Unraveling the Connections: Learning Undirected Graphs in Financial Markets

PhD Defense by

José Vinícius de Miranda Cardoso

Hong Kong University of Science and Technology
Department of Electronic and Computer Engineering

May 24th, 5pm GMT+8, 5am GMT-4, 2023

Based on

Primary Publications

- **NeurIPS'22 Cardoso, J. V. M.**, Ying, J., and Palomar, D. P. "Learning Bipartite Graphs: Heavy Tails and Multiple Components", *Advances in Neural Information Processing Systems*, 14044–14057 (35), 2022
- **NeurIPS'21 Cardoso, J. V. M.**, Ying, J., and Palomar, D. P. "Graphical Models in Heavy-Tailed Markets", *Advances in Neural Information Processing Systems*, 19989–20001 (34), 2021

Relevant Publications

- **NeurIPS'20** Ying, J., **Cardoso, J. V. M.**, and Palomar, D. P. "Nonconvex Sparse Graph Learning under Laplacian Constrained Graphical Model", *Advances in Neural Information Processing Systems*, 7101–7113 (33), 2020
- **NeurIPS'19** Kumar, S., Ying, J., **Cardoso, J. V. M.**, and Palomar, D. P. "Structured Graph Learning Via Laplacian Spectral Constraints" *Advances in Neural Information Processing Systems*, (32), 2019

Contents

1. Introduction & Background
   Motivation, Financial Data, Graphs, Learning Graphs from Data as an Optimization Problem
2. Learning Graphs in Heavy Tailed Markets
   Heavy Tails, k-component Graphs, and Clustering
3. Learning Bipartite Graphs
   Bipartite structure, Stock Classification
4. Conclusion
   Final Remarks

Introduction & Background

Motivation

what?

- how to go from a (financial) dataset $\boldsymbol{X}$ to a graph $\mathcal{G}$?

why?

- **estimating** relationships among (financial) entities $\Rightarrow$ enhance our **understanding** about their behavior

how?

- statistical estimation theory, optimization theory, numerical optimization frameworks

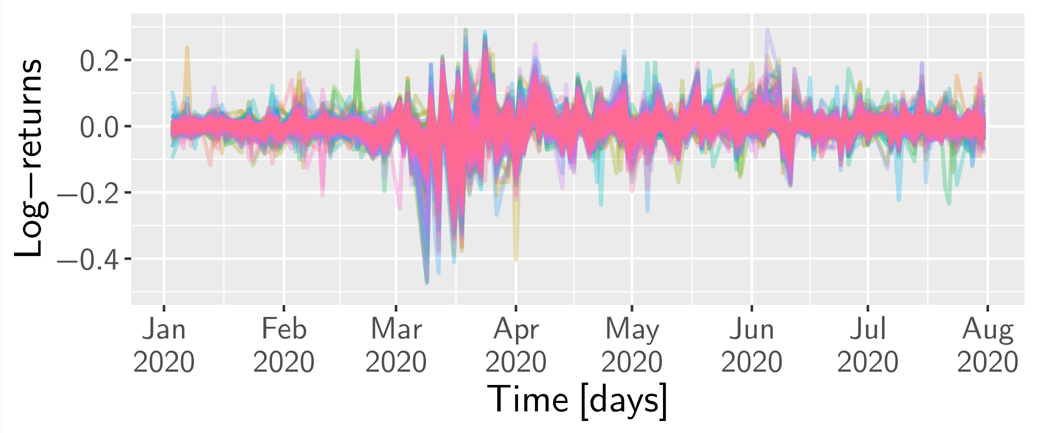
Financial Data

Mathematically:

$$\boldsymbol{X} \in \mathbb{R}^{n \times p}$$

- n is the number of observations
- p is the number of financial instruments

In real-life:

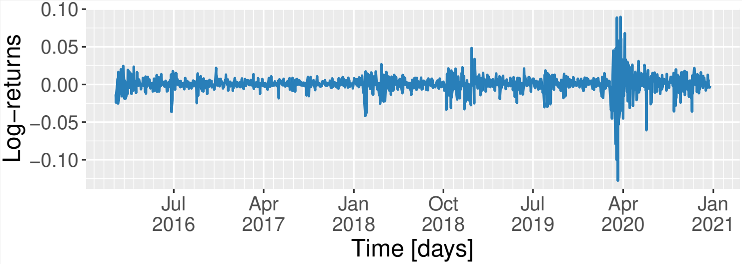


Log−returns
0.2
0.0
−0.2
−0.4
Jan 2020
Feb 2020
Mar 2020
Apr 2020
May 2020
Jun 2020
Jul 2020
Aug 2020
Time [days]

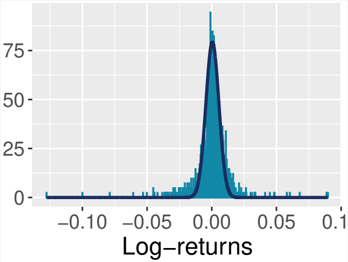
Stylized facts about finance data

Fact #1: Financial data is heavy-tailed

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



(a) S&P500 log-returns.



(b) Histogram of S&P500 log-returns.

S. I. Resnick. Heavy-Tail Phenomena: Probabilistic and Statistical Modeling. Springer, 2007.

Fact #2: Stock markets are modular (stocks are more correlated within their sector)

cf. M. L. de Prado. Machine Learning for Asset Managers (Elements in Quantitative Finance). Cambridge University Press, 2020.

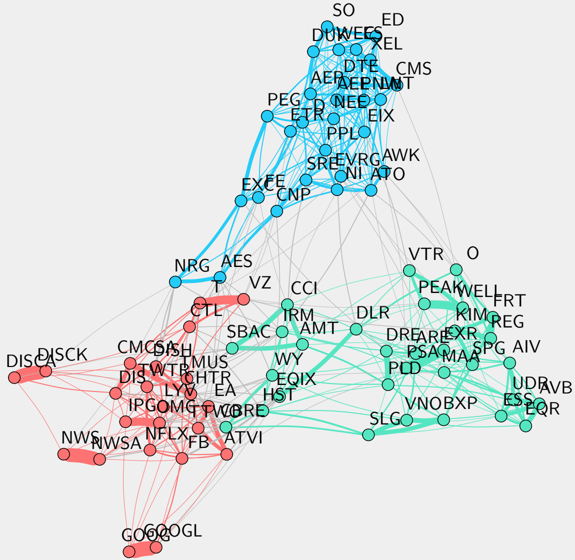
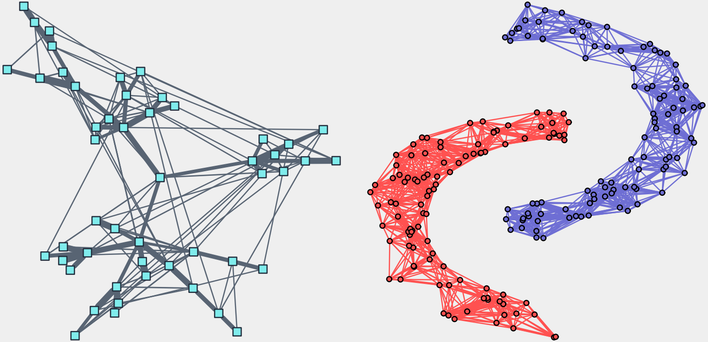


Figure: Graph showing three stock sectors of the US Stock Market, namely: Communication Services, Utilities, and Real Estate.

Graphs

Graphs

- a **set of nodes**
- a **set of edges** connecting these nodes
- many different flavours: directed (undirected), weighted (unweighted), single or multiple components
- in this thesis: **undirected**, **weighted**



Undirected Weighted Graphs

mathematically:

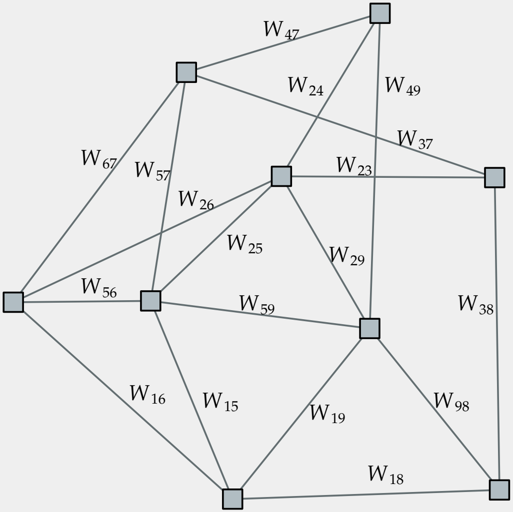
$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$$

- $\mathcal{V} = \{1, 2, \dots, p\}$
- $\mathcal{E} \subseteq \{\{u, v\} : u, v \in \mathcal{V}, u \neq v\}$
- Adjacency Matrix: $\boldsymbol{W} \in \mathbb{R}_+^{p \times p}, \boldsymbol{W} = \boldsymbol{W}^\top, \mathrm{diag}(\boldsymbol{W}) = \mathbf{0}$
- Laplacian Matrix: $\boldsymbol{L} = \mathrm{Diag}(\boldsymbol{W}\mathbb{1}) - \boldsymbol{W}$
- Degree Matrix: $\boldsymbol{D} = \mathrm{Diag}(\boldsymbol{L}) = \mathrm{Diag}(\boldsymbol{W}\mathbb{1})$
- Number of components k: $\mathrm{rank}(\boldsymbol{L}) = p - k$

How to go from data to graphs?

$$\boldsymbol{X} \in \mathbb{R}^{n \times p}$$

- columns of $\boldsymbol{X}$ are **signals** generated at each node
- naive construction: pairwise correlation, norm difference, etc
- pro: **interpretability**
- con: **ignores joint dependencies** among nodes in the whole graph
- there must be a better way...



Laplacian Matrix as Precision Matrix

- **Gaussian Markov Random Field (GMRF) assumption**

$$\boldsymbol{X} \sim \mathcal{N}\left(0, \boldsymbol{L}^{\dagger}\right)$$

(P1) $\boldsymbol{L}\mathbf{1} = \mathbf{0}$

(P2) $L_{ij} = L_{ji} \leq 0 \; \forall i \neq j$

conditional correlation: $-\frac{L_{ij}}{\sqrt{L_{ii}L_{jj}}} \geq 0$

Laplacian Matrix as a Precision Matrix

- Penalized Maximum Likelihood Estimator (Lake and Tenenbaum, 2010; Egilmez et al., 2017; Zhao et al., 2019)

$$\begin{array}{ll} \underset{\boldsymbol{L}\succeq\boldsymbol{0}}{\text{minimize}} & \underbrace{\operatorname{tr}\left(\boldsymbol{L}\boldsymbol{S}\right) - \log\det{}^{*}\left(\boldsymbol{L}\right)}_{\text{negative log-likelihood}} + \underbrace{\alpha h(\boldsymbol{L})}_{\text{regularizer}}, \\ \text{subject to} & \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0},\ L_{ij} = L_{ji} \leq 0, \end{array}$$

- where $\boldsymbol{S} = \frac{1}{n}\boldsymbol{X}^{\top}\boldsymbol{X}$ is the sample covariance matrix
- $\det^{*}$: pseudo-det, product of positive eigenvalues
- $h(\cdot)$ is a regularization function to impose properties on the estimated graph, *e.g.*, sparsity

Maximum Likelihood Estimator

- For connected graphs: $\det^*(\boldsymbol{L}) = \det(\boldsymbol{L} + \boldsymbol{J})$, $\boldsymbol{J} \triangleq \frac{1}{p}\mathbf{1}\mathbf{1}^\top$ (Egilmez et al., 2017)

$$\begin{array}{ll} \underset{\boldsymbol{L} \succeq \mathbf{0}}{\text{minimize}} & \operatorname{tr}(\boldsymbol{L}\boldsymbol{S}) - \log\det(\boldsymbol{L} + \boldsymbol{J}) + \alpha h(\boldsymbol{L}), \\ \text{subject to} & \boldsymbol{L}\mathbf{1} = \mathbf{0},\ L_{ij} = L_{ji} \leq 0, \end{array}$$

- pro: convex problem provided that $h(\cdot)$ is convex
- con: not scalable for disciplined convex programming languages ($p > 100$)
- con: may not be adequate in case $\boldsymbol{X}$ is heavy-tailed distributed
- **challenge**: develop scalable numerical optimization routines
- For $h(\boldsymbol{L}) = \|\boldsymbol{L}\|_1$: Block Coordinate Descent (BCD) (Egilmez et al., 2017), Alternating Direction Method of Multipliers (ADMM) (Zhao et al., 2019)
- For $h(\boldsymbol{L}) =$ concave_regularizer$(\boldsymbol{L})$: Majorization-Minimization (MM) (Sun et al., 2017) + Projected Gradient Descent (PGD) (Ying et al., 2020)

Contents

1. Introduction & Background

   Graphs, learning graphs from data as an optimization problem, and financial data

2. **Learning Graphs in Heavy Tailed Markets**

   State-of-the-art k-component graphs, heavy-tails, and clustering

3. Learning Bipartite Graphs

   Bipartite structure, stock classification

4. Conclusion

   Final remarks and future works

Learning Graphs in Heavy Tailed Markets

State-of-the-art: k-component Graphs

- **Constrained Laplacian Rank** (Nie et al., 2016)
- **key property**: $\text{rank}(\boldsymbol{L}) = p - k$, k is the number of components
- **Two-stage approach:**

1. Obtain an initial adjacency matrix $\boldsymbol{W}_0$ (correlation graph, convex GMRF)
2. Find a projection of $\boldsymbol{W}_0$ that contains k-components:

$$\begin{array}{ll} \underset{\boldsymbol{W}, \boldsymbol{L} \succeq \boldsymbol{0}}{\text{minimize}} & \|\boldsymbol{W} - \boldsymbol{W}_0\|_{\mathsf{F}}^2, \\ \text{subject to} & \boldsymbol{W}\boldsymbol{1} = \boldsymbol{1}, \ \text{rank}(\boldsymbol{L}) = p - k, \\ & \boldsymbol{L} = \text{Diag}\left(\frac{\boldsymbol{W}^\top + \boldsymbol{W}}{2}\right) - \frac{\boldsymbol{W}^\top + \boldsymbol{W}}{2} \end{array}$$

- pro: simple approach with a fast alternating optimization algorithm
- con: graph estimation and k-component identification not done jointly
- con: no statistical foundation

State-of-the-art: k-component Graphs

- **Spectral Regularization** (Kumar et al., 2019)
- **key idea**: $\boldsymbol{L} = \boldsymbol{U}\mathsf{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$

$$\begin{array}{ll}
\underset{\boldsymbol{L},\boldsymbol{U},\boldsymbol{\lambda}}{\text{minimize}} & \underbrace{\operatorname{tr}(\boldsymbol{L}\boldsymbol{S}) - \sum_{i=1}^{p-k}\log(\lambda_i)}_{\text{neg log-likelihood}} + \underbrace{\frac{\eta}{2}\left\|\boldsymbol{L} - \boldsymbol{U}\mathsf{Diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top\right\|_{\mathrm{F}}^2}_{k\text{-component structure}}, \\
\text{subject to} & \boldsymbol{L} \succeq \boldsymbol{0},\ \boldsymbol{L}\boldsymbol{1} = \boldsymbol{0},\ L_{ij} = L_{ji} \le 0, \\
& \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I},\ \boldsymbol{U} \in \mathbb{R}^{p\times(p-k)}, \\
& \boldsymbol{\lambda} \in \mathbb{R}_+^{p-k},\ c_1 < \lambda_1 < \cdots < \lambda_{p-k} < c_2,\ c_1 > 0
\end{array}$$

- pros: statistically motivated, fast BCD optimization algorithm
- cons: **allows isolated nodes**, tunning η is not easy in practice

Graph Operators

- **Laplacian operator** (Kumar et al., 2020) $\mathcal{L} : \mathbb{R}_+^{p(p-1)/2} \to \mathbb{S}_+^p$, which takes a nonnegative vector $\boldsymbol{w}$ and outputs a Laplacian matrix $\boldsymbol{L}$.

Example

For $\boldsymbol{w} = [w_1, w_2, w_3]^\top$, $\mathcal{L}\boldsymbol{w} = \begin{bmatrix} w_1 + w_2 & -w_1 & -w_2 \\ -w_1 & w_1 + w_3 & -w_3 \\ -w_2 & -w_3 & w_2 + w_3 \end{bmatrix}$

- **Degree operator** (Cardoso et al., 2021): $\mathfrak{d}\boldsymbol{w} \triangleq \text{diag}\left(\mathcal{L}\boldsymbol{w}\right)$

Proposed Formulation: Connected Graphs

- **goal**: address the **heavy-tail nature** of financial returns and **leverage** that fact into the problem of **graph learning**
- assuming a **Student-t** data generating process

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

- where $\boldsymbol{\Theta}$ is the so-called Inverse Scatter Matrix modeled as a Laplacian matrix
- robustified version of the MLE for **connected** graphs, *i.e.*

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

Proposed Formulation: k-component Graphs

- rank$(\mathcal{L}\boldsymbol{w}) = p - k$
- Fan's theorem (Fan, 1949):

$$\textstyle\sum_{i=1}^{k} \lambda_i\left(\mathcal{L}\boldsymbol{w}\right) = \underset{\boldsymbol{V}\in\mathbb{R}^{p\times k},\boldsymbol{V}^\top\boldsymbol{V}=\boldsymbol{I}}{\text{minimize}} \ \operatorname{tr}\left(\boldsymbol{V}^\top\mathcal{L}\boldsymbol{w}\boldsymbol{V}\right)$$

- k-component heavy-tailed graph learning:

$$\begin{array}{ll}\underset{\boldsymbol{w}\geq\boldsymbol{0},\boldsymbol{\Theta}\succeq 0,\boldsymbol{V}}{\text{minimize}} & \dfrac{p+\nu}{n}\textstyle\sum_{i=1}^{n}\log\left(1+\dfrac{\boldsymbol{x}_i^\top\mathcal{L}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right) - \log\det{}^*(\boldsymbol{\Theta}) + \underbrace{\eta\operatorname{tr}(\boldsymbol{V}^\top\mathcal{L}\boldsymbol{w}\boldsymbol{V})}_{\text{k-component regularizer}},\\ \text{subject to} & \boldsymbol{\Theta} = \mathcal{L}\boldsymbol{w},\ \text{rank}(\boldsymbol{\Theta}) = p-k,\ \underbrace{\mathfrak{d}\boldsymbol{w} = \boldsymbol{d}}_{\text{avoids isolated nodes}},\ \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I},\ \boldsymbol{V}\in\mathbb{R}^{p\times k}.\end{array}$$

- algorithms are derived from optimization frameworks: ADMM and MM

ADMM + MM Solution

- ADMM **key steps**: 1) divide, 2) relax, and 3) optimize alternatively
  - **build** the Augmented Lagrangian, e.g., connected graph Student-t case:

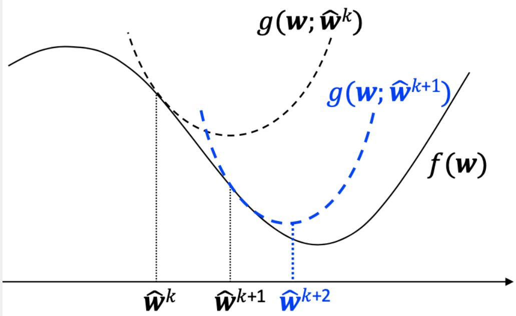
$$L_\rho(\boldsymbol{\Theta}, \boldsymbol{w}, \boldsymbol{Y}, \boldsymbol{y}) = \underbrace{\frac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1 + \frac{\boldsymbol{x}_i^\top \mathcal{L}\boldsymbol{w}\boldsymbol{x}_i}{\nu}\right) - \log\det\left(\boldsymbol{\Theta} + \boldsymbol{J}\right)}_{\text{objective function}}$$

$$+ \underbrace{\langle \boldsymbol{y}, \mathfrak{d}\boldsymbol{w} - \boldsymbol{d}\rangle + \frac{\rho}{2}\left\|\mathfrak{d}\boldsymbol{w} - \boldsymbol{d}\right\|_2^2 + \langle \boldsymbol{Y}, \boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}\rangle + \frac{\rho}{2}\left\|\boldsymbol{\Theta} - \mathcal{L}\boldsymbol{w}\right\|_{\mathrm{F}}^2}_{\text{relaxed constraints}},$$

  - **optimize** over $\boldsymbol{w}, \boldsymbol{\Theta}, \boldsymbol{Y}, \boldsymbol{y}$ in an alternate fashion
  - observation: **not all** constraints have to be relaxed

ADMM + MM Solution

- MM key idea: approximate and solve
  - **approximate** nonconvex terms
  - **solve** the approximated problem
  - **iterate** until convergence
- heavy-tailed formulation:
  - approximate the log function by its 1st-order Taylor expansion
  - $\log\left(1+\frac{t}{b}\right) \leq \frac{t-a}{a+b} + \log\left(1+\frac{a}{b}\right)$
  - results in a sequences of "Gaussianized" problems with weighted sample covariance matrix



Experimental Results

Reproducibility

Open Source Software Package

https://github.com/convexfi/fingraph

Datasets and Benchmark Algorithms

Datasets (Log-returns)

- US Stock Market ($p = 82$ S&P500 stocks, from three sectors, $n = 1006$ daily observations)
- Foregin Exchange ($p = 34$ currencies, $n = 522$ daily observations)
- Cryptocurrencies ($p = 41$ most traded cryptos, $n = 1218$ daily observations)

- data matrix $\boldsymbol{X}$ constructed as:

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

- $P_{i,j}$: is the closing price of the j-th instrument at the i-th day
- sector information on stocks are given by the Global Industry Classification System (GICS) (Morgan Stanley Capital International and S&P Dow Jones, 2018)

Datasets and Benchmark Algorithms

Benchmark Models (Connected Graphs)

- Gaussian formulation with ℓ_1-norm for sparsity (GLE) (Zhao et al., 2019; Egilmez et al., 2017)
- Gaussian formulation with concave regularizer for sparsity (NGL) (Ying et al., 2020)

Benchmark Models (k-component graphs)

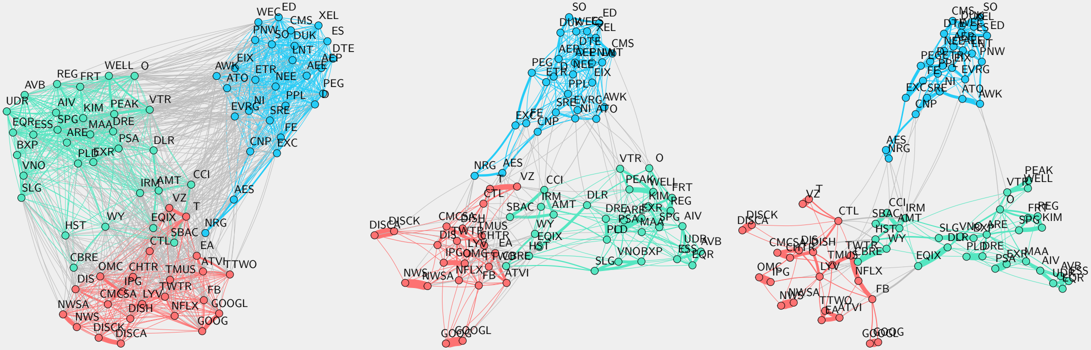
- Constrained Laplacian Rank (CLR) (Nie et al., 2016)
- Gaussian formulation with Spectral Constraints (SGL) (Kumar et al., 2019)

Performance Criteria

- Graph **modularity:** $Q(\mathcal{G}) \triangleq \frac{1}{2|\mathcal{E}|} \sum_{i,j \in \mathcal{V}} \left(\boldsymbol{W}_{ij} - \frac{d_i d_j}{2|\mathcal{E}|} \right) \delta(t_i = t_j)$, where d_i is the degree of the i-th node, t_i is the type (or label) of the i-th node, and $\delta(\cdot)$ is the indicator function

(a) modularity $= 0.1$ **(b)** modularity $= 0.37$ **(c)** modularity $= 0.7$

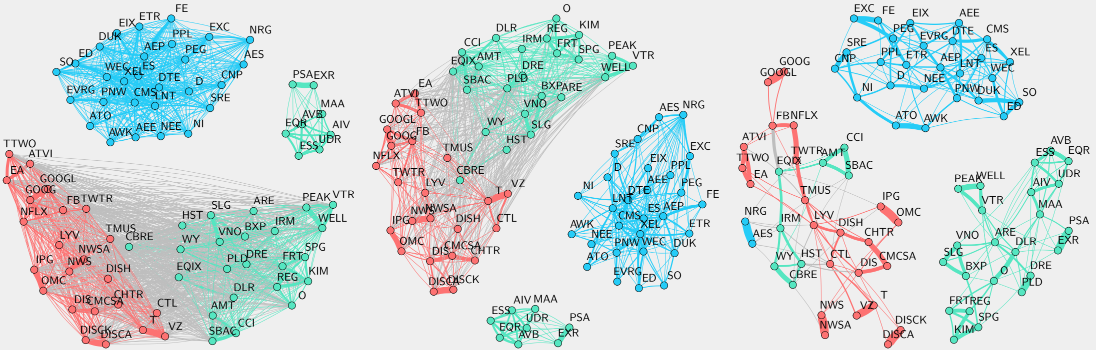
US Stock Market - Connected Graphs



(a) GLE, modularity $= 0.31$ **(b)** NGL, modularity $= 0.49$ **(c)** proposed, modularity $= 0.54$

- our method: **sparser** than Gaussian-based methods and shows **higher** modularity

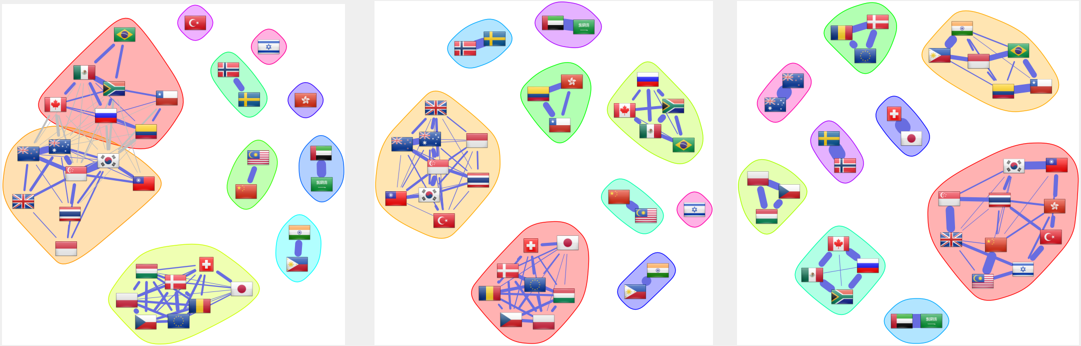
US Stock Market - k-component Graphs



(a) SGL, modularity $= 0.29$ **(b)** CLR, modularity $= 0.33$ **(c)** proposed, modularity $= 0.56$

- our method: clusters are more aligned with industry classification

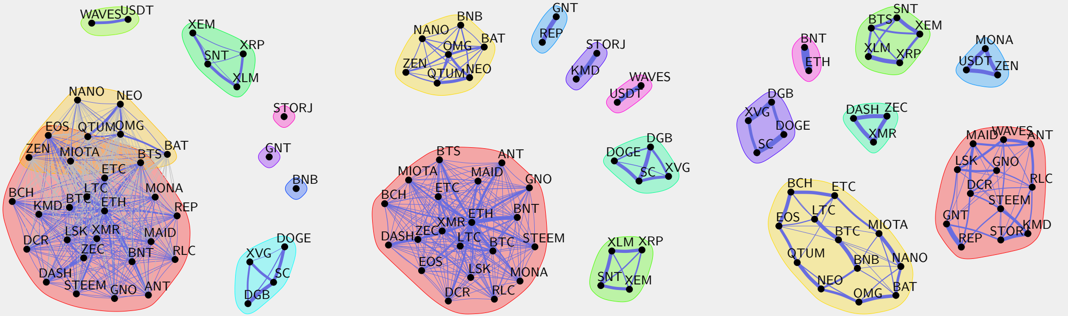
Foreign Exchange - k-component Graphs



(a) SGL, modularity $= 0.62$ **(b)** CLR, modularity $= 0.79$ **(c)** proposed, modularity $= 0.84$

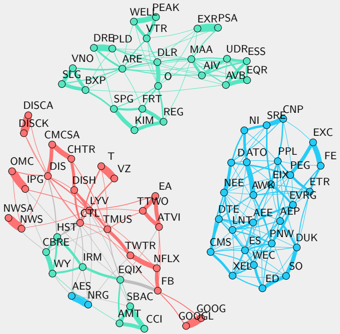
- our method: no isolated nodes, more reasonable clusters ({Australia & New Zealand}, {Hungary, Czech Republic, & Poland})

Cryptocurrencies - k-component Graphs

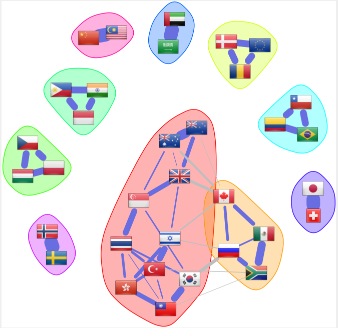


(a) SGL, modularity $= 0.36$ **(b)** CLR, modularity $= 0.66$ **(c)** proposed, modularity $= 0.79$

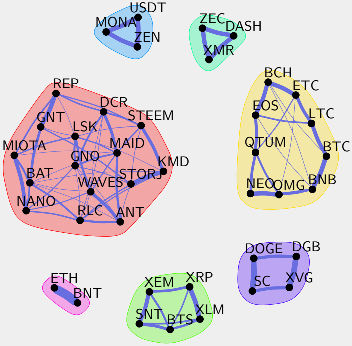
Effect of Initialization



(a) modularity $= 0.56$

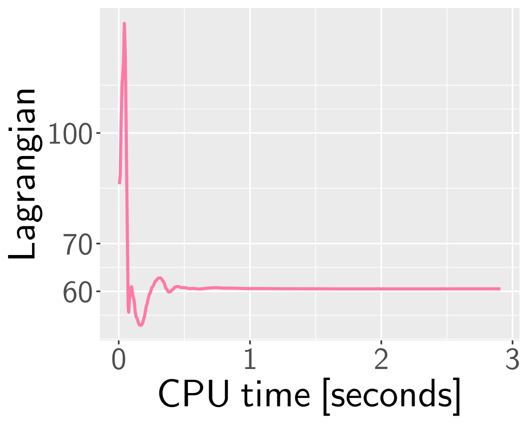


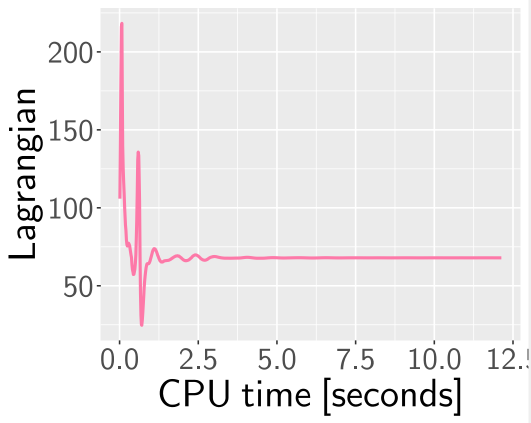
(b) modularity $= 0.80$



(c) modularity $= 0.78$

Empirical Convergence





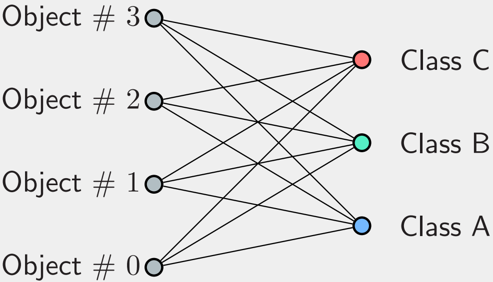
Contents

1. Introduction & Background
   Graphs, learning graphs from data as an optimization problem, and financial data
2. Learning Graphs in Heavy Tailed Markets
   Heavy tails, k-component graphs, and clustering
3. **Learning Bipartite Graphs**
   Bipartite structure, stock classification
4. Conclusion
   Final remarks

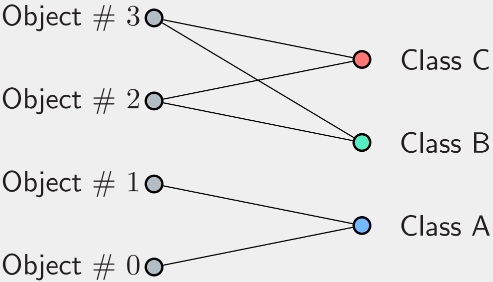
Learning Bipartite Graphs

Bipartite Graphs

- a **single component** bipartite graph:



- a 2-**component** bipartite graph:



Undirected Weighted Bipartite Graphs

$$\mathcal{G} = (\mathcal{V}_r, \mathcal{V}_q, \mathcal{E}, \boldsymbol{L})$$

- $\mathcal{V}_r = \{1, 2, \ldots, r\}$: **objects**
- $\mathcal{V}_q = \{r+1, r+2, \ldots, r+q\}$: **classes**
- $\mathcal{E} \subseteq \{\{u, v\} : u \in \mathcal{V}_r, v \in \mathcal{V}_q\}$
- **Laplacian Matrix:** $\boldsymbol{L} = \begin{bmatrix} \mathrm{Diag}\left(\boldsymbol{B}\mathbf{1}_q\right) & -\boldsymbol{B} \\ -\boldsymbol{B}^\top & \mathrm{Diag}\left(\boldsymbol{B}^\top\mathbf{1}_r\right) \end{bmatrix}, \boldsymbol{B} \in \mathbb{R}_+^{r \times q}$
- B_{ij}: edge weight between object i and class j

State-of-the-art Methods

- **Bipartite Structure** (Nie et al., 2017)

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

- alternating optimization algorithm
- pros: simple optimization that works well in practice
- cons: lacks statistical foundations

State-of-the-art Methods

- **Spectral Regularization** (Kumar et al., 2020)

$$
\begin{aligned}
\underset{\boldsymbol{w}\geq\boldsymbol{0},\boldsymbol{V},\boldsymbol{U},\boldsymbol{\psi},\boldsymbol{\lambda}}{\text{minimize}} \quad & \operatorname{tr}\left(\mathcal{L}\boldsymbol{w}\boldsymbol{S}\right) - \log\det{}^{*}\left(\mathcal{L}\boldsymbol{w}\right) + \underbrace{\frac{\gamma}{2}\left\|\mathcal{A}\boldsymbol{w} - \boldsymbol{U}\mathsf{Diag}(\boldsymbol{\psi})\boldsymbol{U}^{\top}\right\|_{\mathrm{F}}^{2}}_{\text{bipartite structure}} \\
& + \underbrace{\frac{\beta}{2}\left\|\mathcal{L}\boldsymbol{w} - \boldsymbol{V}\mathsf{Diag}(\boldsymbol{\lambda})\boldsymbol{V}^{\top}\right\|_{\mathrm{F}}^{2}}_{k\text{-component structure}}, \\
\text{subject to} \quad & \boldsymbol{U}^{\top}\boldsymbol{U} = \boldsymbol{I}, \boldsymbol{V}^{\top}\boldsymbol{V} = \boldsymbol{I},\ \boldsymbol{\lambda} \in C_{\boldsymbol{\lambda}}, \boldsymbol{\psi} \in C_{\boldsymbol{\psi}} \\
& C_{\boldsymbol{\lambda}} = \left\{\boldsymbol{\lambda} \in \mathbb{R}_{+}^{(p-k)} c_1 \leq \lambda_1, \ldots, \leq \lambda_{p-k} \leq c_2\right\}, \\
C_{\boldsymbol{\psi}} = \{\boldsymbol{\psi} \in \mathbb{R}^{p} : \psi_i = -\psi_{p+1-i},\ & i = 1, 2, \ldots, p, \psi_1 \geq \psi_2 \geq \cdots \geq \psi_p\}.
\end{aligned}
$$

- BCD-like optimization algorithm
- pros: clever idea with statistical foundations!
- cons: tuning γ, β, c_1, and c_2 is difficult, postprocessing often needed!

Proposed Formulations

Connected Bipartite Graphs: Gaussian Case

- Gaussian:

$$\begin{array}{ll} \underset{\boldsymbol{L},\boldsymbol{B}}{\text{minimize}} & \operatorname{tr}\left(\boldsymbol{L}\boldsymbol{S}\right) - \log\det\left(\boldsymbol{L}+\boldsymbol{J}\right), \\ \text{subject to} & \boldsymbol{L} = \begin{bmatrix} \boldsymbol{I}_r & -\boldsymbol{B} \\ -\boldsymbol{B}^\top & \text{Diag}\left(\boldsymbol{B}^\top\mathbf{1}_r\right) \end{bmatrix}, \boldsymbol{B} \geq \mathbf{0}, \boldsymbol{B}\mathbf{1}_q = \mathbf{1}_r, \end{array}$$

- $\boldsymbol{B}\mathbf{1}_q = \mathbf{1}_r$: normalizes the degrees of the set of objects
- **key idea**: simpler formulation in practice by plugging in the equality constraints and using the classical matrix determinant Lemma (Zhang, 2005):

$$\begin{array}{ll} \underset{\boldsymbol{B}\geq\mathbf{0},\boldsymbol{B}\mathbf{1}_q=\mathbf{1}_r}{\text{minimize}} & -\log\det\left(\text{Diag}(\boldsymbol{B}^\top\mathbf{1}_r) + \boldsymbol{J}_{qq} - (\boldsymbol{B}-\boldsymbol{J}_{rq})^\top(\boldsymbol{I}_r+\boldsymbol{J}_{rr})^{-1}(\boldsymbol{B}-\boldsymbol{J}_{rq})\right) \\ & +\operatorname{tr}\left(\boldsymbol{B}\boldsymbol{C}\right) \end{array}$$

- massive **reduction** in computational complexity!
- algorithm: projected gradient descent (Bertsekas, 1999)

Connected Bipartite Graphs: Student-t Case

- Student-t:

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

- like in the Gaussian case, a formulation as a function of $\boldsymbol{B}$ can be obtained:

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

- algorithm: MM to deal with the concave terms

k-component Bipartite Graphs

- Student-t, k-component, bipartite graph

$$\begin{array}{ll}\underset{\boldsymbol{L}\succeq\mathbf{0},\boldsymbol{B}}{\text{minimize}} & \dfrac{p+\nu}{n}\sum_{i=1}^{n}\log\left(1+\dfrac{h_i+\operatorname{tr}\left(\boldsymbol{B}\boldsymbol{G}_i\right)}{\nu}\right)-\log\det^{*}\left(\boldsymbol{L}\right),\\ \text{subject to} & \boldsymbol{L}=\begin{bmatrix}\boldsymbol{I}_r & -\boldsymbol{B}\\ -\boldsymbol{B}^{\top} & \text{Diag}\left(\boldsymbol{B}^{\top}\mathbf{1}_r\right)\end{bmatrix},\ \text{rank}(\boldsymbol{L})=p-k,\ \boldsymbol{B}\geq\mathbf{0},\boldsymbol{B}\mathbf{1}_q=\mathbf{1}_r,\end{array}$$

- algorithmic solution: ADMM + MM

Experimental Results

Reproducibility

Open Source Software Packages

https://github.com/convexfi/bipartite

Experiments

Datasets (Log-returns)

- US Stock Market ($r = 333$ S&P500 stocks $q = 8$ S&P Sector Indices, from Jan. 5th 2016 to Jan. 5th 2021, $n = 1291$ daily observations)
- data matrix $\boldsymbol{X}$ constructed as:

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

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

Benchmark Models

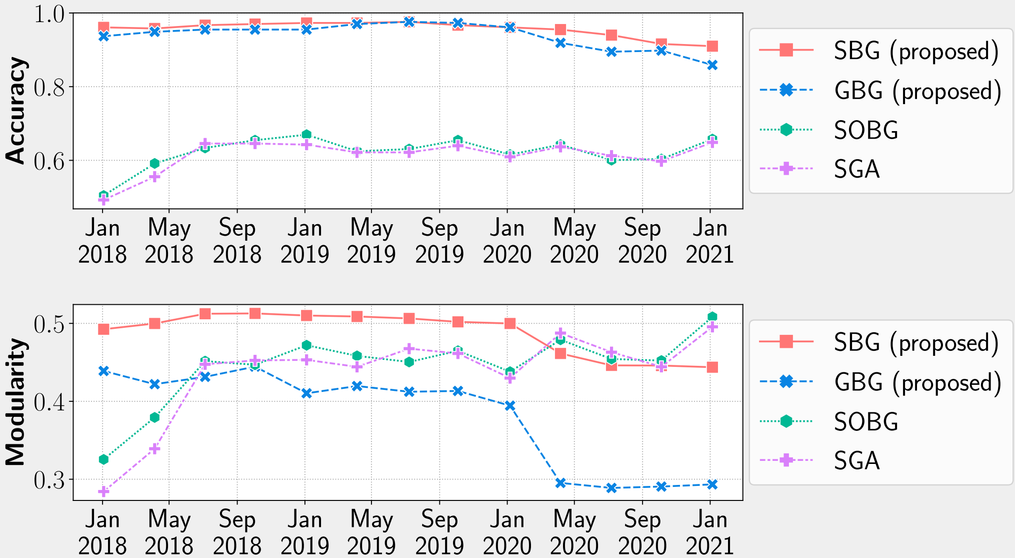
- Bipartite structure: SOBG, connected ($k = 1$) and k-components ($k > 1$) (Nie et al., 2017)
- Spectral regularization methods: SGA (connected), SGLA (k-components) (Kumar et al., 2020)

Experiments

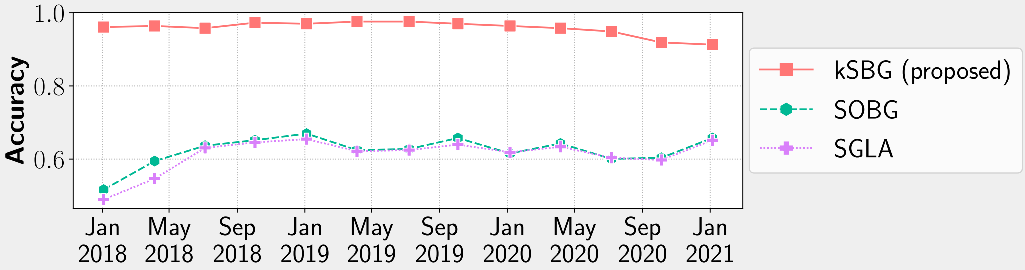
Performance Criteria

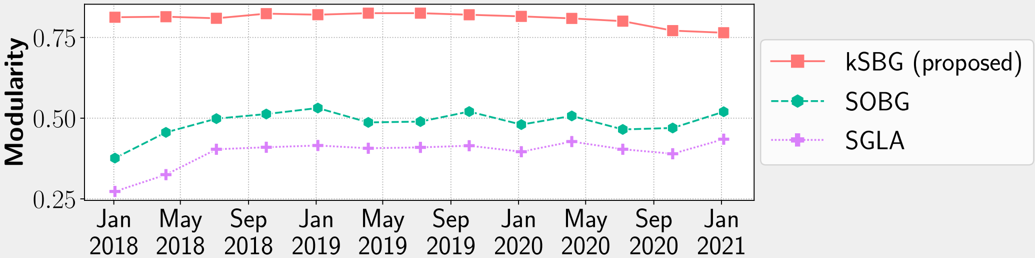
- Graph **modularity**
- Node **accuracy**: fraction of nodes whose sectors agree with those from the Global Industry Classification Standard (GICS) (Morgan Stanley Capital International and S&P Dow Jones, 2018)

Rolling Window Results: Connected Graphs

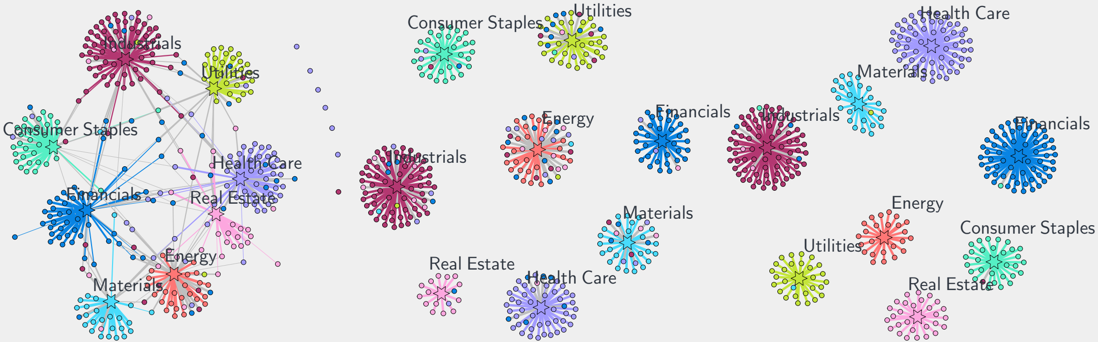


Rolling Window Results: $k = 8$-components



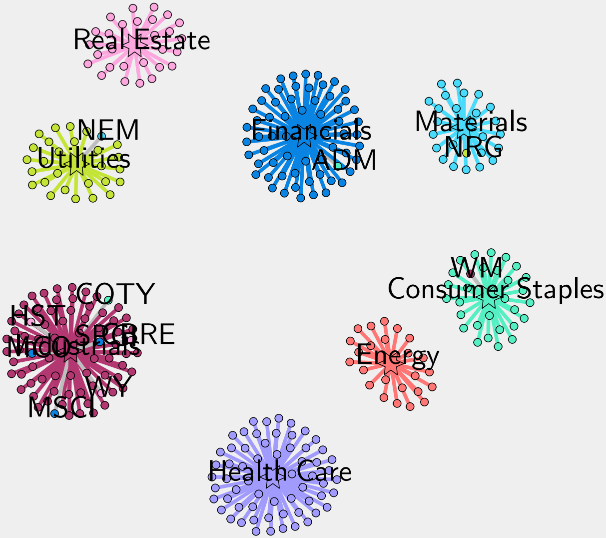


k-component Bipartite Graphs

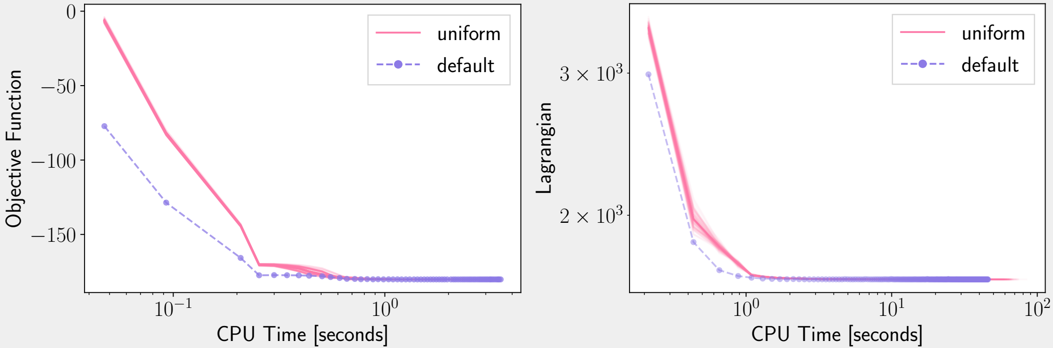


(a) SGLA, accuracy $= 0.77$, modularity $= 0.56$ **(b)** SOBG, accuracy $= 0.75$, modularity $= 0.61$ **(c)** proposed, accuracy $= 0.97$, modularity $= 0.82$

k-component Bipartite Graphs



Robustness to the choice of initial point



Conclusions

Conclusions

- graph learning formulations have received substantial attention from the scientific community in recent years
- **modeled** a financial networks as undirected graphs
- **developed** formulations as well as efficient algorithms to estimate the Laplacian matrix
  - k-component
  - bipartite
  - joint k-component & bipartite
- **worked** on heavy-tailed scenarios envisioning practical applications in finance
- **applied** the estimated graphs into clustering tasks of financial stocks and evaluate their performance via modularity and accuracy
- open source software for research **reproducibility** is made available on GitHub

References I

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

J. V. M. Cardoso, J. Ying, and D. P. Palomar. Graphical models in heavy-tailed markets. In *Advances in Neural Information Processing Systems (NeurIPS'21)*, 2021.

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.

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.

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

References II

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

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

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

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. AAAI Press, 2016.

F. Nie, X. Wang, C. Deng, and H. Huang. Learning a structured optimal bipartite graph for co-clustering. In *Advances on Neural Information Processing Systems (NeurIPS'17)*, page 4132–4141, 2017.

References III

Y. Sun, P. Babu, and D. P. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2017. ISSN 1941-0476.

J. Ying, J. V. M. Cardoso, and D. P. Palomar. Nonconvex sparse graph learning under Laplacian-structured graphical model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 7101–7113, 2020.

F. Zhang. *The Schur Complement and Its Applications*. Springer, 2005.

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.

Thank you very much!