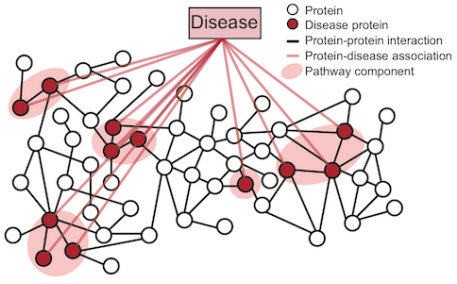
Learning to Learn Graph Topologies

Xingyue (Stacy) Pu[1], Tianyue Cao[2], Xiaoyun Zhang[2], Xiaowen Dong[1], Siheng Chen[2]

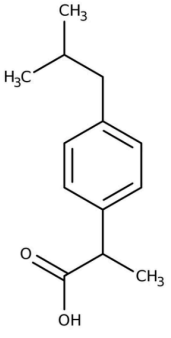
[1]University of Oxford
[2]Shanghai Jiao Tong University

Why Graphs are Important?

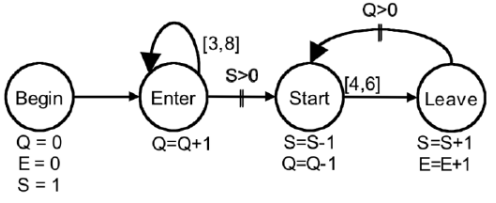
- Graphs are an explicit modeling language for revealing the relational structure in complex domains, helpful to achieve better AI reasoning.



Disease pathways

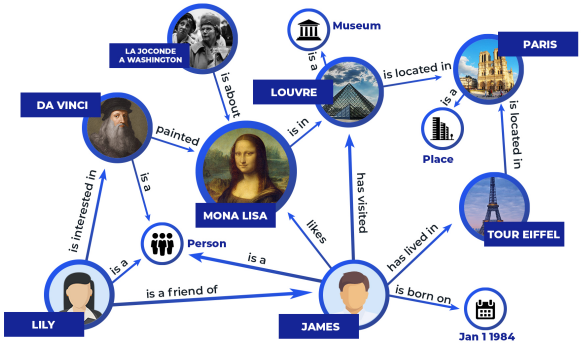


Molecules



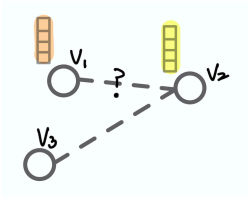
Event graphs

Brain graphs / Neural networks



Knowledge graphs

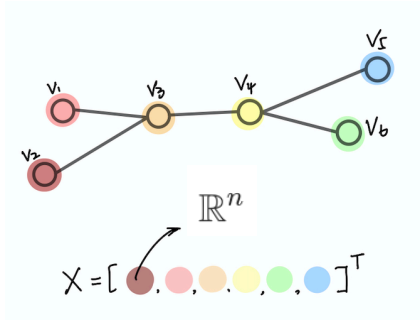
When Graphs are Unknown …



(I) similarity to a graph

$$\mathbf{W}_{1,2} = s(\,,\,)$$

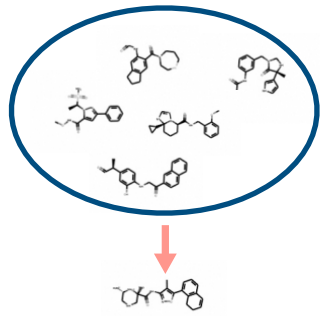
- $s(\cdot,\cdot)$ is an affinity function, e.g. cosine similarity, RBF kernel.
- optionally used with kNN.



(II) data to a graph

Given an observation matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, we assume a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ governs the generative process of $\mathbf{X}$ such that $\mathbf{X} = f(\mathcal{G})$.

- f can be parametric or non-parametric.
- solving the inverse problem f^{-1} to obtain $\mathbf{W}$.
- literature:
  - inverse covariance estimation [1,2]
  - graph signal processing [3] our scope



(III) graphs to a graph

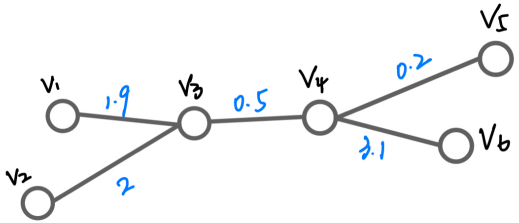
Given graphs sampled from $p_{data}(G)$, where $G \in \mathcal{G}$, we want to learn the distribution $p_{model}(G)$ such that we can generate the graph samples $G \sim p_{model}(G)$.

- deep generative models
- small-scale graph generation: drug molecules
- no signals required

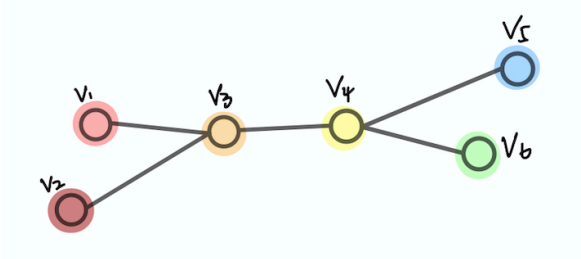
Graph Construction | Graph Learning | Graph Generation

Notations

- Assume a **weighted undirected graph (topology)**: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$

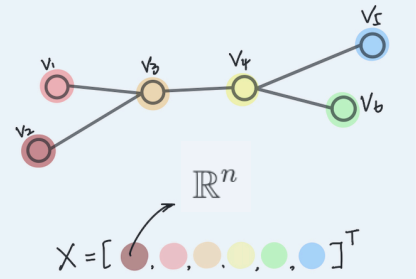


- Weighted adjacency matrix $\mathbf{W}$ where $\mathbf{W}_{ij} > 0$ if $(i, j) \in \mathcal{E}$ and $\mathbf{W}_{ij} = 0$ otherwise.
- Degree matrix $\mathbf{D} = \mathrm{diag}\{d_1, \ldots, d_m\}$ where d_i is the degree of i-th node, i.e. $d_i = \sum_{(i,j)\in\mathcal{E}} \mathbf{W}_{i,j}$
- Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$
- Normalised Laplacian matrix (preserve symmetry): $\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}$



- **node features / graph signals / (structured) data**
  - $\boldsymbol{x} = [\ \ ]^\top \in \mathbb{R}^{|\mathcal{V}|}$
  - can be high-dimensional: $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times n}$

A generic formulation of GL



(II) data to a graph

Given an observation matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, we assume a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ governs the generative process of $\mathbf{X}$ such that $\mathbf{X} = f(\mathcal{G})$.

- f can be parametric or non-parametric.
- solving the inverse problem f^{-1} to obtain $\mathbf{W}$.
- literature:
  - inverse covariance estimation
  - graph signal processing ✦ our scope

- [**Formulation**] Given n graph signals on m nodes: $\mathbf{X}^{m \times n}$, we want to learn a graph structure over which the graph signals are optimally smooth:

$$\min_{\mathbf{L} \in \mathcal{L}} \quad \underbrace{\operatorname{tr}(\mathbf{X}^\top \mathbf{L} \mathbf{X})}_{\text{smoothness}} + \underbrace{\Omega(\mathbf{L})}_{\text{topological priors}} \tag{1}$$

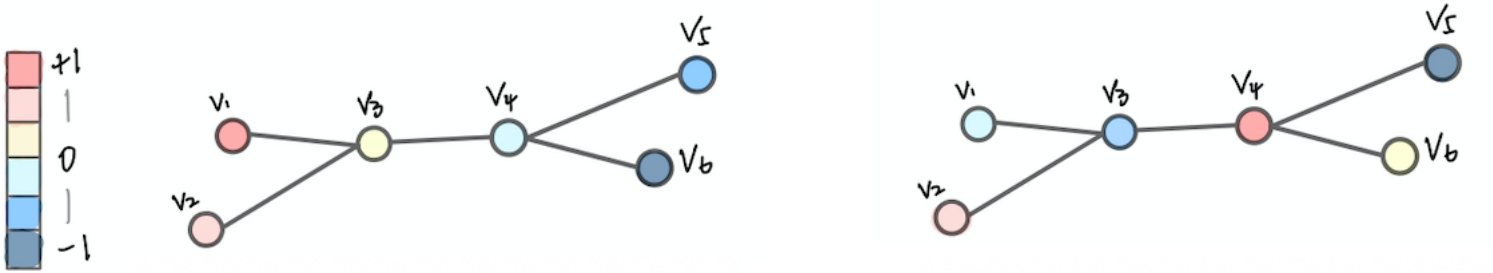
  - A Bayesian interpretation [5]:
    - **Gaussian likelihood**: $\mathbf{x}_i | \mathbf{L} \overset{i.i.d}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{L}^\dagger)$, for $i = 1, 2, \ldots, n$
    - **Topological prior**: $\mathbf{L} \sim p(\mathbf{L})$ and $\log p(\mathbf{L}) = -\Omega(\mathbf{L})$
    - **log-posterior** maximisation is

$$\max_{\mathbf{L}} \ \log \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{L}) + \log p(\mathbf{L})$$

$$\Longrightarrow \max_{\mathbf{L}} \ \underbrace{\log \operatorname{gdet}(\mathbf{L})}_{\substack{\text{treated as a} \\ \text{topological prior,} \\ \text{and combined into } \Omega}} - \operatorname{tr}(\mathbf{X}^\top \mathbf{L} \mathbf{X}) - \Omega(\mathbf{L})$$

Smoothness

- In graph signal processing [3],



$$\text{smoothness:} \quad \mathbf{x}_1^\top \mathbf{L}\mathbf{x}_1 < \mathbf{x}_2^\top \mathbf{L}\mathbf{x}_2$$

Given n graph signals $\mathbf{X}^{m \times n}$, we define the variation of signals on graphs (or equivalently a measure of smoothness) by the Laplacian quadratic form:

$$\mathrm{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X})$$

Reparametrisation

[Formulation] $$\min_{\mathbf{L}\in\mathcal{L}} \underbrace{\mathrm{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X})}_{\text{smoothness}} + \underbrace{\Omega(\mathbf{L})}_{\text{topological priors}} \quad (1)$$

[Reparametrisation] $$\min_{\mathbf{w}} \underbrace{2\mathbf{w}^\top \mathbf{y}}_{\text{smoothness}} + \underbrace{\mathcal{I}_{\{\mathbf{w}\geq 0\}}(\mathbf{w}) + \Omega_1(\mathbf{w}) + \Omega_2(\mathcal{D}\mathbf{w})}_{\text{topological priors}} \quad (2)$$

- reduces the search space by a half
- an extra constraint of PSD of $\mathbf{L}$ is no longer needed

- $\mathrm{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) = \sum_{i,j} \mathbf{W}_{ij} ||\mathbf{x}_i - \mathbf{x}_j||_2^2 = 2\mathbf{w}^\top \mathbf{y}$
- $\mathbf{w} \in \mathbb{R}_+^{m(m-1)/2}$: the vector of edge weights
- $\mathbf{y} \in \mathbb{R}_+^{m(m-1)/2}$: the half-vectorisation of the Euclidean distance matrix
- $\mathcal{I}_{\mathcal{C}}(\mathbf{w}) = 0$ if $\mathbf{w} \in \mathcal{C}$ and $\mathcal{I}_{\mathcal{C}}(\mathbf{w}) = \infty$ otherwise
- $\mathcal{D}\mathbf{w} = \mathbf{W}\mathbf{1} =$ the vector of node degrees

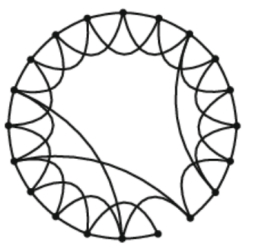
Topological priors

$$\min_{\mathbf{L}\in\mathcal{L}} \underbrace{\mathrm{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X})}_{\text{smoothness}} + \underbrace{\Omega(\mathbf{L})}_{\text{topological priors}} \quad (1)$$

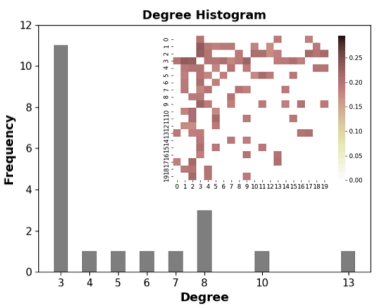
$$\min_{\mathbf{w}} \underbrace{2\mathbf{w}^\top \mathbf{y}}_{\text{smoothness}} + \underbrace{\mathcal{I}_{\{\mathbf{w}\geq 0\}}(\mathbf{w}) + \Omega_1(\mathbf{w}) + \Omega_2(\mathcal{D}\mathbf{w})}_{\text{topological priors}} \quad (2)$$

- Examples of $\Omega(\mathbf{L})$ or $\Omega(\mathbf{w})$:
  - sum barrier [5]: $\Omega(\mathbf{L}) = \lambda|\mathrm{Tr}(\mathbf{L}) - m|$
    - ✦barriers to prevent all-zero solutions
  - log barrier [4]: $\Omega(\mathbf{w}) = -\lambda\mathbf{1}^\top \log(\mathcal{D}\mathbf{w})$
    - ✦prevents isolated nodes
  - smooth edge weights [5]: $\Omega(\mathbf{w}) = \lambda||\mathbf{w}||_2^2$
  - sparse graph [4]: $\Omega(\mathbf{w}) = \lambda||\mathbf{w}||_1$
  - attractive Gaussian Markov random fields [6]: $\Omega(\mathbf{L}) = \log\det(\mathbf{L} + \sigma^2\mathbf{I}) + \lambda\sum_{i\neq j}|\mathbf{L}_{ij}|$

More Complicated Structures?



small-world



scale-free

Can we learn them from graph samples?

☞ **Learning to Learn Graph Topologies (L2G)**

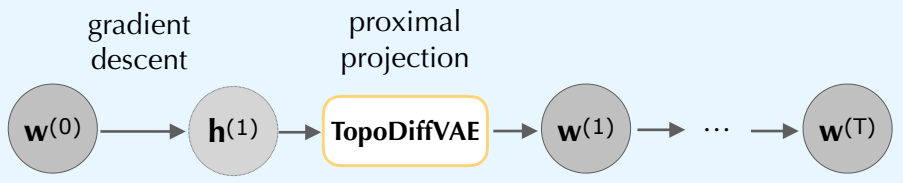
An Overview of L2G

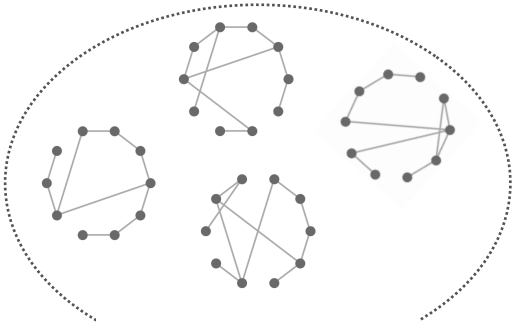
We want to solve: $\min_{\mathbf{w}} 2\mathbf{w}^\top \mathbf{y} + \mathcal{I}_{\{\mathbf{w}\geq 0\}}(\mathbf{w}) + \Omega_1(\mathbf{w}) + \Omega_2(\mathcal{D}\mathbf{w})$ (2)

Iterative Algorithm

1: **Initialisation:** $\mathbf{w}^{(0)} = \mathbf{0}$
2: **for** $t = 1, \ldots, T$ **do**
3: $\quad \mathbf{h}^{(t)} = \mathbf{w}^{(t-1)} - \gamma \nabla_{\mathbf{w}} f(\mathbf{w}^{(t-1)})$
4: $\quad \mathbf{w}^{(t)} = \text{prox}_{\gamma,\Omega}(\mathbf{h}^{(t)})$
5: **end for**

unrolling





training samples

Training (with supervision) to obtain a data-to-graph functional mapping

$$\mathbf{w} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{y})$$

such that

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{w}\sim\mathcal{G}}\left[\sum_{t=1}^{T}\left\{\underbrace{\tau^{T-t}\frac{||\mathbf{w}^{(t)} - \mathbf{w}||_2^2}{||\mathbf{w}||_2^2}}_{\text{reconstruction loss}} + \underbrace{\beta_{\text{KL}}\text{KL}\left(q_\phi(\mathbf{z}|\mathbf{r}_1^{(t)}, \mathbf{w})||\mathcal{N}(\mathbf{0}, \mathbf{I})\right)}_{\text{KL divergence for TopoDiffVAE}}\right\}\right]$$

Unrolling Networks

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

$$\min_{\mathbf{w}} \; 2\mathbf{w}^\top \mathbf{y} + \mathcal{I}_{\{\mathbf{w}\geq 0\}}(\mathbf{w}) - \alpha \mathbf{1}^\top \log(\mathcal{D}\mathbf{w}) + \beta ||\mathbf{w}||_2^2 \qquad (3)$$

iterative algorithm

Algorithm 1 PDS

Input: $\mathbf{y}$, γ, α and β.

Initialisation: $\mathbf{w}^{(0)} = \mathbf{0}, \mathbf{v}^{(0)} = \mathbf{0}$
while $|\mathbf{w}^{(t)} - \mathbf{w}^{(t-1)}| > \epsilon$ **do**
  $\mathbf{r}_1^{(t)} = \mathbf{w}^{(t)} - \gamma(2\beta\mathbf{w}^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{v}^{(t)})$
  $\mathbf{r}_2^{(t)} = \mathbf{v}^{(t)} + \gamma\mathcal{D}\mathbf{w}^{(t)}$
  $\mathbf{p}_1^{(t)} = \text{prox}_{\gamma,\Omega_1}(\mathbf{r}_1^{(t)}) = \max\{\mathbf{0}, \mathbf{r}_1^{(t)}\}$
  $\mathbf{p}_2^{(t)} = \text{prox}_{\gamma,\Omega_2}(\mathbf{r}_2^{(t)})$, where
  $\left(\text{prox}_{\gamma^{(t)},\Omega_2}(\mathbf{r}_2)\right)_i = \frac{r_{2,i} - \sqrt{r_{2,i}^2 + 4\alpha\gamma}}{2}$
  $\mathbf{q}_1^{(t)} = \mathbf{p}_1^{(t)} - \gamma(2\beta\mathbf{p}_1^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{p}_2^{(t)})$
  $\mathbf{q}_2^{(t)} = \mathbf{p}_2^{(t)} + \gamma\mathcal{D}\mathbf{p}_1^{(t)}$
  $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{r}_1^{(t)} + \mathbf{q}_1^{(t)}$
  $\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \mathbf{r}_2^{(t)} + \mathbf{q}_2^{(t)}$
end while
return $\widehat{\mathbf{w}} = \mathbf{w}^{(T)}$

unrolling net

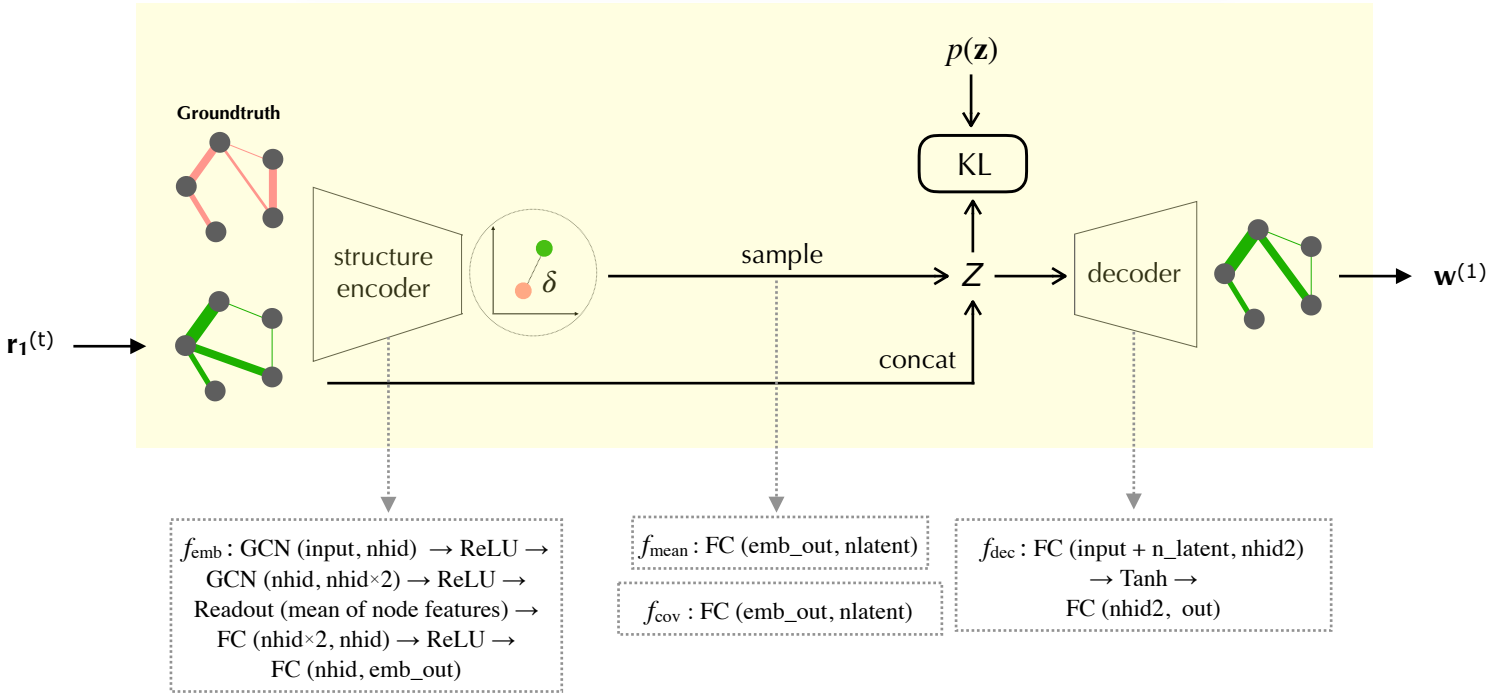
Algorithm 2 Unrolling Net (L2G)

Input: $\mathbf{y}$, T, $\texttt{enhancement}^{(t)} \in \{\texttt{TRUE}, \texttt{FALSE}\}$

Initialisation: $\mathbf{w}^{(0)} = \mathbf{0}, \mathbf{v}^{(0)} = \mathbf{0}$
for $t = 0, 1, \ldots, T$ **do**
  $\mathbf{r}_1^{(t)} = \mathbf{w}^{(t)} - \gamma^{(t)}(2\beta^{(t)}\mathbf{w}^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{v}^{(t)})$
  $\mathbf{r}_2^{(t)} = \mathbf{v}^{(t)} + \gamma^{(t)}\mathcal{D}\mathbf{w}^{(t)}$
  if $\texttt{enhancement}^{(t)}$=TRUE,
    $\mathbf{p}_1^{(t)} = \text{TopoDiffVAE}(\mathbf{r}_1^{(t)})$;
  else,
    $\mathbf{p}_1^{(t)} = \text{prox}_{\gamma,\Omega_1}(\mathbf{r}_1^{(t)}) = \max\{\mathbf{0}, \mathbf{r}_1^{(t)}\}$.
  $\mathbf{p}_2^{(t)} = \text{prox}_{\gamma^{(t)},\Omega_2}(\mathbf{r}_2^{(t)})$, where
  $\left(\text{prox}_{\gamma^{(t)},\Omega_2}(\mathbf{r}_2)\right)_i = \frac{r_{2,i} - \sqrt{r_{2,i}^2 + 4\alpha^{(t)}\gamma^{(t)}}}{2}$
  $\mathbf{q}_1^{(t)} = \mathbf{p}_1^{(t)} - \gamma^{(t)}(2\beta^{(t)}\mathbf{p}_1^{(t)} + 2\mathbf{y} + \mathcal{D}^\top \mathbf{p}_2^{(t)})$
  $\mathbf{q}_2^{(t)} = \mathbf{p}_2^{(t)} + \gamma^{(t)}\mathcal{D}\mathbf{p}_1^{(t)}$
  $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{r}_1^{(t)} + \mathbf{q}_1^{(t)}$
  $\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} - \mathbf{r}_2^{(t)} + \mathbf{q}_2^{(t)}$
end for
return $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(T)} = \widehat{\mathbf{w}}$

Topological Enhancement

Topological Difference VAE (TopoDiffVAE)



$$\boldsymbol{\delta} = f_{\text{emb}}(\mathbf{A}_w) - f_{\text{emb}}(\mathbf{A}_r)$$

$$f_{\text{emb}}(\mathbf{A}) = f_{\text{readout}}\left(\psi\left(\mathbf{A}\psi(\mathbf{A}\mathbf{d}\mathbf{h}_0^\top)\mathbf{H}_1\right)\right)$$

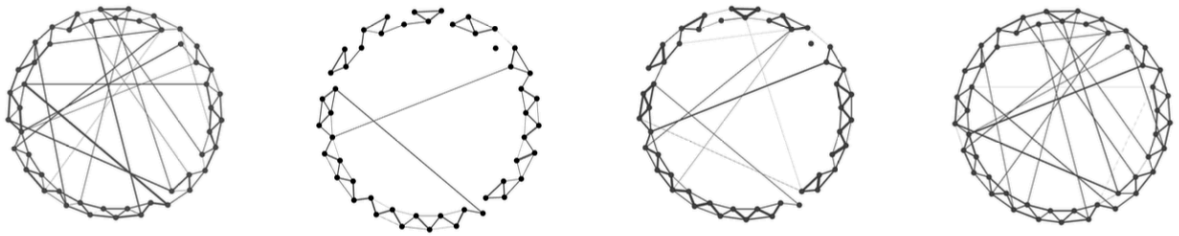
$$f_{\text{mean}}(\boldsymbol{\delta}), \quad \boldsymbol{\sigma} = f_{\text{cov}}(\boldsymbol{\delta})$$

$$\mathbf{z}|\mathbf{r}_1^{(t)}, \mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$$

$$\mathbf{p}_1^{(t)} = f_{\text{dec}}(\text{CONCAT}[\mathbf{r}_1^{(t)}, \mathbf{z}])$$

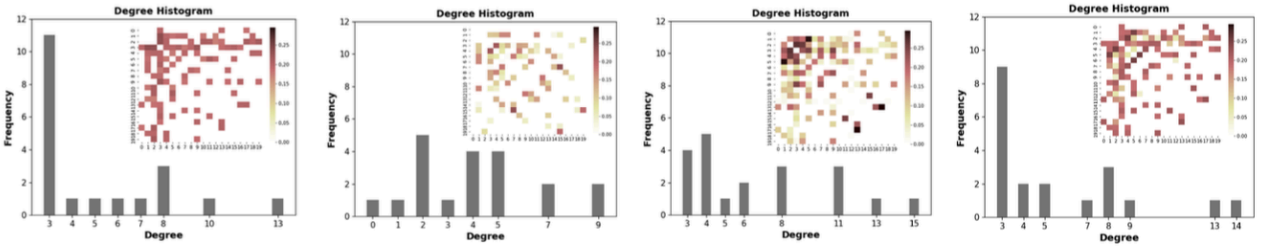
Experimental Results

(1) The ability to learn complex topological priors



(a) groundtruth (b) PDS (c) Unrolling (d) L2G

Figure 8: The ability to recover small-world network



(a) groundtruth (b) PDS (c) Unrolling (d) L2G

Figure 9: The ability to recover scale-free networks

Experimental Results

(2) Comparisons to SOTA and ablation models

Table 1: GMSE* in graph reconstruction

model/graph type	Scale-free (BA)	Random sparse (ER)	Community (SBM)	Small-world (WS)
Iterative algorithm:				
ADMM	0.4094 ± .0120	0.3547 ± .0120	0.3168 ± .0226	0.2214 ± .0151
PDS	0.4033 ± .0072	0.3799 ± .0085	0.3111 ± .0147	0.2180 ± .0117
Learned to optimise:				
GLAD[31]* (NMSE)	1.1230 ± .1756	1.1365 ± .1549	1.4231 ± .2625	0.9999 ± .0001
Deep-graph** [4]	0.8423 ± .0026	0.8179 ± .0269	0.8931 ± .0103	0.8498 ± .0017
Proposed:				
Recurrent Unrolling	0.3018 ± .0080	0.2885 ± .0075	0.2658 ± .0108	0.2007 ± .0081
Unrolling	0.1079 ± .0047	0.0898 ± .0067	0.1199 ± .0064	0.1028 ± .0073
L2G	**0.0594 ± .0044**	**0.0746 ± .0042**	**0.0735 ± .0051**	**0.0513 ± .0060**

Table 2: Structural Metrics of recovered binary graph structure

metric / model	groundtruth	PDS	Deep-Graph [4]	Unrolling	L2G
Scale-free (BA):					
AUC	-	0.934±.009	0.513±.017	0.993±.001	**0.999±.000**
KS test score*	95.31%	65.63%	59.38%	93.75%	**95.31%**
Community (SBM):					
AUC	-	0.926±.006	0.586±.021	0.989±.002	**0.993±.002**
community score	0.463±.002	0.481±.002	0.343±.006	0.458±.001	**0.469±.003**
Small-world (WS):					
AUC	-	0.913±.007	0.529±.010	0.984±.005	**0.991±.000**
average shortest path	2.334±.028	2.656±.204	1.136±.008	2.328±.040	**2.331±.041**
clustering coefficient	0.323±.018	0.514±.035	0.906±.004	0.433±.016	**0.382±.014**

* KS test score is the percentage of test graphs whose degree sequence passes the Kolmogorov -Smirnov test for goodness of fit of a power-law distribution, i.e. p-value > 0.05.

In the reference of this slide [pp.15]:
- PDS[4]
- GLAD[8]
- Deep-graph[9]

Applications in Finance

- **Successfully reveals the sector communities while not missing important inter-sector edges**

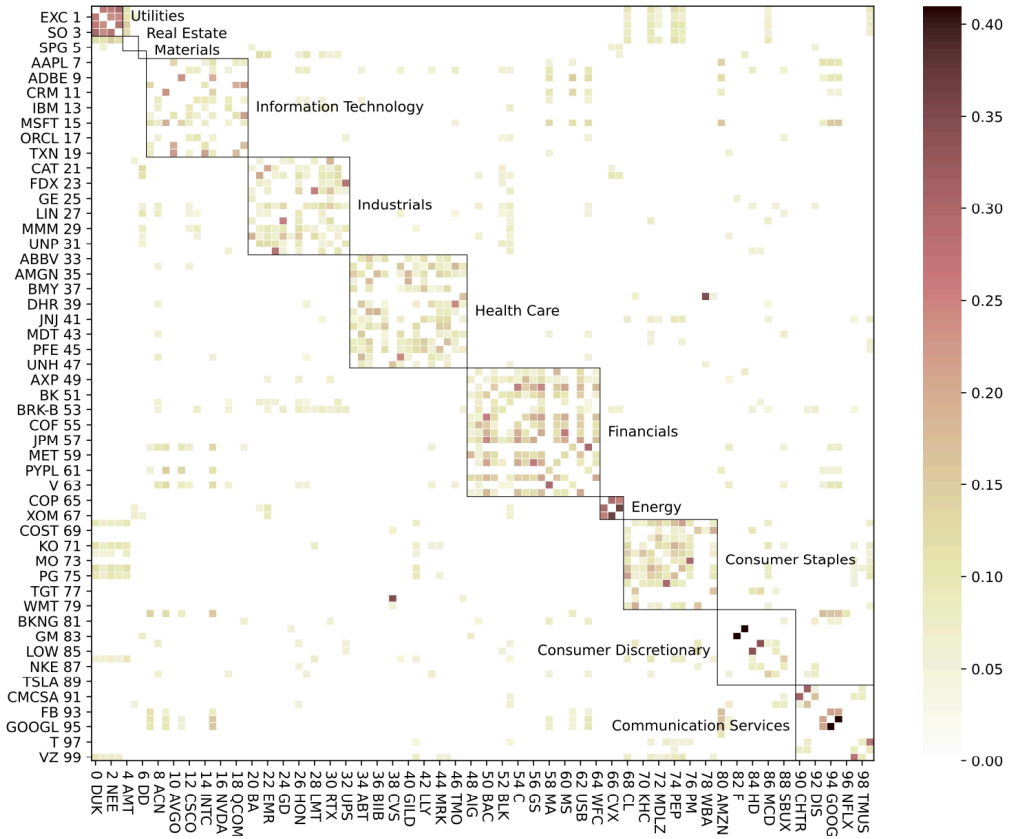


Figure 4: The learned graph adjacency matrix by L2G that reveals the relationship between S&P 100 companies. The rows and columns are sorted by sectors.

References

[1] O. Banerjee, L. El Ghaoui, and A. D'Aspremont, **Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data,** *Journal of Machine Learning Research*, vol. 9, pp. 485– 516, 2008.

[2] B. Rolfs, B. Rajaratnam, D. Guillot, I. Wong, and A. Maleki, **Iterative thresholding algorithm for sparse inverse covariance estimation**, in *Advances in Neural Information Processing Systems*, 2012, pp. 1574– 1582.

[3] Antonio Ortega, Pascal Frossard, Jelena Kovacevic, José MF Moura, and Pierre Vandergheynst. **Graph signal processing: Overview, challenges, and applications.** *Proceedings of the IEEE*, 106(5):808–828, 2018.

[4] V. Kalofolias, **How to learn a graph from smooth signals**, in *Artificial Intelligence and Statistics*, 2016, pp. 920–929.

[5] X.Dong, D.Thanou, P.Frossard, and P.Vandergheynst**, Learning laplacian matrix in smooth graph signal representations**, *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[6] B. Lake and J. Tenenbaum, **Discovering structure by learning sparse graphs**, in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 32, no. 32, 2010.

[8] Harsh Shrivastava, Xinshi Chen, Binghong Chen, Guanghui Lan, Srinivas Aluru, Han Liu, and Le Song. **Glad: Learning sparse graph recovery.** In *International Conference on Learning Representations*, 2020.

[9] Eugene Belilovsky, Kyle Kastner, Gael Varoquaux, and Matthew B. Blaschko. **Learning to discover sparse graphical models,** in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[10] Yahoo Finance! from https://pypi.org/project/yahoofinancials/

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

OXFORD-MAN INSTITUTE

Thanks!
Any Questions?

- Xingyue (Stacy) Pu
- Email: xpu@robots.ox.ac.uk