi$; (2) *Relative error* of degree weights: $\|\mathcal{A}(\boldsymbol{w} - \boldsymbol{w}_{\text{gt}})\|_\xi / \|\mathcal{A}\boldsymbol{w}_{\text{gt}}\|_\xi$ and (3) the *F-measure*: harmonic mean of the precision and recall scores of $\mathbb{1}\{\boldsymbol{W}^* > 0\}$ w.r.t. $\mathbb{1}\{\boldsymbol{W}_{\text{gt}} > 0\}$.

The metrics used to assess a recovered matrix $\widehat{\boldsymbol{X}}$ are: (1) the Root Mean Squared Error (RMSE) defined as

$$\text{RMSE}(\mathcal{S}) = \sqrt{\sum_{(i,j)\in\mathcal{S}} (\widehat{X}_{ij} - M_{ij})^2 / |\mathcal{S}|},$$

where $\mathcal{S} = [\![m]\!] \times [\![n]\!]$ or Ω^c; (2) the normalized RMSE (NRMSE) on the test set, defined as

$$\text{NRMSE} = \frac{\sqrt{\sum_{(i,j)\in\Omega^c} (\widehat{X}_{ij} - M_{ij})^2 / |\Omega^c|}}{\sum_{(i,j)\in\Omega^c} |M_{ij}| / |\Omega^c|}$$

and (3) the normalized deviation (ND) defined as

$$\text{ND} = \frac{\sum_{(i,j)\in\Omega^c} |\widehat{X}_{ij} - M_{ij}|}{\sum_{(i,j)\in\Omega^c} |M_{ij}|}.$$

The Root Relative Squared Error (RRSE) of an univariate time series $\boldsymbol{x} = (x_t)_{t=1,..,T}$ w.r.t. $\boldsymbol{x}^\star$ is defined as

$$\text{RRSE} = \sqrt{\sum_{t=1}^{T} \left(x_t - x_t^\star\right)^2 / \sum_{t=1}^{T} \left(x_t^\star - \bar{x}^\star\right)^2},$$

where $\bar{x}^\star$ denotes the mean value of the time series $\boldsymbol{x}^\star$.

B. Proposition 1

Proof. For any Gram matrix $\boldsymbol{\gamma} \in \mathbb{R}^{m\times m}$, the symmetry of $\boldsymbol{\gamma}$ yields $\text{Tr}(\boldsymbol{\gamma}\boldsymbol{L}) = \sum_{i,j=1}^m \gamma_{ij} L_{ij}$. Moreover, we know from the definition of the graph Laplacian (6) that for $i \in [\![m]\!], L_{ii} = d_i$, where $\boldsymbol{d} = \boldsymbol{W}\mathbf{1}$ and for $i, j \in [\![m]\!], i \neq j, L_{ij} = -W_{ij}$. Hence $\text{Tr}(\boldsymbol{\gamma}\boldsymbol{L}) = \sum_{i=1}^m \gamma_{ii} L_{ii} + \sum_{i\neq j} \gamma_{ij} L_{ij}$ equals

$$\begin{aligned}
&\text{diag}(\boldsymbol{\gamma})^T \boldsymbol{d} - \text{Tr}(\boldsymbol{\gamma}\boldsymbol{W}) \\
&= \frac{1}{2}\big(\text{diag}(\boldsymbol{\gamma})^T \boldsymbol{d} + \boldsymbol{d}^T \text{diag}(\boldsymbol{\gamma})\big) - \text{Tr}(\boldsymbol{\gamma}\boldsymbol{W}) \\
&= \frac{1}{2}\text{Tr}\big\{\boldsymbol{d}\,\text{diag}(\boldsymbol{\gamma})^T + \text{diag}(\boldsymbol{\gamma})\boldsymbol{d}^T\big\} - \text{Tr}(\boldsymbol{\gamma}\boldsymbol{W}). \\
&= \frac{1}{2}\text{Tr}\big\{\boldsymbol{W}\mathbf{1}\text{diag}(\boldsymbol{\gamma})^T + \text{diag}(\boldsymbol{\gamma})\mathbf{1}^T\boldsymbol{W}\big\} - \text{Tr}(\boldsymbol{\gamma}\boldsymbol{W}).
\end{aligned}$$

Finally, by identifying $\boldsymbol{\zeta}$ with $\mathbf{1}\text{diag}(\boldsymbol{\gamma})^T + \text{diag}(\boldsymbol{\gamma})\mathbf{1}^T - 2\boldsymbol{\gamma}$, we have $\text{Tr}(\boldsymbol{\gamma}\boldsymbol{L}) = \frac{1}{2}\text{Tr}(\boldsymbol{\zeta}\boldsymbol{W})$. Note that for this identification of $\boldsymbol{\zeta}$ to work, we simply used the fact that $\text{Tr}(\boldsymbol{W}\mathbf{1}\text{diag}(\boldsymbol{\gamma})^T) = \text{Tr}(\mathbf{1}\text{diag}(\boldsymbol{\gamma})^T\boldsymbol{W})$ in the last equation above. For the first equation above, we notice that $\sum_{i\neq j} \gamma_{ij} L_{ij} = -\sum_{i\neq j} \gamma_{ij} W_{ij} = -\sum_{i,j=1}^m \gamma_{ij} W_{ij} = -\text{Tr}(\boldsymbol{\gamma}\boldsymbol{W})$, since $\text{diag}(\boldsymbol{W}) = \mathbf{0}$ (Section III-A) for graphs without self-loops. □

REFERENCES

[1] J. Bennett and S. Lanning. The Netflix Prize. *KDD Cup and Workshop*, pages 3–6, 2007.

[2] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-Margin Matrix Factorization. *Advances in Neural Information Processing Systems*, 17:1329–1336, 2005.

[3] N. Rao, H.-F. Yu, P. Ravikumar, and I. S. Dhillon. Collaborative Filtering with Graph Information: Consistency and Scalable Methods. In *Advances in Neural Information Processing Systems 28*, pages 2107—-2115. 2015.

[4] H.-F. Yu, N. Rao, and I. S. Dhillon. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *Advances in Neural Information Processing Systems 29*, pages 847–855, 2016.

[5] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Matrix Completion on Graphs. In *NIPS2014 - Robustness in High Dimension*, 2014.

[6] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning Laplacian Matrix in Smooth Graph Signal Representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

[7] V. Kalofolias. How to Learn a Graph from Smooth Signals. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 920–929, Cadiz, Spain, 2016. PMLR.

[8] H. E. Egilmez, E. Pavez, and A. Ortega. Graph Learning from Data under Structural and Laplacian Constraints. 2016.

[9] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[10] J. Lindström. Gaussian Markov Random Fields. pages 1–6, 2014.

[11] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[12] O. Banerjee, L. El Ghaoui, and A. D'Aspremont. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

[13] J. Zhang, H. Liu, Z. Wen, and S. Zhang. A Sparse Completely Positive Relaxation of the Modularity Maximization for Community Detection. 2017.

[14] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone Spectral Projected Gradient Methods on Convex Sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[15] L. Cambier and P.-A. Absil. Robust Low-Rank Matrix Completion by Riemannian Optimization. *SISC, Siam Journal on Scientific Computing*, 38(5):1–25, 2015.

[16] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, aug 2014.

[17] E. N. Gilbert. Random Graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[18] M. Lichman. UCI Machine Learning Repository, 2013.