Nonconvex Graph Learning: Sparsity, Heavy-tails, and Clustering

José Vinícius de M. Cardoso, Jiaxi Ying, and Daniel P. Palomar

January 31, 2022

Abstract

Graph learning has been an active research area that finds applications across a number of fields including finance, health care, social sciences, and so on. In this chapter, we present an overview of recent advancements in the area of learning graphs from data, in particular undirected, weighted graphs. We focus on actual practical requirements needed from such models, *e.g.*, imposing sparsity and handling data with outliers or heavy-tails, as well as showcasing the applicability of these models on tasks such as clustering. We illustrate the performance of state-of-the-art graph learning frameworks on both synthetic and real-world datasets including financial time-series data and handwritten digits image data.

Keywords: graph learning, sparsity, heavy-tail, nonconvex, algorithms

1 Introduction

The current big data era has been enriched with seamless connectivity experienced by devices that generate unprecedented amounts of data at an ever-increasing pace. This interconnected scenario clearly demands robust models that can accurately estimate, detect, and predict relationships among entities that are part of this network.

One way to model such interconnections is via *graphical models* (Lauritzen, 1996). A particular class of graphical models, *undirected* Gaussian graphical models, has found extreme success across a number of practical fields, including biology, finance, etc. Such success is related to its ℓ_1-norm penalized *convex* formulation and the development of the Graphical Lasso algorithm via iterative block coordinate descent methods (Banerjee et al., 2008; Friedman et al., 2008; Wright, 2015). Graphical Lasso is a sparse estimator for the *unconstrained* precision (inverse covariance) matrix of a

multivariate Gaussian distribution. Mathematically, it can be expressed as the following *convex* program:

$$\begin{array}{ll} \underset{\boldsymbol{\Theta}}{\text{minimize}} & \text{tr}\left(\boldsymbol{\Theta}\boldsymbol{S}\right) - \log\det\left(\boldsymbol{\Theta}\right) + \lambda\|\boldsymbol{\Theta}\|_1, \\ \text{subject to} & \boldsymbol{\Theta} \succ \mathbf{0}, \end{array} \tag{1}$$

where $\boldsymbol{S}$ is the sample covariance matrix and $\|\cdot\|_1$ denotes the ℓ_1-norm.

Efficient optimization formulations along with scalable optimization algorithms have been key for performing inference in graphical models. More recently, there has been a growing interest in *constraining* graphical models into a particular family or structure for a variety of practical reasons, including: (i) introducing prior information available from the task at hand or (ii) testing assumptions on the data generating process. Most notably, constraints on the spectral decomposition of the adjacency and Laplacian matrices of a graph have been applied to learn bipartite and k-component graphs for data clustering (Kumar et al., 2020, 2019b; Nie et al., 2016) and sign constraints on the elements of the precision matrix of a Gaussian Markov random field (GMRF) have been employed to learn total positivity models for stock markets (Agrawal et al., 2020; Slawski and Hein, 2015; Wang et al., 2020). From an optimization standpoint, however, such spectral constraints are typically *nonconvex*, which demands the usage of powerful yet scalable optimization algorithms.

While learning the structure of general graphical models is an NP-hard task (Anandkumar et al., 2012), its importance is critical towards understanding and leveraging the information contained in such structures. Learning graphs from data is a fundamental problem in the statistical graph learning and signal processing fields (Dong et al., 2019; Egilmez et al., 2017; Friedman et al., 2008; Lake and Tenenbaum, 2010; Pavez et al., 2018; Witten et al., 2011; Zhao et al., 2019), having a direct impact on applications such as clustering (Hao et al., 2018; Hsieh et al., 2012; Kumar et al., 2020, 2019b; Nie et al., 2016; Sun et al., 2014; Tan et al., 2015), finance (de Prado, 2016; Mantegna, 1999; Marti et al., 2017), network topology inference (Coutino et al., 2019; Mateos et al., 2019; Segarra et al., 2017), graph neural nets (Wu et al., 2019), geometric deep learning (Bronstein et al., 2017), and graph variational autoencoders (Kipf and Welling, 2016; Liu et al., 2018).

In this manuscript, we discuss recent advancements in graph learning estimation methods that were only possible via nonconvex formulations and the subsequent application of optimization frameworks that can handle such nonconvexities. In addition, the algorithms discussed in this chapter are implemented in the R programming language and they are available in

the open-source packages: https://github.com/mirca/sparseGraph and https://github.com/dppalomar/spectralGraphTopology.

1.1 Learning Undirected Graphs

An undirected, weighted graph is denoted as a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, where $\mathcal{V} = \{1, 2, \ldots, p\}$ is the node set, $\mathcal{E} \subseteq \{\{u, v\} : u, v \in \mathcal{V}, u \neq v\}$ is the edge set, that is, a subset of the set of all possible unordered pairs of nodes such that $\{u, v\} \in \mathcal{E}$ iff there exists a link between nodes u and v. $\boldsymbol{W} \in \mathbb{R}_+^{p \times p}$ is the symmetric weighted adjacency matrix that satisfies $W_{ii} = 0, W_{ij} > 0$ iff $\{i, j\} \in \mathcal{E}$ and $W_{ij} = 0$, otherwise. The combinatorial, unnormalized graph Laplacian matrix $\boldsymbol{L}$ is defined, as $\boldsymbol{L} \triangleq \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D} \triangleq \mathsf{Diag}(\boldsymbol{W}\mathbf{1})$ is the degree matrix. A graph is said to be connected if and only if $D_{ii} > 0 \; \forall i$, otherwise the graph is called disconnected.

The basic idea behind learning a graph is to answer the following question: given a data matrix whose columns represent signals (observations) measured at the graph nodes, how can one design a graph that "best" fits such data matrix without possibly any (or with at most partial) knowledge of the underlying graph structure? By "graph structure", it is often understood the Laplacian, adjacency, or incidence matrices of the graph, or even a more general graph shift operator (Marques et al., 2016). In addition, the observed signals need not to live in regular, ordered spaces and can take arbitrary values, such as categorical and numerical, hence the probability distribution of the data can be highly unknown. Figure 1 illustrates such setting.

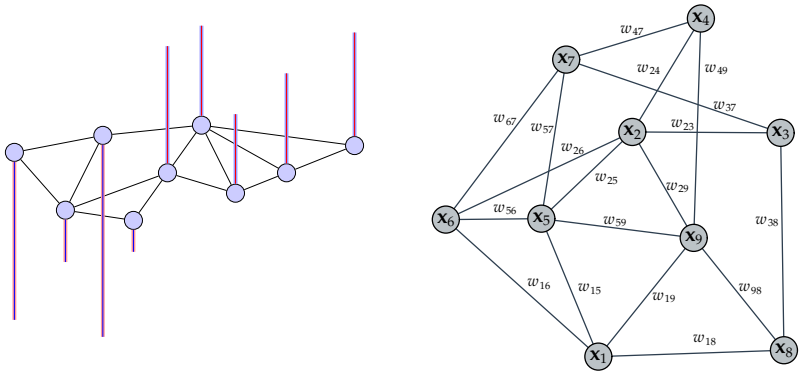
A p-dimensional, real-valued, Gaussian random vector $\boldsymbol{x}$, with mean vector $\mathbb{E}[\boldsymbol{x}] \triangleq \boldsymbol{\mu}$ and rank-deficient precision matrix $\boldsymbol{L}$, is said to form a Laplacian-constrained Gaussian Markov random field (LGMRF) (Kumar et al., 2019b; Ying et al., 2020a,b, 2021) of rank $p - k$, $k \geq 1$, with respect to a graph $\mathcal{G}$, when its probability density function is given as

$$p(\boldsymbol{x}) \propto \sqrt{\det{}^*(\boldsymbol{L})} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{L}(\boldsymbol{x} - \boldsymbol{\mu})\right\}, \tag{2}$$

where $\det^*(\boldsymbol{L})$ is the pseudo-determinant of $\boldsymbol{L}$, *i.e.*, the product of its positive eigenvalues (Knill, 2014).

Assume we are given n observations of $\boldsymbol{x}$, *i.e.*, $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^\top$, $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{x}_i \in \mathbb{R}^{p \times 1}$. The goal of graph learning algorithms is to learn a Laplacian matrix, or equivalently an adjacency matrix, given only the data matrix $\boldsymbol{X}$, *i.e.*, often without any knowledge of the edge set $\mathcal{E}$.

To that end, the penalized maximum likelihood estimator (MLE) of the Laplacian-constrained precision matrix, on the basis of the observed data $\boldsymbol{X}$,



(a) A graph signal.

(b) A graph to be estimated on the basis of its graph signals.

Figure 1: Illustration of a hypothetical signal observed in a graph (Figure 1a). The circles represent the graph nodes, the thin black lines denote the graph edges, indicating the relationships among nodes, whereas the vertical red bars denote the signal intensities measured at each node. Graph learning techniques seek to estimate the underlying graph structure (edge weights W_{ij} in Figure 1b) through the graph signal measurements.

may be formulated as the following optimization program:

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\mathsf{minimize}} & \mathsf{tr}\left(\boldsymbol{L}\boldsymbol{S}\right) - \log\det^*\left(\boldsymbol{L}\right) + h(\boldsymbol{L}), \\ \mathsf{subject\ to} & \boldsymbol{L}\mathbf{1} = \mathbf{0},\ L_{ij} = L_{ji} \leq 0, \end{array} \tag{3}$$

where $\boldsymbol{S}$ is a similarity matrix, *e.g.*, the sample covariance (or correlation) matrix $\boldsymbol{S} \propto \boldsymbol{X}^\top\boldsymbol{X}$, and h is a regularization function to promote certain properties on $\boldsymbol{L}$ such as sparsity or low-rankness. We note that we can express the Laplacian matrix via its linear operator, *i.e.*, $\boldsymbol{L} = \mathcal{L}\boldsymbol{w}$ (Kumar et al., 2019b), where $\boldsymbol{w} \in \mathbb{R}^{p(p-1)/2}$ is the vectorized form of the upper triangular part of the adjacency matrix, also known as the vector of graph weights. In addition, for connected graphs, it follows that $\det^*(\mathcal{L}\boldsymbol{w}) = \det(\mathcal{L}\boldsymbol{w} + \boldsymbol{J})$, $\boldsymbol{J} \triangleq \frac{1}{p}\mathbf{1}\mathbf{1}^\top$ (Egilmez et al., 2017).

Problem (3) is fundamental in the graph signal processing and statistical machine learning fields that has served as a cornerstone for many extensions, primarily those involving the inference of structure onto the Laplacian matrix $\boldsymbol{L}$ (Egilmez et al., 2017; Kumar et al., 2019b; Pavez et al., 2018). Even though Problem (3) is convex, provided we assume a convex choice for h, it is not adequate to be solved by disciplined convex programming languages,

such as cvxpy (Diamond and Boyd, 2016), particularly due to scalability issues related to the computation of the term $\log\det^*(\boldsymbol{L})$ (Egilmez et al., 2017; Zhao et al., 2019). Indeed, recently, considerable efforts have been directed towards the design of scalable, iterative algorithms based on block coordinate descent (Wright, 2015), majorization-minimization (MM) (Sun et al., 2017), and alternating direction method of multipliers (Boyd et al., 2011) to solve Problem (3) in an efficient fashion, *e.g.*, (Egilmez et al., 2017) and (Zhao et al., 2019).

Estimators based on Gaussian assumptions have been proposed for connected graphs (Dong et al., 2016; Egilmez et al., 2017; Kalofolias, 2016; Lake and Tenenbaum, 2010; Zhao et al., 2019). Some of their properties, such as sparsity, are yet being investigated (Ying et al., 2020b, 2021). The authors in (Kumar et al., 2019b) and (Nie et al., 2016) proposed optimization programs for learning the class of k-component graphs, as such class is an appealing model for clustering tasks due to the spectral properties of the Laplacian matrix. However, a major shortcoming in their formulations is the lack of constraints on the degrees of the nodes, which allows for trivial solutions, *i.e.*, graphs with isolated nodes.

In the following sections, we discuss optimization formulations for connected sparse graphs, heavy-tailed graphs, and k-component graphs. The latter is particular appealing for machine learning applications such as clustering. Before diving into the specific formulations, we briefly discuss the optimization frameworks leveraged for the design of optimization algorithms.

1.2 Majorization-minimization: a brief visit

Majorization-minimization (MM) is one of the primary tools to tackle optimization problems, in particular nonconvex ones, due to its flexible framework. In this short section, we briefly revisit the MM recipe.

The MM framework seeks to solve the following general optimization problem:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\mathsf{minimize}} & f(\boldsymbol{x}) \\ \mathsf{subject\ to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{4}$$

where we consider f a continuously differentiable, possibly non-convex function, and $\mathcal{X}$ is an nonempty closed set.

The general idea behind MM is to find a sequence of feasible points $\{\boldsymbol{x}^i\}_{i\in\mathbb{N}}$ by minimizing a sequence of carefully constructed global upper-bounds of f. The popular expectation-maximization (EM) algorithm is a special case of MM (Wu and Lange, 2010).

At point $\boldsymbol{x}^i$, we design a continuous global upper-bound function $g\left(\cdot, \boldsymbol{x}^i\right) : \mathcal{X} \to \mathbb{R}$ such that

$$g\left(\boldsymbol{x}, \boldsymbol{x}^i\right) \geq f(\boldsymbol{x}),\ \forall\ \boldsymbol{x} \in \mathcal{X}. \tag{5}$$

Then, in the minimization step we update $\boldsymbol{x}$ as

$$\boldsymbol{x}^{i+1} \in \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min}\ g(\boldsymbol{x}, \boldsymbol{x}^i). \tag{6}$$

The global upper-bound function $g(\cdot, \boldsymbol{x}^i)$ must satisfy the following conditions in order to guarantee theoretical convergence:

1. $g\left(\boldsymbol{x}, \boldsymbol{x}^i\right) \geq f(\boldsymbol{x})\ \forall\ \boldsymbol{x} \in \mathcal{X}$,
2. $g\left(\boldsymbol{x}^i, \boldsymbol{x}^i\right) = f\left(\boldsymbol{x}^i\right)$,
3. $\nabla g\left(\boldsymbol{x}^i, \boldsymbol{x}^i\right) = \nabla f\left(\boldsymbol{x}^i\right)$,
4. $g(\boldsymbol{x}, \boldsymbol{x}^i)$ is continuous on both $\boldsymbol{x}$ and $\boldsymbol{x}^i$.

A thorough discussion about MM, along with a significant number of its extensions, with practical examples, and comparisons to other optimization frameworks, can be found in (Sun et al., 2017).

1.3 Alternating direction method of multipliers: a brief visit

The alternating direction method of multipliers (ADMM) is a primal-dual framework designed to solve the following class of optimization problems:

$$\begin{array}{ll} \underset{\boldsymbol{x}, \boldsymbol{z}}{\text{minimize}} & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{array} \tag{7}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{z} \in \mathbb{R}^m$ are the optimization variables; $\boldsymbol{A} \in \mathbb{R}^{p \times n}$, $\boldsymbol{B} \in \mathbb{R}^{p \times m}$, and, $\boldsymbol{c} \in \mathbb{R}^p$ are parameters; and f and g are convex, proper, closed, possibly non-differentiable functions.

The central object in the ADMM framework is the augmented Lagrangian function, which is given as

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{y}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}\left\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\right\|_2^2, \tag{8}$$

where ρ is a penalty parameter.

The basic workflow of the ADMM algorithm is summarized in Algorithm 1.

The convergence of ADMM algorithms is attained provided that the following conditions are met:

Algorithm 1: ADMM framework

Data: $\boldsymbol{z}^0$, $\boldsymbol{y}^0$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{c}$, $\rho > 0$
Result: $\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{y}^\star$

$l \leftarrow 0$
while *not converged* **do**
  $\boldsymbol{x}^{l+1} \leftarrow \underset{\boldsymbol{x}\in\mathcal{X}}{\mathsf{argmin}}\ L_\rho\left(\boldsymbol{x}, \boldsymbol{z}^l, \boldsymbol{y}^l\right)$
  $\boldsymbol{z}^{l+1} \leftarrow \underset{\boldsymbol{z}\in\mathcal{Z}}{\mathsf{argmin}}\ L_\rho\left(\boldsymbol{x}^{l+1}, \boldsymbol{z}, \boldsymbol{y}^l\right)$
  $\boldsymbol{y}^{l+1} \leftarrow \boldsymbol{y}^l + \rho\left(\boldsymbol{A}\boldsymbol{x}^{l+1} + \boldsymbol{B}\boldsymbol{z}^{l+1} - \boldsymbol{c}\right)$
  $l \leftarrow l + 1$
end

1. $\mathrm{epi}(f) = \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : f(\boldsymbol{x}) \leq t\}$ and $\mathrm{epi}(g) = \{(\boldsymbol{z}, s) \in \mathbb{R}^m \times \mathbb{R} : g(\boldsymbol{z}) \leq s\}$ are both closed nonempty convex sets;

2. The unaugmented Lagrangian function L_0 has a saddle point.

We refer readers to (Boyd et al., 2011) where elaborate convergence results are discussed.

2 Sparse Graphs

Estimating sparse graph structures are of great applicability in practical scenarios because more often than not real-world graphs are sparse. One alternative to generate sparse graphs is to conduct postprocessing pruning of edges whose weights are below a specified threshold. However, this task may not be adequate because such threshold may not be known *a priori* and it may also lead to disconnected graphs, which may be undesirable. Therefore, it is paramount to include sparsity-promoting regularizers directly into the objective function or constraints of the optimization formulation. The most common sparse-inducing function is the ℓ_1-norm, which has been a key part of Graphical Lasso (Friedman et al., 2008; Witten et al., 2011).

However, the ℓ_1-norm may not always promote sparsity. To see that, consider optimization of a loss function f that attains its minimum at $\boldsymbol{x}^\star$

$$\begin{array}{ll} \underset{\boldsymbol{x}\in\mathbb{R}^p}{\mathsf{minimize}} & f(\boldsymbol{x}), \\ \mathsf{subject\ to} & \boldsymbol{x}^\top\mathbf{1} = 1, \boldsymbol{x} \geq \mathbf{0}. \end{array} \tag{9}$$

It is straightforward to see that changing the objective function to $f(\boldsymbol{x}) + \lambda\|\boldsymbol{x}\|_1$, $\lambda > 0$, does not change the solution of the optimization problem. This

toy problem is to illustrate that careful design choices must be considered before utilizing regularizations in optimization formulations.

Figure 2 illustrates the effect of the ℓ_1-norm on the estimated graphs. We generate $n = 300$ data samples from a tree graph with $p = 50$ nodes, as shown in Figure 2a, as in $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{L}^\dagger)$, where the edge weights in $\boldsymbol{L}$ are uniformly sampled from $U[2, 5]$. We use the data matrix $\boldsymbol{X}$ to recover the graph $\boldsymbol{L}$ by solving Problem 3 with $h(\cdot)$ set as the ℓ_1-norm with regularization hyperparameter λ. It can be observed that, in this scenario, the estimator defined as in Problem (3) yields denser graphs as a the hyperparameter λ increases. More details can be found in (Ying et al., 2020a,b).

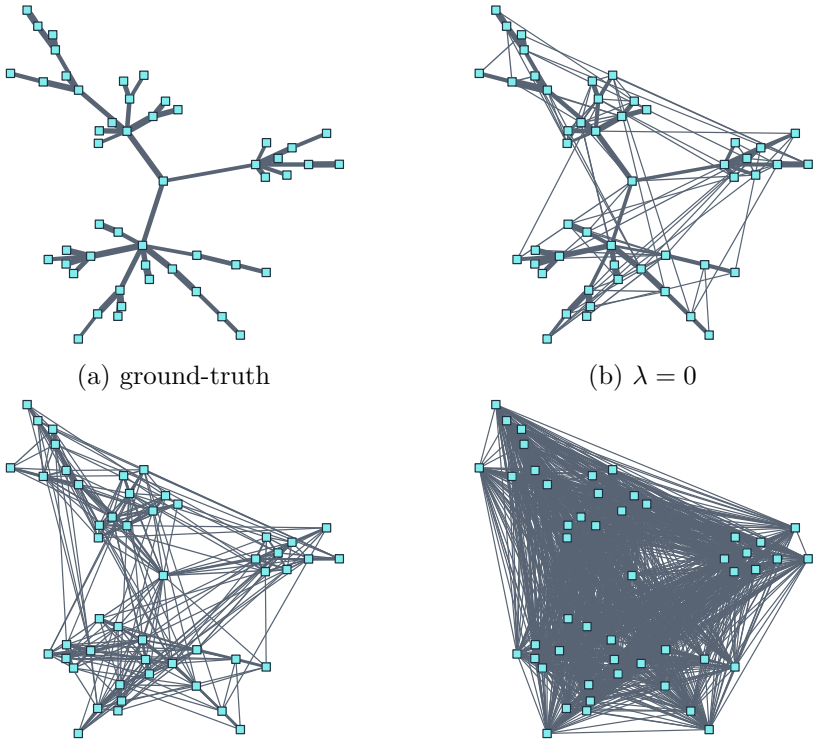
Recently, theoretical and empirical studies on sparse formulations for undirected, weighted graphs have been conducted in (Ying et al., 2020a,b, 2021; Zhang et al., 2020). The main conclusion of these works is that the ℓ_1-norm is unable to learn sparse structures in the settings of Problem (3). Therefore, they rely on concave regularizers such as the minimax concave penalty (MCP) (Zhang, 2010), smoothly clipped absolute deviation (SCAD) (Fan and Li, 2001), or the reweighted ℓ_1-norm (Candès et al., 2008).

The primary shortcoming of these regularizers is that they make the optimization formulation nonconvex. However, efficient optimization framework, such as majorization-minimization (MM) (Sun et al., 2017), can be used to generate a sequence of easy-to-solve convex problems with convergence guarantees. For instance, Ying et al. (2020b, 2021) directly used MM on the optimization formulations with MCP, SCAD, and the logarithm approximation to the ℓ_0-norm, to design an estimator that turns out to be a sequence of constrained quadratic programs. Zhang et al. (2020) used a difference-of-convex algorithm, which can be interpreted as an instance of MM (Sun et al., 2017), to design an estimator for connected graphs. These formulations are identical to that of Problem (3), expect that the function $h(\cdot)$ is either MCP, SCAD, or the logarithm approximation to the ℓ_0-norm.

As a practical example, we illustrate the design an MM-based estimator that solves a sequence of majorized *convex* problems to obtain a sparse graph. The algorithm presented here is analogous to that of Ying et al. (2020b). Recall that the general formulation for graph learning is stated in Problem (3). Assuming that the graph is connected, we have the following formulation

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\mathsf{minimize}} & \mathsf{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) - \log\det\left(\mathcal{L}\boldsymbol{w} + \boldsymbol{J}\right) + h(\boldsymbol{w}), \\ \mathsf{subject\ to} & \boldsymbol{w} \geq \boldsymbol{0}, \end{array} \tag{10}$$

where $h(\cdot)$ is a concave regularizer such as MCP, SCAD (Ying et al., 2020b), or approximations to the ℓ_0-norm (Kumar et al., 2020).



(a) ground-truth (b) $\lambda = 0$

(c) $\lambda = 0.1$ (d) $\lambda = 10$

Figure 2: Estimating Laplacian matrices via ℓ_1-norm regularization for different values of the regularization hyperparameter: (a) ground-truth graph, (b) $\lambda = 0$, (c) $\lambda = 0.1$, and (d) $\lambda = 10$. We notice that increasing the regularization intensity actually makes the estimate graph more dense. The number of nonzero edges in (b), (c) and (d) are 135, 286 and 1225, respectively. The true graph in (a) has 49 edges and the graph in (d) is fully connected. The relative errors of the learned graphs in (b), (c) and (d) are 0.14, 0.64 and 0.99, respectively.

As briefly revisited in Section 1.2, the MM recipe consists of two steps: (1) the majorization step, and the (2) minimization step. For the majorization step, a simple but often effective rule of thumb to construct a majorizer of an objective function is to a take the first-order Taylor approximation of its concave components (Sun et al., 2017). Hence, for the setting of Problem (10), we are required to find the first-order Taylor expansion of $h(\cdot)$, as the other terms are convex. Denoting by $g(\boldsymbol{w}, \boldsymbol{w}^i)$ the majorizer of the objective function in Problem (10) at a point $\boldsymbol{w}^i$, that means constructing g

as follows

$$g(\boldsymbol{w}, \boldsymbol{w}^i) = \mathsf{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) - \log\det\left(\mathcal{L}\boldsymbol{w} + \boldsymbol{J}\right) + \underbrace{h(\boldsymbol{w}^i) + \langle \boldsymbol{w} - \boldsymbol{w}^i, \nabla_{\boldsymbol{w}} h(\boldsymbol{w}^i)\rangle}_{\text{First-order Taylor expansion of } h(\cdot)}. \tag{11}$$

The next step now is to find a minimizer of $g(\cdot, \boldsymbol{w}^i)$, *i.e.*

$$\boldsymbol{w}^{i+1} \in \underset{\boldsymbol{w}\geq \boldsymbol{0}}{\mathsf{arg\,min}}\ \mathsf{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) - \log\det\left(\mathcal{L}\boldsymbol{w} + \boldsymbol{J}\right) + \langle \boldsymbol{w}, \nabla_{\boldsymbol{w}} h(\boldsymbol{w}^i)\rangle. \tag{12}$$

Note that Problem (12) is convex, therefore it can be solved by several optimization algorithms such as block coordinate descent, projected gradient descent (Ying et al., 2020b), and so on. We can also rely on computational convex frameworks, *e.g.* cvxpy (Diamond and Boyd, 2016) for low-dimensional cases. Algorithm 2 summarizes the implementation of an MM algorithm for sparse graph learning similar to that of (Ying et al., 2020b).

In addition, note that Algorithm 2 only depends on the regularizer h through is gradient ∇h. For reference, we list below the gradients of MCP and SCAD:

$$h'_{\mathsf{MCP}}(x) = \begin{cases} \lambda - \dfrac{x}{\gamma} & x \in [0, \gamma\lambda], \\ 0 & x \in [\gamma\lambda, \infty], \end{cases} \tag{13}$$

$$h'_{\mathsf{SCAD}}(x) = \begin{cases} \lambda & x \in [0, \lambda], \\ \dfrac{(\gamma\lambda - x)}{\gamma} - 1 & x \in [\lambda, \gamma\lambda], \\ 0 & x \in [\gamma\lambda, \infty], \end{cases} \tag{14}$$

where γ and λ are positive hyperparameters.

Now, we illustrate the advantages of Algorithm 2 for sparse graph learning in a practical experiment involving financial time series data. More precisely, we perform an experiment with data from returns of stocks belonging to the S&P500 in order to showcase the importance of appropriate sparsity-promoting regularizations while learning graphs. To that end, We collect price data of $p = 85$ stocks belonging to three sectors, namely, Communication Services (red), Utilities (blue), and Real Estate (green) from Apr. 22nd 2019 to Dec. 30th 2020, resulting in $n = 429$ observations.

Figure 3 shows the estimated graphs by Problem (3) using the ℓ_1-norm and MCP, which we respectively denote as "Gaussian MLE" and "Gaussian MLE with MCP". We observe that the modularity value of the MCP-based graph is twice as much as the one based on the ℓ_1-norm. In addition, the

Algorithm 2: Sparse Graph Learning

Data: Similarity matrix $\boldsymbol{S} \in \mathbb{R}^{p \times p}$, initial estimate of the graph weights $\boldsymbol{w}^0$.

Result: Graph Laplacian estimation: $\boldsymbol{\mathcal{L}w}^\star$.

$i \leftarrow 0$

while *convergence criteria* ***not*** *met* **do**

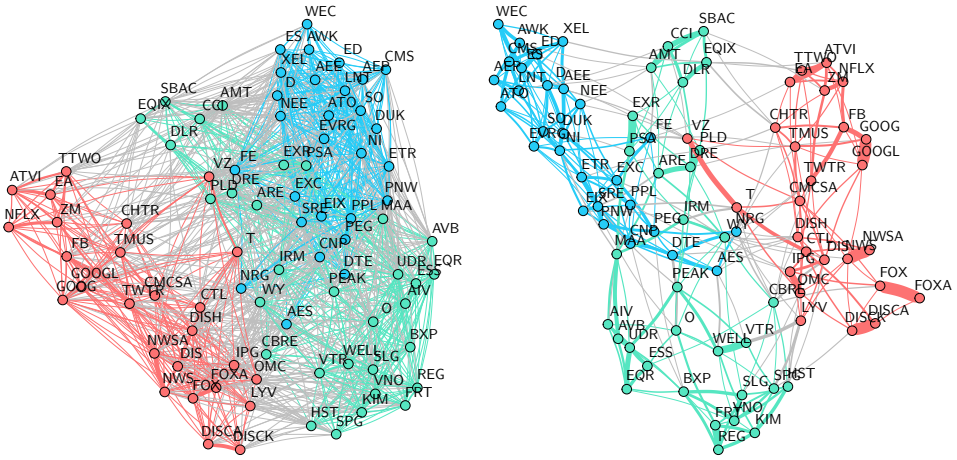
  ▷ update

  $\boldsymbol{w}^{i+1} \in \underset{\boldsymbol{w} \geq \boldsymbol{0}}{\mathsf{arg\ min}}\ \mathsf{tr}\left(\boldsymbol{\mathcal{L}w S}\right) - \log\det\left(\boldsymbol{\mathcal{L}w} + \boldsymbol{J}\right) + \langle \boldsymbol{w}, \nabla_{\boldsymbol{w}} h(\boldsymbol{w}^i)\rangle.$

  $i \leftarrow i + 1$

end

learned graph by MCP is more sparse, which can aid interpretability of the financial network. However, improved results can be obtained if we take into account that the data is actually heavy-tail distributed, which we will discuss in the next section.



(a) Gaussian MLE, $\mathsf{modularity} = 0.23$.

(b) Gaussian MLE with MCP, $\mathsf{modularity} = 0.46$.

Figure 3: Learned graphs of S&P500 stocks during the COVID-19 pandemic.

3 Heavy-tail Graphs

Heavy-tail events and outliers are prevalent in contemporaneous datasets. The appropriate modelling of such events is critical to design performant graph estimators in practical scenarios (Resnick, 2007).

Enforcing sparsity is one possible way to remove spurious conditional correlations between nodes in the presence of data with outliers. However, assuming a principled, heavy-tailed statistical distribution directly brings more benefits, rather than simply imposing arbitrary, non-convex regularizations, because they are often cumbersome to deal with from a theoretical perspective and, in practice, they bring the additional task of tunning hyperparameters, which is often repetitive.

Data from financial instruments, *e.g.*, returns of equities, currencies, and cryptocurrencies can be extremely heavy-tailed. We illustrate that phenomena empirically using returns data from the S&P500 index from different time periods. Figure 4 shows histograms of the log-returns of the S&P500 index in different periods along with a fit of a Gaussian probability density function. As it can be observed, there exists a significant discrepancy between the fitted density and its empirical counterpart especially around the tails of the distribution.

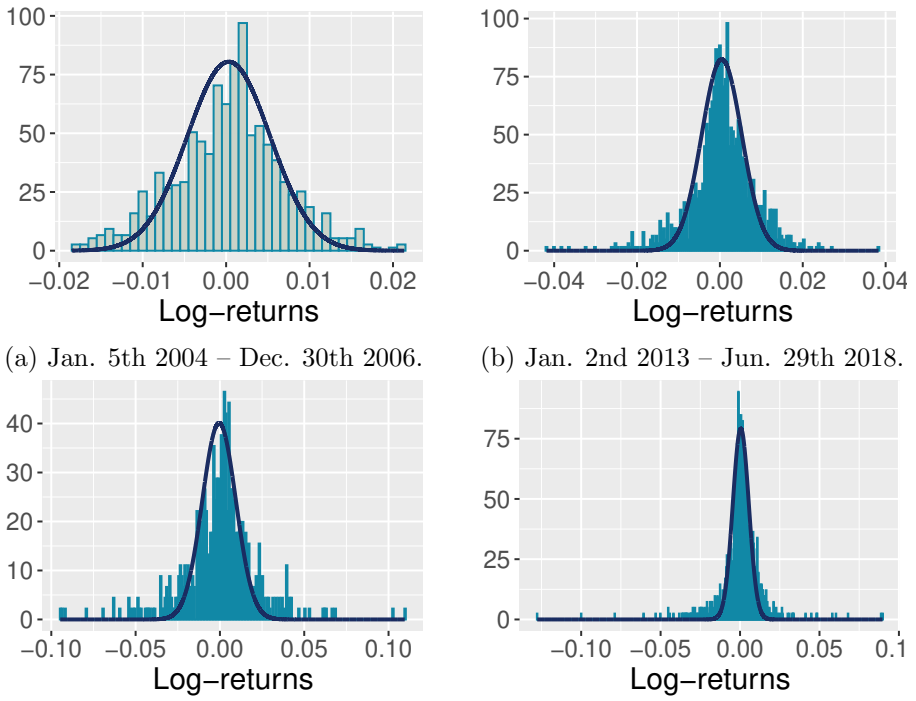
In order to address the inherently heavy-tailed nature of such datasets, the authors in (Cardoso et al., 2021) considered the Student-t distribution under the improper Markov Random field assumption (Rue and Held, 2005) with Laplacian structural constraints, that is, they assume the data generating process to be modeled a multivariate zero-mean Student-t distribution, whose probability density function can be written as

$$p(\boldsymbol{x}) \propto \sqrt{\det{}^*(\boldsymbol{\Theta})} \left(1 + \frac{\boldsymbol{x}^\top \boldsymbol{\Theta} \boldsymbol{x}}{\nu}\right)^{-\frac{\nu+p}{2}}, \ \nu > 2, \tag{15}$$

where $\boldsymbol{\Theta}$ is a positive-semidefinite inverse scatter matrix modeled as a combinatorial graph Laplacian matrix. This results in a robustified version of the MLE for connected graph learning, *i.e.*,

$$\begin{array}{ll} \underset{\boldsymbol{w}\geq\boldsymbol{0},\boldsymbol{\Theta}\succeq\boldsymbol{0}}{\mathsf{minimize}} & \dfrac{p+\nu}{n}\displaystyle\sum_{i=1}^{n}\log\left(1+\frac{\boldsymbol{x}_i^\top\boldsymbol{\mathcal{L}}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right) - \log\det\left(\boldsymbol{\Theta}+\boldsymbol{J}\right), \\ \mathsf{subject\ to} & \boldsymbol{\Theta} = \boldsymbol{\mathcal{L}}\boldsymbol{w},\ \mathfrak{d}\boldsymbol{w} = \boldsymbol{d}, \end{array} \tag{16}$$

where $\mathfrak{d} : \mathbb{R}^{p(p-1)/2} \to \mathbb{R}^p$ is the degree operator defined as $\mathfrak{d}\boldsymbol{w} \triangleq \mathsf{diag}\left(\boldsymbol{\mathcal{L}}\boldsymbol{w}\right)$. The constraint $\mathfrak{d}\boldsymbol{w} = \boldsymbol{d}$ enables the learning of additional graph structures



(a) Jan. 5th 2004 – Dec. 30th 2006.

(b) Jan. 2nd 2013 – Jun. 29th 2018.

(c) Jan. 3rd 2008 – Dec. 31st 2009.

(d) Jan. 5th 2016 – Jul. 20th 2020.

Figure 4: Histograms of the S&P500 log-returns during the aforementioned time periods, where the solid curve represents a Gaussian fit. It can be noticed that the tails of the Gaussian decay much faster than the tails of the empirical histograms, indicating the presence of heavy-tails or outliers.

such as regular graphs and it is crucial for k-component graphs to avoid isolated nodes.

From a theoretical perspective, the Student-t model naturally yields sparse graphs. Comparing the objective function in Problem (16) to that of Problem (3), we note that the Student-t contains a $\log(\cdot)$ term in place of a linear term of the graph weights. The usage of a log function to promote sparsity is closely related to the iteratively reweighted ℓ_1-norm as an approximation for the ℓ_0-norm problem (Candès et al., 2008). Problem (16) is, in general, nonconvex due to the summation of log terms and hence it is challenging to be handled directly. Therefore, the authors in (Cardoso et al., 2021) relied on optimization frameworks such as the alternating direction method of multipliers as well as the majorization-minimization to come up with a convergent algorithm for this problem.

In what follows, we illustrate how to design an ADMM algorithm for Problem (16). Since Problem (16) is nonconvex, elaborate convergence results are required, however, for the sake of simplicity we refer interested readers to check the supplementary material of (Cardoso et al., 2021).

We start by following the ADMM exposition in Section 1.3, then the partial augmented Lagrangian function of Problem (16) can be written as

$$L_\rho(\boldsymbol{\Theta}, \boldsymbol{w}, \boldsymbol{Y}, \boldsymbol{y}) = \frac{p+\nu}{n} \sum_{i=1}^{n} \log\left(1 + \frac{\boldsymbol{x}_{i,*}^\top \mathcal{L}\boldsymbol{w}\boldsymbol{x}_{i,*}}{\nu}\right) - \log\det\left(\boldsymbol{\Theta} + \boldsymbol{J}\right) + \langle \boldsymbol{y}, \mathfrak{d}\boldsymbol{w} - \boldsymbol{d}\rangle + \frac{\rho}{2}\left\|\mathfrak{d}\boldsymbol{w} - \boldsymbol{d}\right\|_2^2 + \langle \boldsymbol{Y}, \boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}\right\|_{\mathrm{F}}^2. \tag{17}$$

The subproblem for $\boldsymbol{\Theta}$ can be written as

$$\boldsymbol{\Theta}^{l+1} = \underset{\boldsymbol{\Theta}\succeq\boldsymbol{0}}{\mathsf{arg\,min}} - \log\det\left(\boldsymbol{\Theta} + \boldsymbol{J}\right) + \langle \boldsymbol{\Theta}, \boldsymbol{Y}^l\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}^l\right\|_{\mathrm{F}}^2. \tag{18}$$

Now, making the simple affine transformation $\boldsymbol{\Omega}^{l+1} = \boldsymbol{\Theta}^{l+1} + \boldsymbol{J}$, we have

$$\boldsymbol{\Omega}^{l+1} = \underset{\boldsymbol{\Omega}\succ\boldsymbol{0}}{\mathsf{arg\,min}} - \log\det\left(\boldsymbol{\Omega}\right) + \langle \boldsymbol{\Omega}, \boldsymbol{Y}^l\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Omega} - \mathcal{L}\boldsymbol{w}^l - \boldsymbol{J}\right\|_{\mathrm{F}}^2, \tag{19}$$

which can be expressed as a proximal operator (Parikh and Boyd, 2014),

$$\boldsymbol{\Omega}^{l+1} = \mathsf{prox}_{\rho^{-1}\left(-\log\,\det(\cdot) + \langle \boldsymbol{Y}^l, \cdot\rangle\right)}\left(\mathcal{L}\boldsymbol{w}^l + \boldsymbol{J}\right), \tag{20}$$

whose closed-form solution is given by Lemma 1.

Lemma 1. The global minimizer of problem (20) is (Danaher et al., 2014; Witten and Tibshirani, 2009)

$$\boldsymbol{\Omega}^{l+1} = \frac{1}{2\rho}\boldsymbol{U}\left(\boldsymbol{\Gamma} + \sqrt{\boldsymbol{\Gamma}^2 + 4\rho\boldsymbol{I}}\right)\boldsymbol{U}^\top, \tag{21}$$

where $\boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{U}^\top$ is the eigenvalue decomposition of $\rho\left(\mathcal{L}\boldsymbol{w}^l + \boldsymbol{J}\right) - \boldsymbol{Y}^l$.

Hence the closed-form solution for (18) is

$$\boldsymbol{\Theta}^{l+1} = \boldsymbol{\Omega}^{l+1} - \boldsymbol{J}. \tag{22}$$

The subproblem for $\boldsymbol{w}$ can be written as

$$\underset{\boldsymbol{w}\geq\boldsymbol{0}}{\mathsf{minimize}}\ \frac{\rho}{2}\boldsymbol{w}^\top\left(\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}\right)\boldsymbol{w} - \left\langle \boldsymbol{w}, \mathcal{L}^*\left(\boldsymbol{Y}^l + \rho\boldsymbol{\Theta}^{l+1}\right) - \mathfrak{d}^*\left(\boldsymbol{y}^l - \rho\boldsymbol{d}\right)\right\rangle + \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1 + \frac{\boldsymbol{x}_{i,*}^\top\mathcal{L}\boldsymbol{w}\boldsymbol{x}_{i,*}}{\nu}\right), \tag{23}$$

where $\mathcal{L}^*$ and $\mathfrak{d}^*$ are the adjoint operators of the Laplacian and degree operators, respectively (Cardoso et al., 2021).

We employ the MM framework to formulate an efficient iterative algorithm to obtain a stationary point of Problem (23). We proceed by constructing a global upper bound of Problem (23). Using the fact that the logarithm is globally upper-bounded by its first-order Taylor expansion, we have

$$\log\left(1+\frac{t}{b}\right) \leq \log\left(1+\frac{a}{b}\right) + \frac{t-a}{a+b}, \forall a \geq 0, t \geq 0, b > 0, \tag{24}$$

which results in the following upper bound:

$$\log\left(1+\frac{\langle \boldsymbol{w}, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle}{\nu}\right) \leq \frac{\langle \boldsymbol{w}, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle}{\langle \boldsymbol{w}^j, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle + \nu} + c_1 \tag{25}$$

where $c_1 = \log\left(1+\frac{\langle \boldsymbol{w}^j, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle}{\nu}\right) - \frac{\langle \boldsymbol{w}^j, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle}{\langle \boldsymbol{w}^j, \mathcal{L}^\star \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle + \nu}$ is a constant.

By upper-bounding the objective function of Problem (23), at point $\boldsymbol{w}^j = \boldsymbol{w}^l$, with (25), the vector of graph weights $\boldsymbol{w}$ can then be updated by solving the following nonnegative, quadratic-constrained, strictly convex problem:

$$\begin{aligned}
\boldsymbol{w}^{j+1} &= \underset{\boldsymbol{w}\geq \boldsymbol{0}}{\mathsf{arg\,min}}\ \frac{\rho}{2}\boldsymbol{w}^\top\left(\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}\right)\boldsymbol{w} - \left\langle \boldsymbol{w}, \mathcal{L}^*\left(\boldsymbol{Y}^l + \rho\boldsymbol{\Theta}^{l+1}\right) - \mathfrak{d}^*\left(\boldsymbol{y}^l - \rho\boldsymbol{d}\right)\right\rangle \\
&\qquad + \frac{p+\nu}{n}\sum_{i=1}^n \frac{\langle \boldsymbol{w}, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle}{\langle \boldsymbol{w}^j, \mathcal{L}^* \boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top\rangle + \nu} \\
&= \underset{\boldsymbol{w}\geq \boldsymbol{0}}{\mathsf{arg\,min}}\ \frac{\rho}{2}\boldsymbol{w}^\top\left(\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}\right)\boldsymbol{w} + \left\langle \boldsymbol{w}, \mathcal{L}^*\left(\tilde{\boldsymbol{S}}^j - \boldsymbol{Y}^l - \rho\boldsymbol{\Theta}^{l+1}\right) + \mathfrak{d}^*\left(\boldsymbol{y}^l - \rho\boldsymbol{d}\right)\right\rangle,
\end{aligned} \tag{26}$$

where $\tilde{\boldsymbol{S}}^j \triangleq \frac{1}{n}\sum_{i=1}^n \frac{(p+\nu)}{\langle \boldsymbol{w}^j, \mathcal{L}^*(\boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top)\rangle + \nu}\boldsymbol{x}_{i,*}\boldsymbol{x}_{i,*}^\top$ is a weighted sample covariance matrix.

Problem (26) is convex quadratic program, and hence can be solved efficiently.

The dual variables $\boldsymbol{Y}$ and $\boldsymbol{y}$ are updated as

$$\boldsymbol{Y}^{l+1} = \boldsymbol{Y}^l + \rho\left(\boldsymbol{\Theta}^{l+1} - \mathcal{L}\boldsymbol{w}^{l+1}\right) \tag{27}$$

and

$$\boldsymbol{y}^{l+1} = \boldsymbol{y}^l + \rho\left(\mathfrak{d}\boldsymbol{w}^{l+1} - \boldsymbol{d}\right). \tag{28}$$

Algorithm 3 summarizes the implementation of an ADMM algorithm to solve Problem (16).

Algorithm 3: Connected Student-t graph learning

Data: Data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, initial estimate of the graph weights $\boldsymbol{w}^0$, desired degree vector $\boldsymbol{d}$, penalty parameter $\rho > 0$, degrees of freedom ν, tolerance $\epsilon > 0$

Result: Laplacian estimation: $\mathcal{L}\boldsymbol{w}^\star$

initialize $\boldsymbol{Y} = \boldsymbol{0}$, $\boldsymbol{y} = \boldsymbol{0}$

$l \leftarrow 0$

while $\max\left(|\boldsymbol{r}^l|\right) > \epsilon$ *or* $\max\left(|\boldsymbol{s}^l|\right) > \epsilon$ **do**

  ▷ update $\boldsymbol{\Theta}^{l+1}$ via (22)

  ▷ update $\boldsymbol{w}^{l+1}$ by iterating the solution of (26)

  ▷ update $\boldsymbol{Y}^{l+1}$ as in (27)

  ▷ update $\boldsymbol{y}^{l+1}$ as in (28)

  ▷ compute residual $\boldsymbol{r}^{l+1} = \boldsymbol{\Theta}^{l+1} - \mathcal{L}\boldsymbol{w}^{l+1}$

  ▷ compute residual $\boldsymbol{s}^{l+1} = \mathfrak{d}\boldsymbol{w}^{l+1} - \boldsymbol{d}$

  $l \leftarrow l + 1$

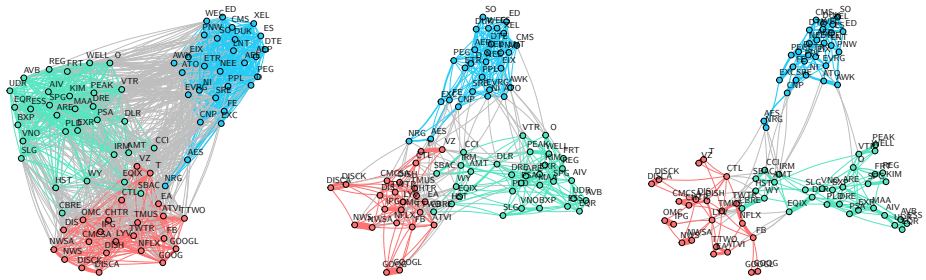
end

To illustrate the benefits of the robustified MLE, we consider learning a financial stock market networks comprised of S&P500 stocks belonging to three sectors, namely, Communication Services (red), Utilities (blue), and Real Estate (green), totalling $p = 82$ stocks, during the time horizon from Jan. 3rd 2014 to Dec. 29th 2017, resulting in $n = 1006$ observations. In order to obtain descriptive insights on this dataset, we measure its degree of heavy-tailedness and annualized volatility[1]. The former is obtained by fitting the degrees of freedom of a Student-t distribution to the matrix of log-returns, whereby we obtain $\nu \approx 5.5$ and $\sigma \approx 21\%$. This scenario can be considered as having a moderate amount of heavy-tailedness.

Figure 5 depicts the learned connected graphs on the aforementioned time periods. It can be readily noticed that the graph learned with the Student-t distribution (Figure 5c) is sparser than those learned with the Gaussian assumption (Figure 5a and 5b), which results from the fact that the Gaussian distribution is more sensitive to outliers. Moreover, the Student-t graph presents a higher degree of interpretability as measured by its modularity

[1]The annualized volatility is computed as $\sigma = \frac{\sqrt{252}}{p}\sum_{i=1}^{p}\sigma_i$, where σ_i is the daily sample standard deviation of the i-th stock.

value. In addition, a larger number of inter-sector connections, as indicated by gray-colored edges, which are often spurious from a practical perspective, are present in the graphs learned using the Gaussian MLE and Gaussian MLE with MCP regularization. Sparsity regularization provides a means to remove edges between nodes in the presence of data with outliers and possibly increasing the modularity of the resulting graph. However, they bring the additional task of tunning hyperparameters, which is often repetitive and impractical for real-time applications. A cleaner graph, without the need for postprocessing or additional regularization, is obtained directly by using the Student-t assumption. More details can be found in (Cardoso et al., 2021).



(a) Gaussian MLE, $\mathsf{modularity} = 0.31$.

(b) Gaussian MLE with MCP, $\mathsf{modularity} = 0.49$.

(c) Student-t MLE, $\mathsf{modularity} = 0.54$.

Figure 5: Learned graphs of S&P500 stocks.

4 Clustering

Recently, Kumar et al. (2020, 2019b); Nie et al. (2016) proposed optimization programs for learning the class of k-component graphs, as such class is an appealing model for clustering tasks due to the spectral properties of the Laplacian matrix. Clustering may be accomplished through graphs by directly taking advantage of the fact that a graph that has k disconnected components must satisfy $\mathsf{rank}(\boldsymbol{L}) = p - k$ (Chung, 1997).

More precisely, Nie et al. (2016) proposed the constrained Laplacian-rank (CLR) algorithm, which works in two-stages. On the first stage it estimates a connected graph and then on the second stage it heuristically projects the graph onto the set of Laplacian matrices of dimension p with rank $p-k$, where k is the given number of graph components. This approach is summarized in the following two stages:

1. Obtain an initial affinity matrix $\boldsymbol{A}^\star$ as the optimal solution of:

$$\begin{array}{ll} \underset{\boldsymbol{A}}{\text{minimize}} & \frac{1}{2}\mathsf{tr}\left(\boldsymbol{A}\boldsymbol{Z}\right) + \frac{\eta}{2}\left\|\boldsymbol{A}\right\|_{\mathrm{F}}^2, \\ \text{subject to} & \mathsf{diag}\left(\boldsymbol{A}\right) = \boldsymbol{0},\ \boldsymbol{A}\boldsymbol{1} = \boldsymbol{1},\ A_{ij} \geq 0\ \forall i, j. \end{array} \tag{29}$$

   Note that Problem (29) is a convex quadratic program, hence it can be readily solved computationally.

2. Find a projection of $\boldsymbol{A}^\star$ such that $\boldsymbol{L}^\star = \mathsf{Diag}(\frac{\boldsymbol{B}^{\star\top}+\boldsymbol{B}^\star}{2}) - \frac{\boldsymbol{B}^{\star\top}+\boldsymbol{B}^\star}{2}$ has rank $p-k$:

$$\begin{array}{ll} \underset{\boldsymbol{B},\boldsymbol{L}\succeq\boldsymbol{0}}{\text{minimize}} & \left\|\boldsymbol{B}-\boldsymbol{A}^\star\right\|_{\mathrm{F}}^2, \\ \text{subject to} & \boldsymbol{B}\boldsymbol{1} = \boldsymbol{1},\ \mathsf{rank}(\boldsymbol{L}) = p-k, \\ & \boldsymbol{L} = \mathsf{Diag}(\frac{\boldsymbol{B}^\top+\boldsymbol{B}}{2}) - \frac{\boldsymbol{B}^\top+\boldsymbol{B}}{2}, \end{array} \tag{30}$$

   where k is the desired number of graph components.

   Note that Problem (30) is nonconvex due to the rank constraint. However, a suitable approximation can be made, which leads to the design of an alternate optimization scheme. We point interested readers to (Nie et al., 2016) for the derivation.

Spectral constraints on the Laplacian matrix are an intuitive way to recover k-component graphs as the multiplicity of its zero eigenvalue, *i.e.*, the nullity of $\boldsymbol{L}$, dictates the number of components of a graph. The first framework to impose structures on the estimated Laplacian matrix under a multivariate Gaussian setting was proposed by Kumar et al. (2020, 2019b), through the use of spectral constraints, as follows:

$$\begin{array}{ll} \underset{\boldsymbol{w}\geq\boldsymbol{0},\boldsymbol{U},\boldsymbol{\lambda}}{\text{minimize}} & \mathsf{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) - \sum_{i=1}^{p-k}\log\left(\lambda_i\right) + \frac{\eta}{2}\left\|\mathcal{L}\boldsymbol{w} - \boldsymbol{U}\mathsf{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top\right\|_{\mathrm{F}}^2, \\ \text{subject to} & \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I},\ \boldsymbol{U}\in\mathbb{R}^{p\times(p-k)}, \\ & \boldsymbol{\lambda}\in\mathbb{R}_+^{p-k},\ c_1 < \lambda_1 < \cdots < \lambda_{p-k} < c_2. \end{array} \tag{31}$$

where the term $\frac{\eta}{2}\left\|\boldsymbol{L} - \boldsymbol{U}\mathsf{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top\right\|_{\mathrm{F}}^2$, often called spectral regularization, is added as a penalty term to indirectly promote $\boldsymbol{L}$ to have the same rank as $\boldsymbol{U}\mathsf{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$, *i.e.*, $p-k$, k is the number of components of the graph to be chosen *a priori*, and $\eta > 0$ is a hyperparameter that controls the penalization on the spectral factorization of $\boldsymbol{L}$, and c_1 and c_2 are positive, real-valued constants employed to promote bounds on the eigenvalues of $\boldsymbol{L}$. While Problem (31) is nonconvex, the authors in (Kumar et al., 2019b)

employed the block MM framework (Sun et al., 2017) to obtain a stationary point. Note that the block MM framework is a natural extension of the MM principle for multiple blocks of variables in a coordinate descent fashion, *i.e.*, the variables are partitioned into blocks and MM is applied to one block while keeping the value of the other blocks fixed. This framework provides a clear flexibility benefit when designing majorization functions. The convergence for the algorithm proposed in (Kumar et al., 2019b) is quite involved, therefore we refer interested readers to their supplementary material. We denote the estimator in Problem (31) as SGL.

We now elaborate on the algorithm proposed by Kumar et al. (2019b). The subproblem for $\boldsymbol{w}$ can be written as

$$\underset{\boldsymbol{w}\geq\boldsymbol{0}}{\text{minimize}} \quad \text{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) + \tfrac{\eta}{2}\left\|\mathcal{L}\boldsymbol{w} - \boldsymbol{U}\text{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top\right\|_{\text{F}}^2, \tag{32}$$

which is a convex QP without closed-form, nonetheless it can be solved via standard QP solvers or projected gradient descent methods.

The subproblem for $\boldsymbol{U}$ can be written as

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{minimize}} \quad & \left\|\mathcal{L}\boldsymbol{w} - \boldsymbol{U}\text{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top\right\|_{\text{F}}^2, \\ \text{subject to} \quad & \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I},\ \boldsymbol{U} \in \mathbb{R}^{p\times(p-k)}, \end{aligned} \tag{33}$$

which can be further simplified as

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{maximize}} \quad & \text{tr}\left(\boldsymbol{U}^\top\mathcal{L}\boldsymbol{w}\boldsymbol{U}\text{Diag}(\boldsymbol{\lambda})\right), \\ \text{subject to} \quad & \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I},\ \boldsymbol{U} \in \mathbb{R}^{p\times(p-k)}. \end{aligned} \tag{34}$$

Problem (34) is an optimization on the orthogonal Stiefel manifold $\text{St}(p, p-k) = \{\boldsymbol{U} \in \mathbb{R}^{p\times p-k} : \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}\}$. From (Absil et al., 2007) the optimizer of (34) is the $p-k$ eigenvectors of $\mathcal{L}\boldsymbol{w}$ associated with the $p-k$ largest eigenvalues of $\mathcal{L}\boldsymbol{w}$.

The subproblem for $\boldsymbol{\lambda}$ is given as

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{minimize}} \quad & -\sum_{i=1}^{p-k}\log\left(\lambda_i\right) + \frac{\eta}{2}\left\|\mathcal{L}\boldsymbol{w} - \boldsymbol{U}\text{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top\right\|_{\text{F}}^2, \\ \text{subject to} \quad & \boldsymbol{\lambda} \in \mathbb{R}_+^{p-k},\ c_1 < \lambda_1 < \cdots < \lambda_{p-k} < c_2, \end{aligned} \tag{35}$$

which can be simplified as

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{minimize}} \quad & -\sum_{i=1}^{p-k}\log\left(\lambda_i\right) + \frac{\eta}{2}\left\|\boldsymbol{u} - \boldsymbol{\lambda}\right\|_2^2, \\ \text{subject to} \quad & \boldsymbol{\lambda} \in \mathbb{R}_+^{p-k},\ c_1 < \lambda_1 < \cdots < \lambda_{p-k} < c_2, \end{aligned} \tag{36}$$

where $\boldsymbol{u} = \mathsf{diag}(\boldsymbol{U}^\top \boldsymbol{\mathcal{L}}\boldsymbol{w}\boldsymbol{U})$.

Problem (36) is a convex optimization problem with isotonic constraints. While it can be solved via general purpose convex solvers, dedicated algorithms are needed for high dimensional cases (Wang et al., 2022).

Algorithm 4 summarizes the scheme of Kumar et al. (2019b) for k-component graph learning.

Algorithm 4: k-component Graph Learning (SGL)

Data: Similarity matrix $\boldsymbol{S} \in \mathbb{R}^{p \times p}$, initial estimate of the graph weights $\boldsymbol{w}^0$, integer number of graph components k.

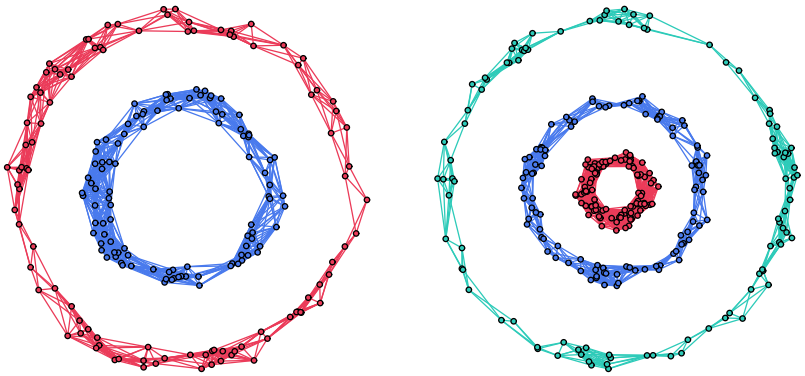
Result: Graph Laplacian estimation: $\boldsymbol{\mathcal{L}}\boldsymbol{w}^\star$.

```
i ← 0
while convergence criteria not met do
    ▷ update w^{i+1} via (32)
    ▷ update U^{i+1} via (34)
    ▷ update λ^{i+1} via (36)
    i ← i + 1
end
```

Note that Problem (31) learns a k-component graph without the need for a two-stage algorithm. However, a clear caveat of this formulation is that it does not control the degrees of the nodes in the graph, which may result in a trivial solution that contains isolated nodes, turning out not to be useful for clustering tasks especially when applied to noisy data sets or to data sets that are not significantly Gaussian distributed. In addition, choosing values for hyperparameters η, c_1, and c_2, is often an intricate task.

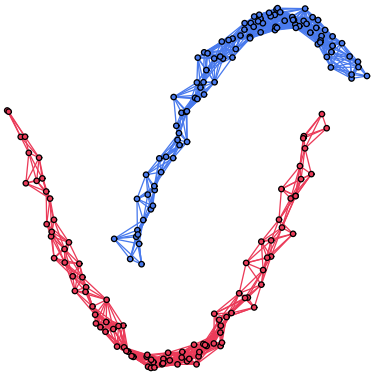
To showcase the capabilities of the estimators CLR and SGL, we perform experiments with both synthetic and real data sets. Figure 6 illustrates the results of clustering synthetic structures via SGL using the spatial coordinates of the nodes as features, with 100 nodes per cluster, *i.e.*, the data matrix is $\boldsymbol{X} \in \mathbb{R}^{100 \times n}$, where n is the number of spatial dimensions and each row of $\boldsymbol{X}$ is associated with a node's coordinates. More precisely, we have $n = 2$ in Figures 6a – 6e and $n = 3$ in Figure 6f. We observe that SGL succeeds in correctly clustering the nodes according to their cluster membership.

As for a real-world dataset, we consider foreign exchange data from the 34 most traded currencies between the period from Jan. 2nd 2019 to Dec. 31st 2020, totalling $n = 522$ observations. The data matrix is composed by the log-returns of the currencies prices with respect to the United States Dollar. Unlike in the experiment involving S&P500 stocks, there is no

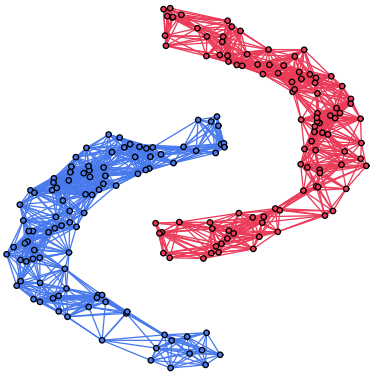


(a) Two circles.

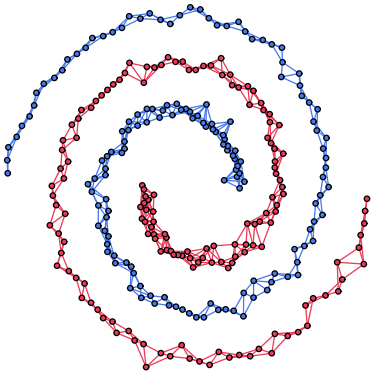
(b) Three circles.



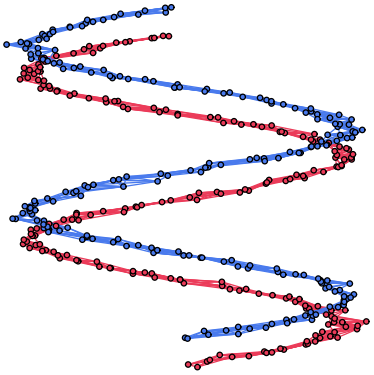
(c) Worms.



(d) Two Moon.



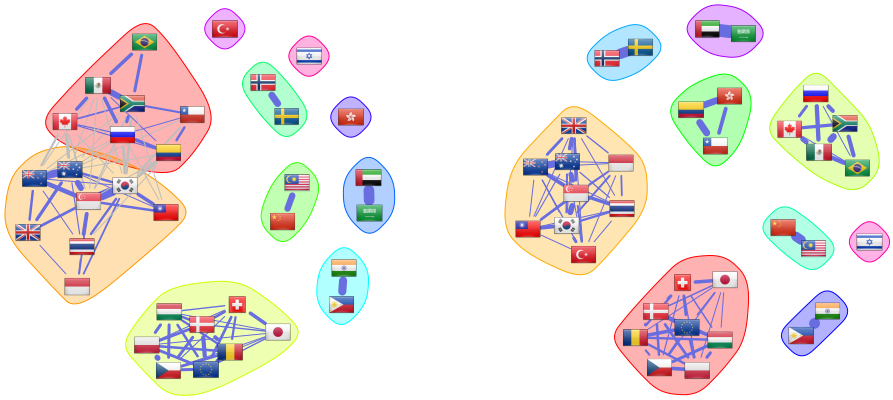
(e) Spirals.



(f) 3D Helix.

Figure 6: The estimator defined in Problem (31) is able to perfectly cluster the data points according to their cluster membership for synthetic structures.

classification standard for currencies, hence we use a community detection algorithm (Clauset et al., 2004) in order to create classes within the learned graph. In particular, the algorithm in Clauset et al. (2004) takes as input the learned Laplacian matrix of the graph and outputs a membership assignment that maximizes the modularity of the graph.



(a) SGL, modularity = 0.62. (b) CLR, modularity = 0.79.

Figure 7: Learned 9-component graphs of currencies.

Figure 7 depicts the learned 9-component graphs of currencies during the time window from Jan. 2nd 2019 to Dec. 31st 2020, where we can readily observe a few meaningful connections between currencies of countries that are geographically close to each other. We can observe that SGL and CLR allow the existence of isolated nodes in the learned graphs, which may not be ideal if we would like to cluster a particular node. More details can be found in (Cardoso et al., 2021).

4.1 Soft-clustering via bipartite graphs

Bipartite graphs are graphs whose node set can be partitioned in two disjoint groups (Chung, 1997). More formally, an undirected, weighted, bipartite graph can be defined as a 4-tuple $\mathcal{G} \triangleq \{\mathcal{V}_r, \mathcal{V}_q, \mathcal{E}, \boldsymbol{W}\}$, where $\mathcal{V}_r \triangleq \{1, 2, \ldots, r\}$ and $\mathcal{V}_q \triangleq \{r+1, r+2, \ldots, r+q\}$ are the node sets associated with a group of objects and classes, respectively; $\mathcal{E} \subseteq \{\{u, v\} : u \in \mathcal{V}_r, v \in \mathcal{V}_q\}$ is the edge set, that is, a subset of the set of all possible unordered pairs of $r+q$ nodes such that $\{u, v\} \in \mathcal{E}$ iff nodes u and v are connected. $\boldsymbol{W} \in \mathbb{R}_+^{p \times p}$ is the symmetric weighted adjacency matrix that satisfies $W_{ii} = 0, W_{ij} > 0$ iff $\{i, j\} \in \mathcal{E}$ and $W_{ij} = 0$, otherwise. Figure 8 displays an instance of such model.

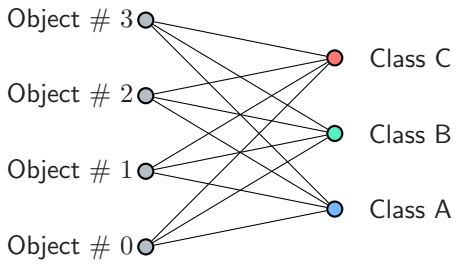


Figure 8: A bipartite graph illustrating the modeling of dependencies between a collection of objects and their classes. The edges of the graph correspond to nonzero elements in its adjacency matrix $\boldsymbol{W}$.

Properties associated with the spectral decomposition of graph matrices have demonstrated prolific advantages that enable learning graphs with specific structures, such as bipartite and k-component, and bipartite k-component graphs (Kumar et al., 2020, 2019a,b; Nie et al., 2017, 2014, 2016). Let $\boldsymbol{W} = \boldsymbol{V}\mathsf{Diag}(\boldsymbol{\psi})\boldsymbol{V}^\top$ be the spectral decomposition of the adjacency matrix. Then, the adjacency matrix of a bipartite graph satisfies the following spectral properties (Godsil and Royle, 2001):

(P1) $\boldsymbol{\psi}$ contains exactly $r - q$ zero elements.

(P2) $\boldsymbol{\psi}$ is anti-symmetric, *i.e.*, $\boldsymbol{\psi} \in C_{\boldsymbol{\psi}}$, where

$$C_{\boldsymbol{\psi}} = \{\boldsymbol{\psi} \in \mathbb{R}^p : \boldsymbol{\psi}_i = -\boldsymbol{\psi}_{2q+1-i},\ c_1 \geq \boldsymbol{\psi}_1 \geq \boldsymbol{\psi}_2 \geq \cdots \geq \boldsymbol{\psi}_q \geq c_2, \\ i = 1, 2, \ldots, q\}, \tag{37}$$

where c_1 and c_2 are positive constants.

Leveraging (P1) and (P2), Kumar et al. (2020, 2019a) proposed the following *nonconvex* program to learn a bipartite graph:

$$\begin{array}{ll} \underset{\boldsymbol{w},\boldsymbol{V},\boldsymbol{\psi}}{\mathsf{minimize}} & \mathrm{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) - \log\det\left(\mathcal{L}\boldsymbol{w} + \boldsymbol{J}\right) + \frac{\gamma}{2}\left\|\mathcal{A}\boldsymbol{w} - \boldsymbol{V}\mathsf{Diag}(\boldsymbol{\psi})\boldsymbol{V}^\top\right\|_{\mathrm{F}}^2, \\ \mathsf{subject\ to} & \boldsymbol{w} \in \mathbb{R}_+^{p(p-1)/2},\ \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}_{2q}, \boldsymbol{V} \in \mathbb{R}^{p\times 2q},\ \boldsymbol{\psi} \in C_{\boldsymbol{\psi}}, \end{array} \tag{38}$$

where $\mathcal{L}$ and $\mathcal{A}$ (Kumar et al., 2020) are the Laplacian and adjacency operators and $\boldsymbol{w}$ is the vector of graph weights. While the estimator proposed in Kumar et al. (2020) does not directly assume knowledge of the partition of the node set, it does require the availability of the number of nodes in each group. We denote the estimator defined by Problem (38) as SGA. The development of

a block MM optimization algorithm to solve Problem (38) is substantially similar to that of Problem (31), hence we omit the mathematical derivations.

Assuming a connected bipartite graph, Nie et al. (2017) proposed a heuristic approach that relies on the availability of a graph similarity matrix $\boldsymbol{B}^0 \in \mathbb{R}^{r\times q}$ and used the fact that the adjacency matrix of a bipartite graph can be written as $\boldsymbol{W} = [\mathbf{0}\ \boldsymbol{B}; \boldsymbol{B}^\top\ \mathbf{0}]$. More precisely, their estimator is obtained as the solution of the following optimization problem:

$$\begin{array}{ll} \underset{\boldsymbol{B}\in\mathbb{R}_+^{r\times q}}{\text{minimize}} & \left\|\boldsymbol{B}-\boldsymbol{B}^0\right\|_{\mathrm{F}}^2, \\ \text{subject to} & \boldsymbol{B}\geq\mathbf{0},\ \boldsymbol{B}\mathbf{1}_q=\mathbf{1}_r. \end{array} \tag{39}$$

Problem (39) is an Euclidean projection of the rows of $\boldsymbol{B}$ onto the probability simplex. While it can be solved efficiently via standard quadratic programming solvers, this approach lacks statistical support and its performance in practical applications heavily depends on the quality of the initial graph similarity matrix. We denote the estimator defined by Problem (39) as SOBG.

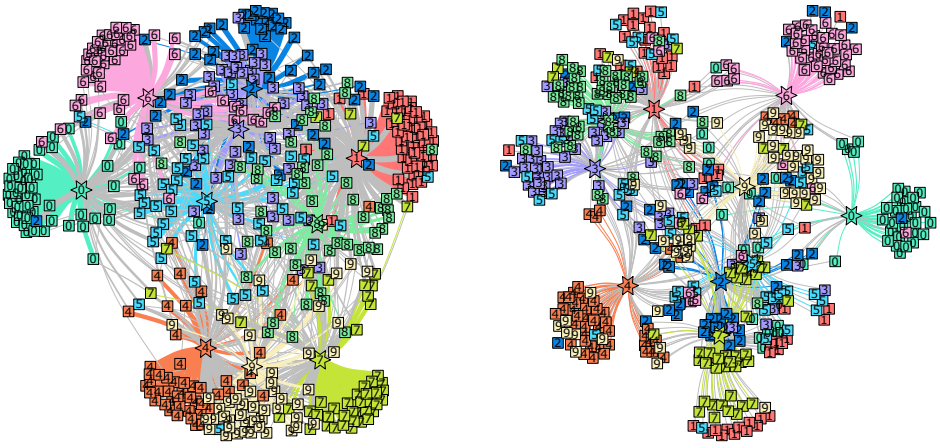
We illustrate the performance of the state-of-the-art algorithms in terms of quantitative measures such as node label accuracy and modularity. The accuracy is computed as the ratio between the number of correctly predicted labels and the number of nodes in the object set, whereas the modularity of a graph is defined as in (Newman, 2006). The modularity measures the strength of division of a graph into groups. A high modularity value means that the nodes from the same group are more likely to be connected.

We conduct a soft-clustering experiment using the MNIST image data (LeCun et al., 2010). While this dataset has been widely investigated in supervised machine (deep) learning settings, which achieved human-like accuracy, we instead use it as a proof of concept for unsupervised soft-clustering via graphical models. MNIST provides a collection of 28×28 grey-scaled images of handwritten digits from 0 to 9. The nodes in the classes set are defined such that every node corresponds to a unique digit, *i.e.*, $\mathcal{V}_q = \{0, 1, \ldots, 9\}$. We assign 500 nodes for the object set, each of which representing an image randomly selected from the testing set. We randomly select the images in a stratified way, such that every label (digit) appears 50 times. The signal for a node $v_i \in \mathcal{V}_q$, associated with digit i, is constructed by averaging 1000 randomly selected images from the training set whose label corresponds to i. We construct the data matrix $\boldsymbol{X}$ by vectorizing and stacking the images column-wise. The quantitative measures are then computed as an average of 50 realizations of this randomized experiment.

Table 1: Performance of the algorithms in soft-clustering hand-written digits.

Algorithm	Accuracy	Modularity
SOBG	0.76 +/- 0.01	0.29 +/- 0.01
SGA	0.62 +/- 0.02	0.35 +/- 0.05

We then proceed to learn graphs by the SOBG and SGA. The final label assigned to the i-th image in the objects set corresponds to $\arg\max_{j\in 1,\ldots,q} B_{ij} - 1$. Figure 9 shows the estimated graphs, while Table 1 provides quantitative results, where we observe that SGA outputs a sparser graph that turns out to have a higher modularity value than that of SOBG, on the other hand SOBG yields a graph that is more accurate.



(a) SOBG. (b) SGA.

Figure 9: Learned bipartite graphs of hand-written digital images in the MNIST dataset. Each color represents a digit. Grey-colored edges represent connections between images of distinct digits.

5 Conclusion

In this chapter, we reviewed state-of-the-art formulations and numerical optimization algorithms for graph learning under different practical requirements, *e.g.*, sparsity and heavy-tails, and different structures such as k-components and bipartite. We illustrated the power of modern graph learning estimators via experiments that included practical real-world datasets such as returns

from financial equities and currencies as well as images of handwritten digits.

6 Acknowledgements

This work was supported by the Hong Kong GRF 16207019 research grant.

References

Absil, P.-A., R. Mahony, and R. Sepulchre (2007). *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press.

Agrawal, R., U. Roy, and C. Uhler (2020, 09). Covariance matrix estimation under total positivity for portfolio selection. *Journal of Financial Econometrics*.

Anandkumar, A., V. Y. F. Tan, F. Huang, and A. S. Willsky (2012). High-dimensional Gaussian graphical model selection: Walk summability and local separation criterion. *Journal of Machine Learning Research 13*(1), 2293–2337.

Banerjee, O., L. E. Ghaoui, and A. d'Aspremont (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research 9*(15), 485–516.

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning 3*(1), 1–122.

Bronstein, M. M., J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst (2017). Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine 34*(4), 18–42.

Candès, E. J., M. B. Wakin, and S. P. Boyd (2008). Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications 14*(5), 877–905.

Cardoso, J. V. M., J. Ying, and D. P. Palomar (2021). Graphical models in heavy-tailed markets. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chung, F. R. K. (1997). *Spectral Graph Theory*, Volume 92. CBMS Regional Conference Series in Mathematics.

Clauset, A., M. E. J. Newman, and C. Moore (2004). Finding community structure in very large networks. *Physical Review E 70*, 066111.

Coutino, M., E. Isufi, T. Maehara, and G. Leus (2019). State-space network topology identification from partial observations. In *arXiv: 1906.10471*.

Danaher, P., P. Wang, and D. M. Witten (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society Series B 76*(2), 373–397.

de Prado, M. L. (2016). Building diversified portfolios that outperform out of sample. *The Journal of Portfolio Management 42*(4), 59–69.

Diamond, S. and S. Boyd (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research 17*(83), 1–5.

Dong, X., D. Thanou, P. Frossard, and P. Vandergheynst (2016). Learning Laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing 64*(23), 6160–6173.

Dong, X., D. Thanou, M. Rabbat, and P. Frossard (2019). Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine 36*(3), 44–63.

Egilmez, H. E., E. Pavez, and A. Ortega (2017). Graph learning from data under Laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing 11*(6), 825–841.

Fan, J. and R. Li (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association 96*(456), 1348–1360.

Friedman, J., T. Hastie, and R. Tibshirani (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics 9*, 432–41.

Godsil, C. and G. Royle (2001). *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer Science.

Hao, B., W. W. Sun, Y. Liu, and G. Cheng (2018). Simultaneous clustering and estimation of heterogeneous graphical models. *Journal of Machine Learning Research 18*(217), 1–58.

Hsieh, C., A. Banerjee, I. S. Dhillon, and P. K. Ravikumar (2012). A divide-and-conquer method for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems (NeurIPS'12)*, pp. 2330–2338.

Kalofolias, V. (2016). How to learn a graph from smooth signals. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, Volume 51, pp. 920–929.

Kipf, T. N. and M. Welling (2016). Variational graph auto-encoders. In *Bayesian Deep Learning Workshop, Advances in Neural Information Processing Systems (NeurIPS)*.

Knill, O. (2014). Cauchy–Binet for pseudo-determinants. *Linear Algebra and its Applications 459*, 522 – 547.

Kumar, S., J. Ying, J. V. M. Cardoso, and D. P. Palomar (2020). A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research 21*, 1–60.

Kumar, S., J. Ying, J. V. de M. Cardoso, and D. P. Palomar (2019a). Bipartite structured Gaussian graphical modeling via adjacency spectral priors. In *53rd Annual Asilomar Conference on Signals, Systems, and Computers*.

Kumar, S., J. Ying, J. V. de M. Cardoso, and D. P. Palomar (2019b). Structured graph learning via laplacian spectral constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lake, B. M. and J. B. Tenenbaum (2010). Discovering structure by learning sparse graph. In *Proceedings of the 33rd Annual Cognitive Science Conference*.

Lauritzen, S. L. (1996). *Graphical models*, Volume 17. Clarendon Press.

LeCun, Y., C. Cortes, and C. Burges (2010). MNIST handwritten digit database. In *ATT Labs [Online]*. https://yann.lecun.com/exdb/mnist.

Liu, Q., M. Allamanis, M. Brockschmidt, and A. L. Gaunt (2018). Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B 11*(1), 193–197.

Marques, A. G., S. Segarra, G. Leus, and A. Ribeiro (2016). Sampling of graph signals with successive local aggregations. *IEEE Transactions on Signal Processing 64*(7), 1832–1843.

Marti, G., F. Nielsen, M. Bińkowski, and P. Donnat (2017). A review of two decades of correlations, hierarchies, networks and clustering in financial markets. In *arXiv: 1703.00485*.

Mateos, G., S. Segarra, A. G. Marques, and A. Ribeiro (2019). Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine 36*(3), 16–43.

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America 103*, 8577–8582.

Nie, F., X. Wang, C. Deng, and H. Huang (2017). Learning a structured optimal bipartite graph for co-clustering. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Neurips'17, pp. 4132–4141.

Nie, F., X. Wang, and H. Huang (2014). Clustering and projected clustering with adaptive neighbors. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '14.

Nie, F., X. Wang, M. I. Jordan, and H. Huang (2016). The constrained Laplacian rank algorithm for graph-based clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 1969–1976.

Parikh, N. and S. Boyd (2014). Proximal algorithms. *Foundations and Trends in Optimization 1*(3), 127–239.

Pavez, E., H. E. Egilmez, and A. Ortega (2018). Learning graphs with monotone topology properties and multiple connected components. *IEEE Transactions on Signal Processing 66*(9), 2399–2413.

Resnick, S. I. (2007). *Heavy-Tail Phenomena: Probabilistic and Statistical Modeling*. Springer-Verlag New York.

Rue, H. and L. Held (2005). *Gaussian Markov Random Fields: Theory And Applications*. Chapman & Hall/CRC.

Segarra, S., A. G. Marques, G. Mateos, and A. Ribeiro (2017). Network topology inference from spectral templates. *IEEE Transactions on Signal and Information Processing over Networks 3*(3), 467–483.

Slawski, M. and M. Hein (2015). Estimation of positive definite m-matrices and structure learning for attractive gaussian markov random fields. *Linear Algebra and its Applications 473*, 145 – 179.

Sun, S., Y. Zhu, and J. Xu (2014). Adaptive variable clustering in Gaussian graphical models. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, Volume 33, pp. 931–939.

Sun, Y., P. Babu, and D. P. Palomar (2017). Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing 65*(3), 794–816.

Tan, K. M., D. Witten, and A. Shojaie (2015). The cluster graphical Lasso for improved estimation of Gaussian graphical models. *Computational Statistics & Data Analysis 85*, 23 – 36.

Wang, X., J. Ying, J. V. M. Cardoso, , and D. P. Palomar (2022). Efficient algorithms for general isotone optimization. In *The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*.

Wang, Y., U. Roy, and C. Uhler (2020). Learning high-dimensional gaussian graphical models under total positivity without adjustment of tuning parameters. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Volume 108, pp. 2698–2708.

Witten, D. M., J. H. Friedman, and N. Simon (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics 20*(4), 892–900.

Witten, D. M. and R. Tibshirani (2009). Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society. Series B (Statistical Methodology) 71*(3), 615–636.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming 151*, 3–34.

Wu, T. T. and K. Lange (2010). The MM alternative to EM. *Statistical Science 25*(4), 492–505.

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu (2019). A comprehensive survey on graph neural networks. *arXiv e-prints: 1901.00596*.

Ying, J., J. V. M. Cardoso, and D. P. Palomar (2020a). Does the ℓ_1-norm Learn a Sparse Graph under Laplacian Constrained Graphical Models? *arXiv e-prints: 2006.14925*.

Ying, J., J. V. M. Cardoso, and D. P. Palomar (2020b). Nonconvex sparse graph learning under Laplacian-structured graphical model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ying, J., J. V. M. Cardoso, and D. P. Palomar (2021). Minimax estimation of Laplacian constrained precision matrices. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, Volume 130 of *Proceedings of Machine Learning Research*, pp. 3736–3744. PMLR.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics 38*(2), 894 – 942.

Zhang, Y., K.-C. Toh, and D. Sun (2020). Learning graph laplacian with mcp.

Zhao, L., Y. Wang, S. Kumar, and D. P. Palomar (2019). Optimization algorithms for graph laplacian estimation via ADMM and MM. *IEEE Transactions on Signal Processing 67*(16), 4231–4244.