Graphical Models in Heavy-Tailed Markets

Anonymous Author(s)
Affiliation
Address
email

Abstract

Heavy-tailed statistical distributions have long been considered a more realistic statistical model for the data generating process in financial markets in comparison to their Gaussian counterpart. Nonetheless, mathematical nuisances, including nonconvexities, involved in estimating graphs in heavy-tailed settings pose a significant challenge to the practical design of algorithms for graph learning. In this work, we present graph learning estimators based on the Markov Random Field framework that assume a Student-t data generating process. We design scalable numerical algorithms, via the alternating direction method of multipliers, to learn both connected and k-component graphs along with their theoretical convergence guarantees. The proposed methods outperform state-of-the-art benchmarks in an extensive series of practical experiments with publicly available data from the S&P500 index, foreign exchanges, and cryptocurrencies.

1 Introduction

Graph learning frameworks are often designed based on the assumption that the observed graph signals are Gaussian distributed [1–5]. While such assumption for graphical models has found great success in many practical areas, which includes brain network analysis [6], psychological networks [7], and single-cell sequencing [8], it inherently neglects scenarios where there may exist outliers or the underlying data is naturally heavy-tailed distributed. As a consequence, those methods often lack robustness and may not succeed in capturing a meaningful representation of the underlying graph.

Data from financial instruments are well-known examples of such scenarios where heavy-tailedness and skewedness are present [9–14]. In addition, there has been a growing interest in methods for estimating graphical models in financial markets, which hence demands the development of scalable and robust learning algorithms [15].

Perhaps one of the most prominent applications, clustering financial time-series via graph techniques has been an active research topic [15–19]. Nonetheless, current techniques rely on the assumption that the underlying graph has a tree structure, which does bring advantages due to its hierarchical clustering properties, but also have been shown to be unstable [20–22] and not suitable when the data is not Gaussian distributed [23].

Motivated by practical challenging applications in finance, such as clustering of financial instruments and network estimation, we investigate the problem of learning graph matrices whose structure follow that of a Laplacian matrix of an undirected weighted graph for which the data generating process is assumed to be Student-t distributed. In particular, the main contributions of this paper are as follows:

- We propose a novel formulation for learning undirected weighted graphs under the assumption that the data generating process is Student-t distributed. We solve the underlying learning problem via a carefully designed numerical algorithm based on the Alternating Direction Method of Multipliers (ADMM), along with the establishment of its theoretical

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

convergence guarantees. We note that the proposed algorithm can be easily extended to account for additional linear constraints on the graph weights.

- We extend the proposed framework to account for heavy-tails and k-component graphs simultaneously, which enables a novel method for clustering financial time-series.
- We present extensive practical results, with real-world data from stock markets, 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.

Notation: Given a symmetric matrix $\boldsymbol{A}$, $\lambda_i(\boldsymbol{A})$ and $\lambda_{\mathsf{max}}(\boldsymbol{A})$ denote the i-th smallest eigenvalue and the maximum eigenvalue of $\boldsymbol{A}$, respectively. The Moore-Penrose inverse of $\boldsymbol{A}$ is denoted as $\boldsymbol{A}^\dagger$. The Frobenius norm of a matrix $\boldsymbol{A}$ is denoted as $\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\mathsf{tr}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}$. The operator $\mathsf{Diag} : \mathbb{R}^p \to \mathbb{R}^{p\times p}$ creates a diagonal matrix with the elements of an input vector along its diagonal. The operator $\mathsf{diag} : \mathbb{R}^{p\times p} \to \mathbb{R}^p$ extracts the diagonal of a square matrix. For $\boldsymbol{x} \in \mathbb{R}^p$, $\|\boldsymbol{x}\|_2$ stands for the Euclidean norm of $\boldsymbol{x}$ and $\|\boldsymbol{x}\|_\infty = \mathsf{sup}_i |x_i|$. $(\boldsymbol{x})^+$ denotes the projection on to the nonnegative orthant, *i.e.*, $(\boldsymbol{x})^+ = \mathsf{max}\left(\boldsymbol{0}, \boldsymbol{x}\right)$.

2 Background & Related Works

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

A p-dimensional, real-valued, Gaussian random variable $\boldsymbol{x}$, with mean vector $\mathbb{E}\left[\boldsymbol{x}\right] \triangleq \boldsymbol{\mu}$ and rank-deficient precision matrix $\boldsymbol{L}$, is said to form a Laplacian Gaussian Markov Random Field (LGMRF) [5, 24–26] of rank $p - k$, $k \geq 1$, with respect to a graph $\mathcal{G}$, when its probability density function is given as

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

where $\det^*(\boldsymbol{L})$ is the pseudo-determinant of $\boldsymbol{L}$, *i.e.*, the product of its positive eigenvalues [27].

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

To that end, the penalized Maximum Likelihood Estimator (MLE) of the Laplacian-constrained precision matrix of $\boldsymbol{x}$, on the basis of the observed data $\boldsymbol{X}$, is:

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\mathsf{minimize}} & \mathsf{tr}\left(\boldsymbol{L}\boldsymbol{S}\right) - \log\det{}^*\left(\boldsymbol{L}\right) + h(\boldsymbol{L}), \\ \mathsf{subject\ to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0},\ L_{ij} = L_{ji} \leq 0, \end{array} \tag{2}$$

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

Even though Problem (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 [28], particularly due to scalability issues related to the computation of the term $\log\det^*(\boldsymbol{L})$ [5, 29]. Indeed, recently, considerable efforts have been directed towards the design of scalable, iterative algorithms based on Block Coordinate Descent [30], Majorization-Minimization (MM) [31], and ADMM [32] to solve Problem (2) in an efficient fashion, *e.g.*, [5] and [29].

Estimators based on Gaussian assumptions have been proposed for connected graphs [2–5, 29]. Some of their properties, such as sparsity, are yet being investigated [33, 34]. The authors in [35] and [36] 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.

3 Proposed Formulations & Algorithms

In this section, we propose 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.*, $\boldsymbol{L} = \mathcal{L}\boldsymbol{w}$ [35], where $\boldsymbol{w} \in \mathbb{R}^{p(p-1)/2}$ is the vectorized form of the upper triangular part of the adjacency matrix, also known as the vector of graph weights. In addition, we use the fact that, for connected graphs, it follows that $\det^*(\mathcal{L}\boldsymbol{w}) = \det(\mathcal{L}\boldsymbol{w} + \boldsymbol{J})$, $\boldsymbol{J} \triangleq \frac{1}{p}\mathbf{1}\mathbf{1}^\top$ [5].

In order to address the inherent heavy-tailed nature of financial market data [37], we consider the Student-t distribution under the Improper Markov Random Field assumption [24] with Laplacian structural constraints, that is, we assume the data generating process to be modeled as multivariate zero-mean Student-t distribution, whose probability density function can be written as

$$p(\boldsymbol{x}) \propto \sqrt{\det{}^*(\boldsymbol{\Theta})} \left(1 + \frac{\boldsymbol{x}^\top \boldsymbol{\Theta} \boldsymbol{x}}{\nu}\right)^{-\frac{\nu+p}{2}}, \ \nu > 2, \tag{3}$$

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.*,

$$\begin{array}{ll} \underset{\boldsymbol{w}\geq\mathbf{0},\boldsymbol{\Theta}\succeq\mathbf{0}}{\text{minimize}} & \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{\boldsymbol{x}_i^\top\mathcal{L}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right) - \log\det\left(\boldsymbol{\Theta}+\boldsymbol{J}\right), \\ \text{subject to} & \boldsymbol{\Theta} = \mathcal{L}\boldsymbol{w},\ \mathfrak{d}\boldsymbol{w} = \boldsymbol{d}, \end{array} \tag{4}$$

where $\mathfrak{d} : \mathbb{R}^{p(p-1)/2} \to \mathbb{R}^p$ is the degree operator defined as $\mathfrak{d}\boldsymbol{w} \triangleq \mathsf{diag}\left(\mathcal{L}\boldsymbol{w}\right)$. The constraint $\mathfrak{d}\boldsymbol{w} = \boldsymbol{d}$ enables the learning of additional graph structures such as regular graphs and it is crucial for k-component graphs, as discussed in Section 3.2.

From a theoretical perspective, the Student-t model naturally yields sparse graphs. Comparing the objective function in Problem (4) to that of Problem (2), we note that the Student-t contains a $\log(\cdot)$ term in place of a linear term of the graph weights. The usage of a log function to promote sparsity is closely related to the iteratively reweighted ℓ_1-norm as an approximation for the ℓ_0-norm problem [38]. Problem (4) 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.

3.1 ADMM Solution

The partial augmented Lagrangian function of Problem (4) is given as

$$\begin{aligned} L_\rho(\boldsymbol{\Theta}, \boldsymbol{w}, \boldsymbol{Y}, \boldsymbol{y}) = &\frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{\boldsymbol{x}_i^\top\mathcal{L}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right) - \log\det\left(\boldsymbol{\Theta}+\boldsymbol{J}\right) \\ &+ \langle \boldsymbol{y}, \mathfrak{d}\boldsymbol{w} - \boldsymbol{d}\rangle + \frac{\rho}{2}\left\|\mathfrak{d}\boldsymbol{w} - \boldsymbol{d}\right\|_2^2 + \langle \boldsymbol{Y}, \boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}\right\|_{\mathrm{F}}^2, \end{aligned} \tag{5}$$

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

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

$$\boldsymbol{\Theta}^{l+1} = \underset{\boldsymbol{\Theta}\succeq\mathbf{0}}{\arg\min} \ -\log\det\left(\boldsymbol{\Theta}+\boldsymbol{J}\right) + \langle \boldsymbol{\Theta}, \boldsymbol{Y}^l\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}^l\right\|_{\mathrm{F}}^2, \tag{6}$$

whose closed-form solution is given by Lemma 1.

Lemma 1 *The global minimizer of problem* (6) *is [39, 40]*

$$\boldsymbol{\Theta}^{l+1} = \frac{1}{2\rho}\boldsymbol{U}\left(\boldsymbol{\Gamma} + \sqrt{\boldsymbol{\Gamma}^2 + 4\rho\boldsymbol{I}}\right)\boldsymbol{U}^\top - \boldsymbol{J}, \tag{7}$$

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

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

$$\underset{\boldsymbol{w}\geq\boldsymbol{0}}{\text{minimize}} \ \frac{\rho}{2}\boldsymbol{w}^\top\left(\mathfrak{d}^*\mathfrak{d}+\mathcal{L}^*\mathcal{L}\right)\boldsymbol{w}-\left\langle \boldsymbol{w}, \mathcal{L}^*\left(\boldsymbol{Y}^l+\rho\boldsymbol{\Theta}^{l+1}\right)-\mathfrak{d}^*\left(\boldsymbol{y}^l-\rho\boldsymbol{d}\right)\right\rangle + \frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\frac{\boldsymbol{x}_i^\top\mathcal{L}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right), \tag{8}$$

where $\mathfrak{d}^*$ and $\mathcal{L}^*$ are the adjoint operators of the degree and Laplacian operators, respectively.

In general, subproblem (8) is nonconvex due to the concave nature of the logarithm function. Hence, we resort to the MM method [41] to find a stationary point of subproblem (8). We proceed by constructing a global upper-bound of the objective function of (8) at point $\boldsymbol{w}^j \in \mathbb{R}_+^{p(p-1)/2}$ as

$$g(\boldsymbol{w},\boldsymbol{w}^j) = g(\boldsymbol{w}^j,\boldsymbol{w}^j) + \langle \boldsymbol{w}-\boldsymbol{w}^j, \nabla_{\boldsymbol{w}} f(\boldsymbol{w}^j)\rangle + \frac{\mu}{2}\left\|\boldsymbol{w}-\boldsymbol{w}^j\right\|_2^2, \tag{9}$$

where f is the objective function in the minimization in (8), its gradient is given as $\nabla_{\boldsymbol{w}} f(\boldsymbol{w}^j) = \boldsymbol{a}^j + \boldsymbol{b}^j$, where

$$\boldsymbol{a}^j = \mathcal{L}^*\left(\tilde{\boldsymbol{S}}^j - \boldsymbol{Y}^l - \rho\left(\boldsymbol{\Theta}^{l+1}-\mathcal{L}\boldsymbol{w}^j\right)\right), \tag{10}$$

$$\boldsymbol{b}^j = \mathfrak{d}^*\left(\boldsymbol{y}^l - \rho\left(\boldsymbol{d}-\mathfrak{d}\boldsymbol{w}^j\right)\right), \tag{11}$$

where $\tilde{\boldsymbol{S}}^j \triangleq \frac{1}{n}\sum_{i=1}^{n}\frac{(p+\nu)\boldsymbol{x}_i\boldsymbol{x}_i^\top}{\langle \boldsymbol{w}^j, \mathcal{L}^*(\boldsymbol{x}_i\boldsymbol{x}_i^\top)\rangle+\nu}$ is a weighted sample covariance matrix, and $\mu = \rho\lambda_{\mathsf{max}}\left(\mathfrak{d}^*\mathfrak{d}+\mathcal{L}^*\mathcal{L}\right)$, and the maximum eigenvalue of $\mathfrak{d}^*\mathfrak{d}+\mathcal{L}^*\mathcal{L}$ is given by Lemma 2, whose proof is presented in the Supplementary Material.

Lemma 2 *The maximum eigenvalue of the matrix $\mathfrak{d}^*\mathfrak{d}+\mathcal{L}^*\mathcal{L}$ is given as*

$$\lambda_{\mathsf{max}}\left(\mathfrak{d}^*\mathfrak{d}+\mathcal{L}^*\mathcal{L}\right) = 2(2p-1). \tag{12}$$

The vector of graph weights $\boldsymbol{w}$ can then be updated by minimizing the function g constructed in (9), which is tantamount to solving the following nonnegative, quadratic-constrained, strictly convex problem:

$$\boldsymbol{w}^{j+1} = \underset{\boldsymbol{w}\geq\boldsymbol{0}}{\arg\min}\ \rho(2p-1)\left\|\boldsymbol{w}-\boldsymbol{w}^j\right\|_2^2 + \langle \boldsymbol{w}, \boldsymbol{a}^j+\boldsymbol{b}^j\rangle, \tag{13}$$

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

$$\boldsymbol{w}^{j+1} = \left(\boldsymbol{w}^j - \frac{\boldsymbol{a}^j+\boldsymbol{b}^j}{2\rho(2p-1)}\right)^+, \tag{14}$$

that is, a projected gradient descent step with learning rate $(2\rho(2p-1))^{-1}$. Thus, we iterate (14) in order to obtain a stationary point, $\boldsymbol{w}^{l+1}$, of Problem (8). In practice, we observe that a few iterations are sufficient to retrieve $\boldsymbol{w}^{l+1}$.

The dual variables $\boldsymbol{Y}$ and $\boldsymbol{y}$ are updated via gradient ascent steps. Algorithm 1 summarizes the implementation to find a stationary point of Problem (4). We present Theorem 3, proved in the Supplementary Material, which establishes the convergence of Algorithm 1.

Theorem 3 *Algorithm 1 converges subsequently for any sufficiently large ρ, that is, the sequence $\left\{\left(\boldsymbol{\Theta}^l, \boldsymbol{w}^l, \boldsymbol{Y}^l, \boldsymbol{y}^l\right)\right\}$ generated by Algorithm 1 has at least one limit point, and each limit point is a stationary point of Problem* (4).

3.2 An extension to k-component graphs

The graph learning formulation proposed in (4) is applicable to learn connected graphs. Learning graphs with k components poses a considerably higher challenge, as the dimension of the nullspace of the Laplacian matrix $\mathcal{L}\boldsymbol{w}$ is equal to the number of components of the graph [42]. One way to

Algorithm 1: Student-t Graph Learning

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

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

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

$l \leftarrow 0$

while $\left\|\boldsymbol{r}^l\right\|_\infty > \epsilon \; or \; \left\|\boldsymbol{s}^l\right\|_\infty > \epsilon$ **do**

  ▷ update $\boldsymbol{\Theta}^{l+1}$ via (7)

  ▷ update $\boldsymbol{w}^{l+1}$ by iterating (14)

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

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

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

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

  $l \leftarrow l + 1$

end

achieve this requirement is by imposing the constraint $\mathsf{rank}\left(\mathcal{L}\boldsymbol{w}\right) = p - k$, which is nonconvex and nondifferentiable, in the maximum likelihood problem generated by (3). We instead relax this rank constraint by noting that via Fan's theorem [43], we have

$$\sum_{i=1}^{k} \lambda_i\left(\mathcal{L}\boldsymbol{w}\right) = \underset{\boldsymbol{V} \in \mathbb{R}^{p \times k}, \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}}{\mathsf{minimize}} \; \mathsf{tr}\left(\boldsymbol{V}^\top \mathcal{L}\boldsymbol{w}\boldsymbol{V}\right). \tag{15}$$

Thus, by using the right hand side of (15) as a regularization term, we are able formulate the following optimization problem to learn a Student-t k-component graph:

$$\begin{aligned} \underset{\boldsymbol{w} \geq \boldsymbol{0}, \boldsymbol{\Theta} \succeq 0, \boldsymbol{V}}{\mathsf{minimize}} \quad & \frac{p+\nu}{n} \sum_{i=1}^{n} \log\left(1 + \frac{\boldsymbol{x}_i^\top \mathcal{L}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right) - \log\det{}^*\left(\boldsymbol{\Theta}\right) + \eta \mathsf{tr}(\mathcal{L}\boldsymbol{w}\boldsymbol{V}\boldsymbol{V}^\top), \\ \mathsf{subject\ to} \quad & \boldsymbol{\Theta} = \mathcal{L}\boldsymbol{w}, \; \mathsf{rank}(\boldsymbol{\Theta}) = p - k, \mathfrak{d}\boldsymbol{w} = \boldsymbol{d}, \; \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}, \; \boldsymbol{V} \in \mathbb{R}^{p \times k}. \end{aligned} \tag{16}$$

The partial augmented Lagrangian function of Problem (16) can be expressed as

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

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

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

which is nearly the same as (6). Its solution is obtained as

$$\boldsymbol{\Theta}^{l+1} = \frac{1}{2\rho}\boldsymbol{U}\left(\boldsymbol{\Gamma} + \sqrt{\boldsymbol{\Gamma}^2 + 4\rho\boldsymbol{I}}\right)\boldsymbol{U}^\top, \tag{19}$$

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

The subproblem to obtain $\boldsymbol{w}^{l+1}$ is virtually the same as in (8) except for the additional linear term $\eta \mathsf{tr}(\mathcal{L}\boldsymbol{w}\boldsymbol{V}^l\boldsymbol{V}^{l\top})$. Hence, its update is also a projected gradient descent step, alike (14) where

$$\boldsymbol{a}^j \triangleq \mathcal{L}^*\left(\tilde{\boldsymbol{S}}^j + \eta\boldsymbol{V}^l\boldsymbol{V}^{l\top} - \boldsymbol{Y}^l - \rho\left(\boldsymbol{\Theta}^{l+1} - \mathcal{L}\boldsymbol{w}^j\right)\right). \tag{20}$$

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

$$\underset{\boldsymbol{V} \in \mathbb{R}^{p \times k}, \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}}{\mathsf{minimize}} \quad \mathsf{tr}\left(\boldsymbol{V}^\top \mathcal{L}\boldsymbol{w}^{l+1}\boldsymbol{V}\right), \tag{21}$$

whose closed-form solution is given by the k eigenvectors associated with the k smallest eigenvalues of $\mathcal{L}\boldsymbol{w}^{l+1}$ [44, 45]. Algorithm 2 summarizes the implementation to find a stationary point of Problem (16), and its convergence is established through Theorem 4, whose proof is presented in the Supplementary Material.

Theorem 4 *Algorithm 2 converges subsequently for any sufficiently large ρ, that is, the sequence $\left\{\left(\boldsymbol{\Theta}^l, \boldsymbol{w}^l, \boldsymbol{V}^l, \boldsymbol{Y}^l, \boldsymbol{y}^l\right)\right\}$ generated by Algorithm 2 has at least one limit point, and each limit point is a stationary point of Problem* (16).

Algorithm 2: k-component Student-t graph learning

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

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

initialize $\boldsymbol{Y} = \boldsymbol{0}$, $\boldsymbol{y} = \boldsymbol{0}$
$l \leftarrow 0$
while $\left\|\boldsymbol{r}^l\right\|_\infty > \epsilon$ *or* $\left\|\boldsymbol{s}^l\right\|_\infty > \epsilon$ **do**
  ▷ update $\boldsymbol{\Theta}^{l+1}$ via (19)
  ▷ update $\boldsymbol{w}^{l+1}$ as in (14) with $\boldsymbol{a}^j$ given in (20)
  ▷ update $\boldsymbol{V}^{l+1}$ as in (21)
  ▷ update $\boldsymbol{Y}^{l+1} = \boldsymbol{Y}^l + \rho\left(\boldsymbol{\Theta}^{l+1} - \mathcal{L}\boldsymbol{w}^{l+1}\right)$
  ▷ update $\boldsymbol{y}^{l+1} = \boldsymbol{y}^l + \rho\left(\mathfrak{d}\boldsymbol{w}^{l+1} - \boldsymbol{d}\right)$
  ▷ compute residual $\boldsymbol{r}^{l+1} = \boldsymbol{\Theta}^{l+1} - \mathcal{L}\boldsymbol{w}^{l+1}$
  ▷ compute residual $\boldsymbol{s}^{l+1} = \mathfrak{d}\boldsymbol{w}^{l+1} - \boldsymbol{d}$
  $l \leftarrow l + 1$
end

4 Experiments

To evaluate the performance of the proposed 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 $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, where n is the number of log-return observations and p is the number of instruments, as

$$X_{i,j} = \log P_{i,j} - \log P_{i-1,j}, \tag{22}$$

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

Benchmarks: We compare our proposed algorithms with state-of-the-art, Gaussian distribution-based methods for connected graphs, namely GLE [29] and NGL [33], which use ℓ_1-norm and minimax concave penalty regularizations, respectively; and CLR [36] and SGL [35] that consider k-component graphs. For a fair comparison among algorithms, we set the degree vector $\boldsymbol{d}$ equal to $\boldsymbol{1}$ for the proposed algorithms, *i.e.*, we do not consider any prior information on the degree of nodes. In our ADMM algorithms, we set the penalty parameter to $\rho = 1$ and the hyperparameter η in (16) 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.

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[1] 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) [47, 48]. [2]

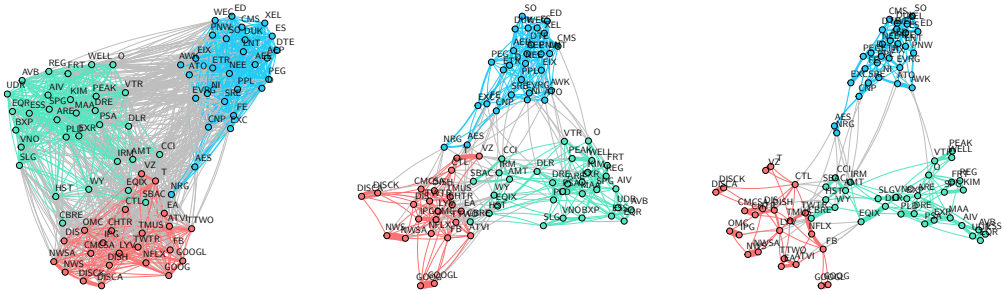
[1] The modularity measures the strength of division of a graph into groups [46].

[2] More often than not, stocks have impacts on multiple sectors, *e.g.*, the evident case of technology companies, whose influence affect the prices of stocks not only in their own sector, but spans across multiple sectors. Therefore, GICS or other sector classification systems cannot be considered as absolute ground-truth labels.

4.1 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), totalling $p = 82$ stocks, during the time horizon from Jan. 3rd 2014 to Dec. 29th 2017, resulting in $n = 1006$ observations. In order to obtain descriptive insights on this dataset, we measure its degree of heavy-tailedness and annualized volatility[3]. The former is obtained by fitting the degrees of freedom of a Student-t distribution to the matrix of log-returns, whereby we obtain $\nu \approx 5.5$ and $\sigma \approx 21\%$. This scenario can be considered as having a moderate amount of heavy-tailedness.

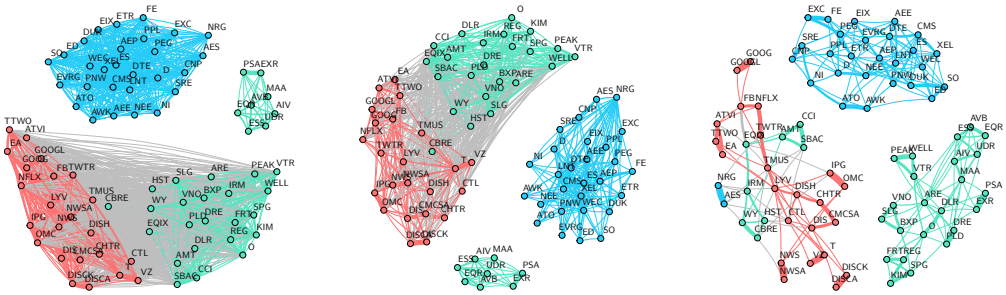
Figure 1 depicts the learned connected graphs on the aforementioned time periods. It can be readily noticed that the graph learned with the Student-t distribution (Figure 1c) is sparser than those learned with the Gaussian assumption (Figure 1a and 1b), which results from the fact that the Gaussian distribution is more sensitive to outliers. Moreover, the Student-t graph presents a higher degree of interpretability as measured by its modularity 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-t assumption.



(a) GLE, modularity $= 0.31$. (b) NGL, modularity $= 0.49$. (c) Algorithm 1, modularity $= 0.54$.

Figure 1: Learned graphs of S&P500 stocks.

Figure 2 illustrates the learned 3-component graphs during the time from Jan. 3rd 2014 until Dec. 29th 2017. We can notice that SGL (Figure 2a) and CLR (Figure 2b) 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) are uncharacteristic of the actual expected behavior in stock markets. Figure 2c displays the graph learned by the proposed Algorithm 2, 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.



(a) SGL, modularity $= 0.29$. (b) CLR, modularity $= 0.33$. (c) Algorithm 2, modularity $= 0.56$.

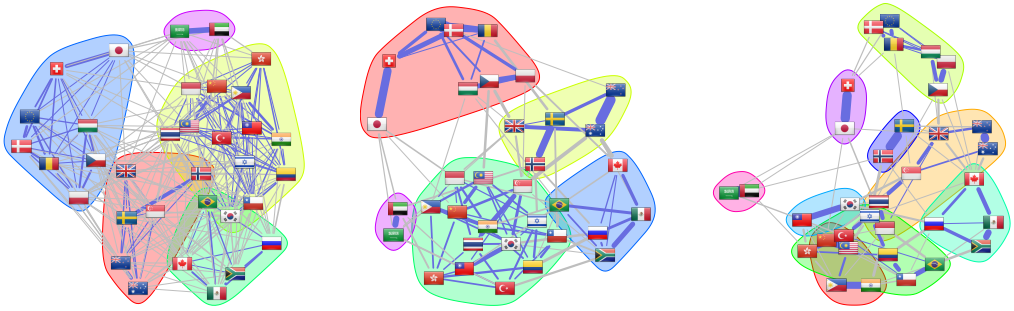
Figure 2: Learned 3-component graphs of S&P500 stocks.

[3]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.

4.2 Communities in Foreign Exchange Markets

We query foreign exchange data from the 34 most traded currencies between the period from Jan. 2nd 2019 to Dec. 31st 2020, totalling $n = 522$ observations. The data matrix is composed by the log-returns of the currencies prices with respect to the United States Dollar. Similar to the previous experiment, we compute the degrees of freedom of a Student-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 [49] in order to create classes within the learned graph. In particular, the algorithm in [49] takes as input the learned Laplacian matrix of the graph and outputs a membership assignment that maximizes the modularity of the graph.

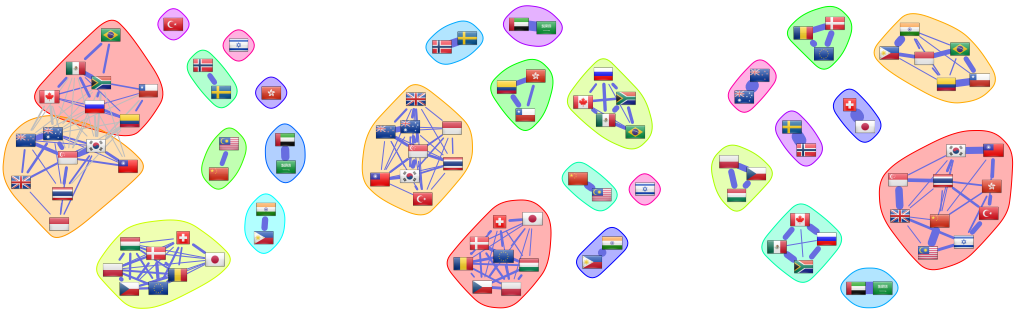
Figure 3 displays the learned graphs. As it can be observed, the Student-t graph (Figure 3c) is sparser, more interpretable, and has a higher modularity value than that of the Gaussian-based graphs (Figure 3a and 3b). In addition, the expected correlation between currencies of locations geographically close to each other, *e.g.*, {Hong Kong SAR, China}, {Taiwan, South Korea}, and {Poland, Czech Republic} are significantly more evident for the Student-t graph.



(a) GLE, modularity $= 0.34$. (b) NGL, modularity $= 0.46$. (c) Algorithm 1, modularity $= 0.58$.

Figure 3: Learned connected graphs of currencies.

Figure 4 depicts the learned 9-component graphs of currencies during the time window from Jan. 2nd 2019 to Dec. 31st 2020. It can be observed that the graph learned by the proposed Algorithm 2 (Figure 4c) presents a finer structure and a higher modularity value than those learned by SGL (Figure 4a) and CLR (Figure 4b). In addition, the learned graph in Figure 4c presents more reasonable clusters such as {New Zealand, Australia} and {Poland, 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. In our proposed algorithm, we avoid such solutions by imposing linear constraints on the degree of the graph.



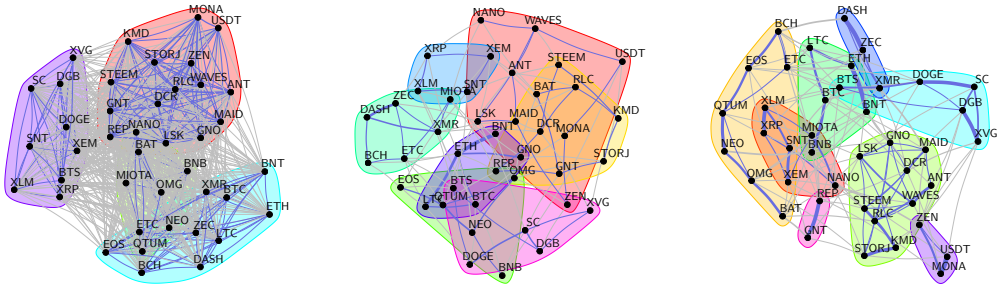
(a) SGL, modularity $= 0.62$. (b) CLR, modularity $= 0.79$. (c) Algorithm 2, modularity $= 0.84$.

Figure 4: Learned 9-component graphs of currencies.

4.3 Communities in Cryptocurrencies

We query daily prices of the $p = 41$ most traded cryptocurrencies during the period starting from Aug. 1st 2017 to Dec. 1st 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.

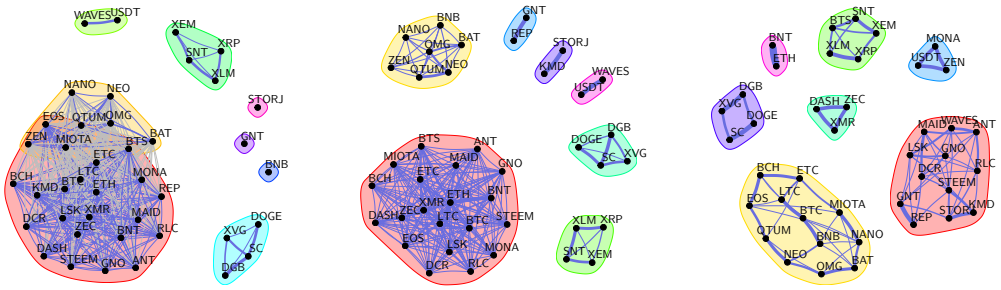
Figure 5 shows the learned graphs by GLE, NGL, and our proposed Algorithm 1. While the graphs in Figures 5a and 5b present a small modularity value and contain a large number of edges, which impairs interpretability, the resulting proposed graph in Figure 5c reveals a refined representation of the interactions 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 Figure 5c.



(a) GLE, modularity $= 0.19$. (b) NGL, modularity $= 0.40$. (c) Algorithm 1, modularity $= 0.52$.

Figure 5: Learned connected graphs of cryptocurrencies.

Figure 6 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 (Figure 6a) and CLR contains a large number of spurious connections in the main cluster (Figure 6b), whereas the graph learned via Algorithm 2 (Figure 6c) has the largest modularity value. Interestingly, while all three methods agree to cluster {Dogecoin (DOGE), Verge (XVG), Siacoin (SC), DigiByteCoin (DGB)}, which may be related to their similar initial release dates, only our proposed algorithm clusters together the coins that mainly focus on privacy and anonymity features, *i.e.*, {Monero (XMR), Zcash (ZEC), DASH}.



(a) SGL, modularity $= 0.36$ (b) CLR, modularity $= 0.66$. (c) Algorithm 2, modularity $= 0.79$.

Figure 6: Learned 7-component graphs of cryptocurrencies.

5 Conclusions

Heavy-tails are prevalent in time-series of financial markets. Yet, they have been little explored in the context of Laplacian graphical models. In this paper, we have proposed optimization programs to learn graphical models with Laplacian constraints assuming that the data generating process is Student-t distributed. The formulations follow a maximum likelihood approach of a Markov Random Field, for which we designed ADMM algorithms that converge to a stationary point of the resulting nonconvex problems. The proposed algorithms showed significant gains, measured via the modularity values of the estimated graphs, 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.

References

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

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

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

[4] V. Kalofolias. How to learn a graph from smooth signals. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 920–929, 2016.

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

[6] Stephen M. Smith, Karla L. Miller, Gholamreza Salimi-Khorshidi, Matthew Webster, Christian F. Beckmann, Thomas E. Nichols, Joseph D. Ramsey, and Mark W. Woolrich. Network modelling methods for fmri. *NeuroImage*, 54(2):875 – 891, 2011.

[7] S. Epskamp, D. Borsboom, and E. I. Fried. Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50:195–212, 2018.

[8] O. Stegle, S. A. Teichmann, and J. C. Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, 16:133–145, 2015.

[9] C. Gourieroux and A. Monfort. *Time Series and Dynamic Models*. Themes in Modern Econometrics. Cambridge University Press, 1997.

[10] R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1:223–236, 2001.

[11] R. S. Tsay. *Analysis of Financial Time Series*. Wiley, 3rd edition, 2010.

[12] A. C. Harvey. *Dynamic models for volatility and heavy tails: with applications to financial and economic time series*. Cambridge University Press, 2013.

[13] Y. Feng and D. Palomar. A signal processing perspective on financial engineering. *Foundations and Trends in Signal Processing*, 9:1–231, 2015.

[14] Nassim Dehouche. Scale matters: The daily, weekly and monthly volatility and predictability of bitcoin, gold, and the s&p 500, 2021.

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

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

[17] C. Dose and S. Cincotti. Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications*, 355(1):145 – 151, 2005.

[18] G. Marti, S. Andler, F. Nielsen, and P. Donnat. Clustering financial time series: How long is enough? In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[19] G. Marti, F. Nielsen, P. Donnat, and S. Andler. On clustering financial time series: a need for distances between dependent random variables. *Computational Information Geometry*, pages 149–174, 2017.

[20] G. Carlsson and F. Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research*, 11:1425–1470, 2010.

[21] V. Lemieux, P. S. Rahmdel, R. Walker, B. L. W. Wong, and M. Flood. Clustering techniques and their effect on portfolio formation and risk analysis. In *Proceedings of the International Workshop on Data Science for Macro-Modeling*, page 1–6, 2014.

[22] G. Marti, P. Very, P. Donnat, and F. Nielsen. A proposal of a methodological framework with experimental guidelines to investigate clustering stability on financial time series. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 32–37, 2015.

[23] P. Donnat, G. Marti, and P. Very. Toward a generic representation of random variables for machine learning. *Pattern Recognition Letters*, 70:24–31, 2016.

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

[25] M. Slawski and M. Hein. Estimation of positive definite m-matrices and structure learning for attractive Gaussian Markov random fields. *Linear Algebra and its Applications*, 473:145–179, 2015.

[26] J. Ying, J. V. de M. Cardoso, and D. P. Palomar. Does the ℓ_1-norm learn a sparse graph under Laplacian constrained graphical models? *arXiv e-prints: 2006.14925*, June 2020.

[27] O. Knill. Cauchy–Binet for pseudo-determinants. *Linear Algebra and its Applications*, 459:522–547, 2014.

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

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

[30] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151:3–34, 2015.

[31] David Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004.

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

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

[34] J. Ying, J. V. M. Cardoso, and D. P. Palomar. Minimax estimation of Laplacian constrained precision matrices. In *24th International Conference on Artificial Intelligence and Statistics (AISTATS'21)*, 2021.

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

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

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

[38] Emmanuel J. Candès, Michael B. Wakin, and Stephen P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

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

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

[41] Y. Wald, N. Noy, G. Elidan, and A. Wiesel. Globally optimal learning for structured elliptical losses. In *Advances in Neural Information Processing Systems (NeurIPS'19)*, 2019.

[42] F. R. K. Chung. *Spectral Graph Theory*, volume 92. CBMS Regional Conference Series in Mathematics, 1997.

[43] K. Fan. On a theorem of Weyl concerning eigenvalues of linear transformations I. *Proceedings of the National Academy of Sciences*, 35(11):652–655, 1949.

[44] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[45] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2007.

[46] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103, 2006.

[47] Standard & Poor's. Global Industry Classification Standard (GICS). *Tech Report*, 2006.

[48] Morgan Stanley Capital International and S&P Dow Jones. Revisions to the global industry classification standard (GICS) structure, 2018.

[49] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70, Dec 2004.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] No obvious negative societal impact is caused by our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs of the theorems are available in the Supplementary Material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and instructions on how to reproduce all the experiments are available in the Supplementary Material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] All the assets used in this work are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The data we are using do not contain personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]