Portfolio Optimization

Financial Data: Graphs

Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.*
Cambridge University Press.

portfoliooptimizationbook.com

Latest update: April 14, 2025

Outline

Abstract

Graphs are a powerful mathematical tool for representing data and relationships between entities in various fields, including biology, finance, machine learning, and social networks. With the increasing availability of large datasets, graph-based analysis is crucial for understanding the structure of networks generating the data. In practice, the underlying graph structure is often unknown and must be inferred from the data. Over the past two decades, numerous graph learning algorithms have been proposed, with recent advances focusing on financial data applications (Palomar 2025, chap. 5).

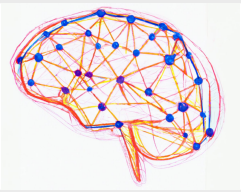
Outline

Graphs

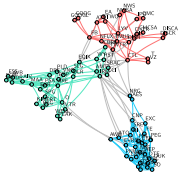
- **Graphs in Various Domains:**
  - Serve as a fundamental tool for modeling data across diverse applications.
  - Enable understanding of large networks' structure and data geometry visualization.
- **Graph Examples Across Domains:**
  - **Social Media:** Models individual behaviors and influences using online activities.
  - **Brain Activity:** Correlates brain sensors, utilizing fMRI data.
  - **Financial Stocks:** Shows company interdependencies in markets, with stock prices and volumes.
  - **Currency & Cryptocurrency:** Summarizes interdependencies in foreign and crypto markets, using economic data.

Examples of graphs in different applications

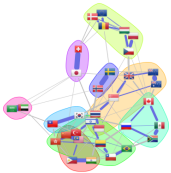
Social media graph



Brain activity graph



Financial stock graph

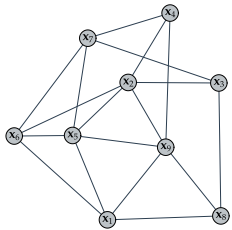


Financial currency graph

Terminology

The basic elements of a graph are

- *nodes*: corresponding to the entities or variables; and
- *edges*: encoding the relationships between entities or variables.



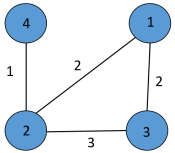
- **Graph Mathematical Structure:**
  - **Nodes:** $\mathcal{V} = \{1, 2, 3, \ldots, p\}$
  - **Edges:** $\mathcal{E} = \{(1,2), (1,3), \ldots, (i,j), \ldots\}$
  - **Weight Matrix W:** strength of relationships between nodes.

Graph matrices

- **Adjacency Matrix $\boldsymbol{W}$:** Directly characterizes a graph.
  - $[\boldsymbol{W}]_{ij} = w_{ij}$ if $(i,j) \in \mathcal{E}$, else 0; $W_{ij} = 0$ if $i = j$.
  - Symmetric $\boldsymbol{W}$ implies undirected graph.
- **Connectivity Matrix $\boldsymbol{C}$:** Binary version of the adjacency matrix.
  - $[\boldsymbol{C}]_{ij} = 1$ if $(i,j) \in \mathcal{E}$, else 0; $C_{ij} = 0$ if $i = j$.
  - Describes adjacency matrix connectivity pattern.
- **Degree Matrix $\boldsymbol{D}$:** Diagonal matrix with node degrees $\boldsymbol{d} = (d_1, \ldots, d_p)$.
  - Degree $d_i = \sum_j W_{ij}$.
  - $\boldsymbol{D} = \text{Diag}(\boldsymbol{W}\mathbf{1})$.
- **Laplacian Matrix $\boldsymbol{L}$:** Defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$.
  - Symmetric, positive semidefinite: $\boldsymbol{L} \succeq \mathbf{0}$.
  - Zero eigenvalue with all-one vector $\mathbf{1}$: $\boldsymbol{L}\mathbf{1} = \mathbf{0}$.
  - Degree vector on diagonal: $\text{diag}(\boldsymbol{L}) = \boldsymbol{d}$.
  - Zero eigenvalues number indicates graph's connected components.
  - Measures graph signal smoothness: $\boldsymbol{x}^\mathsf{T}\boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2$.

Example of a toy undirected graph

- **Graph Definition:**
  - Nodes: $\mathcal{V} = \{1, 2, 3, 4\}$
  - Edges: $\mathcal{E} = \{(1,2), (2,1), (1,3), (3,1), (2,3), (3,2), (2,4), (4,2)\}$
  - Weights: $w_{12} = w_{21} = 2,\ w_{13} = w_{31} = 2,\ w_{23} = w_{32} = 3,\ w_{24} = w_{42} = 1$



- The connectivity, adjacency, and Laplacian graph matrices are

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{W} = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 3 & 1 \\ 2 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{L} = \begin{bmatrix} 4 & -2 & -2 & 0 \\ -2 & 6 & -3 & -1 \\ -2 & -3 & 5 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}.$$

Outline

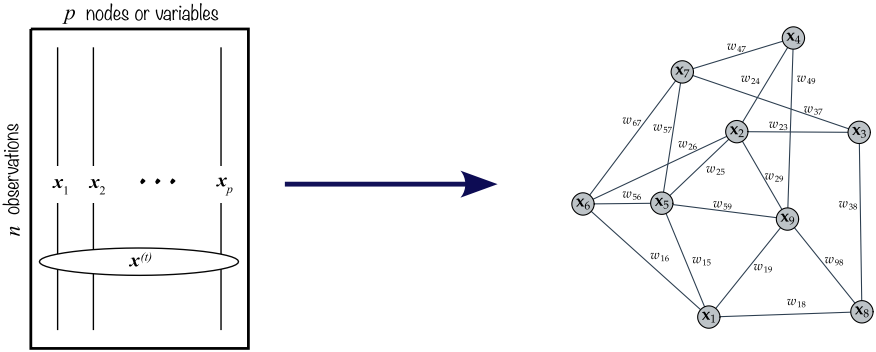
Learning graphs

- **Graph Structure in Applications:**
  - Directly observable in social networks (nodes as users, edges as friendships).
  - Must be inferred in gene graphs, brain activity graphs, financial graphs.
- **Graph Learning Methods:**
  - Range from heuristic techniques to statistically sound approaches.
  - Utilize physical interpretation or estimation theory.
- **Data Matrix as Starting Point:**

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p] \in \mathbb{R}^{n \times p},$$

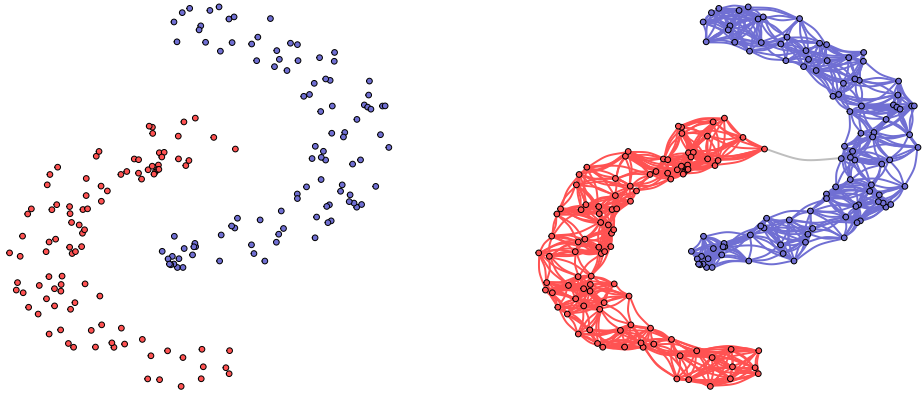
  - Columns represent signals of variables/nodes.
  - p: number of variables/nodes, n (T in financial time series): number of observations.
- **Goal in Graph Learning:**
  - Transition from data matrix $\boldsymbol{X}$ to graph description $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$.

Learning graphs from data



Learning graphs from data

Illustration of graph learning for the “two-moon” dataset:



Development in graph learning

- **Pioneering Work:**
  - (Mantegna 1999): First data-driven graphs in financial markets using correlation graphs.
- **Foundational Textbooks:**
  - (Lauritzen 1996; Kolaczyk 2009): Comprehensive understanding of graph theory.
- **Introductory and Overview Articles:**
  - (Mateos et al. 2019; Dong et al. 2019): For basics and overview of graph learning.
- **Basic Graph Learning Algorithms:**
  - Found in (Lake and Tenenbaum 2010; Egilmez, Pavez, and Ortega 2017; Zhao et al. 2019).

Development in graph learning

- **Structured Graph Learning:**
  - Spectral constraints approach: (Kumar et al. 2019, 2020).
  - Sparsity in graphs: (Ying, Cardoso, and Palomar 2020).
  - Convex formulation for bipartite graphs: (Cardoso, Ying, and Palomar 2022b).
- **Graph Learning with Financial Data:**
  - General guidelines: (Cardoso and Palomar 2020).
  - Learning under heavy tails: (Cardoso, Ying, and Palomar 2021).
  - Bipartite-structured graphs for clustering: (Cardoso, Ying, and Palomar 2022b); overview in (Cardoso, Ying, and Palomar 2022a).
- **Comprehensive Literature Overview:**
  - On financial graphs in the past two decades: (Marti et al. 2021).

Learning graphs from similarity measures

- **Graph Inference based on Adjacency Matrix W:**
  - Use similarity or scoring functions to measure connectivity strength.
  - Different functions lead to different graphs.
- **Illustrative Simple Methods:**
  - **Thresholded Distance Graph:**
    - Connect nodes i and j ($w_{ij} = 1$) if $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq \gamma$ (threshold).
    - Otherwise, $w_{ij} = 0$.
  - **Gaussian Graph:**
    - Connect all pairs $i \neq j$ with weights:

$$w_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right),$$

    - σ^2 controls neighborhood size.

Learning graphs from similarity measures

- **Illustrative Simple Methods:**
  - **k-Nearest Neighbors (k-NN) Graph:**
    - Connect nodes i and j ($w_{ij} = 1$) if $\boldsymbol{x}_i$ is among k closest to $\boldsymbol{x}_j$ or vice-versa.
    - Otherwise, $w_{ij} = 0$.
  - **Feature Correlation Graph:**
    - Use pairwise feature correlation for $i \neq j$:

$$w_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j.$$

    - If signals normalized ($\|\boldsymbol{x}_i\|^2 = 1$), Euclidean distance relates to correlation:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = 2 \times (1 - \boldsymbol{x}_i^\top \boldsymbol{x}_j).$$

- **Limitations of Heuristic Methods:**
  - Measure connectivity independently for each pair.
  - May not perform well, especially for time series data.
  - Better to measure connectivity jointly for all pairs.

Learning graphs from signal smoothness

- **Variance of Graph Signals:**
  - For p-dimensional signal $\boldsymbol{x}$ on p nodes: $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2$.
  - For n observations in data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$:

$$\sum_{t=1}^{n} (\boldsymbol{x}^{(t)})^{\mathsf{T}}\boldsymbol{L}\boldsymbol{x}^{(t)} = \mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^{\mathsf{T}}\right).$$

- **Graph Learning Problem Formulation:**
  - Minimize signal variance to find the generating graph.
  - Choose between graphs $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ by comparing variances: $\mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}_1\boldsymbol{X}^{\mathsf{T}}\right)$ vs. $\mathrm{Tr}\left(\boldsymbol{X}\boldsymbol{L}_2\boldsymbol{X}^{\mathsf{T}}\right)$.
- **Graph Learning Optimization:**
  - Determine graph $\mathcal{G}$ that minimizes signal variance.
  - Include regularization for graph properties (sparsity, energy, volume).

Learning graphs from signal smoothness

- **Problem Formulation in Terms of $\boldsymbol{L}$:**

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{minimize}} & \text{Tr}\left(\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^{\text{T}}\right) + \gamma h_L(\boldsymbol{L}) \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array}$$

  - γ: regularization level, $h_L(\boldsymbol{L})$: regularization function.
- **Problem Formulation in Terms of $\boldsymbol{W}$:**

$$\begin{array}{ll} \underset{\boldsymbol{W}}{\text{minimize}} & \frac{1}{2}\text{Tr}(\boldsymbol{W}\boldsymbol{Z}) + \gamma h_W(\boldsymbol{W}) \\ \text{subject to} & \text{diag}(\boldsymbol{W}) = \mathbf{0}, \quad \boldsymbol{W} = \boldsymbol{W}^{\text{T}} \geq \mathbf{0}, \end{array}$$

  - $h_W(\boldsymbol{W})$: regularization function for adjacency matrix.
- **Convexity and Complexity:**
  - Formulations are convex if regularization terms are convex.
  - Formulation in terms of $\boldsymbol{L}$ not more complex than $\boldsymbol{W}$ despite $\boldsymbol{L} \succeq \mathbf{0}$ constraint.

Learning graphs from graphical model networks

- **Graph Data as Multivariate Distribution:**
  - Graph signals $\boldsymbol{x}^{(t)}$ assumed to follow a distribution, e.g., Gaussian: $\boldsymbol{x}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
  - T: number of observations.
- **Sample Covariance Matrix $\boldsymbol{S}$:**

$$\boldsymbol{S} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})(\boldsymbol{x}^{(t)} - \boldsymbol{\mu})^{\mathsf{T}} = \frac{1}{T}(\boldsymbol{X} - \bar{\boldsymbol{X}})^{\mathsf{T}}(\boldsymbol{X} - \bar{\boldsymbol{X}}),$$

  - $\bar{\boldsymbol{X}}$: matrix with mean vector $\boldsymbol{\mu}$ along each row.
- **Graphical Model Estimation:**
  - Inverse sample covariance matrix $\boldsymbol{S}^{-1}$ historically used to determine graph structure.

Graph learning methods

- **Correlation Networks:**
  - Use pairwise correlations to measure node similarity.
  - Drawback: High correlation may result from common dependencies on other nodes.
- **Partial Correlation Networks:**
  - Measure direct dependency between two nodes, conditioned on other nodes.
  - Information contained in precision matrix $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$.
  - Conditional correlation between nodes i and j: $-\Theta_{ij}/\sqrt{\Theta_{ii}\Theta_{jj}}$.
  - Conditional independence if $\Theta_{ij} = 0$.
  - Nonzero off-diagonal entries of $\boldsymbol{\Theta}$ indicate graph edges.
- **Graphical LASSO (GLASSO):**
  - Estimates sparse precision matrix $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$.
  - Regularized maximum likelihood estimation:

$$\underset{\boldsymbol{\Theta} \succ \mathbf{0}}{\text{maximize}} \quad \log\det(\boldsymbol{\Theta}) - \text{Tr}(\boldsymbol{\Theta}\boldsymbol{S}) - \rho\|\boldsymbol{\Theta}\|_{1,\text{off}},$$

  - ρ: controls sparsity level.

Graph learning methods

- **Laplacian-structured GLASSO:**
  - Adapts GLASSO for Gaussian Markov random fields (GMRFs) with Laplacian constraints:

$$\underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} \quad \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{1,\text{off}},$$

  - Constraints ensure Laplacian matrix properties.
- **Sparse GMRF Graphs:**
  - Addresses the limitation of dense graphs produced by ℓ_1-norm regularization:

$$\underset{\boldsymbol{L}\succeq\mathbf{0}}{\text{maximize}} \quad \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}},$$

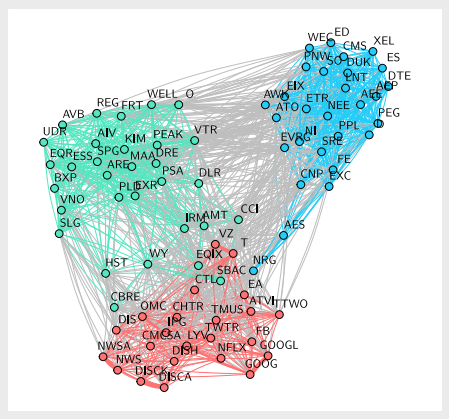
  - $\|\boldsymbol{L}\|_{0,\text{off}}$: promotes true sparsity.
  - Reweighted ℓ_1-norm regularization:
    - Approximates $\|\boldsymbol{L}\|_{0,\text{off}}$ with a smooth concave function, then a convex weighted ℓ_1-norm.
    - Effective for sparse graph learning.

Numerical experiments

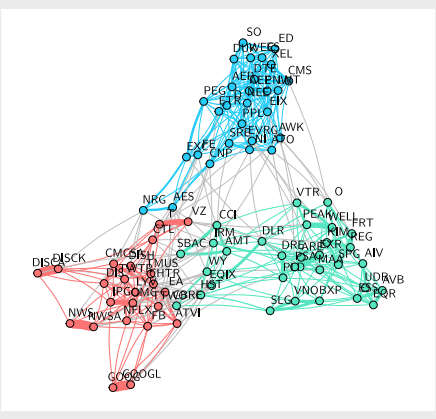
- **Data Overview:**
  - Three years of S&P 500 stock price data (2016-2019) from Industrials, Consumer Staples, and Energy sectors.
  - Data matrix $\boldsymbol{X} \in \mathbb{R}^{T \times N}$ with log-returns of N assets.
- **Normalization:**
  - Assets normalized to have volatility one.
  - Normalization equivalent to using correlation matrix instead of covariance matrix.
  - Ensures uniform dynamic ranges and unit measurements.
- **Financial Assets Correlation:**
  - High correlation typically due to market or other factors.
  - Precision and Laplacian matrices interpret partial correlation, removing common factor effects.
- **Graph Learning Methods for Time Series:**
  - GMRF-based methods preferred for enforcing sparsity:
    - Laplacian-structured GLASSO with ℓ_1-norm.
    - Sparse GMRF graph with ℓ_0 penalty, practically solved via reweighted ℓ_1-norm.
  - Reweighted ℓ_1-norm iterative method shows superior performance.

Numerical experiments

Effect of sparsity regularization term on financial graphs:



GMRF graph (l_1−norm)



GMRF graph (reweighted l_1−norm)

Outline

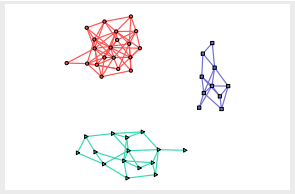
Learning structured graphs

- **Challenges with Structured Graphs:**
  - Learning graphs with specific structures is generally NP-hard.
  - Designing a universal algorithm for structured graphs is challenging.

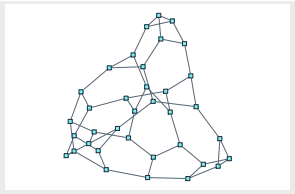
- **Common Types of Structured Graphs:**
  - **Multi-component or k-component graph:** Contains clusters, useful for classification.
  - **Regular graph:** Each node has the same number of neighbors, useful for balanced graphs.
  - **Modular graph:** Satisfies shortest path distance properties among triplets of nodes, useful for social network analysis.
  - **Bipartite graph:** Two types of nodes with inter-connections only.
  - **Grid graph:** Nodes follow a rectangular grid or two-dimensional lattice.
  - **Tree graph:** Undirected graph with exactly one path between any two vertices, resembling a branching structure.

Learning structured graphs

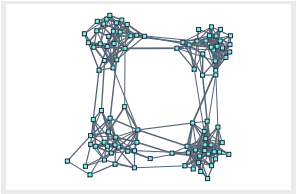
Types of structured graphs:



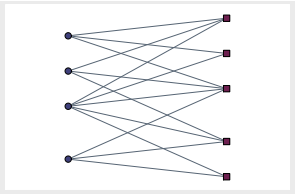
Multi−component graph



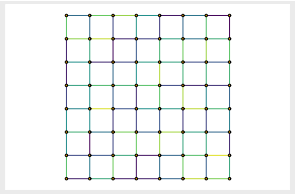
Regular graph



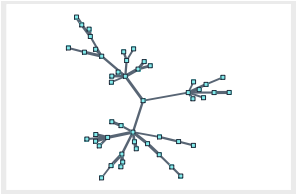
Modular graph



Bipartite graph



Grid graph



Tree graph

Learning structured graphs

- **Controlling Structural Constraints:**
  - Some constraints, like those in grid or regular graphs, are simpler to manage by fixing adjacency and Laplacian matrix elements or controlling node degrees.
  - Constraints like the number of neighbors involve nonconvex conditions.

- **Spectral Properties for Structural Constraints:**
  - More complex structural constraints can be characterized by spectral properties of Laplacian and adjacency matrices.
  - Spectral properties can be enforced in graph learning formulations to achieve desired structures.

k-component graphs

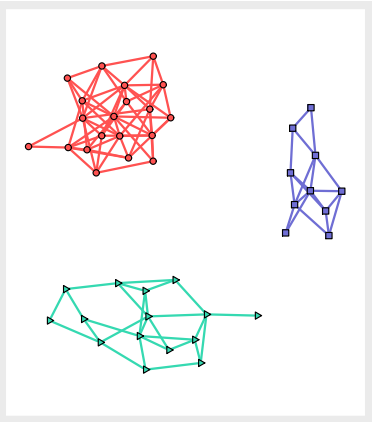
- **Characterization:**
  - A k-component graph is defined by its Laplacian matrix having k zero eigenvalues.
  - This indicates the graph is divided into k distinct clusters or components.
- **Eigenvalue Decomposition:**
  - Laplacian matrix $\boldsymbol{L}$ decomposed as $\boldsymbol{L} = \boldsymbol{U}\text{Diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)\boldsymbol{U}^\top$.
  - $\boldsymbol{U}$ contains eigenvectors, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p$ are eigenvalues in increasing order.
  - For k-component graph: $\lambda_1 = \cdots = \lambda_k = 0$.
- **Low-Rank Property:**
  - $\text{rank}(\boldsymbol{L}) = p - k$, indicating a nonconvex constraint in optimization.
- **Practical Handling via Ky Fan's Theorem:**
  - Enforce $\sum_{i=1}^{k} \lambda_i(\boldsymbol{L}) = 0$ to capture the low-rank property.
  - Optimization involves matrix $\boldsymbol{F}$ as a variable:

$$\sum_{i=1}^{k} \lambda_i(\boldsymbol{L}) = \min_{\boldsymbol{F} \in \mathbb{R}^{p \times k}: \boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}} \text{Tr}(\boldsymbol{F}^\top \boldsymbol{L} \boldsymbol{F}),$$

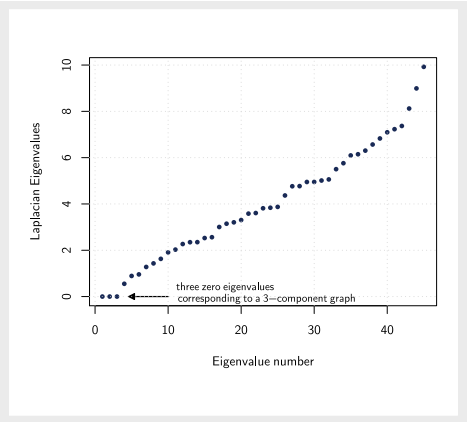
  - This approach simplifies handling the nonconvex constraint of low-rank Laplacian matrices.

k-component graphs

Example of a 3-component graph (3 clusters) with corresponding Laplacian matrix eigenvalues (3 zero eigenvalues):



Graph



Laplacian matrix eigenvalues

Low-rank graph learning formulations

- **Low-Rank Constraint Handling:**
  - Low-rank property $\text{rank}(\boldsymbol{L}) = p - k$ is nonconvex.
  - Enforce $\sum_{i=1}^{k} \lambda_i(\boldsymbol{L}) = 0$ using Ky Fan's theorem.
  - Alternate minimization between $\boldsymbol{L}$ and $\boldsymbol{F}$ for optimization.
- **Regularized Optimization Formulation:**
  - Introduce low-rank constraint as regularization in objective:

$$\underset{\boldsymbol{L},\boldsymbol{F}}{\text{minimize}} \quad f(\boldsymbol{L}) + \gamma \text{Tr}(\boldsymbol{F}^{\mathsf{T}} \boldsymbol{L} \boldsymbol{F}),$$

  - $\boldsymbol{F}$ optimized to be eigenvectors corresponding to k smallest eigenvalues of $\boldsymbol{L}$.
- **Low-Rank Graph Approximation Example:**
  - Given graph Laplacian $\boldsymbol{L}_0$, approximate with low-rank $\boldsymbol{L}$:

$$\underset{\boldsymbol{L} \succeq \boldsymbol{0}, \boldsymbol{F}}{\text{minimize}} \quad \|\boldsymbol{L} - \boldsymbol{L}_0\|_{\text{F}}^2 + \gamma \text{Tr}\left(\boldsymbol{F}^{\mathsf{T}} \boldsymbol{L} \boldsymbol{F}\right),$$

  - Subject to Laplacian constraints and $\boldsymbol{F}^{\mathsf{T}} \boldsymbol{F} = \boldsymbol{I}$.

Low-rank graph learning formulations

- **Low-Rank Sparse GMRF Graphs Example:**
  - Incorporate low-rank regularization into sparse GMRF graph learning:

$$\underset{\boldsymbol{L}\succeq\mathbf{0},\boldsymbol{F}}{\text{maximize}} \quad \log \text{gdet}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} - \gamma\text{Tr}\left(\boldsymbol{F}^{\mathsf{T}}\boldsymbol{L}\boldsymbol{F}\right),$$

  - Subject to Laplacian constraints and $\boldsymbol{F}^{\mathsf{T}}\boldsymbol{F} = \boldsymbol{I}$.
- **Avoiding Trivial Solutions:**
  - To prevent isolated nodes, control node degrees with constraint $\text{diag}(\boldsymbol{L}) = \mathbf{1}$.
- **R Package for Spectral Constraints:**
  - `spectralGraphTopology` provides functions for graph learning with spectral constraints.

Bipartite graphs

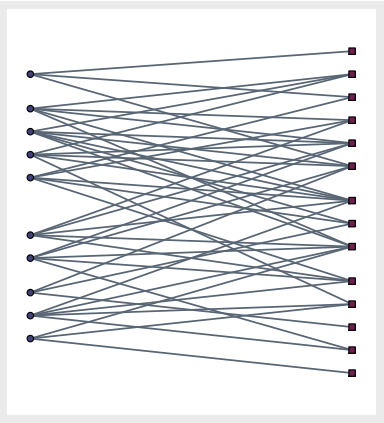
- **Adjacency Matrix Symmetric Eigenvalues:**
  - Bipartite graphs have adjacency matrix eigenvalues symmetric around zero.
  - Eigenvalue decomposition: $\boldsymbol{W} = \boldsymbol{V}\text{Diag}(\psi_1, \psi_2, \ldots, \psi_p)\boldsymbol{V}^\top$.
  - Symmetry condition: $\psi_i = -\psi_{p-i}, \quad \forall i$.
- **Laplacian Matrix Structure:**
  - Alternative characterization via Laplacian matrix:

$$\boldsymbol{L} = \begin{bmatrix} \text{Diag}(\boldsymbol{B}\boldsymbol{1}) & -\boldsymbol{B} \\ -\boldsymbol{B}^\top & \text{Diag}(\boldsymbol{B}^\top\boldsymbol{1}) \end{bmatrix},$$

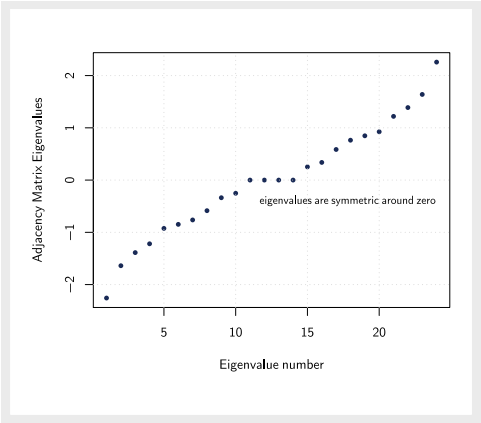
  - $\boldsymbol{B} \in \mathbb{R}_+^{r\times q}$ represents edge weights between two node types.
  - Constructed $\boldsymbol{L}$ inherently satisfies $\boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, L_{ij} = L_{ji} \leq 0, \forall i \neq j$.
- **Optimization Challenges:**
  - Enforcing symmetric eigenvalues is nonconvex and complex.
  - Laplacian matrix structure provides a more practical approach to enforce bipartite properties.
- **R Package for Bipartite Graphs:**
  - `finbipartite` offers methods for solving bipartite graph problems.

Bipartite graphs

Example of a bipartite graph with corresponding adjacency matrix eigenvalues:



Graph



Adjacency matrix eigenvalues

Bipartite graph learning formulations

- **Bipartite Graph Approximation:**
  - Goal: Find closest bipartite graph approximation to a given graph $\boldsymbol{L}_0$.
  - Formulation:

$$\begin{array}{ll} \underset{\boldsymbol{L},\boldsymbol{B}}{\text{minimize}} & \|\boldsymbol{L}-\boldsymbol{L}_0\|_{\mathrm{F}}^2 \\ \text{subject to} & \boldsymbol{L}=\begin{bmatrix} \mathrm{Diag}(\boldsymbol{B}\mathbf{1}) & -\boldsymbol{B} \\ -\boldsymbol{B}^{\mathrm{T}} & \mathrm{Diag}(\boldsymbol{B}^{\mathrm{T}}\mathbf{1}) \end{bmatrix} \\ & \boldsymbol{B}\geq\mathbf{0}, \quad \boldsymbol{B}\mathbf{1}=\mathbf{1}. \end{array}$$

  - Objective: Minimize Frobenius norm difference between $\boldsymbol{L}$ and $\boldsymbol{L}_0$.
  - Constraints: Enforce bipartite structure and non-negativity of $\boldsymbol{B}$.

Bipartite graph learning formulations

- **Bipartite Graphs from Sparse GMRFs:**
  - Goal: Learn a sparse bipartite graph under a GMRF framework.
  - Formulation:

$$\begin{aligned} \underset{\boldsymbol{L}\succeq\mathbf{0},\boldsymbol{B}}{\text{maximize}} \quad & \log \operatorname{gdet}(\boldsymbol{L}) - \operatorname{Tr}(\boldsymbol{L}\boldsymbol{S}) - \rho\|\boldsymbol{L}\|_{0,\text{off}} \\ \text{subject to} \quad & \boldsymbol{L} = \begin{bmatrix} \operatorname{Diag}(\boldsymbol{B}\mathbf{1}) & -\boldsymbol{B} \\ -\boldsymbol{B}^{\top} & \operatorname{Diag}(\boldsymbol{B}^{\top}\mathbf{1}) \end{bmatrix} \\ & \boldsymbol{B} \geq \mathbf{0}, \quad \boldsymbol{B}\mathbf{1} = \mathbf{1}. \end{aligned}$$

  - Objective: Maximize log-determinant of $\boldsymbol{L}$ minus trace term and sparsity penalty.
  - Constraints: Enforce bipartite structure, non-negativity, and uniform distribution of connections in $\boldsymbol{B}$.

Numerical experiments

- **Data Overview:**
  - S&P 500 stock price data from 2016-2019 for Industrials, Consumer Staples, and Energy sectors of the S&P 500 index.
- **Stock Classification into Sectors and Industries:**
  - Stocks grouped into sectors and industries for investment diversification.
  - Classification criteria vary, including production-oriented and market-oriented approaches.
- **Major Sector Classification Systems:**
  - **GICS:** Developed by MSCI and S&P for classifying companies into industry groups, sectors, and sub-industries.
  - **ICB:** Created by Dow Jones and FTSE for categorizing companies into industries, supersectors, sectors, and subsectors.
  - **TRBC:** Thomson Reuters' system for classifying companies into economic sectors, business sectors, and industries.

Numerical experiments

- Major sector classification systems:

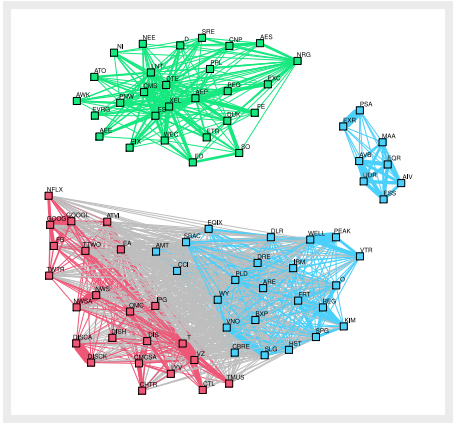
Level/System	GICS	ICB	TRBC
1st	11 Sectors	10 Industries	10 Economic Sectors
2nd	24 Industry Groups	19 Supersectors	28 Business Sectors
3rd	68 Industries	41 Sectors	56 Industry Groups
4th	157 Sub-Industries	114 Subsectors	136 Industries

- **Relevance of Classification Systems:**
  - Each system groups stocks differently, with some similarities.
  - Not always clear which classification is most relevant for portfolio investment.
- **Data-Oriented Approach to Stock Classification:**
  - With data availability, traditional classification systems can be bypassed.
  - Learn stock graph from data to discover natural clusters and relationships.
  - Enforce a k-component graph for automatic clustered graph generation.

Numerical experiments: Two-stage versus joint design of k-component financial graphs

- **Two-Stage Approach:**
  - **Step 1:** Learn a connected graph using GMRF design (e.g., GLASSO or sparse GMRF).
  - **Step 2:** Perform a low-rank approximation to the learned graph.
- **Joint Approach:**
  - Enforce the low-rank property directly in the GMRF formulation.
  - Expected to outperform the two-stage approach by integrating all constraints and objectives from the start.
- **Comparison and Effectiveness:**
  - Joint design significantly superior to two-stage design.
  - Demonstrated by the clear difference in outcomes when employing each approach.
- **Considerations for Low-Rank Graph Learning:**
  - Importance of controlling node degrees to avoid trivial solutions or isolated nodes.
  - Ensures the practical utility and connectivity of the learned graph.

Numerical experiments: Two-stage versus joint design of k-component financial graphs



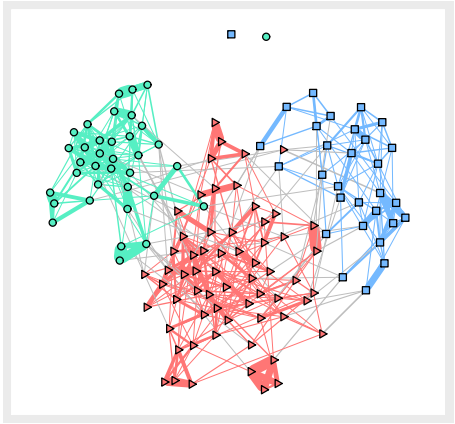
Two−stage procedure

Joint procedure

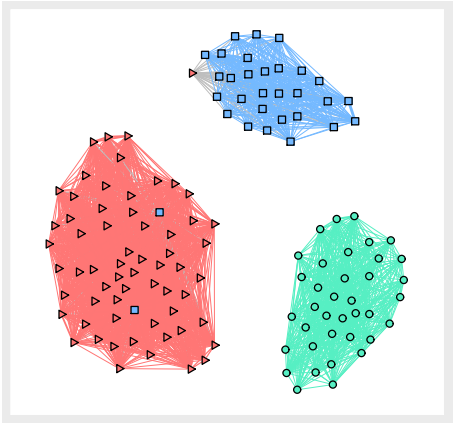
Numerical experiments: Isolated nodes in low-rank designs for k-component graphs

- **Issue with Isolated Nodes:**
  - Imposing a low-rank structure to learn a k-component graph can result in isolated nodes.
  - These nodes artificially increase the number of clusters by being disconnected.
- **Solution:**
  - Control the degrees of nodes to ensure they are nonzero and preferably balanced.
  - Prevents the occurrence of isolated nodes, maintaining connectivity within the graph.
- **Impact of Degree Control:**
  - Significant improvement in graph structure by avoiding isolated nodes.
  - Ensures more meaningful and connected clusters in the learned graph.

Numerical experiments: Isolated nodes in low-rank (clustered) designs for k-component financial graphs



Without degree control



With degree control

Outline

Transition from Gaussian to Heavy-Tailed Graphs

- **Gaussian Maximum Likelihood Estimation (MLE) based on:**

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right),$$

leading to optimization problem for Laplacian matrix $\boldsymbol{L}$:

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \operatorname{Tr}(\boldsymbol{L}\boldsymbol{S}) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j. \end{array}$$

- **Heavy-Tailed Distribution Model:** Student t distribution:

$$f(\boldsymbol{x}) \propto \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}} \left(1 + \frac{1}{\nu}(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)^{-\frac{p+\nu}{2}},$$

leading to a new optimization problem:

$$\underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\text{maximize}} \quad \log \operatorname{gdet}(\boldsymbol{L}) - \frac{p+\nu}{T}\sum_{t=1}^{T} \log\left(1 + \frac{1}{\nu}(\boldsymbol{x}^{(t)})^\mathsf{T}\boldsymbol{L}\boldsymbol{x}^{(t)}\right)$$

Learning heavy-tailed graphs

- **Majorization-Minimization (MM) Framework:**
  - Nonconvex heavy-tailed problem solved iteratively using MM.
  - Logarithm upper bound used for simplification:

$$\log\left(1+\frac{1}{\nu}(\boldsymbol{x}^{(t)})^{\mathsf{T}}\boldsymbol{L}\boldsymbol{x}^{(t)}\right) \leq \log\left(1+\frac{1}{\nu}(\boldsymbol{x}^{(t)})^{\mathsf{T}}\boldsymbol{L}_0\boldsymbol{x}^{(t)}\right) + \frac{\nu+(\boldsymbol{x}^{(t)})^{\mathsf{T}}\boldsymbol{L}\boldsymbol{x}^{(t)}}{\nu+(\boldsymbol{x}^{(t)})^{\mathsf{T}}\boldsymbol{L}_0\boldsymbol{x}^{(t)}} - 1.$$

- **Iterative Gaussianized Problem Solving:**
  - MM algorithm solves sequence of Gaussianized problems:

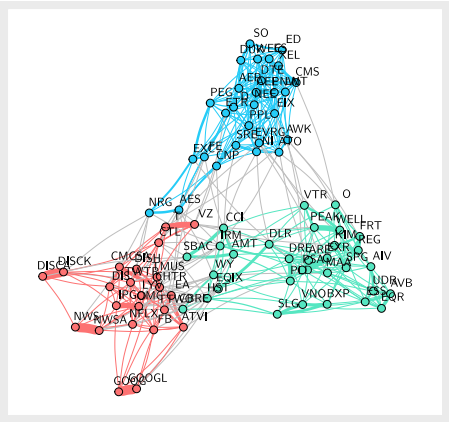
$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\text{maximize}} & \log \operatorname{gdet}(\boldsymbol{L}) - \operatorname{Tr}(\boldsymbol{L}\boldsymbol{S}^k) \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}, \quad L_{ij} = L_{ji} \leq 0, \quad \forall i \neq j, \end{array}$$

    where $\boldsymbol{S}^k$ is a weighted sample covariance matrix.
- **R Package for Heavy-Tailed Graph Learning:**
  - `fingraph` R package provides algorithms for learning from heavy-tailed data.
  - Functions include `learn_regular_heavytail_graph()` and `learn_kcomp_heavytail_graph()` for solving heavy-tailed graph learning problems with additional constraints.

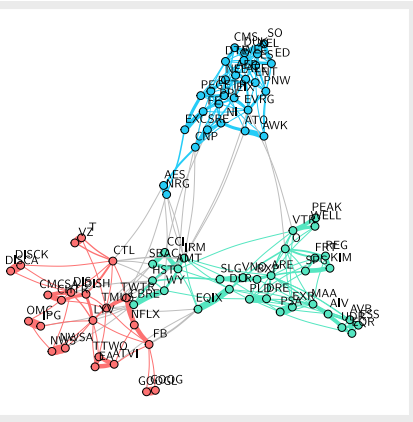
Numerical experiments: From Gaussian to heavy-tailed graphs

- **Stock Price Data Analysis:**
  - Data from three S&P 500 sectors: Industrials, Consumer Staples, Energy.
  - Time period: 2016-2019.
- **Comparison of Graphical Models:**
  - Gaussian MRF with concave sparsity regularizer used for Gaussian case.
  - Heavy-tailed MRF formulation used for non-Gaussian case.
- **Results and Conclusions:**
  - Next figure shows MRF results under Gaussian and heavy-tailed assumptions.
  - Heavy-tailed graphs are more suitable for financial data analysis.

Numerical experiments: Gaussian versus heavy-tailed graph learning with stocks



Sparse Gaussian MRF graph

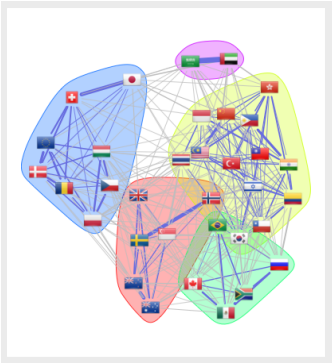


Heavy−tailed MRF graph

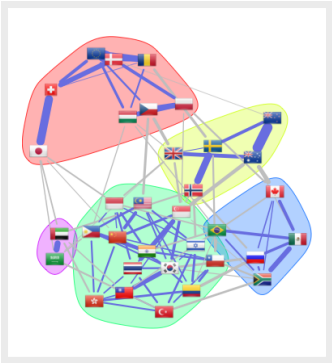
Numerical experiments: k-component graphs

- **FX Market Data Analysis:**
  - Data: 34 most traded currencies from Jan. 2nd, 2019 to Dec. 31st, 2020.
  - Data matrix: Log-returns of currency prices relative to USD.
- **Graph Learning Comparisons:**
  - GMRF with ℓ_1-norm regularizer vs. concave sparsity regularizer.
  - Heavy-tailed MRF formulation for non-Gaussian data.
  - GMRF with ℓ_1-norm does not produce sparse graphs.
  - Concave sparsity regularizer and heavy-tailed MRF yield cleaner graphs.
  - Heavy-tailed MRF highlights expected correlations between geographically close currencies (e.g., {Hong Kong, China}, {Taiwan, South Korea}, {Poland, Czech Republic}).
- **k-Component Graphs for Clustering:**
  - 9-component graphs show clear clustering.
  - Heavy-tailed MRF graph provides clearer, more reasonable clusters (e.g., {New Zealand, Australia}, {Poland, Czech Republic, Hungary}).
  - Heavy-tailed MRF with low-rank structure avoids isolated nodes, unlike GMRF formulations.

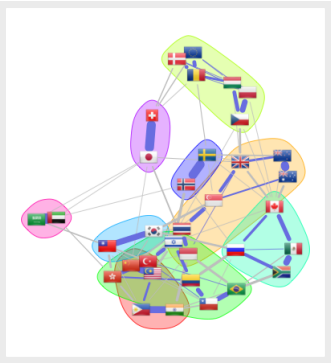
Numerical experiments: Gaussian versus heavy-tailed graph learning with FX



GMRF graph (l_1−norm)

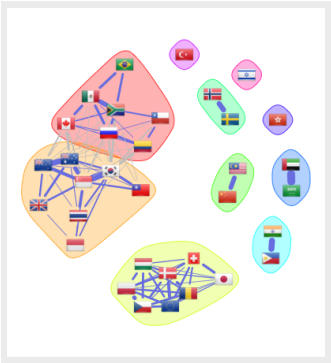


GMRF graph (reweighted l_1−norm)

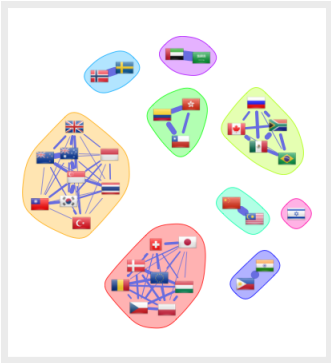


Heavy−tailed MRF graph

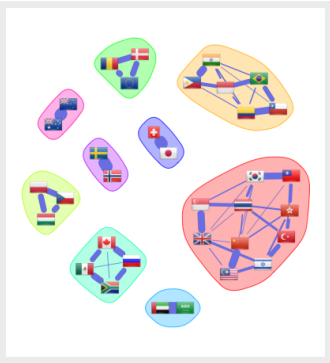
Numerical experiments: Gaussian versus heavy-tailed multi-component graph learning with FX



GMRF graph (l_1−norm)



GMRF graph (reweighted l_1−norm)



Heavy−tailed MRF graph

Outline

Learning time-varying graphs

- **Dynamic vs. Static Graphs:**
  - Static graphs do not capture time variations in dynamic systems (e.g., financial markets).
  - Dynamic graphs are essential for understanding time-varying behaviors.
- **Challenges in Learning Dynamic Graphs:**
  - Formulation complexity and increased variable count.
  - Limited literature due to these challenges.
- **Naive Approach for Dynamic Graphs:**
  - Divide observations into T chunks, learn graphs $\boldsymbol{L}_t$ independently.
  - Drawbacks: fewer observations per chunk, potential lack of time-consistency.

Learning time-varying graphs

- **Time-Consistent Dynamic Graph Learning:**
  - Regularization term $d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t)$ ensures smooth transitions.
  - Frobenius norm: $d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t) = \|\boldsymbol{L}_{t-1} - \boldsymbol{L}_t\|_{\mathrm{F}}^2$.
  - ℓ_1-norm: $d(\boldsymbol{L}_{t-1}, \boldsymbol{L}_t) = \|\boldsymbol{L}_{t-1} - \boldsymbol{L}_t\|_1$.
- **Dynamic Graph Learning Formulation:**
  - Objective: Minimize trace and log-det terms, plus regularization for time-consistency.
  - Constraints ensure positive semi-definiteness and structural conditions.
  - Hyper-parameter δ controls time-consistency level.
- **Solution and Estimation:**
  - Dynamic graph solution $\hat{\boldsymbol{L}}_{t|T}$ considers all T chunks (with look-ahead bias).
  - Rolling window approach for causal estimate $\hat{\boldsymbol{L}}_{t|t}$ avoids look-ahead bias.

Outline

Summary

The first application of graphical models in financial markets goes back to 1999, using a simple correlation graph. Since then, numerous methods have been proposed.

Among the methods herein overviewed, only a few are suitable for financial time series and produce desirable graphs:

- Sparse GMRF graphs are a good starting point.
- k-component graphs with low-rank structure and degree control can be used for data-driven asset clustering.
- Due to the heavy-tailed nature of financial data, heavy-tailed graph models are more suitable than Gaussian ones and can be solved iteratively.

References I

Cardoso, J. V. M., and D. P. Palomar. 2020. "Learning Undirected Graphs in Financial Markets." In *Proceedings of the 54th Asilomar Conference on Signals, Systems and Computers*. Pacific Grove, CA, USA.

Cardoso, J. V. M., J. Ying, and D. P. Palomar. 2021. "Graphical Models for Heavy-Tailed Markets." In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Virtual.

———. 2022a. "Nonconvex Graph Learning: Sparsity, Heavy-Tails, and Clustering." In *Signal Processing and Machine Learning Theory, Digital Signal Processing Series*. Elsevier.

———. 2022b. "Learning Bipartite Graphs: Heavy Tails and Multiple Components." In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. New Orleans, LA, USA.

Dong, X., D. Thanou, M. Rabbat, and P. Frossard. 2019. "Learning Graphs from Data: A Signal Representation Perspective." *IEEE Signal Processing Magazine* 36 (3): 44–63.

Egilmez, H. E., E. Pavez, and A. Ortega. 2017. "Graph Learning from Data Under Laplacian and Structural Constraints." *IEEE Journal of Selected Topics in Signal Processing* 11 (6): 825–41.

Kolaczyk, E. D. 2009. *Statistical Analysis of Network Data: Methods and Models*. New York: Springer-Verlag.

References II

Kumar, S., J. Ying, J. V. M. Cardoso, and D. P. Palomar. 2019. "Structured Graph Learning via Laplacian Spectral Constraints." In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada.

———. 2020. "A Unified Framework for Structured Graph Learning via Spectral Constraints." *Journal of Machine Learning Research (JMLR)*, 1–60.

Lake, B., and J. Tenenbaum. 2010. "Discovering Structure by Learning Sparse Graphs." In *Proceedings of the 33rd Annual Cognitive Science Conference*.

Lauritzen, S. 1996. *Graphical Models*. Oxford: Oxford University Press.

Mantegna, R. N. 1999. "Hierarchical Structure in Financial Markets." *The European Physical Journal B-Condensed Matter and Complex Systems* 11: 193–97.

Marti, G., F. Nielsen, M. Binkowski, and P. Donnat. 2021. "A Review of Two Decades of Correlations, Hierarchies, Networks and Clustering in Financial Markets." In *Progress in Information Geometry*, edited by F. Nielsen, 245–74. Springer.

Mateos, G., S. Segarra, A. G. Marques, and A. Ribeiro. 2019. "Connecting the Dots." *IEEE Signal Processing Magazine* 36 (3): 16–43.

References III

Palomar, D. P. 2025. *Portfolio Optimization: Theory and Application*. Cambridge University Press.

Ying, J., J. V. M. Cardoso, and D. P. Palomar. 2020. "Nonconvex Sparse Graph Learning Under Laplacian Constrained Graphical Model." In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Virtual.

Zhao, L., Y. Wang, S. Kumar, and D. P. Palomar. 2019. "Optimization Algorithms for Graph Laplacian Estimation via ADMM and MM." *IEEE Transactions on Signal Processing* 67 (16): 4231–44.