Chapter 10
Learning Graphs from Heavy-Tailed Data

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

Abstract In this chapter, we present advancements in graph learning for elliptical distributed data. More precisely, we model the inverse scatter matrix of a multivariate Student's t-distribution as a Laplacian matrix associated to a graph whose node features (or signals) are observable. We design numerical algorithms, via the alternating direction method of multipliers, to learn connected, k-component, bipartite, and k-component bipartite graphs suitable to be applied in datasets that contain outliers or whose assumption on the data-generating process is that it follows a multivariate Student's t-distribution. We measure the performance of graph learning algorithms in terms of graph modularity and node accuracy.

10.1 Introduction

Graph learning from data has been a problem of critical importance for the statistical graph learning and graph signal processing fields [21, 27, 36, 40, 54, 70, 79], with direct impact on applied areas such as unsupervised learning, clustering [31, 34, 38, 39, 49, 63, 66], applied finance [17, 42, 43], network topology inference [15, 44, 57, 58], and community detection [10, 26, 41]. In addition, graph matrices play a fundamental role in graph neural networks [52, 73].

The basic idea behind graph learning is to answer the following question: given a data matrix $\mathbf{X}$ whose columns represent signals (observations) measured at the graph nodes, how can one estimate a graph structure that best fits such data matrix without possibly any (or with at most partial) knowledge of the underlying graph structure? This question is often answered on the basis of the assumption that the observed graph signals are Gaussian distributed [20, 21, 27, 36, 39, 40, 59, 76, 79].

J. V. de Miranda Cardoso (✉) · J. Ying · D. P. Palomar
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong SAR, China
e-mail: jvdmc@connect.ust.hk

J. Ying
e-mail: jx.ying@connect.ust.hk

D. P. Palomar
e-mail: palomar@ust.hk

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

J.-P. Delmas et al. (eds.), *Elliptically Symmetric Distributions in Signal Processing and Machine Learning*, https://doi.org/10.1007/978-3-031-52116-4_10

While such an assumption for graphical models has found great success in many practical areas, which includes brain network analysis [60], psychological networks [22], and single-cell sequencing [62], it inherently neglects scenarios where there may exist outliers or the underlying data is naturally heavy-tailed distributed, such as financial time series [14, 18, 24, 30, 32, 67]. As a consequence, those methods often lack robustness and may not succeed in capturing a meaningful representation of the underlying graph [25].

Motivated by practical challenging applications such as time-series clustering and network estimation, this chapter investigates the problem of learning graph matrices whose structure follows that of a Laplacian matrix of an undirected weighted graph for which the data generating process is assumed to be Student's t-distributed. In particular, the main contributions of this chapter, which are largely based on novel material published at [8, 9], are as follows:

- We discuss a novel formulation for learning undirected weighted graphs under the assumption that the data-generating process is Student's t-distributed. We solve the underlying learning problem via a carefully designed numerical algorithm based on the alternating direction method of multipliers (ADMM). We note that this algorithm can be easily extended to account for additional linear constraints on graph weights.
- We extend the previously discussed framework to account for heavy tails, k-component graphs, as well as bipartite graphs, which enables a novel method for clustering financial time series.
- We present extensive practical results, with real-world data from the US stock market, foreign exchanges, and cryptocurrencies, that showcase clear advantages of including heavy-tail assumptions into graph learning frameworks when compared to state-of-the-art, Gaussian-based methods.
- Along with the methods discussed in this chapter, we release R packages named fingraph[1] and bipartite,[2] containing fast, unit-tested code that implements the algorithms discussed thereafter.

10.2 Background

An undirected, weighted graph is denoted as a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, 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. $\mathbf{W} \in \mathbb{R}_+^{p \times p}$ 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 $\mathbf{L}$ is defined, as $\mathbf{L} \triangleq \mathbf{D} - \mathbf{W}$, where $\mathbf{D} \triangleq \mathsf{Diag}(\mathbf{W1})$ is the degree matrix.

[1] https://github.com/convexfi/fingraph.

[2] https://github.com/convexfi/bipartite.

A p-dimensional, real-valued, Gaussian random vector $\mathbf{x}$, with mean vector $\mathbb{E}[\mathbf{x}] \triangleq \boldsymbol{\mu}$ and rank-deficient precision matrix $\mathbf{L}$, is said to form a Laplacian constrained Gaussian Markov random field (LGMRF) [38, 75–77] of rank $p-k, k \geq 1$, with respect to a graph $\mathcal{G}$, when its probability density function is given as

$$p(\mathbf{x}) \propto \sqrt{\det^*(\mathbf{L})} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{L}(\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{10.1}$$

where $\det^*(\mathbf{L})$ is the pseudo-determinant of $\mathbf{L}$, i.e., the product of its positive eigenvalues [37].

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

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

$$\begin{aligned} &\underset{\mathbf{L} \succeq \mathbf{0}}{\text{minimize}} \;\; \operatorname{tr}(\mathbf{L}\mathbf{S}) - \log\det^*(\mathbf{L}) + h(\mathbf{L}), \\ &\text{subject to } \mathbf{L}\mathbf{1} = \mathbf{0}, \;\; L_{ij} = L_{ji} \leq 0, \end{aligned} \tag{10.2}$$

where $\mathbf{S}$ is a similarity matrix, e.g., the sample covariance (or correlation) matrix $\mathbf{S} \propto \mathbf{X}^\top\mathbf{X}$, and h is a regularization function to promote certain properties on $\mathbf{L}$ such as sparsity or low-rankness.

Even though Problem (10.2) 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 [19], particularly due to scalability issues related to the computation of the term $\log\det^*(\mathbf{L})$ [21, 79]. Indeed, recently, considerable efforts have been directed toward the design of scalable, iterative algorithms based on block coordinate descent [71], majorization–minimization (MM) [35, 65], and ADMM [5] to solve Problem (10.2) in an efficient fashion, e.g., [21, 79].

Estimators based on Gaussian assumptions have been proposed for connected graphs [20, 21, 36, 40, 79]. Some of their properties, such as sparsity, are yet being investigated [76, 77]. The authors in [38, 49] 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.

Elliptical losses along with linear structural constraints that retain the positive-definiteness of the estimated covariance matrix have been proposed in the literature [64, 68]. In this work, however, we address the case of Laplacian constraints, which lead to positive-semidefinite precision matrices, and nonconvex k-component structural constraints.

10.3 Brief Detour: Algorithmic Frameworks for Optimization

The graph learning algorithms that will be discussed in the preceding section rely heavily on two well-established iterative numerical optimization frameworks, namely: Alternating Direction Method of Multipliers (ADMM) [5] and Majorization–Minimization (MM) [51, 65]. In the following subsections, we provide a brief overview about ADMM and MM.

10.3.1 Alternating Direction Method of Multipliers (ADMM)

ADMM is a primal–dual framework designed to solve the following class of optimization problems:

$$\begin{aligned} &\underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \;\; f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{subject to } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned} \tag{10.3}$$

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

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

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \tag{10.4}$$

where ρ is a penalty parameter.

The basic workflow of the ADMM algorithm is summarized in Algorithm 10.1.

Algorithm 10.1 ADMM framework

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

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

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

1. $\text{epi}(f) = \{(\mathbf{x}, t) \in \mathbb{R}^n \times \mathbb{R} : f(\mathbf{x}) \leq t\}$ and $\text{epi}(g) = \{(\mathbf{z}, s) \in \mathbb{R}^m \times \mathbb{R} : g(\mathbf{z}) \leq s\}$ are both closed nonempty convex sets.
2. The unaugmented Lagrangian function L_0 has a saddle point.

We refer readers to [5] where elaborate convergence results are discussed.

10.3.2 Majorization–Minimization (MM)

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

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \;\; f(\mathbf{x}) \\ &\text{subject to } \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{10.5}$$

where we consider f a smooth, possibly non-convex function.

The general idea behind MM is to find a sequence of feasible points $\left\{\mathbf{x}^i\right\}_{i \in \mathbb{N}}$ by minimizing a sequence of carefully constructed global upper-bounds of f. The popular expectation–maximization (EM) algorithm is a special case of MM [72].

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

$$g\left(\mathbf{x}, \mathbf{x}^i\right) \geq f(\mathbf{x}), \;\; \forall\, \mathbf{x} \in \mathcal{X}. \tag{10.6}$$

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

$$\mathbf{x}^{i+1} \in \underset{\mathbf{x} \in \mathcal{X}}{\arg\min}\, g(\mathbf{x}, \mathbf{x}^i). \tag{10.7}$$

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

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

A thorough discussion about MM, along with a significant number of its extensions, with practical examples, can be found in [65].

10.4 Algorithms

In this section, we discuss optimization formulations and an iterative algorithm to learn a Laplacian matrix from heavy-tailed assumptions. With that goal, we express the Laplacian matrix via its linear operator, *i.e.*, $\mathbf{L} = \mathcal{L}\mathbf{w}$ [38], where $\mathbf{w} \in \mathbb{R}^{p(p-1)/2}$ is the vectorized form of the upper triangular part of the adjacency matrix, also known as the vector of graph weights. The formal definition of the Laplacian operator $\mathcal{L}$ is as follows:

Definition 10.1 (*Laplacian operator*) The Laplacian operator [39] $\mathcal{L} : \mathbb{R}^{p(p-1)/2} \to \mathbb{R}^{p\times p}$, which takes a nonnegative vector $\mathbf{w}$ and outputs a Laplacian matrix $\mathcal{L}\mathbf{w}$, is defined as

$$[\mathcal{L}\mathbf{w}]_{ij} = \begin{cases} -w_{i+s(j)}, & \text{if} \quad i > j, \\ [\mathcal{L}\mathbf{w}]_{ji}, & \text{if} \quad i < j, \\ -\sum_{i\neq j} [\mathcal{L}\mathbf{w}]_{ij}, & \text{if} \quad i = j, \end{cases} \tag{10.8}$$

where $s(j) = \frac{j-1}{2}(2p - j) - j$.

In addition, we use the fact that, for connected graphs, it follows that $\det^*(\mathcal{L}\mathbf{w}) = \det(\mathcal{L}\mathbf{w} + \mathbf{J})$, $\mathbf{J} \triangleq \frac{1}{p}\mathbf{1}\mathbf{1}^\top$ [21].

In order to address the inherent heavy-tailed nature of financial market data [55], we consider the Student's t-distribution under the improper Markov random field assumption [56] with Laplacian structural constraints, that is, we assume the data generating process to be modeled as a multivariate zero-mean Student's t-distribution, whose probability density function can be written as

$$p(\mathbf{x}) \propto \sqrt{\det^*(\boldsymbol{\Theta})} \left(1 + \frac{\mathbf{x}^\top \boldsymbol{\Theta}\mathbf{x}}{\nu}\right)^{-\frac{\nu+p}{2}}, \quad \nu > 2, \tag{10.9}$$

where $\boldsymbol{\Theta}$ is a positive-semidefinite inverse scatter matrix modeled as a combinatorial graph Laplacian matrix and ν is the number of degrees of freedom, which measures the rate of decay of the tails.

This results in a robustified version of the MLE for connected graph learning, i.e.,

$$\begin{aligned} &\underset{\mathbf{w}\geq\mathbf{0},\boldsymbol{\Theta}\geq\mathbf{0}}{\text{minimize}} \quad \frac{p+\nu}{n} \sum_{i=1}^{n} \log\left(1 + \frac{\mathbf{x}_i^\top \mathcal{L}\mathbf{w}\mathbf{x}_i}{\nu}\right) - \log\det\left(\boldsymbol{\Theta} + \mathbf{J}\right), \\ &\text{subject to } \boldsymbol{\Theta} = \mathcal{L}\mathbf{w}, \ \mathfrak{d}\mathbf{w} = \mathbf{d}, \end{aligned} \tag{10.10}$$

where $\mathfrak{d} : \mathbb{R}^{p(p-1)/2} \to \mathbb{R}^p$ is the degree operator defined as follows:

Definition 10.2 (*Degree operator*) The degree operator $\mathfrak{d} : \mathbb{R}^{p(p-1)/2} \to \mathbb{R}^p$, which takes a nonnegative vector $\mathbf{w}$ and outputs the diagonal of a Laplacian matrix, is defined as

$$\mathfrak{d}\mathbf{w} = \mathsf{diag}(\mathcal{L}\mathbf{w}). \tag{10.11}$$

The constraint $\mathfrak{d}\mathbf{w} = \mathbf{d}$ in Problem (10.10) enables the learning of additional graph structures such as weighted regular graphs and it is crucial for k-component graphs.

From a theoretical perspective, the Student's t model naturally yields sparse graphs. Comparing the objective function in Problem (10.10) to that of Problem (10.2), we note that the Student's 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 [6]. Problem (10.10) is, in general, nonconvex due to the summation of log terms and hence it is challenging to be considered directly. Hence, we design an iterative algorithm based on the ADMM framework.

10.4.1 ADMM Solution

The partial augmented Lagrangian function of Problem (10.10) is given as

$$L_\rho(\boldsymbol{\Theta}, \mathbf{w}, \mathbf{Y}, \mathbf{y}) = \frac{p+\nu}{n} \sum_{i=1}^{n} \log\left(1 + \frac{\mathbf{x}_i^\top \mathcal{L}\mathbf{w}\mathbf{x}_i}{\nu}\right) - \log\det(\boldsymbol{\Theta} + \mathbf{J}) + \langle \mathbf{y}, \mathfrak{d}\mathbf{w} - \mathbf{d}\rangle + \frac{\rho}{2}\|\mathfrak{d}\mathbf{w} - \mathbf{d}\|_2^2 + \langle \mathbf{Y}, \boldsymbol{\Theta} - \mathcal{L}\mathbf{w}\rangle + \frac{\rho}{2}\|\boldsymbol{\Theta} - \mathcal{L}\mathbf{w}\|_{\mathrm{F}}^2, \tag{10.12}$$

where $\mathbf{Y}$ and $\mathbf{y}$ are the dual variables associated with the equality constraints $\boldsymbol{\Theta} = \mathcal{L}\mathbf{w}$ and $\mathfrak{d}\mathbf{w} = \mathbf{d}$, respectively. Note that we deal with the constraints $\mathbf{w} \geq \mathbf{0}$ and $\boldsymbol{\Theta} \succeq \mathbf{0}$ directly, hence there are no dual variables associated with them.

Given $\mathbf{w}^l$ and $\mathbf{Y}^l$, the subproblem for $\boldsymbol{\Theta}$ can be written as

$$\boldsymbol{\Theta}^{l+1} = \underset{\boldsymbol{\Theta}\succeq\mathbf{0}}{\arg\min} \; -\log\det(\boldsymbol{\Theta} + \mathbf{J}) + \langle \boldsymbol{\Theta}, \mathbf{Y}^l\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\mathbf{w}^l\right\|_{\mathrm{F}}^2, \tag{10.13}$$

whose closed-form solution is given by Lemma 10.1.

Lemma 10.1 *The global minimizer of Problem* (10.13) *is [16, 69]*

$$\boldsymbol{\Theta}^{l+1} = \frac{1}{2\rho}\mathbf{U}\left(\boldsymbol{\Gamma} + \sqrt{\boldsymbol{\Gamma}^2 + 4\rho\mathbf{I}}\right)\mathbf{U}^\top - \mathbf{J}, \tag{10.14}$$

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

Given $\boldsymbol{\Theta}^{l+1}$, $\mathbf{Y}^l$, and $\mathbf{y}^l$, the subproblem for $\mathbf{w}$ can be formulated as

$$\begin{aligned}\underset{\mathbf{w}\geq\mathbf{0}}{\text{minimize}}\ \ & \frac{\rho}{2}\mathbf{w}^\top\left(\mathfrak{d}^*\mathfrak{d}+\mathcal{L}^*\mathcal{L}\right)\mathbf{w}-\left\langle\mathbf{w},\mathcal{L}^*\left(\mathbf{Y}^l+\rho\boldsymbol{\Theta}^{l+1}\right)-\mathfrak{d}^*\left(\mathbf{y}^l-\rho\mathbf{d}\right)\right\rangle \\ & +\frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{\mathbf{x}_i^\top\mathcal{L}\mathbf{w}\mathbf{x}_i}{\nu}\right),\end{aligned} \tag{10.15}$$

where $\mathfrak{d}^*$ and $\mathcal{L}^*$ are the adjoint operators of the degree and Laplacian operators, respectively, which are defined as follows:

Definition 10.3 (*Adjoint of Laplacian operator*) The adjoint of Laplacian operator [39] $\mathcal{L}^* : \mathbb{R}^{p\times p} \to \mathbb{R}^{p(p-1)/2}$ is defined as

$$\left(\mathcal{L}^*\mathbf{P}\right)_{s(i,j)} = \mathbf{P}_{i,i} - \mathbf{P}_{i,j} - \mathbf{P}_{j,i} + \mathbf{P}_{j,j}, \tag{10.16}$$

where $s(i, j) = i - j + \frac{j-1}{2}(2p - j), i > j$.

Definition 10.4 (*Adjoint of degree operator*) The adjoint of degree operator $\mathfrak{d}^* : \mathbb{R}^p \to \mathbb{R}^{p(p-1)/2}$ is given as

$$\left(\mathfrak{d}^*\mathbf{y}\right)_{s(i,j)} = \mathbf{y}_i + \mathbf{y}_j, \tag{10.17}$$

where $s(i, j) = i - j + \frac{j-1}{2}(2p - j), i > j$.

In general, subproblem (10.15) is nonconvex due to the concave nature of the logarithm function. Hence, we resort to the MM method [65] to find a stationary point of subproblem (10.15). We proceed by constructing a global upper-bound of the objective function of (10.15) at point $\mathbf{w}^j \in \mathbb{R}_+^{p(p-1)/2}$ as

$$g(\mathbf{w}, \mathbf{w}^j) = g(\mathbf{w}^j, \mathbf{w}^j) + \langle \mathbf{w} - \mathbf{w}^j, \nabla_{\mathbf{w}} f(\mathbf{w}^j)\rangle + \frac{\mu}{2}\left\|\mathbf{w} - \mathbf{w}^j\right\|_2^2, \tag{10.18}$$

where f is the objective function of subproblem (10.15), its gradient is given as $\nabla_{\mathbf{w}} f(\mathbf{w}^j) = \mathbf{a}^j + \mathbf{b}^j$, where

$$\mathbf{a}^j = \mathcal{L}^*\left(\tilde{\mathbf{S}}^j - \mathbf{Y}^l - \rho\left(\boldsymbol{\Theta}^{l+1} - \mathcal{L}\mathbf{w}^j\right)\right), \tag{10.19}$$

$$\mathbf{b}^j = \mathfrak{d}^*\left(\mathbf{y}^l - \rho\left(\mathbf{d} - \mathfrak{d}\mathbf{w}^j\right)\right), \tag{10.20}$$

where $\tilde{\mathbf{S}}^j \triangleq \frac{1}{n}\sum_{i=1}^{n}\frac{(p+\nu)\mathbf{x}_i\mathbf{x}_i^\top}{\langle \mathbf{w}^j, \mathcal{L}^*(\mathbf{x}_i\mathbf{x}_i^\top)\rangle + \nu}$ is a weighted sample covariance matrix, and $\mu = \rho\lambda_{\max}\left(\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}\right)$, and the maximum eigenvalue of $\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}$ is given by Lemma 10.2 as follows:

Lemma 10.2 *The maximum eigenvalue of the matrix* $\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}$ *is given as*

$$\lambda_{\max}\left(\mathfrak{d}^*\mathfrak{d} + \mathcal{L}^*\mathcal{L}\right) = 2(2p-1). \tag{10.21}$$

The vector of graph weights $\mathbf{w}$ can then be updated by minimizing the function g constructed in (10.18), which is tantamount to solving the following nonnegative, quadratic-constrained, strictly convex problem:

$$\mathbf{w}^{j+1} = \underset{\mathbf{w}\geq\mathbf{0}}{\arg\min}\ \rho(2p-1)\left\|\mathbf{w}-\mathbf{w}^j\right\|_2^2 + \langle \mathbf{w}, \mathbf{a}^j + \mathbf{b}^j\rangle, \tag{10.22}$$

whose unique solution can be readily obtained via its KKT optimality conditions and is given as

$$\mathbf{w}^{j+1} = \left(\mathbf{w}^j - \frac{\mathbf{a}^j + \mathbf{b}^j}{2\rho(2p-1)}\right)^+, \tag{10.23}$$

that is, a projected gradient descent step with learning rate $(2\rho(2p-1))^{-1}$. Thus, we iterate (10.23) in order to obtain a stationary point, $\mathbf{w}^{l+1}$, of Problem (10.15). In practice, we observe that a few iterations are sufficient to retrieve $\mathbf{w}^{l+1}$.

The dual variables $\mathbf{Y}$ and $\mathbf{y}$ are updated via gradient ascent steps. Algorithm 10.2 summarizes the implementation to find a stationary point of Problem (10.10).

Algorithm 10.2 Student-t Graph Learning

Data: Data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, initial estimate of the graph weights $\mathbf{w}^0$, desired degree vector $\mathbf{d}$, penalty parameter $\rho > 0$, degrees of freedom $\nu > 2$, convergence tolerance $\epsilon > 0$
Result: Graph Laplacian estimation: $\mathcal{L}\mathbf{w}^\star$

initialize $\mathbf{Y} = \mathbf{0}$, $\mathbf{y} = \mathbf{0}$
$l \leftarrow 0$
while $\left\|\mathbf{r}^l\right\|_\infty > \epsilon$ *or* $\left\|\mathbf{s}^l\right\|_\infty > \epsilon$ **do**
  ▷ update $\mathbf{\Theta}^{l+1}$ via (10.14)
  ▷ update $\mathbf{w}^{l+1}$ by iterating (10.23)
  ▷ update $\mathbf{Y}^{l+1} = \mathbf{Y}^l + \rho\left(\mathbf{\Theta}^{l+1} - \mathcal{L}\mathbf{w}^{l+1}\right)$
  ▷ update $\mathbf{y}^{l+1} = \mathbf{y}^l + \rho\left(\mathfrak{d}\mathbf{w}^{l+1} - \mathbf{d}\right)$
  ▷ compute residual $\mathbf{r}^{l+1} = \mathbf{\Theta}^{l+1} - \mathcal{L}\mathbf{w}^{l+1}$
  ▷ compute residual $\mathbf{s}^{l+1} = \mathfrak{d}\mathbf{w}^{l+1} - \mathbf{d}$
  $l \leftarrow l + 1$
end

10.4.2 Extension to k-Component Graphs

The graph learning formulation presented in (10.10) is applicable to learn connected graphs. Learning graphs with k components, k assumed to be known, poses a con-

siderably higher challenge, as the dimension of the nullspace of the Laplacian matrix $\mathcal{L}\mathbf{w}$ is equal to the number of components of the graph [11]. One way to achieve this requirement is by imposing the constraint $\mathsf{rank}\,(\mathcal{L}\mathbf{w}) = p - k$, which is nonconvex and nondifferentiable, in the maximum likelihood problem generated by (10.44). We instead relax this rank constraint by noting that via Fan's theorem [23], we have

$$\sum_{i=1}^{k} \lambda_i \left(\mathcal{L}\mathbf{w}\right) = \underset{\mathbf{V}\in\mathbb{R}^{p\times k}, \mathbf{V}^\top\mathbf{V}=\mathbf{I}}{\mathsf{minimize}}\ \mathrm{tr}\left(\mathbf{V}^\top \mathcal{L}\mathbf{w}\mathbf{V}\right). \tag{10.24}$$

Thus, by using the right-hand side of (10.24) as a regularization term, we are able to formulate the following optimization problem to learn a Student's t k-component graph:

$$\begin{aligned} &\underset{\mathbf{w}\geq\mathbf{0}, \boldsymbol{\Theta}\succeq 0, \mathbf{V}}{\mathsf{minimize}}\ \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{\mathbf{x}_i^\top \mathcal{L}\mathbf{w}\mathbf{x}_i}{\nu}\right) - \log\det{}^*\left(\boldsymbol{\Theta}\right) + \eta\mathrm{tr}(\mathcal{L}\mathbf{w}\mathbf{V}\mathbf{V}^\top), \\ &\mathsf{subject\ to}\ \boldsymbol{\Theta} = \mathcal{L}\mathbf{w},\ \mathsf{rank}(\boldsymbol{\Theta}) = p-k, \mathfrak{d}\mathbf{w} = \mathbf{d},\ \mathbf{V}^\top\mathbf{V} = \mathbf{I},\ \mathbf{V}\in\mathbb{R}^{p\times k}. \end{aligned} \tag{10.25}$$

The partial augmented Lagrangian function of Problem (10.25) can be expressed as

$$\begin{aligned} L_\rho(\boldsymbol{\Theta}, \mathbf{w}, \mathbf{V}, \mathbf{Y}, \mathbf{y}) = &\frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{\mathbf{x}_i^\top \mathcal{L}\mathbf{w}\mathbf{x}_i}{\nu}\right) - \log\det{}^*\left(\boldsymbol{\Theta}\right) + \eta\mathrm{tr}\left(\mathcal{L}\mathbf{w}\mathbf{V}\mathbf{V}^\top\right) \\ &+ \langle \mathbf{y}, \mathfrak{d}\mathbf{w} - \mathbf{d}\rangle + \frac{\rho}{2}\left\|\mathfrak{d}\mathbf{w} - \mathbf{d}\right\|_2^2 + \langle \mathbf{Y}, \boldsymbol{\Theta} - \mathcal{L}\mathbf{w}\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\mathbf{w}\right\|_{\mathrm{F}}^2. \end{aligned} \tag{10.26}$$

Given $\mathbf{w}^l$ and $\mathbf{Y}^l$, the subproblem for $\boldsymbol{\Theta}$ can be written as

$$\boldsymbol{\Theta}^{l+1} = \underset{\mathsf{rank}(\boldsymbol{\Theta})=p-k}{\arg\min}\ -\log\det{}^*(\boldsymbol{\Theta}) + \langle \boldsymbol{\Theta}, \mathbf{Y}^l\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\mathbf{w}^l\right\|_{\mathrm{F}}^2, \tag{10.27}$$

which is nearly the same as (10.13). Its solution is obtained as

$$\boldsymbol{\Theta}^{l+1} = \frac{1}{2\rho}\mathbf{U}\left(\boldsymbol{\Gamma} + \sqrt{\boldsymbol{\Gamma}^2 + 4\rho\mathbf{I}}\right)\mathbf{U}^\top, \tag{10.28}$$

except that now $\mathbf{U}\boldsymbol{\Gamma}\mathbf{U}^\top$ is the eigenvalue decomposition of $\rho\mathcal{L}\mathbf{w}^l - \mathbf{Y}^l$, with $\boldsymbol{\Gamma}$ having the largest $p-k$ eigenvalues along its diagonal and $\mathbf{U}\in\mathbb{R}^{p\times(p-k)}$ contains the corresponding eigenvectors.

The subproblem to obtain $\mathbf{w}^{l+1}$ is virtually the same as in (10.15) except for the additional linear term $\eta\mathrm{tr}(\mathcal{L}\mathbf{w}\mathbf{V}^l\mathbf{V}^{l\top})$. Hence, its update is also a projected gradient descent step, alike (10.23) where

$$\mathbf{a}^j \triangleq \mathcal{L}^*\left(\tilde{\mathbf{S}}^j + \eta\mathbf{V}^l\mathbf{V}^{l\top} - \mathbf{Y}^l - \rho\left(\boldsymbol{\Theta}^{l+1} - \mathcal{L}\mathbf{w}^j\right)\right). \tag{10.29}$$

Given $\mathbf{w}^{l+1}$, we have the following subproblem for $\mathbf{V}$:

$$\underset{\mathbf{V}\in\mathbb{R}^{p\times k},\mathbf{V}^\top\mathbf{V}=\mathbf{I}}{\text{minimize}} \ \text{tr}\left(\mathbf{V}^\top\mathcal{L}\mathbf{w}^{l+1}\mathbf{V}\right), \tag{10.30}$$

whose closed-form solution is given by the k eigenvectors associated with the k smallest eigenvalues of $\mathcal{L}\mathbf{w}^{l+1}$ [1, 33]. Algorithm 10.3 summarizes the implementation to find a stationary point of Problem (10.25).

Algorithm 10.3 k-component Student-t graph learning

Data: Data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, initial estimate of the graph weights $\mathbf{w}^0$, number of graph components k, desired degree vector $\mathbf{d}$, degrees of freedom ν, rank hyperparameter $\eta > 0$, penalty parameter $\rho > 0$, tolerance $\epsilon > 0$
Result: Laplacian estimation: $\mathcal{L}\mathbf{w}^\star$

initialize $\mathbf{Y} = \mathbf{0}$, $\mathbf{y} = \mathbf{0}$
$l \leftarrow 0$
while $\left\|\mathbf{r}^l\right\|_\infty > \epsilon$ *or* $\left\|\mathbf{s}^l\right\|_\infty > \epsilon$ **do**
  ▷ update $\boldsymbol{\Theta}^{l+1}$ via (10.28)
  ▷ update $\mathbf{w}^{l+1}$ as in (10.23) with $\mathbf{a}^j$ given in (10.29)
  ▷ update $\mathbf{V}^{l+1}$ as in (10.30)
  ▷ update $\mathbf{Y}^{l+1} = \mathbf{Y}^l + \rho\left(\boldsymbol{\Theta}^{l+1} - \mathcal{L}\mathbf{w}^{l+1}\right)$
  ▷ update $\mathbf{y}^{l+1} = \mathbf{y}^l + \rho\left(\mathfrak{d}\mathbf{w}^{l+1} - \mathbf{d}\right)$
  ▷ compute residual $\mathbf{r}^{l+1} = \boldsymbol{\Theta}^{l+1} - \mathcal{L}\mathbf{w}^{l+1}$
  ▷ compute residual $\mathbf{s}^{l+1} = \mathfrak{d}\mathbf{w}^{l+1} - \mathbf{d}$
  $l \leftarrow l + 1$
end

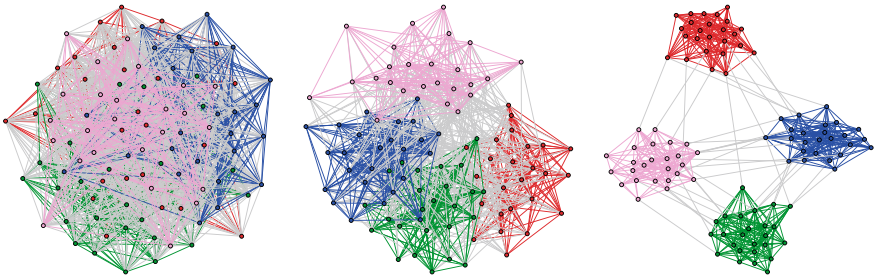
10.5 Numerical Experiments

In order to evaluate the performance of the graph learning algorithms, we perform experiments using historical daily price time series data, available in Yahoo! Finance™, from financial instruments in three scenarios: (i) stocks belonging to the S&P500 index, (ii) foreign exchange markets, and (iii) cryptocurrencies. We start by constructing the log-returns data matrix, i.e., a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, where n is the number of log-returns observations and p is the number of instruments, as

$$X_{i,j} = \log P_{i,j} - \log P_{i-1,j}, \tag{10.31}$$

where $P_{i,j}$ is the closing price of the j-th instrument at the i-th day.

We note that, in all the experiments, we have $\frac{n}{p} > 1$, meaning that we are not in a high-dimensional setting.



(a) modularity = 0.1. (b) modularity = 0.37. (c) modularity = 0.7.

Fig. 10.1 Visualizations of graphs with different values of modularity

Benchmarks: We compare algorithms developed in this chapter with state-of-the-art, Gaussian distribution-based methods for connected graphs, namely GLE [21, 79] and NGL [76], which use ℓ_1-norm and minimax concave penalty regularizations for sparsity inducing solutions, respectively; and CLR [49] and SGL [38] that consider k-component graphs. For a fair comparison among algorithms, we set the degree vector $\mathbf{d}$ equal to $\mathbf{1}$, i.e., we do not consider any prior information on the degree of nodes.

The figure of merit used to quantify the performance of the graph learning methods is the modularity of the estimated graph.

Definition 10.5 (*Modularity*) The modularity of a graph $\mathcal{G}$ [47] is defined as

$$\text{modularity}(\mathcal{G}) \triangleq \frac{1}{2|\mathcal{E}|} \sum_{i,j \in \mathcal{V}} \left(\mathbf{W}_{ij} - \frac{d_i d_j}{2|\mathcal{E}|} \right) \mathbb{I}(t_i = t_j), \tag{10.32}$$

where d_i is the weighted degree of the i-th node, i.e., $d_i \triangleq [\mathfrak{d}(\mathbf{w})]_i$, t_i is the type (or label) of the i-th node, and $\mathbb{I}(\cdot)$ is the indicator function.

The graph modularity measures the strength of separability of a graph into groups [47]. For illustrative purposes, Fig. 10.1 shows the structure of graphs with different modularity values. It is observed that highly modular graphs present a community-like structure, which is often the expected behavior for networks of financial instruments such as stocks.

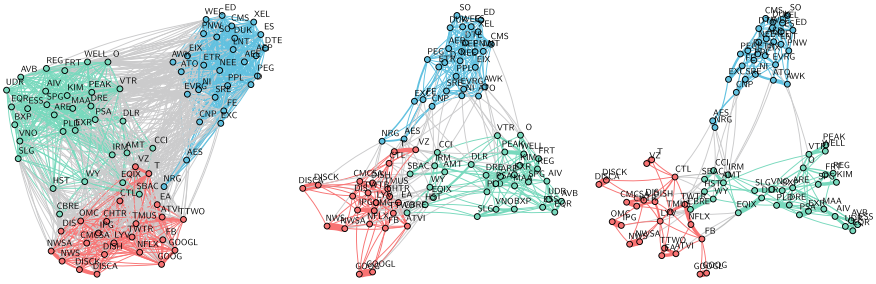
In our ADMM algorithms, we set the penalty parameter to $\rho = 1$ and the hyperparameter η in (10.25) is adaptively increased until the rank constraint is satisfied. For GLE and NGL, we use grid search on the sparsity hyperparameter such that the resulting graph yields the highest modularity value. The graph weights in Algorithms 10.2 and 10.3 are initialized using the same procedure as in [39]. Finally, the choice of k is guided by the number of sectors that we consider in the S&P500. For other experiments, the choice of k is somewhat arbitrary, but cross-validation approaches could be applied.

Our goal with the experiments that follow is to verify whether the heavy-tail assumption provides an improved version of the learned graph, which is evaluated based on the modularity of the estimated graph and the graph visualization. In addition, for the task of clustering stocks, we analyze whether the learned graphs agree with industry standards of sector classification set by the Global Industry Classification Standard (GICS)[3] [46, 61].

10.5.1 Communities in S&P500 Stocks

In this experiment, we consider S&P500 stocks belonging to three sectors, namely, Communication Services (red), Utilities (blue), and Real Estate (green), totaling $p = 82$ stocks, during the time horizon from January 3, 2014, to December 29, 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.[4] The former is obtained by fitting the degrees of freedom of a Student's 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 10.2 depicts the learned connected graphs on the aforementioned time periods. It can be readily noticed that the graph learned with the Student's t-distribution (Fig. 10.2c) is sparser than those learned with the Gaussian assumption (Fig. 10.2a and b), which results from the fact that the Gaussian distribution is more sensitive to outliers. Moreover, the Student's t graph presents a higher degree of interpretability

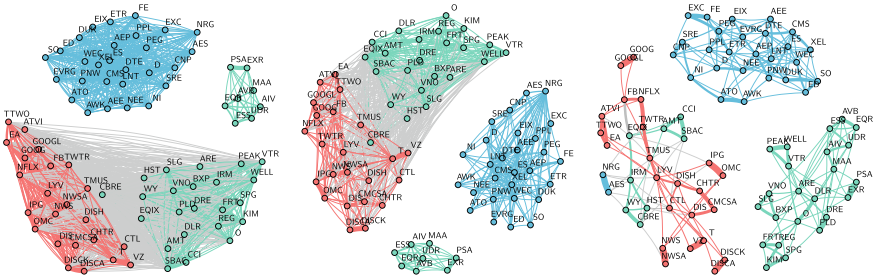


(a) GLE, modularity = 0.31. (b) NGL, modularity = 0.49. (c) Algorithm 10.2, modularity = 0.54.

Fig. 10.2 Learned graphs of S&P500 stocks

[3] More often than not, stocks have impacts on multiple sectors, e.g., the evident case of technology companies, whose influence not only affects the prices of stocks not only in their own sector but also spans across multiple sectors. Therefore, GICS or other sector classification systems might not always be representative systems for ground-truth labels.

[4] 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.



(a) SGL, modularity = 0.29. (b) CLR, modularity = 0.33. (c) Algorithm 10.3, modularity = 0.56.

Fig. 10.3 Learned 3-component graphs of S&P500 stocks

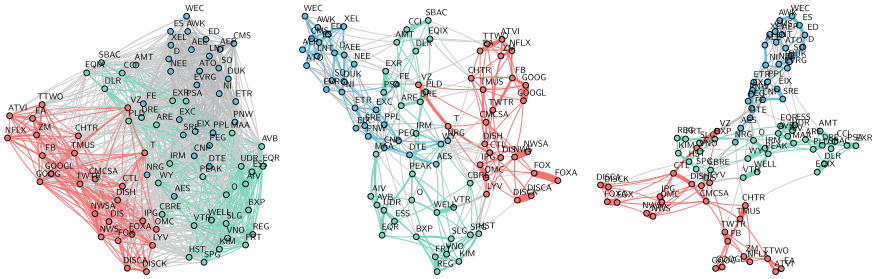
as measured by its modularity value. In addition, a larger number of inter-sector connections, as indicated by gray-colored edges, which are often spurious from a practical perspective, are present in the graphs learned by NGL and GLE. Sparsity regularization provides a means to remove edges between 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's t assumption.

Figure 10.3 illustrates the learned 3-component graphs during the time from January 3, 2014, to December 29, 2017. We can notice that SGL (Fig. 10.3a) and CLR (Fig. 10.3b) are unable to separate the stocks in a way that agrees with their sector information as given by GICS. In addition, the high number of spurious connections (gray-colored edges) is uncharacteristic of the actual expected behavior in stock markets. Figure 10.3c displays the graph learned by Algorithm 10.3, where it presents not only a higher modularity value but also a sparser, more plausible representation of an actual network of stocks with three sectors.

10.5.1.1 Impact of COVID-19 in the US Stock Market

We collect price data of $p = 85$ stocks belonging to three sectors, namely, Communication Services (red), Utilities (blue), and Real Estate (green) from April 22, 2019, to December 30, 2020, resulting in $n = 429$ observations. The degrees of freedom and annualized volatility during this period, which includes the financial crisis caused by the COVID-19 pandemic, were $\nu \approx 2.89$ and $\sigma \approx 41\%$, indicating a strongly heavy-tailed scenario.

In Fig. 10.4, we observe that the modularity values of the Gaussian-based graphs are severely reduced as compared to the results presented in Fig. 10.2, while the opposite occurs with the Student's t graph. That may be explained by the increase in



(a) GLE, modularity = 0.23. (b) NGL, modularity = 0.46. (c) Algorithm 10.2, modularity = 0.56.

Fig. 10.4 Learned graphs of S&P500 stocks during the COVID-19 pandemic

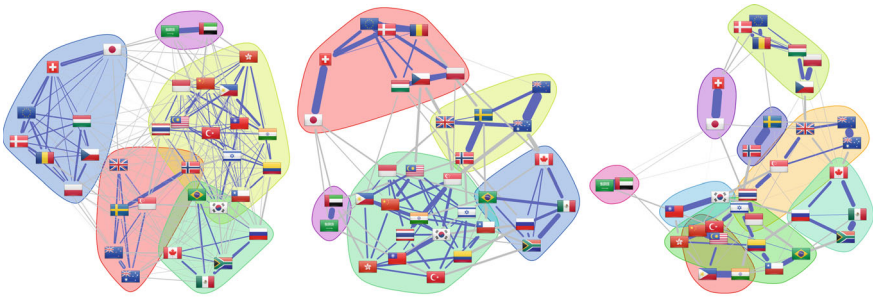
variability and outliers in the data during the financial crisis caused by the COVID-19 pandemic, which is better modeled by a Student's t-distribution. In addition, the learned graph in Fig. 10.4c clearly displays expected behaviors such as the strong correlation between pairs of companies including {Zoom Video Communications Inc., Netflix Inc.}, and {Twitter Inc., Facebook Inc.}, which are not as evident in Fig. 10.4a and b.

10.5.2 Communities in Foreign Exchange Markets

We query foreign exchange data from the 34 most traded currencies between the period from January 2, 2019, to December 31, 2020, totaling $n = 522$ observations. The data matrix is composed by the log-returns of the currency prices with respect to the United States Dollar. Similar to the previous experiment, we compute the degrees of freedom of a Student's t-distribution fitted to the log-returns data matrix, whereby we obtain $\nu \approx 4.6$, which represents a scenario with considerable amount of heavy-tailedness. Unlike in the experiment involving S&P500 stocks, there are no classification standard for currencies, hence we rely on a community detection algorithm [12] in order to create classes within the learned graph. In particular, the algorithm in [12] takes as input the learned Laplacian matrix of the graph and outputs a membership assignment that maximizes the modularity of the graph.

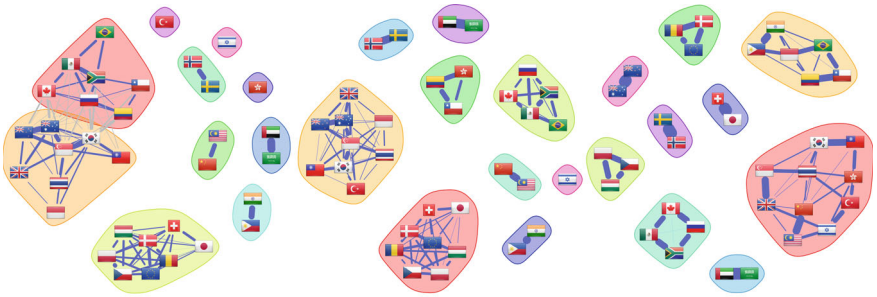
Figure 10.5 displays the learned graphs. As it can be observed, the Student's t graph (Fig. 10.5c) is sparser, more interpretable, and has a higher modularity value than that of the Gaussian-based graphs (Fig. 10.5a and b). In addition, the expected correlation between currencies of locations geographically close to each other, e.g., {Hong Kong SAR, China}, {Taiwan, South Korea}, and {Poland, Czech Republic} are significantly more evident for the Student's t graph.

Figure 10.6 depicts the learned 9-component graphs of currencies during the time window from January 2, 2019, to December 31, 2020. It can be observed that the



(a) GLE, modularity = 0.34. (b) NGL, modularity = 0.46. (c) Algorithm 10.2, modularity = 0.58.

Fig. 10.5 Learned connected graphs of currencies



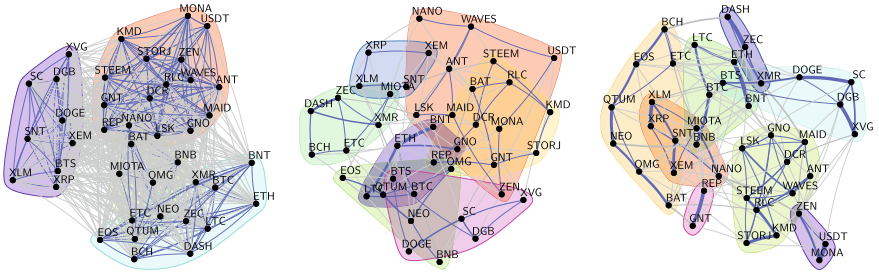
(a) SGL, modularity = 0.62. (b) CLR, modularity = 0.79. (c) Algorithm 10.3, modularity = 0.84.

Fig. 10.6 Learned 9-component graphs of currencies

graph learned by Algorithm 10.3 (Fig. 10.6c) presents a finer structure and a higher modularity value than those learned by SGL (Fig. 10.6a) and CLR (Fig. 10.6b). In addition, the learned graph in Fig. 10.6c presents more reasonable clusters such as {New Zealand, Australia} and {Poland, the Czech Republic, Hungary}, which are not separated in the Gaussian-based graphs. More critically, we can observe that SGL and CLR allow the existence of isolated nodes in the learned graphs. Algorithm 10.3 avoids such solutions by imposing linear constraints on the degree of the graph.

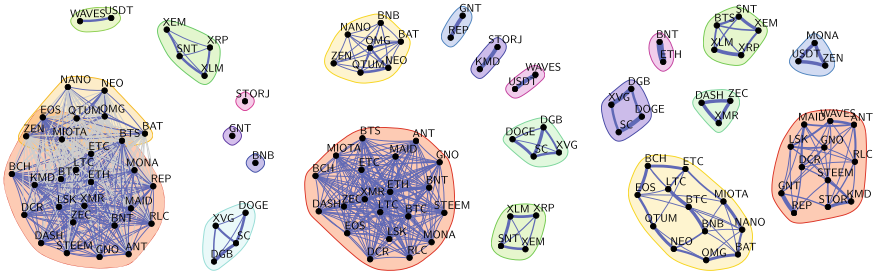
10.5.3 Communities in Cryptocurrencies

We query daily prices of the $p = 41$ most traded cryptocurrencies during the period starting from August 1, 2017, to December 1, 2020, which amounts to $n = 1218$ observations. The degrees of freedom during this time frame was measured as $\nu \approx 3$, which is tantamount to a strong heavy-tail scenario.



(a) GLE, modularity = 0.19. (b) NGL, modularity = 0.40. (c) Algorithm 10.2, modularity = 0.52.

Fig. 10.7 Learned connected graphs of cryptocurrencies

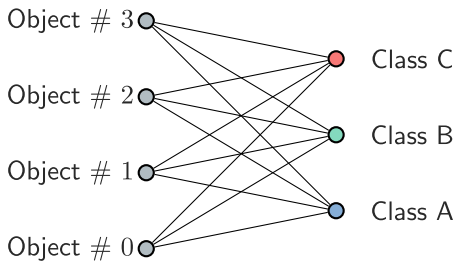


(a) SGL, modularity = 0.36 (b) CLR, modularity = 0.66. (c) Algorithm 10.3, modularity = 0.79.

Fig. 10.8 Learned 7-component graphs of cryptocurrencies

Figure 10.7 shows the learned graphs by GLE, NGL, and Algorithm 10.2. While the graphs in Fig. 10.7a and b present a small modularity value and contain a large number of edges, which impairs interpretability, the resulting graph in Fig. 10.7c reveals a refined representation of the interactions between pairs of cryptocurrencies, which is possibly more aligned with the actual market scenario. As an example, the link between Bitcoin (BTC) and Litecoin (LTC), a Bitcoin spinoff established in 2011, is substantially more evident in Fig. 10.7c.

Figure 10.8 shows the learned 7-component graphs of cryptocurrencies during the aforementioned time window. As in the previous experiments with foreign exchange data, SGL shows isolated nodes in the learned graph (Fig. 10.8a) and CLR contains a large number of spurious connections in the main cluster (Fig. 10.8b), whereas the graph learned via Algorithm 10.3 (Fig. 10.8c) has the largest modularity value. Interestingly, while all three methods agree to cluster {Dogecoin (DOGE), Verge (XVG), Siacoin (SC), and DigiByteCoin (DGB)}, which may be related to their similar initial release dates, only Algorithm 10.3 clusters together the coins that mainly focus on privacy and anonymity features, i.e., {Monero (XMR), Zcash (ZEC), DASH}.



(a) Connected (single-component) bipartite graph.

Fig. 10.9 Bipartite graph illustrating the modeling of dependencies between a collection of objects and their classes. The edges correspond to nonzero elements of $\mathbf{W}$

10.6 Learning Bipartite Graphs

In this section, we focus our attention on learning a special class of graphs: bipartite graphs. An undirected, weighted, bipartite graph can be defined as a 4-tuple $\mathcal{G} \triangleq \{\mathcal{V}_r, \mathcal{V}_q, \mathcal{E}, \mathbf{W}\}$, 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 to a group of objects and classes, respectively, where $p \triangleq r+q$ is the total number of nodes, and we assume that $r \gg q$, i.e., there exist many more objects than classes; $\mathcal{E} \subseteq \{\{u, v\} : u \in \mathcal{V}_r, v \in \mathcal{V}_q\}$ is the edge set, that is, a subset of the set of all possible unordered pairs of nodes such that $\{u, v\} \in \mathcal{E}$ iff nodes u and v are connected; $\mathbf{W} \in \mathbb{R}_+^{p \times p}$ 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 weighted Laplacian matrix of a graph is defined as $\mathbf{L} \triangleq \mathsf{Diag}(\mathbf{W1}) - \mathbf{W}$. Figure 10.9 displays an instance of this model.

The Laplacian matrix $\mathbf{L}$ of a bipartite graph can be written as

$$\mathbf{L} = \begin{bmatrix} \mathsf{Diag}\left(\mathbf{B1}_q\right) & -\mathbf{B} \\ -\mathbf{B}^\top & \mathsf{Diag}\left(\mathbf{B}^\top \mathbf{1}_r\right) \end{bmatrix}, \tag{10.33}$$

where $\mathbf{B} \in \mathbb{R}_+^{r \times q}$ contains the edge weights between the nodes of objects and the nodes of classes. Note that $\mathbf{L}$ satisfies $\mathbf{L1} = \mathbf{0}$ and $L_{ij} = L_{ji} \leq 0 \; \forall \, i \neq j$ by definition.

In what follows, we briefly discuss the two main approaches proposed in the literature to tackle the estimation of bipartite graphs. They basically follow two avenues: (*i*) exploiting the bipartite structure via (10.33) as in [48] and (*ii*) exploiting the structure of the spectral properties of graph matrices as in [39].

Bipartite Structure. The authors in [50] proposed the following optimization problem to learn a k-component bipartite graph from a given bipartite graph weights $\mathbf{A} \in \mathbb{R}^{r \times q}$. More precisely, their estimator is obtained as the solution of the following optimization problem:

$$\begin{aligned} &\underset{\mathbf{B}, \mathbf{V} \in \mathbb{R}^{p \times k}}{\mathsf{minimize}} \;\; \|\mathbf{B} - \mathbf{A}\|_{\mathrm{F}}^2 + \eta \mathrm{tr}\left(\mathbf{V}^\top \mathbf{L} \mathbf{V}\right), \\ &\mathsf{subject\ to} \;\; \mathbf{B} \geq \mathbf{0}, \; \mathbf{B1}_q = \mathbf{1}_r, \; \mathbf{V}^\top \mathbf{V} = \mathbf{I}_k, \end{aligned} \tag{10.34}$$

where $\mathbf{L}$ depends on $\mathbf{B}$ through (10.33), $\eta > 0$ is a hyperparameter that promotes the rank of $\mathbf{L}$ to be $p - k$, and $\mathbf{A}$ can be constructed from the correlation between nodes of objects and classes.

An alternating minimization approach was proposed in [50] to solve Problem (10.34). For a fixed value of $\mathbf{V}$, it reduces to a constrained quadratic program, which can be solved efficiently via standard quadratic programming solvers. On the other hand, for a fixed $\mathbf{B}$, the optimal solution of $\mathbf{V}$ are the k eigenvectors of $\mathbf{L}$ associated to its smallest eigenvalues. A shortcoming of this formulation is it lacks statistical support, which makes it difficult to draw statistical conclusions about the learned graph as well as to consider this model under different statistical assumptions.

Spectral Regularization. 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 graphs [38, 39, 48–50]. Let $\mathbf{A} = \mathbf{U}\mathsf{Diag}(\boldsymbol{\psi})\mathbf{U}^\top$ and $\mathbf{L} = \mathbf{V}\mathsf{Diag}(\boldsymbol{\lambda})\mathbf{V}^\top$ be the spectral decomposition of the adjacency and Laplacian matrices, respectively. For a k-component graph, the eigenvalues of $\mathbf{L}$ belong to the set

$$C_{\boldsymbol{\lambda}} = \left\{\boldsymbol{\lambda} \in \mathbb{R}^p : \{\lambda_i = 0\}_{i=1}^k, c_1 \leq \lambda_{k+1} \leq \lambda_{k+2} \leq \cdots \leq \lambda_p \leq c_2\right\},$$

where c_1 and c_2 are positive constants. On the other hand, the adjacency matrix of a bipartite graph satisfies the following spectral properties [28]: (i) $\boldsymbol{\psi}$ contains exactly $r - q$ zero elements and (ii) $\boldsymbol{\psi}$ is anti-symmetric, i.e., $\boldsymbol{\psi} \in C_{\boldsymbol{\psi}}$, where

$$C_{\boldsymbol{\psi}} = \left\{\boldsymbol{\psi} \in \mathbb{R}^p : \psi_i = -\psi_{p+1-i},\ i = 1, 2, \ldots, p, \psi_1 \geq \psi_2 \geq \cdots \geq \psi_p\right\}.$$

By combining these facts, the authors in [39] proposed the following optimization program to learn a k-component bipartite graph based on the GMRF framework:

$$\begin{aligned} \underset{\mathbf{w}\geq\mathbf{0},\mathbf{V},\mathbf{U},\boldsymbol{\psi},\boldsymbol{\lambda}}{\text{minimize}}\ & \operatorname{tr}(\mathcal{L}\mathbf{w}\mathbf{S}) - \log\det{}^*(\mathcal{L}\mathbf{w}) + \tfrac{\gamma}{2}\left\|\mathcal{A}\mathbf{w} - \mathbf{U}\mathsf{Diag}(\boldsymbol{\psi})\mathbf{U}^\top\right\|_{\mathrm{F}}^2 \\ & + \tfrac{\beta}{2}\left\|\mathcal{L}\mathbf{w} - \mathbf{V}\mathsf{Diag}(\boldsymbol{\lambda})\mathbf{V}^\top\right\|_{\mathrm{F}}^2, \\ \text{subject to}\ & \mathbf{U}^\top\mathbf{U} = \mathbf{I},\ \mathbf{U} \in \mathbb{R}^{p\times p},\ \psi \in C_\psi,\ \mathbf{V}^\top\mathbf{V} = \mathbf{I},\ \mathbf{V} \in \mathbb{R}^{p\times p},\ \boldsymbol{\lambda} \in C_{\boldsymbol{\lambda}}, \end{aligned} \tag{10.35}$$

where $\mathcal{L}$ and $\mathcal{A}$ are the Laplacian and adjacency operators and $\mathbf{w}$ is the vector of graph weights [39]. Although this model is statistically sound, we note that this approach does not always guarantee that $\mathcal{A}\mathbf{w}^\star$ is a bipartite graph because the equality $\mathcal{A}\mathbf{w}^\star = \mathbf{U}^\star\mathsf{Diag}(\boldsymbol{\psi}^\star)\mathbf{U}^{\star\top}$ is often not met, further requiring postprocessing of $\mathcal{A}\mathbf{w}^\star$, which is often problematic in practical settings. In addition, the lack of constraints on the node degrees allows trivial solutions with isolated nodes, which is also undesired in practice. The method in [39] does not require knowledge of the partition of the node set; however, it does assume that the number of nodes in each set is known.

10.6.1 Gaussian Bipartite Graphs

In what follows, we discuss methods and algorithms for learning bipartite graphs, starting off with the simple case of a connected bipartite graph under Gaussian settings. It is important to look at the Gaussian scenario first, as this case will serve as a building block for more elaborate formulations that include additional statistical assumptions, such as heavy tails, and its extension to k-component bipartite graphs.

Noting that the adjacency matrix of a bipartite graph can be partitioned as $\mathbf{W} = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0} \end{bmatrix}$ [28], where $\mathbf{B} \in \mathbb{R}_+^{r \times q}$, we can formulate its MLE as the following convex optimization program:

$$\begin{aligned} \underset{\mathbf{L}, \mathbf{B} \geq \mathbf{0}}{\text{minimize}} \quad & \operatorname{tr}(\mathbf{LS}) - \log\det(\mathbf{L} + \mathbf{J}), \\ \text{subject to} \quad & \mathbf{L} = \begin{bmatrix} \mathsf{Diag}(\mathbf{B}\mathbf{1}_q) & -\mathbf{B} \\ -\mathbf{B}^\top & \mathsf{Diag}(\mathbf{B}^\top \mathbf{1}_r) \end{bmatrix}, \end{aligned} \tag{10.36}$$

where we have used the fact that, for a connected graph, $\det^*(\mathbf{L}) = \det(\mathbf{L} + \mathbf{J})$, $\mathbf{J} \triangleq \frac{1}{p}\mathbf{1}\mathbf{1}^\top$ [21].

We can reformulate Problem (10.36) by taking advantage of the symmetry of the similarity matrix $\mathbf{S}$, i.e., $\begin{bmatrix} \mathbf{S}_{rr} & \mathbf{S}_{rq} \\ \mathbf{S}_{rq}^\top & \mathbf{S}_{qq} \end{bmatrix}$, where $\mathbf{S}_{rq} \in \mathbb{R}^{r \times q}$ contains the pairwise similarities between the nodes in the set of objects $\mathcal{V}_r$ and the nodes in the set of classes $\mathcal{V}_q$. In particular, we have

$$\operatorname{tr}(\mathbf{LS}) = \mathsf{diag}(\mathbf{S})^\top \begin{bmatrix} \mathbf{B}\mathbf{1}_q \\ \mathbf{B}^\top \mathbf{1}_r \end{bmatrix} - 2\operatorname{tr}(\mathbf{B}\mathbf{S}_{rq}^\top) = \operatorname{tr}\left(\mathbf{B}\left(\mathbf{1}_q \mathbf{s}_{1:r}^\top + \mathbf{s}_{r+1:p}\mathbf{1}_r^\top - 2\mathbf{S}_{rq}^\top\right)\right), \tag{10.37}$$

where $\mathbf{s} \triangleq \mathsf{diag}(\mathbf{S})$.

Plugging (10.33) and (10.37) in (10.36), we arrive at the following formulation to learn a connected bipartite graph:

$$\begin{aligned} \underset{\mathbf{B} \geq \mathbf{0}}{\text{minimize}} \; & \operatorname{tr}\left(\mathbf{B}\left(\mathbf{1}_q \mathbf{s}_{1:r}^\top + \mathbf{s}_{r+1:p}\mathbf{1}_r^\top - 2\mathbf{S}_{rq}^\top\right)\right) \\ & - \log\det\left(\begin{bmatrix} \mathsf{Diag}(\mathbf{B}\mathbf{1}_q) + \mathbf{J}_{rr} & -\mathbf{B} + \mathbf{J}_{rq} \\ -\mathbf{B}^\top + \mathbf{J}_{qr} & \mathsf{Diag}(\mathbf{B}^\top \mathbf{1}_r) + \mathbf{J}_{qq} \end{bmatrix}\right). \end{aligned} \tag{10.38}$$

We discuss an optimization algorithm based on the PGD framework [2, 3] that amounts to two steps: (1) computation of a descent update and (2) projection onto the feasible set.

A common way to compute a descent direction is using the gradient of the objective function. The gradient of the objective function in Problem (10.38) would involve the computation of the inverse of the matrix inside the log-determinant term, which in general costs $O(p^3)$. By leveraging the block structure of the Laplacian matrix of

bipartite graphs, and using the classical matrix determinant lemma for block matrices that relates the determinant of a block matrix with its Schur complement [78, p. 4], we have the following equality:

$$\log\det\left(\begin{bmatrix}\mathsf{Diag}(\mathbf{B}\mathbf{1}_q)+\mathbf{J}_{rr} & -\mathbf{B}+\mathbf{J}_{rq}\\ -\mathbf{B}^\top+\mathbf{J}_{qr} & \mathsf{Diag}(\mathbf{B}^\top\mathbf{1}_r)+\mathbf{J}_{qq}\end{bmatrix}\right)=\log\det\left(\mathsf{Diag}(\mathbf{B}\mathbf{1}_q)+\mathbf{J}_{rr}\right)+$$
$$\log\det\left(\mathsf{Diag}(\mathbf{B}^\top\mathbf{1}_r)+\mathbf{J}_{qq}-(-\mathbf{B}^\top+\mathbf{J}_{qr})(\mathsf{Diag}(\mathbf{B}\mathbf{1}_q)+\mathbf{J}_{rr})^{-1}(-\mathbf{B}+\mathbf{J}_{rq})\right). \tag{10.39}$$

In practice, we also would like to normalize the degrees of the nodes in the objects group, so that the edge weights can be directly interpreted as the degree of membership of an object to a particular class. Mathematically, this can be achieved by a simple linear constraint $\mathbf{B}\mathbf{1}_q=\mathbf{1}_r$.

Plugging the above constraint and equality (10.39) into (10.38), we have the following formulation:

$$\underset{\mathbf{B}\geq\mathbf{0},\mathbf{B}\mathbf{1}_q=\mathbf{1}_r}{\text{minimize}}\ \mathrm{tr}\,(\mathbf{B}\mathbf{C})-\log\det\left(\mathsf{Diag}(\mathbf{B}^\top\mathbf{1}_r)+\mathbf{J}_{qq}-(\mathbf{B}-\mathbf{J}_{rq})^\top(\mathbf{I}_r+\mathbf{J}_{rr})^{-1}(\mathbf{B}-\mathbf{J}_{rq})\right), \tag{10.40}$$

where $\mathbf{C}\triangleq\left(\mathbf{s}_{r+1:p}\mathbf{1}_r^\top-2\mathbf{S}_{rq}^\top\right)$.

Let $f(\mathbf{B})\triangleq-\log\det(g(\mathbf{B}))+\mathrm{tr}\,(\mathbf{B}\mathbf{C})$ be the objective function of Problem (10.40), where $g(\mathbf{B})\triangleq\mathsf{Diag}(\mathbf{B}^\top\mathbf{1}_r)+\mathbf{J}_{qq}-(\mathbf{B}-\mathbf{J}_{rq})^\top(\mathbf{I}_r+\mathbf{J}_{rr})^{-1}(\mathbf{B}-\mathbf{J}_{rq})$. Then the gradient of $f(\mathbf{B})$ can be computed as

$$\nabla f(\mathbf{B})=\mathbf{1}_r\left[\mathsf{diag}\left(-g(\mathbf{B})^{-1}\right)\right]^\top-2(\mathbf{I}_r+\mathbf{J}_{rr})^{-1}(\mathbf{B}-\mathbf{J}_{rq})\left(-g(\mathbf{B})^{-1}\right)^\top+\mathbf{C}^\top. \tag{10.41}$$

The PGD update is formulated as

$$\mathbf{B}^{l+1}=\underset{\mathbf{B}\geq\mathbf{0},\mathbf{B}\mathbf{1}_q=\mathbf{1}_r}{\arg\min}\ \left\|\mathbf{B}-\left(\mathbf{B}^l-\alpha_l\nabla f(\mathbf{B}^l)\right)\right\|_{\mathrm{F}}^2\triangleq P_\triangle\left(\mathbf{B}^l-\alpha_l\nabla f(\mathbf{B}^l)\right). \tag{10.42}$$

Problem (10.42) is an Euclidean projection of the rows of $\mathbf{B}^l-\alpha_l\nabla f(\mathbf{B}^l)$ onto the probability simplex. The unique solution to Problem (10.42) can be found efficiently via several algorithms [4, 45, 53] whose theoretical worst-case complexity is $O(rq^2)$ but their observed practical complexity is $O(rq)$ [13].

Finally, to validate the point $\mathbf{B}^{l+1}$ and to adaptively update the learning rate α_l, we check the following backtracking condition [2]:

$$f\left(\mathbf{B}^{l+1}\right)\leq f\left(\mathbf{B}^l\right)+\left\langle\nabla f\left(\mathbf{B}^l\right),\mathbf{B}^{l+1}-\mathbf{B}^l\right\rangle+\frac{1}{2\alpha_l}\|\mathbf{B}^{l+1}-\mathbf{B}^l\|_{\mathrm{F}}^2. \tag{10.43}$$

In practice, we decrease the learning rate $\alpha_l\in(0,1)$ until condition (10.43) is satisfied or convergence has been achieved. Algorithm 10.4 summarizes the scheme for learning a bipartite graph from a Gaussian MRF assumption. Since the objective

function of Problem (10.40) is convex and its feasible set is compact, Algorithm 10.4 converges to a global minimum [3].

Algorithm 10.4 Gaussian bipartite graph learning (GBG)

Data: Similarity matrix $\mathbf{S}$, initial feasible estimate of the graph weights $\mathbf{B}^0$, initial learning rate $\alpha_0 > 0$, tolerance $\epsilon > 0$.
Result: Optimal bipartite graph: $\mathbf{B}^\star$

```
while l ≤ maxiter do
    ▷ B^{l+1} ← P_△(B^l − α_l ∇f(B^l)), where α_l is chosen such that (10.43) is satisfied
    if ‖B^{l+1} − B^l‖_F / ‖B^l‖_F ≤ ϵ then
        return B^{l+1}
    end
end
```

10.6.2 Student's t Bipartite Graphs

Outliers and heavy-tailed events are pervasive in modern datasets [55]. To account for this phenomenon, instead of assuming a Gaussian generative process, as done in [39] and in Problem (10.36), we assume the data-generating process to be modeled by a multivariate zero-mean Student's t-distribution, whose probability density function can be written as

$$p(\mathbf{x}) \propto \sqrt{\det{}^*(\boldsymbol{\Theta})} \left(1 + \frac{\mathbf{x}^\top \boldsymbol{\Theta} \mathbf{x}}{\nu}\right)^{-(\nu+p)/2}, \tag{10.44}$$

where $\boldsymbol{\Theta}$ is a positive-semidefinite inverse scatter matrix modeled as a combinatorial graph Laplacian matrix, and $\nu > 2$ is the number of degrees of freedom that controls the rate of decay of the tails. More precisely, as $\nu \to \infty$ the tails of (10.44) approach those of a Gaussian distribution.

This results in a robustified version of the MLE for connected graph learning, i.e.,

$$\begin{aligned} &\underset{\mathbf{L},\mathbf{B}}{\text{minimize}} \quad -\log\det(\mathbf{L}+\mathbf{J}) + \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\tfrac{1}{\nu}\mathbf{x}_i^\top\mathbf{L}\mathbf{x}_i\right), \\ &\text{subject to } \mathbf{L} = \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right) \end{bmatrix}, \ \mathbf{B} \geq \mathbf{0}, \ \mathbf{B}\mathbf{1}_q = \mathbf{1}_r. \end{aligned} \tag{10.45}$$

We can further simplify the objective function of Problem (10.45) by noting that

$$\mathbf{x}_i^\top \mathbf{L}\mathbf{x}_i = \mathbf{x}_i^\top \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right) \end{bmatrix} \mathbf{x}_i = h_i + \operatorname{tr}\left(\mathbf{B}\mathbf{G}_i\right), \tag{10.46}$$

where $\mathbf{G}_i \triangleq \operatorname{diag}(\mathbf{x}_i\mathbf{x}_i^\top)_{r+1:p}\mathbf{1}_r^\top - 2\left(\mathbf{x}_i\mathbf{x}_i^\top\right)_{rq}^\top$ and $h_i \triangleq \mathbf{1}_r^\top \operatorname{diag}(\mathbf{x}_i\mathbf{x}_i^\top)_{1:r}$.

Plugging (10.39) and (10.46) into (10.45), we have the following optimization program to learn a heavy-tailed bipartite graph:

$$\begin{aligned}\underset{\mathbf{B}\geq\mathbf{0},\ \mathbf{B}\mathbf{1}_q=\mathbf{1}_r}{\text{minimize}}\ & -\log\det\left(\mathsf{Diag}(\mathbf{B}^\top\mathbf{1}_r)+\mathbf{J}_{qq}-(\mathbf{B}-\mathbf{J}_{rq})^\top(\mathbf{I}_r+\mathbf{J}_{rr})^{-1}(\mathbf{B}-\mathbf{J}_{rq})\right)\\ &+\frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{h_i+\mathrm{tr}\,(\mathbf{B}\mathbf{G}_i)}{\nu}\right).\end{aligned} \tag{10.47}$$

Problem (10.47) is, in general, nonconvex due to the summation over concave terms and hence it is difficult to be dealt with directly. To tackle this issue, we leverage the MM framework that generates a sequence of updates, where in each iteration an upper-bounded subproblem is solved [64, 65].

To find such upper-bound, we use the fact that the logarithm term in Problem (10.47) can be globally upper-bounded by its first-order Taylor expansion, i.e., for all $a\geq 0, t\geq 0, b>0$, we have $\log\left(1+\frac{t}{b}\right)\leq\log\left(1+\frac{a}{b}\right)+\frac{t-a}{a+b}$, hence, at a point $\mathbf{B}^j$, we have that

$$\log\left(1+\frac{h_i+\mathrm{tr}\,(\mathbf{B}\mathbf{G}_i)}{\nu}\right)\leq\frac{\mathrm{tr}\,(\mathbf{B}\mathbf{G}_i)}{\nu+h_i+\mathrm{tr}\left(\mathbf{B}^j\mathbf{G}_i\right)}+c,$$

where $c\triangleq\log\left(1+\frac{h_i+\mathrm{tr}\left(\mathbf{B}^j\mathbf{G}_i\right)}{\nu}\right)-\frac{\mathrm{tr}\left(\mathbf{B}^j\mathbf{G}_i\right)}{\nu+h_i+\mathrm{tr}\left(\mathbf{B}^j\mathbf{G}_i\right)}$ is a constant that does not depend on $\mathbf{B}$.

Hence, to find an update $\mathbf{B}^{j+1}$, we iterate the solution of the following upper-bounded convex subproblem:

$$\begin{aligned}\mathbf{B}^{j+1}=\ &\underset{\mathbf{B}}{\arg\min}\quad -\log\det\left(\mathsf{Diag}(\mathbf{B}^\top\mathbf{1}_r)+\mathbf{J}_{qq}-(\mathbf{B}-\mathbf{J}_{rq})^\top(\mathbf{I}_r+\mathbf{J}_{rr})^{-1}(\mathbf{B}-\mathbf{J}_{rq})\right)\\ &\qquad\quad +\mathrm{tr}\left(\mathbf{B}\mathbf{M}^j\right),\\ &\text{subject to}\ \ \mathbf{B}\geq\mathbf{0},\ \mathbf{B}\mathbf{1}_q=\mathbf{1}_r,\end{aligned} \tag{10.48}$$

where $\mathbf{M}^j\triangleq\frac{p+\nu}{n}\sum_{i=1}^{n}\frac{\mathbf{G}_i}{\nu+h_i+\mathrm{tr}\left(\mathbf{B}^j\mathbf{G}_i\right)}$. We observe in practice that a few (≈ 5) iterations are sufficient for convergence.

Problem (10.48) has the same format as Problem (10.40) and it can be readily solved by Algorithm 10.4. The learning scheme for Student's t bipartite graphs is summarized in Algorithm 10.5. The sequence $\left\{\mathbf{B}^l\right\}$ generated by Algorithm 10.5 converges to the set of stationary points of Problem (10.47). This result directly follows from the convergence results of the MM framework in [65].

Algorithm 10.5 Student-t bipartite graph learning (SBG)

Data: Data matrix $\mathbf{X}$, initial feasible estimate of the graph weights $\mathbf{B}^0$, initial learning rate $\alpha_0 > 0$, tolerance $\epsilon > 0$, degree of freedoms $\nu > 2$.
Result: Learned bipartite graph: $\mathbf{B}^{l+1}$

```
while l ≤ maxiter do
    ▷ update B^{l+1} by iterating (10.48)
    if ‖B^{l+1} − B^l‖_F / ‖B^l‖_F ≤ ϵ then
        return B^{l+1}
    end
end
```

10.6.3 k-Component Student's t Bipartite Graphs

Graphs with multiple components can be learned by introducing rank constraints on the Laplacian matrix [39, 49, 50]. More precisely, the number of zero eigenvalues of the Laplacian matrix amounts to the number of graph components, i.e., $\mathsf{rank}(\mathbf{L}) = p - k$, where k is the number of components.

The approach taken by [50] approximates the rank constraint by applying Fan's theorem [23] as follows:

$$\sum_{i=1}^{k} \lambda_i \left(\mathbf{L}\right) = \underset{\mathbf{V}\in\mathbb{R}^{p\times k}, \mathbf{V}^\top\mathbf{V}=\mathbf{I}_k}{\mathsf{minimize}} \ \mathrm{tr}\left(\mathbf{V}^\top\mathbf{L}\mathbf{V}\right), \tag{10.49}$$

where the right-hand side of (10.49) can be used as a regularization term (*cf.* Problem (10.34)). On the other hand, the authors in [39] introduced the quadratic relaxation $\left\|\mathbf{L} - \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top\right\|_{\mathrm{F}}^2$. Those formulations [39, 50] can be conveniently solved by coordinate descent methods. However, they introduce additional variables to be optimized, extra hyperparameters to be tuned, and do not guarantee that the rank constraint on $\mathbf{L}$ is satisfied.

In the method presented here, we directly apply the rank constraint into the optimization formulation. More precisely, using the Student's t-case, we formulate the following optimization problem to learn a k-component bipartite graph:

$$\begin{aligned}
&\underset{\mathbf{L}\geq\mathbf{0},\mathbf{B}}{\mathsf{minimize}} \ \ \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{h_i+\mathrm{tr}\left(\mathbf{B}\mathbf{G}_i\right)}{\nu}\right) - \log\det{}^*\left(\mathbf{L}\right), \\
&\mathsf{subject\ to}\ \mathbf{L} = \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \mathsf{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right) \end{bmatrix}, \ \mathsf{rank}(\mathbf{L}) = p-k, \ \mathbf{B}\geq\mathbf{0}, \mathbf{B}\mathbf{1}_q = \mathbf{1}_r.
\end{aligned} \tag{10.50}$$

Unlike the case for connected bipartite graphs in Problem (10.38), the pseudo-determinant term in (10.50) cannot be simplified through the block matrix determinant lemma [78]. Therefore, we design an iterative algorithm based on the ADMM framework [5] for Problem (10.50).

10.6.3.1 ADMM Solution

The augmented Lagrangian of Problem (10.50) can be written as

$$L_\rho(\mathbf{L},\mathbf{B},\mathbf{Y}) = \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{h_i+\operatorname{tr}(\mathbf{B}\mathbf{G}_i)}{\nu}\right) - \log\det{}^*(\mathbf{L}) + \left\langle \mathbf{L} - \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right)\end{bmatrix}, \mathbf{Y}\right\rangle + \frac{\rho}{2}\left\|\mathbf{L} - \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right)\end{bmatrix}\right\|_{\mathrm{F}}^2, \quad (10.51)$$

where $\rho > 0$ is a penalty hyperparameter.

For fixed $\mathbf{B}$ and $\mathbf{Y}$, the subproblem for $\mathbf{L}$ can be written as

$$\mathbf{L}^\star = \underset{\operatorname{rank}(\mathbf{L})=p-k}{\arg\min} -\log\det{}^*(\mathbf{L}) + \langle \mathbf{L},\mathbf{Y}\rangle + \frac{\rho}{2}\left\|\mathbf{L} - \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right)\end{bmatrix}\right\|_{\mathrm{F}}^2, \quad (10.52)$$

whose closed-form solution is given by Lemma 10.3.

Lemma 10.3 *The global minimizer of Problem* (10.52) *is [16, 69]*

$$\mathbf{L}^\star = \frac{1}{2\rho}\mathbf{R}\left(\mathbf{\Gamma} + \sqrt{\mathbf{\Gamma}^2 + 4\rho\mathbf{I}}\right)\mathbf{R}^\top, \quad (10.53)$$

where $\mathbf{R}\mathbf{\Gamma}\mathbf{R}^\top$ *is the eigenvalue decomposition of* $\rho\begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right)\end{bmatrix} - \mathbf{Y}$, *with* Γ *having the largest* $p-k$ *eigenvalues along its diagonal and* $\mathbf{R} \in \mathbb{R}^{p\times(p-k)}$ *containing the corresponding eigenvectors.*

Fixing $\mathbf{L}$ and $\mathbf{Y}$, the subproblem for $\mathbf{B}$ is

$$\mathbf{B}^\star = \underset{\mathbf{B}\geq\mathbf{0},\mathbf{B}\mathbf{1}=\mathbf{1}}{\arg\min} \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{h_i+\operatorname{tr}(\mathbf{B}\mathbf{G}_i)}{\nu}\right) + \frac{\rho}{2}\left\|\mathbf{L} - \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right)\end{bmatrix}\right\|_{\mathrm{F}}^2 - \left\langle \begin{bmatrix} \mathbf{I}_r & -\mathbf{B} \\ -\mathbf{B}^\top & \operatorname{Diag}\left(\mathbf{B}^\top\mathbf{1}_r\right)\end{bmatrix}, \mathbf{Y}\right\rangle, \quad (10.54)$$

which can be simplified as

$$\mathbf{B}^\star = \underset{\mathbf{B}\geq\mathbf{0},\mathbf{B}\mathbf{1}=\mathbf{1}}{\arg\min} \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{h_i+\operatorname{tr}(\mathbf{B}\mathbf{G}_i)}{\nu}\right) + \operatorname{tr}(\mathbf{B}\mathbf{H}) + \rho\|\mathbf{B}\|_{\mathrm{F}}^2 + \frac{\rho}{2}\mathbf{1}_r^\top\mathbf{B}\mathbf{B}^\top\mathbf{1}_r, \quad (10.55)$$

where $\mathbf{H} \triangleq 2\mathbf{Y}_{rq}^\top - \mathbf{y}_{r+1:p}\mathbf{1}_r^\top - \rho\left(\mathbf{l}_{r+1:p}\mathbf{1}_r^\top - 2\mathbf{L}_{rq}^\top\right)$, $\mathbf{y}_{r+1:p} \triangleq \operatorname{diag}(\mathbf{Y})_{r+1:p}$, and $\mathbf{l}_{r+1:p} \triangleq \operatorname{diag}(\mathbf{L})_{r+1:p}$.

Leveraging the MM framework to Problem (10.55), like it was applied to Problem (10.47), we can find an upper-bounded subproblem at point $\mathbf{B}^j$ as follows:

$$\mathbf{B}^{j+1} = \underset{\mathbf{B}\geq\mathbf{0},\mathbf{B}\mathbf{1}=\mathbf{1}}{\arg\min}\ \mathrm{tr}\left(\mathbf{B}\left(\mathbf{H}+\mathbf{M}^j\right)\right) + \rho\left\|\mathbf{B}\right\|_{\mathrm{F}}^2 + \frac{\rho}{2}\mathbf{1}_r^\top\mathbf{B}\mathbf{B}^\top\mathbf{1}_r. \tag{10.56}$$

Subproblem (10.56) is strongly convex, thus it can be solved efficiently via convex solvers. Finally, to find an update $\mathbf{B}^\star$, we iterate the solution of Problem (10.56). The dual variable $\mathbf{Y}$ is updated approximately via gradient ascent.

Algorithm 10.6 summarizes the discussed scheme to learn k-component bipartite graphs.

Algorithm 10.6 Student-t k-component bipartite graph learning (kSBG)

Data: Data matrix $\mathbf{X}$, number of components k, initial feasible estimate of the graph weights $\mathbf{B}^0$, penalty hyperparameter $\rho > 0$, tolerance $\epsilon > 0$, degrees of freedom $\nu > 2$.
Result: Learned bipartite graph: $\mathbf{B}^{l+1}$

while $l \leq$ maxiter **do**
  ▷ update $\mathbf{L}^{l+1}$ via (10.53)
  ▷ update $\mathbf{B}^{l+1}$ by iterating (10.56)
  ▷ update $\mathbf{Y}^{l+1} = \mathbf{Y}^l - \rho\left(\mathbf{L}^{l+1} - \begin{bmatrix}\mathbf{I}_r & -\mathbf{B}^{l+1} \\ -\mathbf{B}^{l+1\top} & \mathrm{Diag}\left(\mathbf{B}^{l+1\top}\mathbf{1}_r\right)\end{bmatrix}\right)$
  if $\|\mathbf{B}^{l+1} - \mathbf{B}^l\|_{\mathrm{F}}/\|\mathbf{B}^l\|_{\mathrm{F}} \leq \epsilon$ **then**
    return $\mathbf{B}^{l+1}$
  end
end

10.6.4 Experimental Results

We conduct experiments to illustrate the performance of the discussed algorithms in terms of quantitative measures such as node label accuracy and graph modularity. Accuracy is computed as the ratio between the number of correctly predicted node labels and the number of nodes in the objects set, whereas modularity measures the strength of division of a graph into groups [47]. A high modularity value means that the nodes from the same group are more likely to be connected.

Benchmarks: We compare the presented algorithms with state-of-the-art methods in two settings: (*i*) connected bipartite graphs, where SOBG ($k = 1$) [50] and SGA [39] are the benchmarks, and (*ii*) k-component bipartite graphs, where SOBG ($k > 1$) and SGLA [39] act as benchmarks. In our Algorithm 10.6, we set the hyperparameter $\rho = 1$ and the relative tolerance $\epsilon = 10^{-5}$. The hyperparameter of SOBG was tunned according to the heuristic procedure described in [50]. The presented algorithms and benchmarks were implemented using the R programming language.

For SGA and SGLA, we relied on their official implementation via the package spectralGraphTopology [7].

10.6.4.1 Returns of S&P500 Stocks

We perform experiments using publicly available, historical daily prices, queried from Yahoo! Finance™, of $r = 333$ stocks listed in the S&P500 Index and $q = 8$ sectors indexes, namely: Consumer Staples, Energy, Financials, Health Care, Industrials, Materials, Real Estate, and Utilities. Following our notation, the node set of classes $\mathcal{V}_q$ represents the sectors indexes, whereas the stocks are represented by the node set of objects $\mathcal{V}_r$.

Stock sector labels are available through the Global Industry Classification Standard (GICS) [46, 61]. Arguably the most well-recognized, industry-standard stock classification system, GICS relies on a fixed classification structure that may not fully capture the actual impact of a particular stock at particular time. With the intent of solving this shortcoming, we employ the learned bipartite graphs to extend the current industry stock classification system from a static classification to a dynamic soft clustering, in which a particular node (stock) may belong to different clusters (sectors) with different degrees of membership that can change over time. Our goal is to perform soft clustering of stocks according to the market sectors. To that end, we recall that each row of $\mathbf{B}$ is a point belonging to the probability simplex in $\mathbb{R}^q$, which represents the level of membership of a stock to a particular set of sectors. The final sector assigned to the i-th stock corresponds to $\arg\max_{j\in 1,\dots,q} B_{ij}$.

We start by constructing the log-returns data matrix, i.e., a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, where n is the number of log-returns observations and p is the number of instruments, as $X_{i,j} = \log P_{i,j} - \log P_{i-1,j}$, where $P_{i,j}$ is the closing price of the j-th instrument at the i-th day. We collect data from January 5, 2016, to January 5, 2021, totaling $n = 1291$ daily observations. Then, we proceed to learn connected bipartite graphs via the benchmarks and discussed methods using the observations $\mathbf{X}$ in a rolling-window fashion.[5] More precisely, we use a window of length 504 d (2 years in terms of stock market days) and shift this window by 63 d (3 months in terms of stock market days).

Figure 10.10 shows quantitative measurements that reveal the higher performance of SBG algorithm both in terms of accuracy and modularity. That may be explained by the fact that SBG adopts a multivariate Student's t-distribution, which is well known to better model financial datasets in comparison to Gaussian assumptions.

Furthermore, we evaluate the performance of the k-component bipartite estimators under the same settings as previously described, where we set $k = 8$, i.e., k is the number of sectors. Figure 10.11 reveals that kSBG presents a better performance across the whole time period both in terms of accuracy and modularity when com-

[5] An estimate for the degree of freedoms ν, required by Algorithms 10.5 and 10.6, is obtained by fitting a univariate Student's t-distribution to the log-returns of the S&P500 index during the corresponding time period.

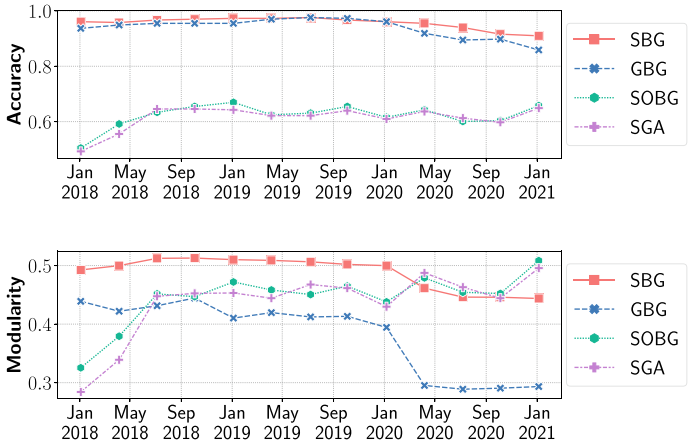


Fig. 10.10 Performance of the estimators for connected bipartite graphs of S&P500 stocks

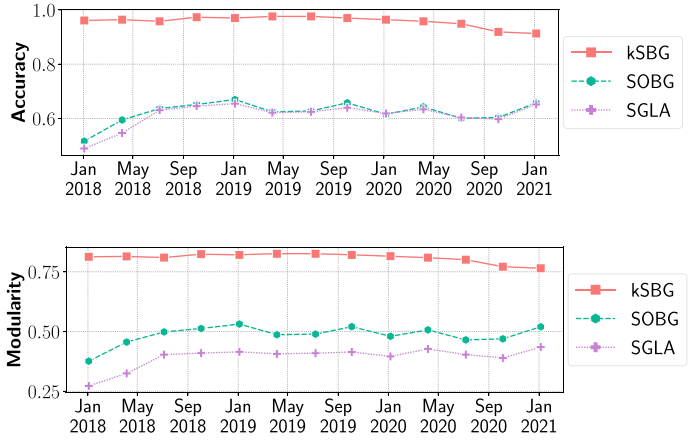
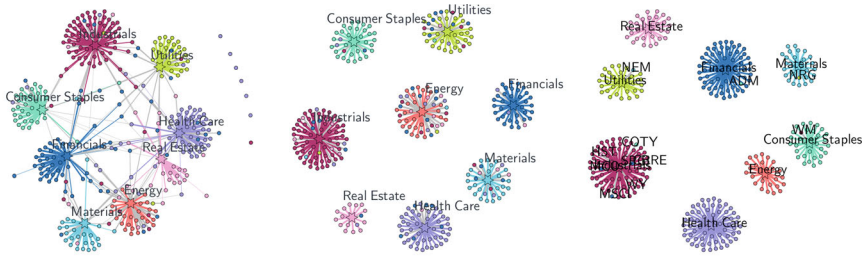


Fig. 10.11 Performance of the estimators for 8-component bipartite graphs of S&P500 stocks

pared to SGLA and SOBG. Interestingly, in both Figs. 10.10 and 10.11, we observe a decline in accuracy and modularity of the methods GBG, SBG, and kSBG, starting in January 2020, which can be explained by the impact of the COVID-19 pandemic on the US stock market. This reveals that our methods may be able to identify periods of economic turmoil and suggests that, during such times, standards such as GICS may not be the best reference for stock market sector classification. This insight may be critical for investors who rely on diversification of their portfolios through information on the sectors [74].

Figure 10.12 shows the k-component bipartite models learned using the whole dataset. We observe that SGLA allows the existence of isolated nodes, which degrades performance. The method kSBG avoids that issue by imposing a linear



(a) SGLA (Kumar et al., 2020). accuracy = 0.77, modularity = 0.56.

(b) SOBG (Nie et al., 2017). accuracy = 0.75, modularity = 0.61.

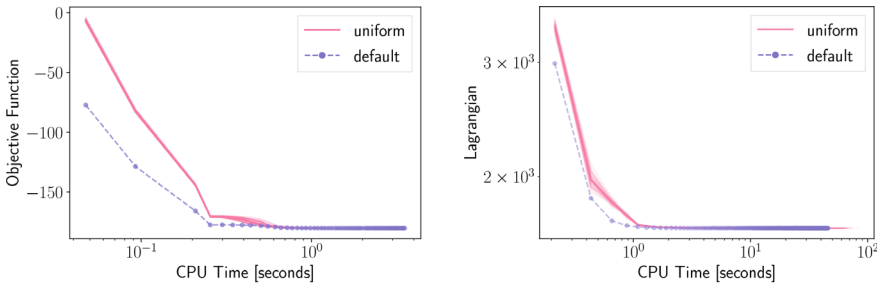
(c) kSBG. accuracy = 0.97, modularity = 0.82.

Fig. 10.12 Graphs learned by different algorithms in the S&P500 stock returns dataset

constraint on the node degrees. In addition, kSBG yields a more intuitive representation of the financial network than SOBG as verified by the accuracy and modularity values. Finally, we note that kSGD algorithm "misclassifies" a few nodes such as the company ADM, which is originally classified by GICS as a Consumer Staples company but our algorithm places it in the Financials sector. A closer look at the fundamentals of ADM reveals that it is an active commodities trading participant, which then justifies it being classified as a Financials company. This application showcases the benefits of using data-driven classification algorithms, such as the one discussed in this chapter, which can be used to inform investors that prioritize diversification of their portfolios [29].

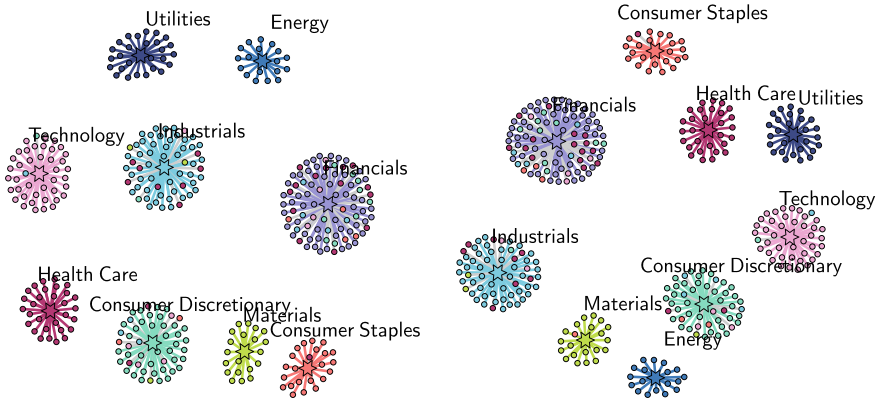
10.6.4.2 Robustness to Initialization

The performance of iterative algorithms may be affected by their initial point, especially when dealing with nonconvex problems as in Algorithms 10.5 and 10.6. Hence, we perform an experiment to compare two initialization schemes: (*i*) uniform, the graph weights are randomly sampled, i.e., $B_{ij}^0 \sim \mathsf{Uniform}(0, 1)$, and (*ii*) default, $\mathbf{B}^0 \propto \max\left(\mathbf{0}, -\boldsymbol{\Xi}_{rq}\right)$, where $\mathbf{S}^{-1} = \left[\boldsymbol{\Xi}_{rr}\ \boldsymbol{\Xi}_{rq}; \boldsymbol{\Xi}_{rq}^\top\ \boldsymbol{\Xi}_{qq}\right]$ is the inverse of the sample correlation matrix, as discussed in more details in [39]. In both schemes, we normalize the rows of $\mathbf{B}^0$ such that $\mathbf{B}^0\mathbf{1} = \mathbf{1}$ holds. For this experiment, we consider log-returns of $r = 362$ stocks and $q = 9$ stock sectors from October 5, 2005, to December 30, 2015, totaling $n = 2577$ observations. Figure 10.13 shows the result of 100 realizations of Algorithms 10.5 and 10.6. It can be observed that Algorithms 10.5 and 10.6 are not sensitive to the choice of an initial point, which is an important feature for practical applications. Finally, Fig. 10.14a and b show the learned graphs from kSBG with different initial points, where we observe that the performance of kSBG is largely unaffected.



(a) Algorithm 10.5 (SBG). (b) Algorithm 10.6 (kSBG).

Fig. 10.13 Convergence trend of the presented algorithms for different initial points



(a) default initial point. accuracy = 0.825, modularity = 0.691.

(b) uniform initial point. accuracy = 0.820, modularity = 0.687.

Fig. 10.14 Visualizations of the graphs learned by kSBG with distinct initial points

10.7 Conclusions

Heavy tails are prevalent in time series of financial markets. Yet, they have been little explored in the context of Laplacian constrained graphical models. In this chapter, we have discussed optimization programs to learn graphical models with Laplacian constraints assuming that the data generating process is Student's t-distributed. The formulations follow a maximum-likelihood approach of a Markov random field, for which we designed ADMM algorithms. The algorithms presented in this chapter showed significant gains, measured via the modularity values of the estimated graphs as well as node accuracy, when compared to state-of-the-art counterparts in real-world scenarios that involved data from the US stock market, foreign exchange markets, and cryptocurrencies.

Acknowledgements This work was supported by the Hong Kong GRF 16207820 research grant.

References

1. Absil, P.-A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton, NJ (2007)
2. Beck, A.: First-Order Methods in Optimization. Society for Industrial and Applied Mathematics (SIAM), MOS-SIAM Series on Optimization (2017)
3. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1999)
4. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
5. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
6. Candès, E.J., Wakin, M.B., Boyd, S.P.: Enhancing sparsity by reweighted ℓ_1 minimization. J. Fourier Anal. Appl. **14**(5), 877–905 (2008)
7. Cardoso, J.V.M., Palomar, D.: spectralGraphTopology: learning graphs from data via spectral constraints. In: The Comprehensive R Archive Network (2019). https://CRAN.R-project.org/package=spectralGraphTopology
8. Cardoso, J.V.M., Ying, J., Palomar, D.P.: Graphical models in heavy-tailed markets. In: Advances in Neural Information Processing Systems (NeurIPS'21) (2021)
9. Cardoso, J.V.M., Ying, J., Palomar, D.P.: Learning bipartite graphs: heavy tails and multiple components. In: Advances in Neural Information Processing Systems (NeurIPS'22) (2022)
10. Chen, Z., Li, L., Bruna, J.: Supervised community detection with graph neural networks. In: International Conference on Learning Representations (ICLR'19) (2019)
11. Chung, F.R.K.: Spectral Graph Theory. In: CBMS Regional Conference Series in Mathematics, vol. 92 (1997)
12. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E **70** (2004)
13. Condat, L.: Fast projection onto the simplex and the ℓ_1 ball. Math. Program. **158** (2016)
14. Cont, R.: Empirical properties of asset returns: stylized facts and statistical issues. Quant. Financ. **1**, 223–236 (2001)
15. Coutino, M., Isufi, E., Maehara, T., Leus, G.: State-space network topology identification from partial observations (2019). arXiv:1906.10471
16. Danaher, P., Wang, P., Witten, D.M.: The joint graphical lasso for inverse covariance estimation across multiple classes. J. Roy. Stat. Soc. B **76**(2), 373–397 (2014)
17. de Prado, M.L.: Building diversified portfolios that outperform out of sample. J. Portf. Manag. **42**(4), 59–69 (2016)
18. Dehouche, N.: Scale matters: the daily, weekly and monthly volatility and predictability of bitcoin, gold, and the S&P 500 (2021). arXiv:2103.00395
19. Diamond, S., Boyd, S.: CVXPY: a Python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
20. Dong, X., Thanou, D., Frossard, P., Vandergheynst, P.: Learning Laplacian matrix in smooth graph signal representations. IEEE Trans. Signal Process. **64**(23), 6160–6173 (2016)
21. Egilmez, H.E., Pavez, E., Ortega, A.: Graph learning from data under Laplacian and structural constraints. IEEE J. Sel. Top. Signal Process. **11**(6), 825–841 (2017)
22. Epskamp, S., Borsboom, D., Fried, E.I.: Estimating psychological networks and their accuracy: a tutorial paper. Behav. Res. Methods **50**, 195–212 (2018)
23. Fan, K.: On a theorem of Weyl concerning eigenvalues of linear transformations I. Proc. Natl. Acad. Sci. **35**(11), 652–655 (1949)
24. Feng, Y., Palomar, D.: A signal processing perspective on financial engineering. Found. Trends Signal Process. **9**, 1–231 (2015)
25. Finegold, M., Drton, M.: Robust graphical modeling with t-distributions. In: Conference on Uncertainty in Artificial Intelligence (2009)
26. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3), 75–174 (2010)
27. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**, 432–41 (2008)

28. Godsil, C., Royle, G.: Algebraic Graph Theory. In: Graduate Texts in Mathematics, Springer Science (2001)
29. Goetzmann, N.W., Kumar, A.: Equity Portfolio Diversification. Rev. Financ. **12**(3), 433–463 (2008)
30. Gourieroux, C., Monfort, A.: Time Series and Dynamic Models. In: Themes in Modern Econometrics. Cambridge University Press (1997)
31. Hao, B., Sun, W.W., Liu, Y., Cheng, G.: Simultaneous clustering and estimation of heterogeneous graphical models. J. Mach. Learn. Res. **18**(217), 1–58 (2018)
32. Harvey, A.C.: Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series. Cambridge University Press (2013)
33. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press (1985)
34. Hsieh, C., Banerjee, A., Dhillon, I.S., Ravikumar, P.K.: A divide-and-conquer method for sparse inverse covariance estimation. In: Advances in Neural Information Processing Systems (NeurIPS'12), pp. 2330–2338 (2012)
35. Hunter, D., Lange, K.: A tutorial on MM algorithms. Am. Stat. **58**(1), 30–37 (2004)
36. Kalofolias, V.: How to learn a graph from smooth signals. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, vol. 51, pp. 920–929. PMLR (2016)
37. Knill, O.: Cauchy-Binet for pseudo-determinants. Linear Algebra Appl. **459**, 522–547 (2014)
38. Kumar, S., Ying, J., Cardoso, J.V.M., Palomar, D.P.: Structured graph learning via Laplacian spectral constraints. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
39. Kumar, S., Ying, J., Cardoso, J.V.M., Palomar, D.P.: A unified framework for structured graph learning via spectral constraints. J. Mach. Learn. Res. **21**, 1–60 (2020)
40. Lake, B.M., Tenenbaum, J.B.: Discovering structure by learning sparse graph. In: Proceedings of the 33rd Annual Cognitive Science Conference (2010)
41. Li, Y., Sha, C., Huang, X., Zhang, Y.: Community detection in attributed graphs: an embedding approach. In: AAAI Conference on Artificial Intelligence (AAAI'18) (2018)
42. Mantegna, R.N.: Hierarchical structure in financial markets. Eur. Phys. J. B **11**(1), 193–197 (1999). ISSN 1434-6028
43. Marti, G., Nielsen, F., Bińkowski, M., Donnat, P.: A review of two decades of correlations, hierarchies, networks and clustering in financial markets (2017). arXiv: 1703.00485
44. Mateos, G., Segarra, S., Marques, A.G., Ribeiro, A.: Connecting the dots: identifying network structure via graph signal processing. IEEE Signal Process. Mag. **36**(3), 16–43 (2019)
45. Michelot, C.: A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. J. Optim. Theory Appl. **50**(1), 195–200 (1986). July
46. Morgan Stanley Capital International and S&P Dow Jones. Revisions to the global industry classification standard (GICS) structure (2018)
47. Newman, M.E.J.: Modularity and community structure in networks. In: Proceedings of the National Academy of Sciences of the United States of America **103** (2006)
48. Nie, F., Wang, X., Huang, H.: Clustering and projected clustering with adaptive neighbors. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'14) (2014)
49. Nie, F., Wang, X., Jordan, M.I., Huang, H.: The constrained Laplacian rank algorithm for graph-based clustering. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, pp. 1969–1976. AAAI Press (2016)
50. Nie, F., Wang, X., Deng, C., Huang, H.: Learning a structured optimal bipartite graph for co-clustering. In: Advances on Neural Information Processing Systems (NeurIPS'17), pp. 4132–4141 (2017)
51. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. In: Society for Industrial and Applied Mathematics (2000)
52. Pal, S., Regol, F., Coates, M.: Bayesian graph convolutional neural networks using non-parametric graph learning. In: International Conference on Learning Representations (ICLR'19) (2019)
53. Palomar, D.P., Fonollosa, J.R.: Practical algorithms for a family of water filling solutions. IEEE Trans. Signal Process. **53**(2), 686–695 (2005)

54. Pavez, E., Egilmez, H.E., Ortega, A.: Learning graphs with monotone topology properties and multiple connected components. IEEE Trans. Signal Process. **66**(9), 2399–2413 (2018)
55. Resnick, S.I.: Heavy-Tail Phenomena: Probabilistic and Statistical Modeling. Springer, New York (2007)
56. Rue, H., Held, L.: Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability). Chapman & Hall/CRC (2005)
57. Segarra, S., Marques, A.G., Mateos, G., Ribeiro, A.: Network topology inference from spectral templates. IEEE Trans. Signal Inf. Process. Netw. **3**(3), 467–483 (2017)
58. Shafipour, R., Mateos, G.: Online topology inference from streaming stationary graph signals with partial connectivity information. Algorithms **13**(9) (2020)
59. Slawski, M., Hein, M.: Estimation of positive definite M-matrices and structure learning for attractive Gaussian Markov random fields. In: Linear Algebra and its Applications, pp. 145–179 (2015)
60. Smith, S.M., Miller, K.L., Salimi-Khorshidi, G., Webster, M., Beckmann, C.F., Nichols, T.E., Ramsey, J.D., Woolrich, M.W.: Network modelling methods for FMRI. Neuroimage **54**(2), 875–891 (2011)
61. Standard & Poor's. Global Industry Classification Standard (GICS). Technical Report (2006)
62. Stegle, O., Teichmann, S.A., Marioni, J.C.: Computational and analytical challenges in single-cell transcriptomics. Nat. Rev. Genet. **16**, 133–145 (2015)
63. Sun, S., Zhu, Y., Xu, J.: Adaptive variable clustering in Gaussian graphical models. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, vol. 33, pp. 931–939. PMLR (2014)
64. Sun, Y., Babu, P., Palomar, D.P.: Robust estimation of structured covariance matrix for heavy-tailed elliptical distributions. IEEE Trans. Signal Process. **64**(14), 3576–3590 (2016)
65. Sun, Y., Babu, P., Palomar, D.P.: Majorization-minimization algorithms in signal processing, communications, and machine learning. IEEE Trans. Signal Process. **65**(3), 794–816 (2017). ISSN 1941-0476
66. Tan, K.M., Witten, D., Shojaie, A.: The cluster graphical Lasso for improved estimation of Gaussian graphical models. Comput. Stat. Data Anal. **85**, 23–36 (2015). ISSN 0167-9473
67. Tsay, R.S.: Analysis of Financial Time Series. Wiley, 3rd edn (2010)
68. Wald, Y., Noy, N., Elidan, G., Wiesel, A.: Globally optimal learning for structured elliptical losses. In: Advances in Neural Information Processing Systems (NeurIPS'19) (2019)
69. Witten, D.M., Tibshirani, R.: Covariance-regularized regression and classification for high dimensional problems. J. R. Stat. Society. Ser. B (Stat. Methodol.) **71**(3), 615–636 (2009)
70. Witten, D.M., Friedman, J.H., Simon, N.: New insights and faster computations for the graphical lasso. J. Comput. Graph. Stat. **20**(4), 892–900 (2011)
71. Wright, S.J.: Coordinate descent algorithms. Math. Program. **151**, 3–34 (2015)
72. Wu, T.T., Lange, K.: The MM alternative to EM. Stat. Sci. **25**(4), 492–505 (2010)
73. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu., P.S.: A comprehensive survey on graph neural networks (2019). arXiv: 1901.00596
74. Yang, Y., Zhao, L., Chen, L., Wang, C., Han, J.: Portfolio optimization with idiosyncratic and systemic risks for financial networks (2021). arXiv: 2111.11286
75. Ying, J., Cardoso, J.V.M., Palomar, D.P.: Does the ℓ_1-norm learn a sparse graph under Laplacian constrained graphical models? (2020a). arXiv: 2006.14925
76. Ying, J., Cardoso, J.V.M., Palomar, D.P.: Nonconvex sparse graph learning under Laplacian-structured graphical model. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 7101–7113 (2020)
77. Ying, J., Cardoso, J.V.M., Palomar, D.: Minimax estimation of Laplacian constrained precision matrices. In: 24th International Conference on Artificial Intelligence and Statistics (2021)
78. Zhang, F.: The Schur Complement and Its Applications. Springer (2005)
79. Zhao, L., Wang, Y., Kumar, S., Palomar, D.P.: Optimization algorithms for graph Laplacian estimation via ADMM and MM. IEEE Trans. Signal Process. **67**(16), 4231–4244 (2019)