Signal Processing and Deep Learning over Graphs

Sundeep Prabhakar Chepuri

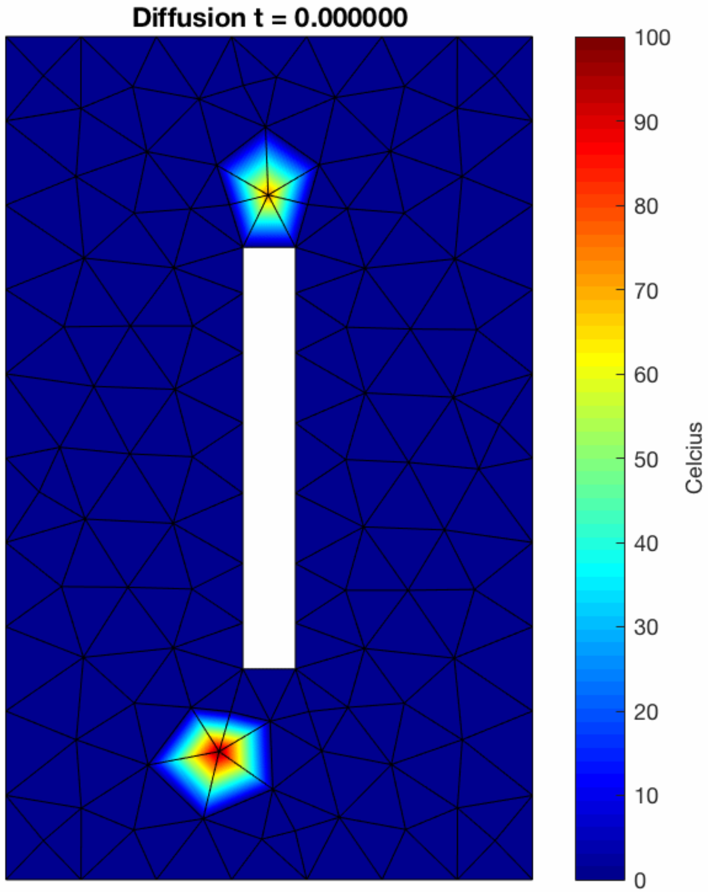
Email: spchepuri@iisc.ac.in

Acknowledgements: Sai Kiran Kadambari, Siddartha Reddy, Amarlingam Madapu, Guillermo Ortiz-Jiménez, Mario Coutino, Geert Leus, Santiago Segarra, Antonio Marques.

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Roadmap

- Introduction and context
- Signal processing on graphs
- Active Learning, semi-supervised learning, or signal reconstruction
- Multi-domain (tensor) signal reconstruction over product graphs
- Sparse sampler design
- Graph learning or topology inference
- Geometric deep learning (CNNs, RNNs, GANs)
- Conclusions, Q&A

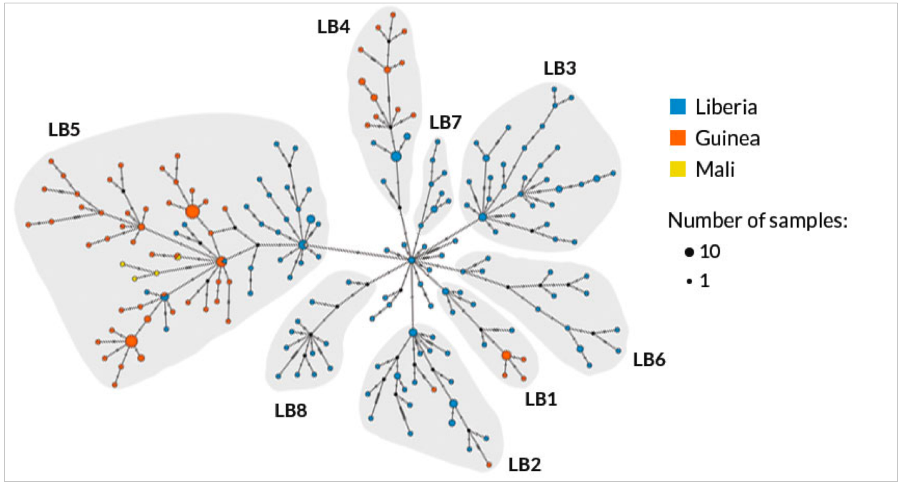


Frozen metal plate with cavity
excited with two hotspots

1854 Cholera outbreak in the City
of Soho, London

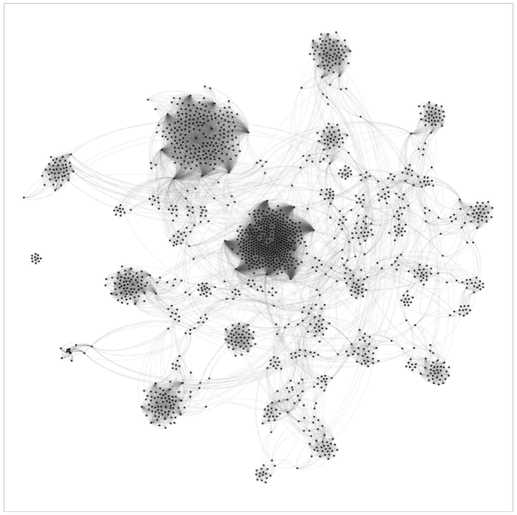
How to optimally deploy sensors?

Temperature on Earth’s surface



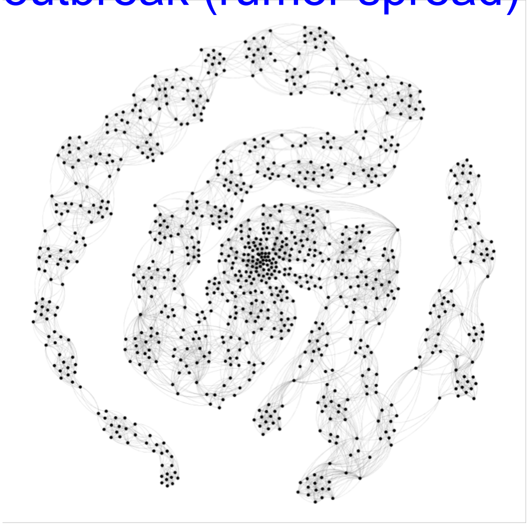
Epidemic network
- Ebola outbreak (rumor spread)

3D point clouds (Kinect, LiDAR)



Movies graph

◇



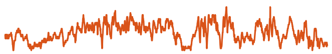
Social network

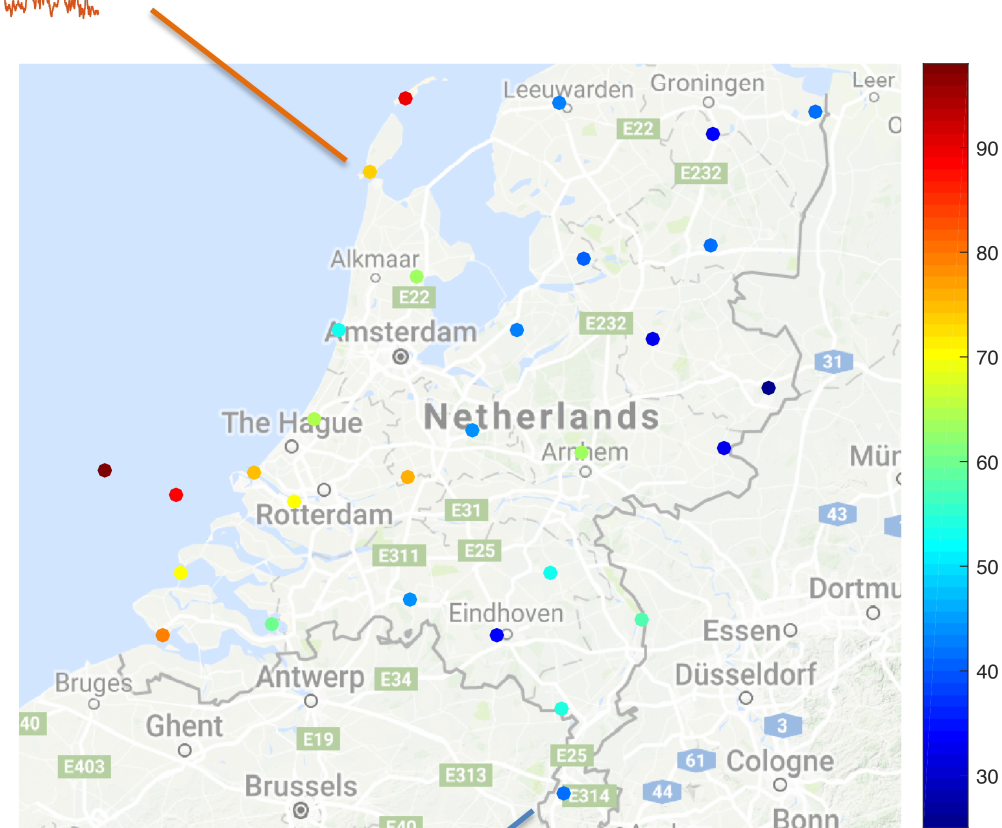
Recommender systems

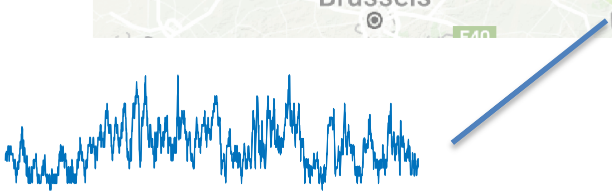
Design sparse samplers taking into account the underlying topology

Graph learning or topology inference

Construct/estimate graphs from data and for a specific task





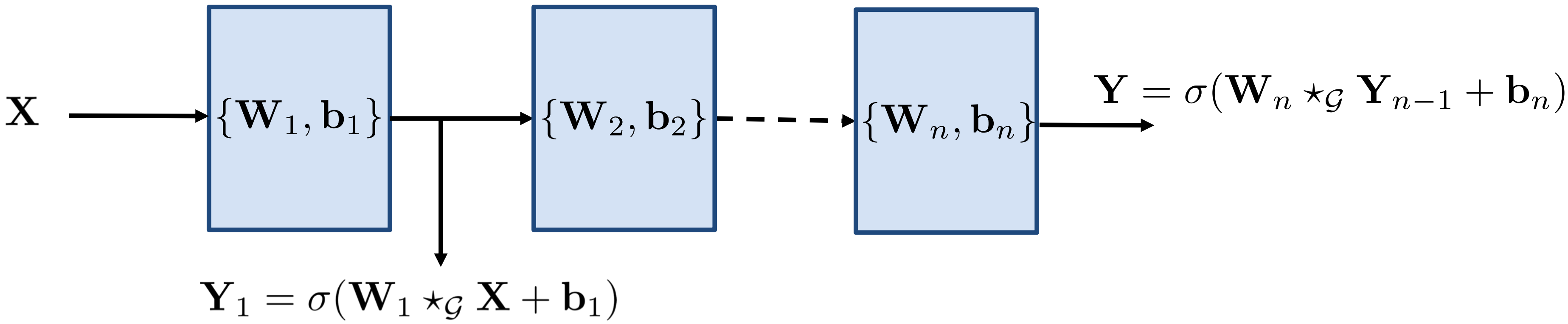


Wind speed data from 30 stations

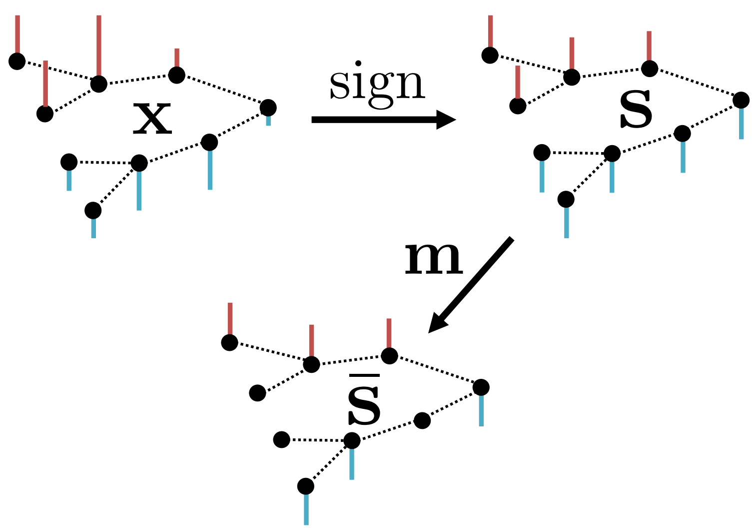
[Source: KNMI, Netherlands]

"Learn a sparse graph that sufficiently explains the data"

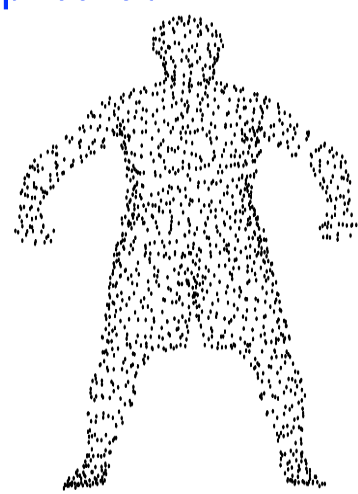
Geometric deep learning



- Lack of models, but many available examples
- Optimization underlying the inference task is complicated



PU learning (Yes/no response)



Dynamic 3D point cloud

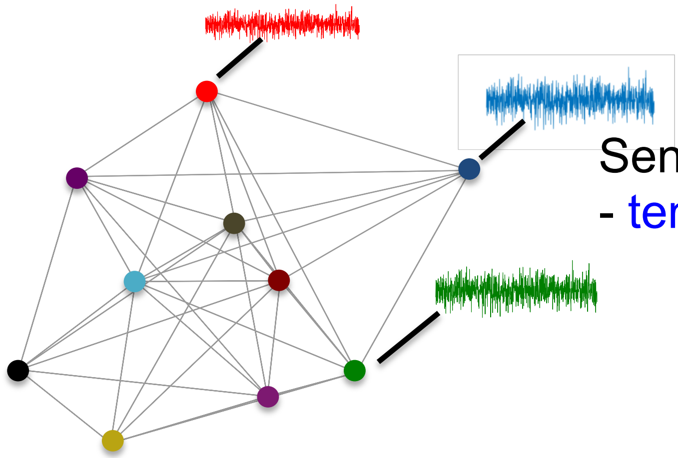
In this tutorial

We will cover the following three aspects:

1. Sparse sampling or active learning over graphs
2. Graph learning or topology inference
3. Geometric deep learning

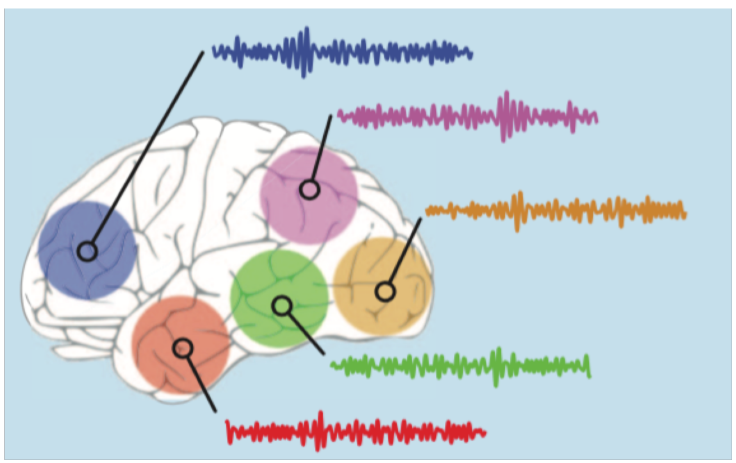
Graph Signal Processing

- D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” IEEE Signal Process. Mag., vol. 30, no. 3, pp. 83–98, 2013.
- A. Sandryhaila and J. M. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” IEEE Signal Process. Mag., vol. 31, no. 5, pp. 80–90, 2014.



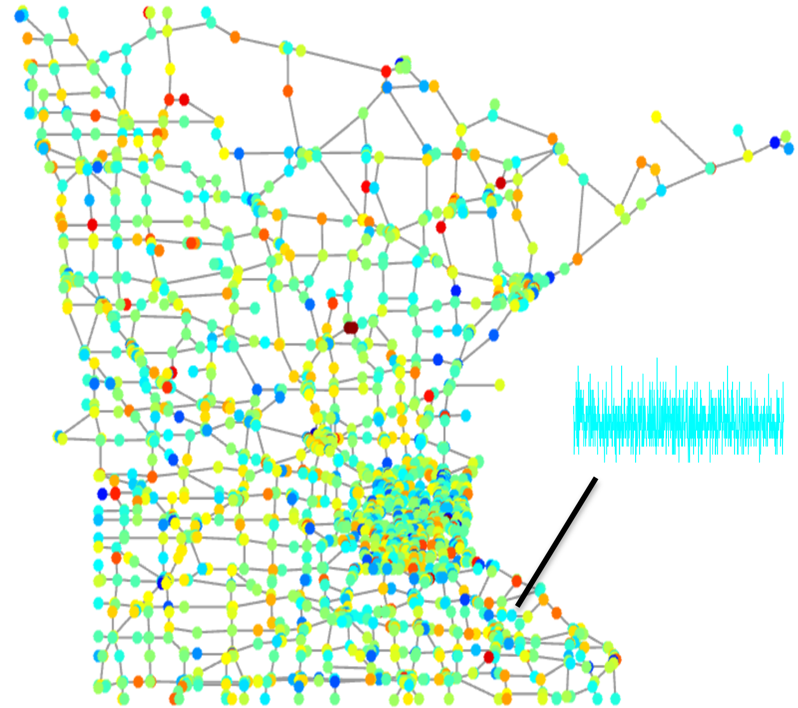
Sensing networks

- temp., pressure, air quality monitoring



Brain networks

- fMRI time series, EEG signals



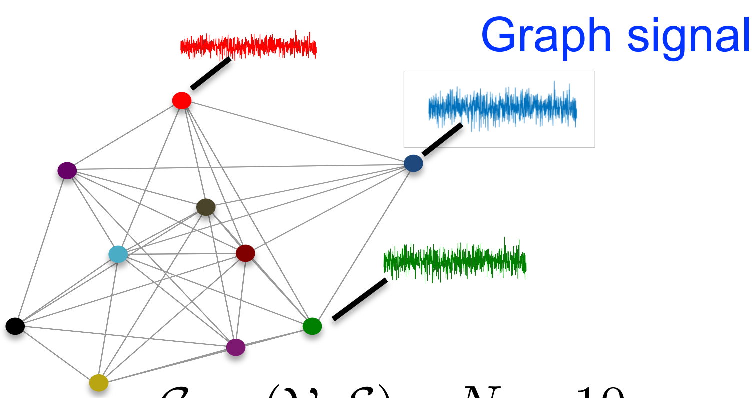
Transport networks

- # vehicles crossing a junction

Signals and random processes on graphs

Graphs and graph signals

- Datasets with *irregular support* can be represented using a graph

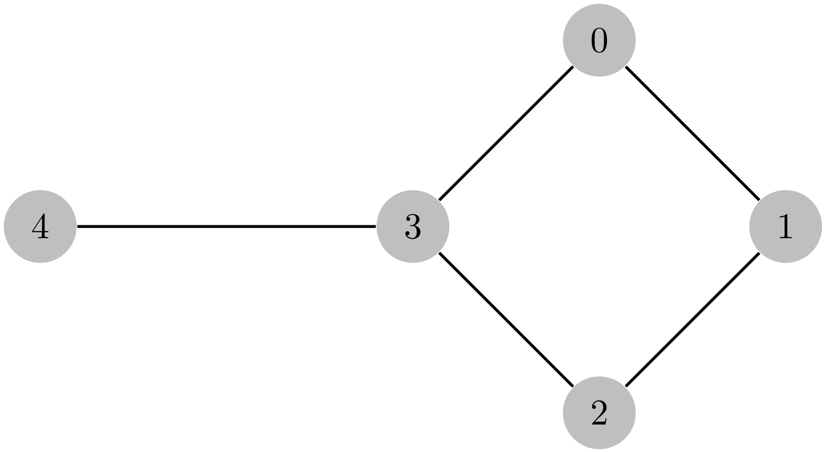


$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \quad N = 10$

- $\mathcal{V}$ is the set of nodes
- $\mathcal{E}$ is the set of edges
- $\boldsymbol{x} \in \mathbb{R}^N$ represents the graph signal

- Graph is represented using the matrix $\boldsymbol{S} \in \mathbb{R}^{N \times N}$
  - $[\boldsymbol{S}]_{i,j}$ is nonzero only if $i = j$ and/or $(i, j) \in \mathcal{E}$
  - $\boldsymbol{S}$ could be graph Laplacian, adjacency matrix, or ...
  - $\boldsymbol{S}$ is referred to as the graph-shift operator

Graph Laplacian



$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$

$$= \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

diagonal degree matrix adjacency matrix

- For an *undirected graph*, $\boldsymbol{L}$ is symmetric

$$\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^H$$
$$= [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N] \operatorname{diag}(\lambda_1, \cdots, \lambda_N) [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N]^H$$

- $\boldsymbol{L}\mathbf{1} = \mathbf{0}$, so

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$$

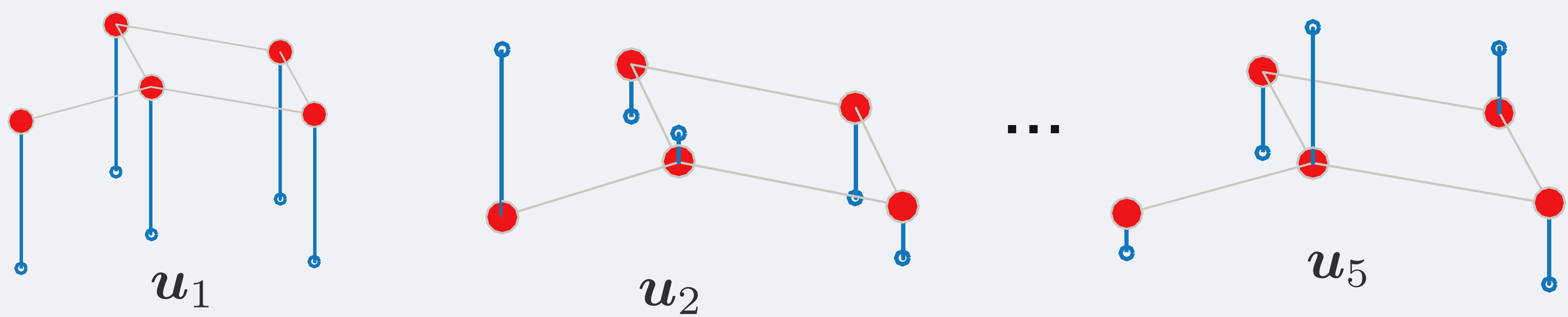
Graph Laplacian - eigenmodes

Frequency interpretation of the eigenvectors (viewed as signals on graphs)

eigenvalues

eigenvectors

$$\boldsymbol{\lambda} = \begin{bmatrix} 0 \\ 0.8299 \\ 2 \\ 2.6889 \\ 4.4812 \end{bmatrix} \quad \boldsymbol{U} = \begin{bmatrix} -0.4472 & -0.2560 & 0.7071 & 0.2422 & -0.4193 \\ -0.4472 & -0.4375 & 0 & -0.7031 & 0.3380 \\ -0.4472 & -0.2560 & -0.7071 & 0.2422 & -0.4193 \\ -0.4472 & 0.1380 & 0 & 0.5362 & 0.7024 \\ -0.4472 & 0.8115 & 0 & -0.3175 & -0.2018 \end{bmatrix}$$

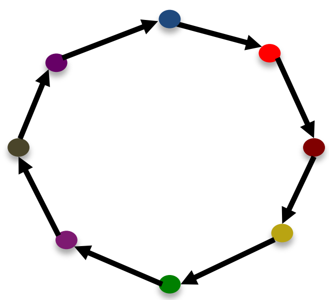


DC (no zero crossing) two zero crossings five zero crossings

Sign transitions of eigenvectors increase with eigenvalues

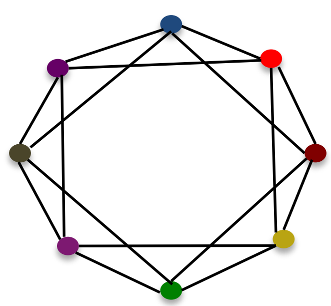
Time-domain as a graph

- The DFT and the traditional frequency grid is obtained by the adjacency matrix of the cycle graph



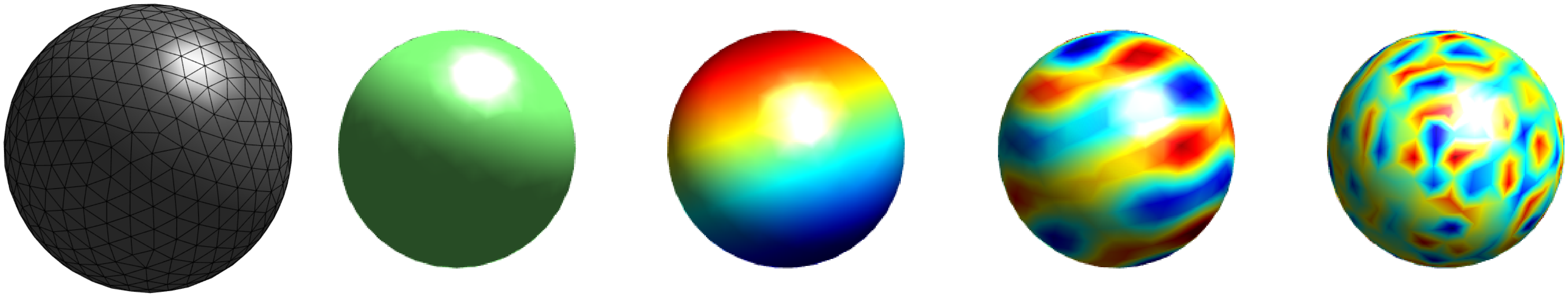
$$\boldsymbol{S} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Any circulant graph in principle leads to the DFT as the graph Fourier transform

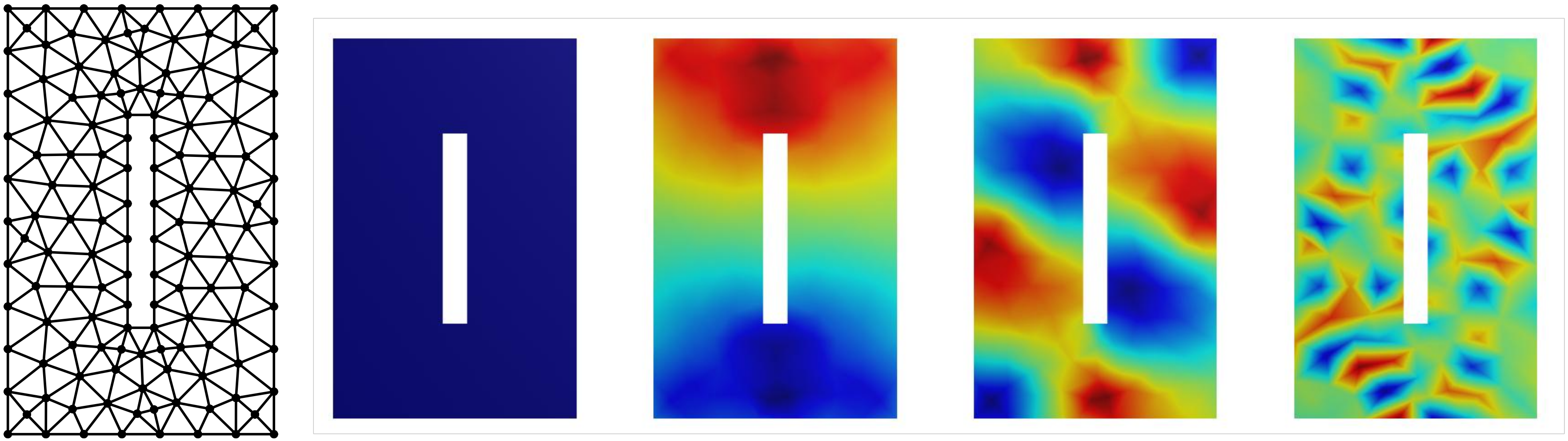


$$\boldsymbol{S} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Fourier-like basis on meshes



(Laplace's) spherical harmonics



Fourier-like oscillating modes of the metal plate with cavity

Fourier-like orthogonal basis

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^H$$

$$= [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N] \, \mathrm{diag}(\lambda_1, \cdots, \lambda_N) \, [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N]^H$$

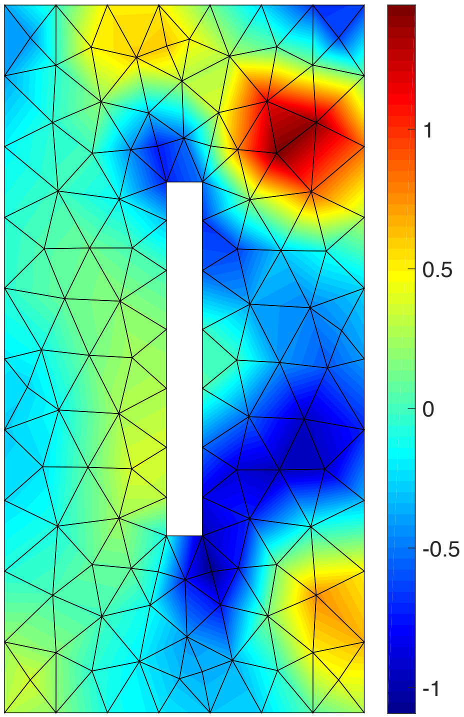
Fourier-like basis for the graph — Spectrum of the graph

- Holds for graph Laplacians and adjacency matrices
  - Frequency interpretation based on zero crossings or total variation
- For undirected graphs
  - Eigenvalues are all real (*graph-shift operator is symmetric*)
- For directed graphs with normal S
  - Eigenvalues occur in complex conjugate pairs

Graph Fourier transform

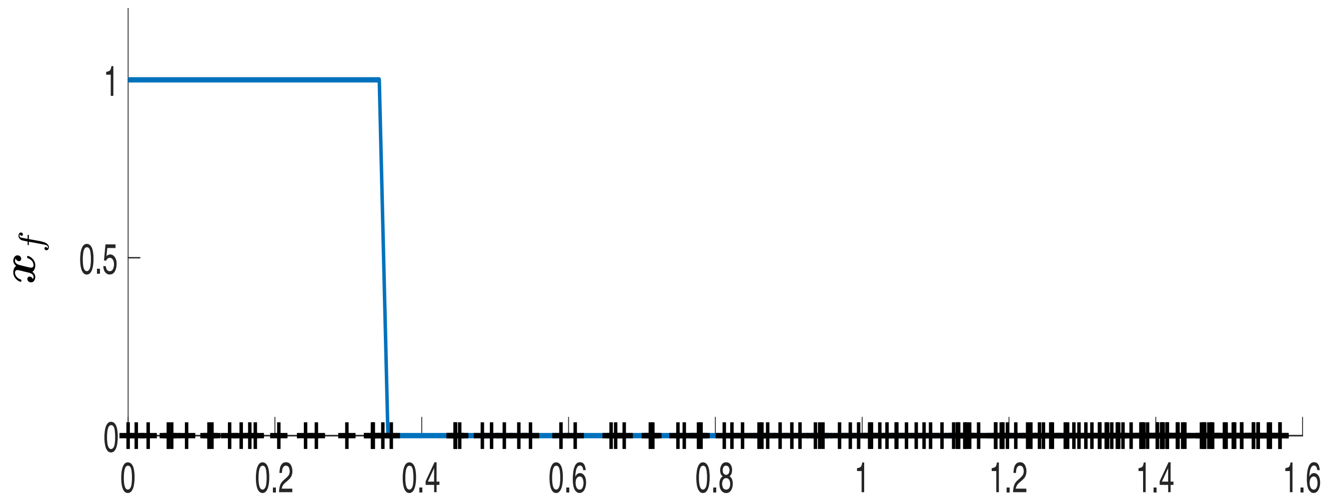
Decomposition of the (graph) signal $\boldsymbol{x} \in \mathbb{R}^N$ w.r.t. the orthonormal basis $\boldsymbol{U}$

$$\boldsymbol{x}_f := \boldsymbol{U}^H \boldsymbol{x} \Leftrightarrow \boldsymbol{x} =: \boldsymbol{U}\boldsymbol{x}_f$$



Field distribution

$\boldsymbol{x}$ is the field values measured at mesh points



Laplacian eigenvalues
(non-uniform discrete frequency grid)

Graph filters

- *Graph filters* (polynomial of the *graph-shift* operator) can be used to modify the frequency content of graph signals

$$\boldsymbol{H} = \sum_{l=0}^{L-1} h_l \boldsymbol{S}^l = \boldsymbol{U} \left(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l \right) \boldsymbol{U}^H = \boldsymbol{U} \text{diag}(\boldsymbol{h}_f) \boldsymbol{U}^H$$

Shift invariant: $\boldsymbol{H}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{H}$ and distributable: $\boldsymbol{x}_l = \boldsymbol{S}\boldsymbol{x}_{l-1}$

- Filter design using least squares, by solving the following linear system

$$\begin{bmatrix} h_{f,1} \\ h_{f,2} \\ \vdots \\ h_{f,N} \end{bmatrix} = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{L-1} \\ 1 & \lambda_2 & \cdots & \lambda_1^{L-1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \lambda_N & \cdots & \lambda_N^{L-1} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{L-1} \end{bmatrix}$$

Graph filters

- *Graph filters* (polynomial of the *graph-shift* operator) can be used to modify the frequency content of graph signals

$$\boldsymbol{H} = \sum_{l=0}^{L-1} h_l \boldsymbol{S}^l = \boldsymbol{U}\left(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l\right)\boldsymbol{U}^H = \boldsymbol{U}\text{diag}(\boldsymbol{h}_f)\boldsymbol{U}^H$$

Shift invariant: $\boldsymbol{HS} = \boldsymbol{SH}$ and distributable: $\boldsymbol{x}_l = \boldsymbol{S}\boldsymbol{x}_{l-1}$

- Vertex-domain vs. frequency-domain implementation

Vertex-domain implementation: $\boldsymbol{y} = \boldsymbol{Hx}$

Frequency-domain implementation: $\boldsymbol{y}_f = \text{diag}(\boldsymbol{h}_f)\boldsymbol{x}_f$

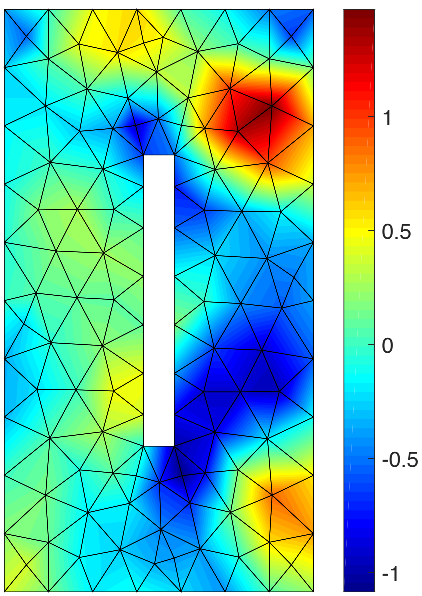
  - No fast GFT implementations
  - Parametrized filter implementation in the vertex-domain is possible

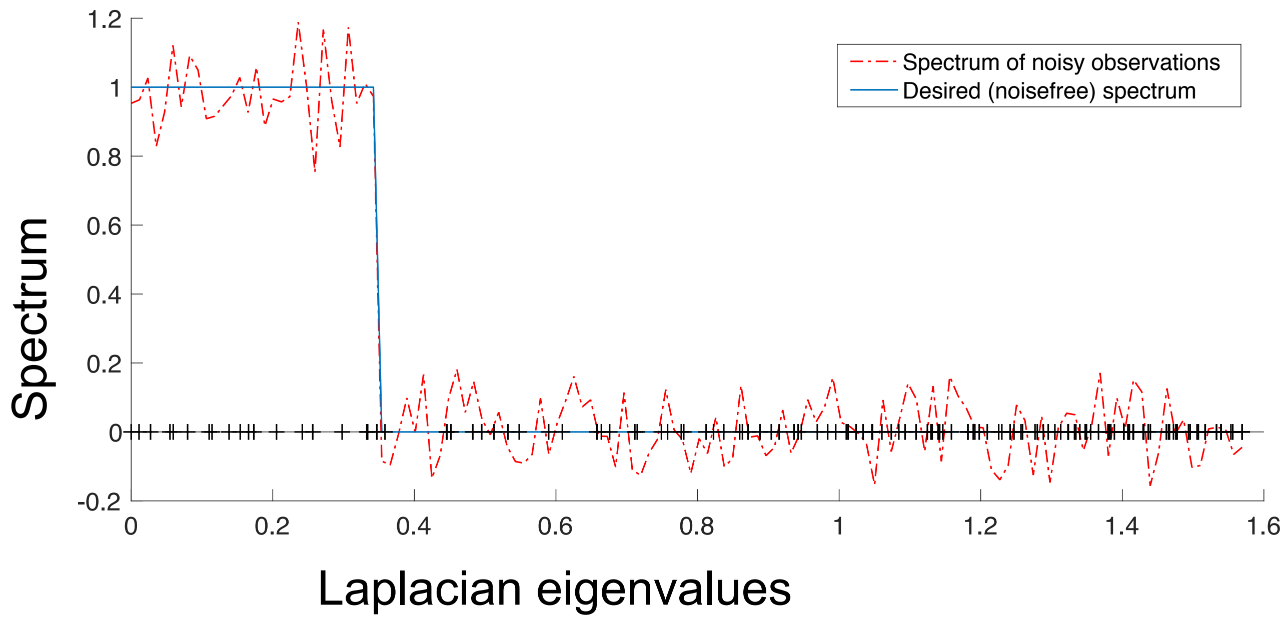
Graph filters

- *Graph filters* (polynomial of the *graph-shift* operator) can be used to modify the frequency content of graph signals

$$\boldsymbol{H} = \sum_{l=0}^{L-1} h_l \boldsymbol{S}^l = \boldsymbol{U}\left(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l\right)\boldsymbol{U}^H = \boldsymbol{U}\text{diag}(\boldsymbol{h}_f)\boldsymbol{U}^H$$

Denoising example:





Graph Signal Sampling

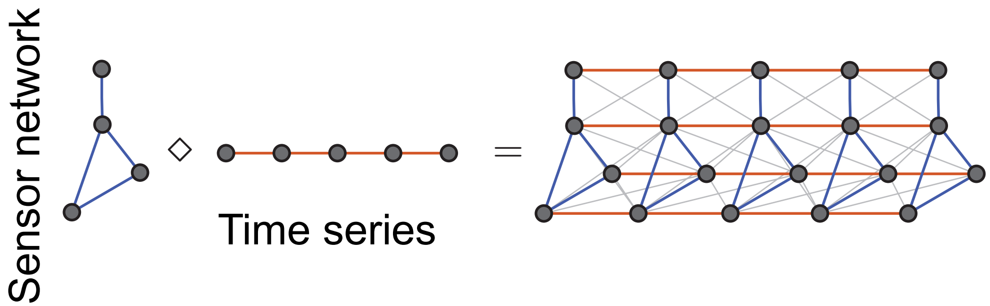
- S.P. Chepuri, Y. Eldar and G. Leus. Graph Sampling With and Without Input Priors. In Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2018), Calgary, Canada, April 2018.

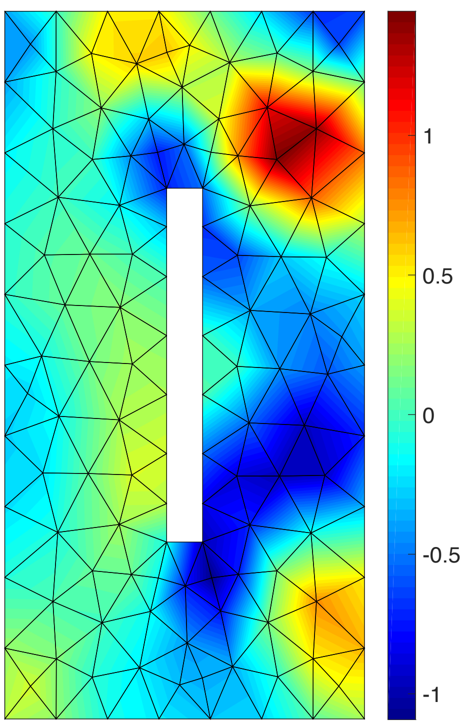
- S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, “Discrete signal processing on graphs: Sampling theory,” IEEE TSP, vol. 63, no. 24, pp. 6510–6523, Dec. 2015.

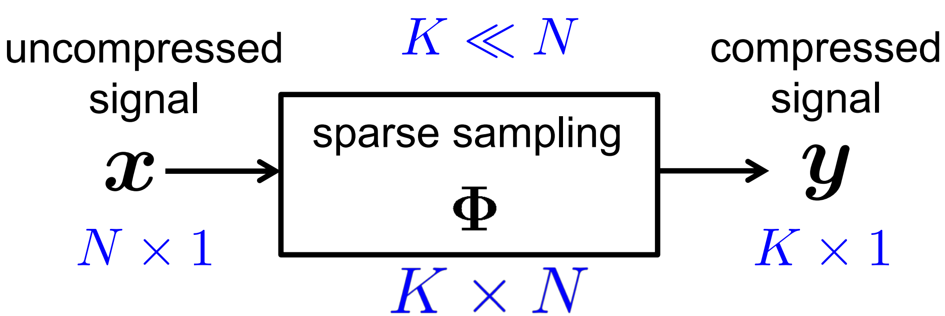
- D. Romero, M. Ma, and G.B. Giannakis. Kernel-Based Reconstruction of Graph Signals, IEEE TSP, vol. 65, no. 3, pp. 764–778, Feb 2017.

Sparse sampling on irregular domains

Active learning or semi-supervised learning







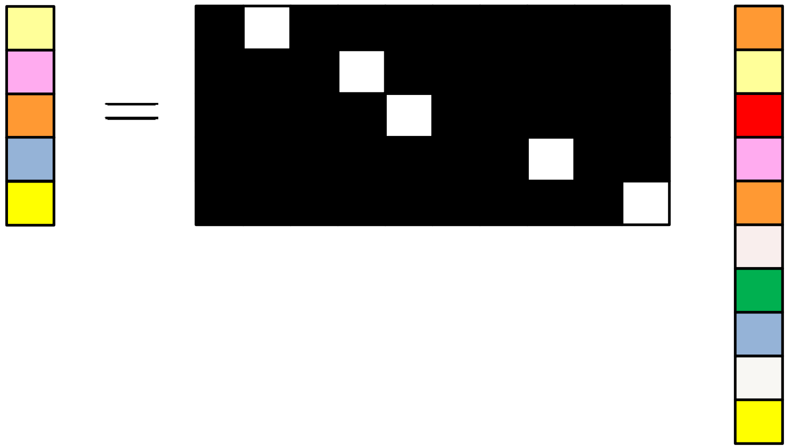
Given y estimate x

What is sparse sampling?

$$\boldsymbol{\Phi}(\boldsymbol{w}) \in \{0,1\}^{K \times N}$$

$\boldsymbol{y}$

$$\boldsymbol{R}_{\boldsymbol{y}} = \mathrm{E}\left\{\boldsymbol{y}\boldsymbol{y}^H\right\}$$



$\boldsymbol{x}$

$$\boldsymbol{R}_{\boldsymbol{x}} = \mathrm{E}\left\{\boldsymbol{x}\boldsymbol{x}^H\right\}$$

- Sampling matrix is determined by the sampling vector/set

$$\boldsymbol{w} = [w_1, w_2, \ldots, w_N]^T \in \{0,1\}^N \quad \text{or} \quad \mathcal{S} = \{n | w_n = 1, n = 1, 2, \ldots, N\}$$

$w_m = (0)1$ sample or vertex is (not) selected

- Sparse sampling structure
  - only one nonzero entry per row
  - many zero columns

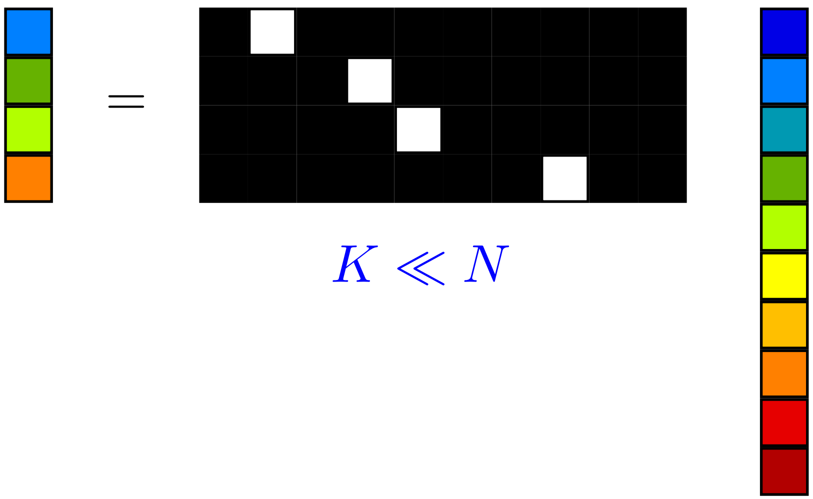
S.P. Chepuri and G. Leus. Sparse Sensing for Statistical Inference. *Foundations and Trends in Signal Processing, Vol. 9: No. 3–4, pp 233-368, Dec. 2016.*

Why sparse sampling or active learning?

- **Economical** constraints (hardware cost)
- Limited **physical space**
- Limited data **storage space**
- **Labelling** is expensive
- Reduce **communications bandwidth**
- Reduce **processing overhead**

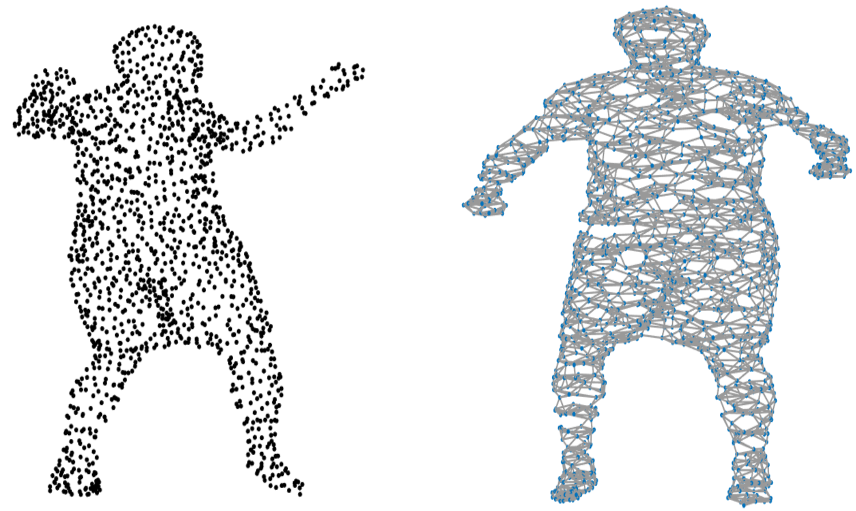
Sparse graph sampling

$\boldsymbol{y}$ $\quad \boldsymbol{\Phi} \in \{0,1\}^{K \times N}$ $\quad \boldsymbol{x}$



$K \ll N$

graph signal



Given y estimate x

signal: 3D points, which are displacements of graph nodes

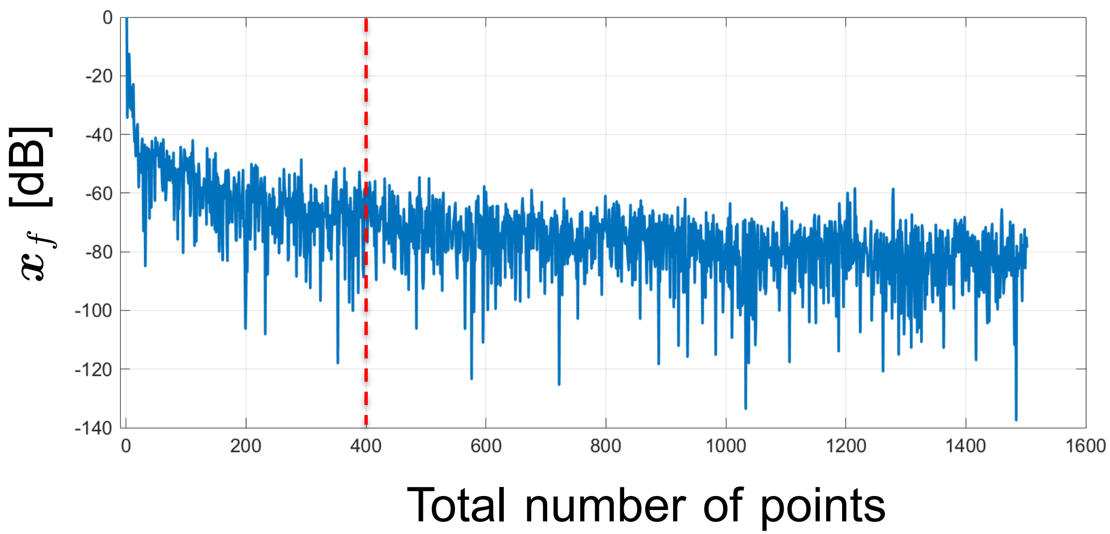
Bandlimited graph signals – subspace prior

Suppose the support of the sparse $\boldsymbol{x}_f$ is known

$$\boldsymbol{x} = \boldsymbol{U}\boldsymbol{x}_f = \left[\ \boldsymbol{U}_{\mathsf{BL}}\ \middle|\ \star\ \right] \begin{bmatrix} \tilde{\boldsymbol{x}}_f \\ \hline \boldsymbol{0} \end{bmatrix} \quad \Leftrightarrow \quad \boldsymbol{x} = \boldsymbol{U}_{\mathsf{BL}}\tilde{\boldsymbol{x}}_f$$

$\boldsymbol{U}_{\mathsf{BL}}$: $N \times L$; $\tilde{\boldsymbol{x}}_f$: $L \times 1$

$\boldsymbol{x} \in$ range$(\boldsymbol{U}_{\mathsf{BL}})$ —a known L-dimensional subspace



Bandlimited graph signals – subspace prior

With sparse sampling, we get K equations in L unknowns

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{U}_{\mathsf{BL}}\tilde{\boldsymbol{x}}_f$$

If the matrix $\boldsymbol{\Phi}\boldsymbol{U}_{\mathsf{BL}}$ has full column rank, i.e, range$(\boldsymbol{U}_{\mathsf{BL}}) \cap$ null$(\boldsymbol{\Phi}) = \{0\}$:

Least squares solution: $\widehat{\tilde{\boldsymbol{x}}}_f = (\boldsymbol{\Phi}\boldsymbol{U}_{\mathsf{BL}})^{\dagger}\boldsymbol{y}$

Design of $\boldsymbol{\Phi}$ crucial for the least-squares solution to be unique

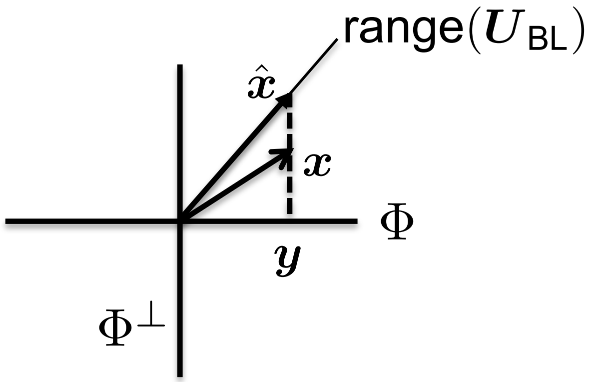
Bandlimited graph signals – subspace prior

- With sparse sampling, we get K equations in L unknowns

$$\boldsymbol{y} = \boldsymbol{\Phi x} = \boldsymbol{\Phi U}_{\mathsf{BL}} \tilde{\boldsymbol{x}}_f$$

- *Oblique projection* of x onto the range$(\boldsymbol{U}_{\mathsf{BL}})$ and along the null$(\boldsymbol{\Phi})$

$$\hat{\boldsymbol{x}} = \boldsymbol{U}_{\mathsf{BL}} (\boldsymbol{U}_{\mathsf{BL}}^H \boldsymbol{\Phi}^T \boldsymbol{\Phi U}_{\mathsf{BL}})^{-1} \boldsymbol{U}_{\mathsf{BL}}^H \boldsymbol{\Phi}^T \boldsymbol{\Phi x} = \boldsymbol{E}_{U_{\mathsf{BL}} \Phi^{\perp}} \boldsymbol{x}$$

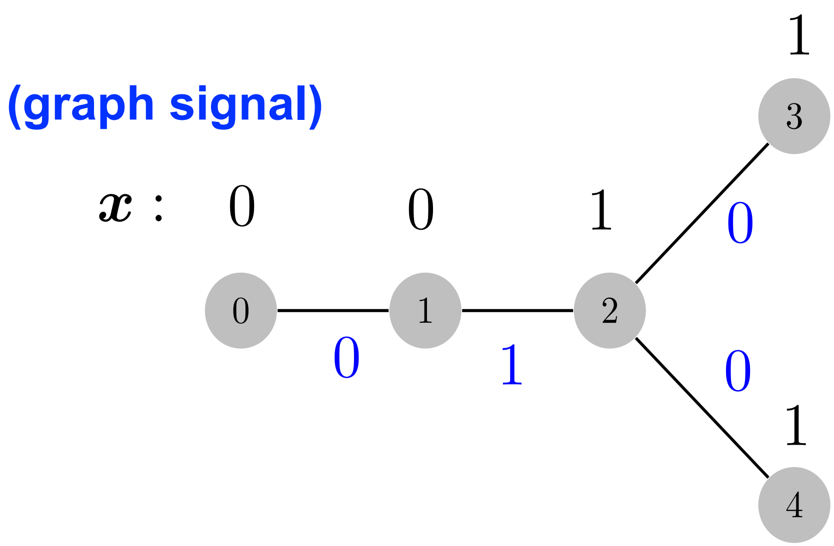


- A more interesting case, perhaps is, when the support is not known!

Reconstruction with smoothness prior

- Assume x is smooth with respect to the underlying graph or has small

$$\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2$$



$$\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2$$

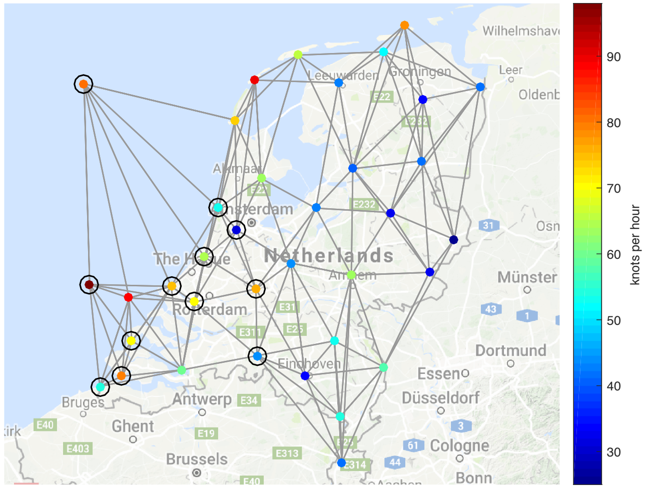
$$= 1$$

Sum of squares of differences across edges

Reconstruction with smoothness prior

- When the prior subspace is not known, we can be ***consistent*** (cf. interpolation)

$$\boldsymbol{\Phi}\boldsymbol{x} = \boldsymbol{\Phi}\hat{\boldsymbol{x}}$$



- Assume $\boldsymbol{x}$ is smooth with respect to the underlying graph or has small

- Equality constrained quadratic program

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{x}^H \boldsymbol{L}\boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{\Phi}\boldsymbol{x} = \boldsymbol{y}$$

Solution: $$\begin{bmatrix} \boldsymbol{L} + \boldsymbol{\Phi}^T\boldsymbol{\Phi} & \boldsymbol{\Phi}^T \\ \boldsymbol{\Phi} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}^T\boldsymbol{y} \\ \boldsymbol{y} \end{bmatrix}$$

If $\text{null}(\boldsymbol{L}) \cap \text{null}(\boldsymbol{\Phi}) = \{0\}$, then $\hat{\boldsymbol{x}} = \tilde{\boldsymbol{L}}(\boldsymbol{\Phi}\tilde{\boldsymbol{L}})^{-1}\boldsymbol{y}$

$$\tilde{\boldsymbol{L}} = (\boldsymbol{L} + \boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$$

Sampling via graph filtering

Sparse sampling in spectral domain:

- Suppose sampling operator collects the first K contiguous frequencies
- Sampling and interpolation operations can be implemented via graph filters

$$\hat{\boldsymbol{x}} = \boldsymbol{H}_{\text{interp}} \boldsymbol{H}_{\text{samp}} \boldsymbol{x}.$$

- Subspace prior

diagonal matrix

$$\boldsymbol{\Phi} = \boldsymbol{E}_K \boldsymbol{U}^H \Rightarrow \boldsymbol{H}_{\text{samp}} = \boldsymbol{\Phi}^H \boldsymbol{\Phi} = \boldsymbol{U} \boldsymbol{E}_K^T \boldsymbol{E}_K \boldsymbol{U}^H \qquad \boldsymbol{E}_K = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_K]$$

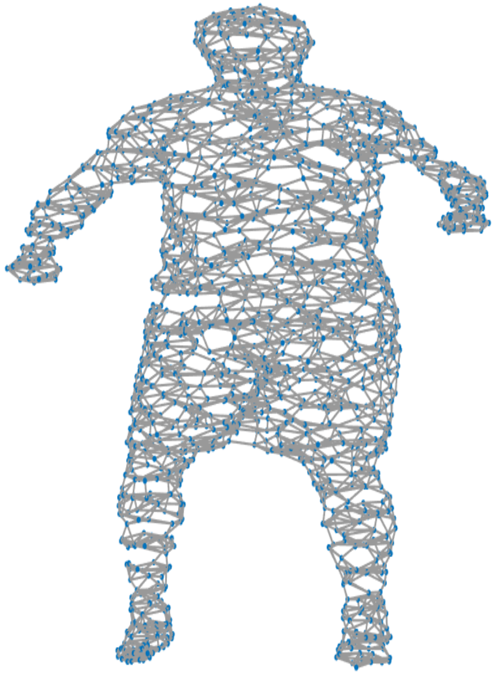
$$\boldsymbol{H}_{\text{interp}} = \boldsymbol{U}_{\text{BL}} \boldsymbol{H}_{f,\text{interp}} \boldsymbol{U}_{\text{BL}}^H \qquad \boldsymbol{H}_{f,\text{interp}}^{-1} = \boldsymbol{U}_{\text{BL}}^H \boldsymbol{H}_{\text{samp}} \boldsymbol{U}_{\text{BL}} \text{ (diagonal)}$$

- Smoothness prior

$$\boldsymbol{H}_{f,\text{samp}} = \boldsymbol{E}_K^T [\boldsymbol{E}_K (\boldsymbol{\Lambda} + \boldsymbol{E}_K^T \boldsymbol{E}_K)^{-1} \boldsymbol{E}_K^T]^{-1} \boldsymbol{E}_K \quad \text{(diagonal)}$$

$$\boldsymbol{H}_{\text{interp}} = \boldsymbol{U} (\boldsymbol{\Lambda} + \boldsymbol{E}_K^T \boldsymbol{E}_K)^{-1} \boldsymbol{U}^H$$

Numerical experiments



Graph (K-nearest neighbor)

Original signal (3D points)

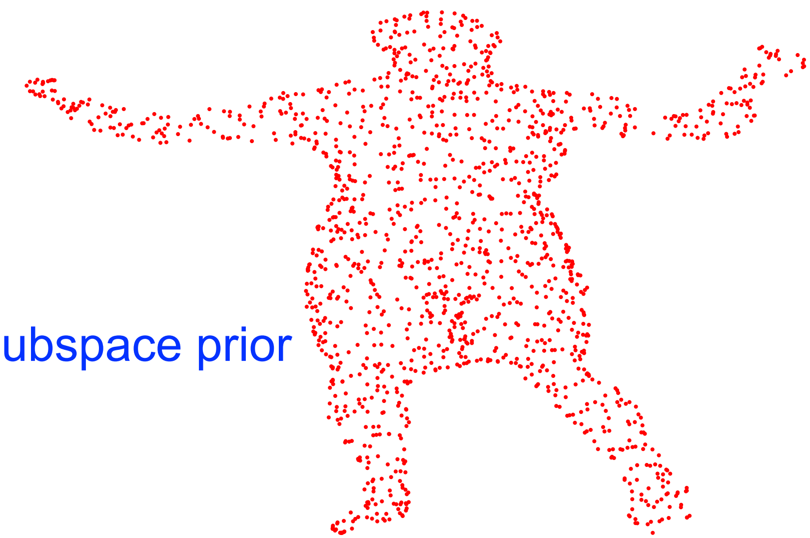
$N = 1502$, $K = 600$, $K/N \approx 40\%$ compression

Numerical experiments

Original signal

$N = 1502,\ K = 600,\ K/N \approx 40\%$ compression

Subspace prior



Smoothness prior

Sampling diffusion fields over graphs

- S. Reddy and S.P. Chepuri. Sampling and Reconstruction of Diffusive Fields on Graphs. *GlobalSIP 2019*, Ottawa, Canada.
- A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, “Sampling of graph signals with successive local aggregations,” IEEE TSP, vol. 64, no. 7, pp. 1832–1834, Arp. 2016.

Sampling diffusion processes

- Let us consider the heat equation

$$\frac{\partial x(t, \mathbb{D})}{\partial t} = -\nabla^2 x(t, \mathbb{D})$$

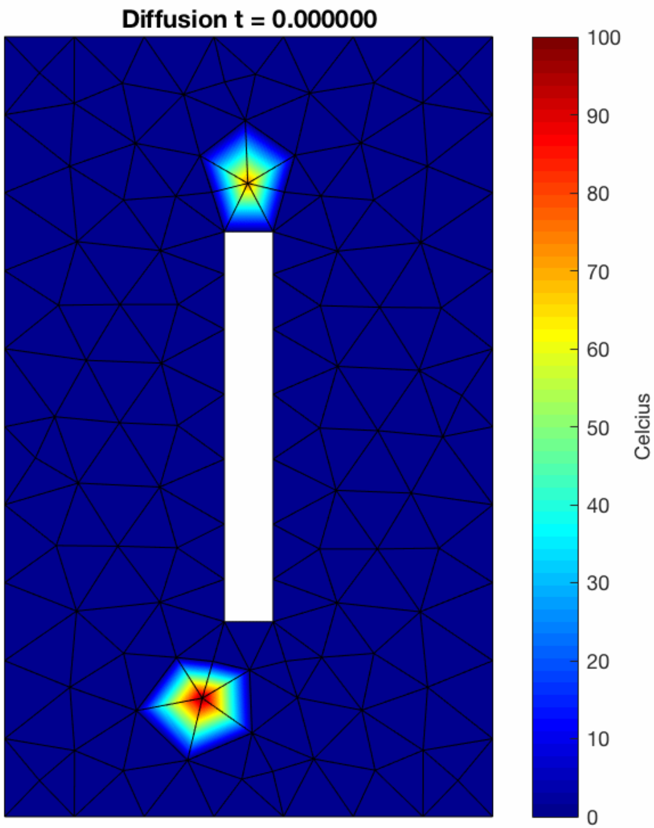
- Often, we approximate complicated manifolds with a mesh (e.g., Delaunay mesh)

$$\frac{\partial \boldsymbol{x}(t)}{\partial t} = -\boldsymbol{L}\boldsymbol{x}(t)$$

Solution:

$$\boldsymbol{x}(t) = e^{-t\boldsymbol{L}}\boldsymbol{x}(0) = \boldsymbol{U}e^{-t\boldsymbol{\Lambda}}\boldsymbol{U}^H\boldsymbol{x}(0)$$

- Initial condition can be computed by observing all the mesh points "once" for some $t > 0$



Frozen metal plate with cavity
initial condition: two spikes

Design structured (sparse) space-time samplers

Sampling diffusion processes

- Sample $\boldsymbol{x}(t) = e^{-t\boldsymbol{L}}\boldsymbol{x}(0)$ at times $t_1 \leq t_2 \leq \cdots \leq t_T$

$$\begin{aligned} \boldsymbol{x}(t_k) &= e^{-t_k \boldsymbol{L}}\boldsymbol{x}(0) \\ &= \boldsymbol{U} \begin{bmatrix} e^{-\lambda_1 t_k} & & & \\ & e^{-\lambda_2 t_k} & & \\ & & \ddots & \\ & & & e^{-\lambda_N t_k} \end{bmatrix} \boldsymbol{\theta} = \boldsymbol{U}\text{diag}(\boldsymbol{\theta})\boldsymbol{a}(t_k) \end{aligned}$$

with $\boldsymbol{\theta} = \boldsymbol{U}^H \boldsymbol{x}(0)$ and $\boldsymbol{a}(t_k) = [e^{-\lambda_1 t_k}, \ldots, e^{-\lambda_N t_k}]^T$

- Stacking all the space-time samples

$$\boldsymbol{X} = \boldsymbol{U}\text{diag}(\boldsymbol{\theta})\boldsymbol{A}^T \qquad \boldsymbol{A} = [\boldsymbol{a}(t_1), \cdots \boldsymbol{a}(t_T)]^T$$

- Sparse space-time sampling amounts to observing a few mesh points at a few time instances

Given $\boldsymbol{L}$ and $\boldsymbol{Y} = \boldsymbol{\Phi}_s \boldsymbol{X} \boldsymbol{\Phi}_t^T$ find the initial condition $\boldsymbol{\theta}$

Sampling diffusion processes

- On vectorizing $\boldsymbol{Y} = \boldsymbol{\Phi}_s \boldsymbol{X} \boldsymbol{\Phi}_t = \boldsymbol{\Phi}_s \boldsymbol{U} \text{diag}(\boldsymbol{\theta}) \boldsymbol{A}^T \boldsymbol{\Phi}_t^T$

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{\Phi}_t \boldsymbol{A} \circ \boldsymbol{\Phi}_s \boldsymbol{U})\boldsymbol{\theta} \\ &= (\boldsymbol{\Phi}_t \otimes \boldsymbol{\Phi}_s)(\boldsymbol{A} \circ \boldsymbol{U})\boldsymbol{\theta} \end{aligned}$$

$\boldsymbol{y} : K_t K_s \times 1$, $\boldsymbol{\Phi}_t : K_t \times T$, $\boldsymbol{\Phi}_s : K_s \times N$ $\quad$ $\text{vec}(\boldsymbol{A}\text{diag}(\boldsymbol{d})\boldsymbol{B}) = (\boldsymbol{B}^T \circ \boldsymbol{A})\boldsymbol{d}$

$\otimes$: Kronecker product; $\circ$: Khatri-Rao (columnwise Kronecker) product

If the matrix $\boldsymbol{\Phi}_t \boldsymbol{A} \circ \boldsymbol{\Phi}_s \boldsymbol{U}$ has full column rank, which requires $K_t K_s \geq N$:

Least squares solution: $\widehat{\boldsymbol{\theta}} = [\boldsymbol{\Phi}_t \boldsymbol{A} \circ \boldsymbol{\Phi}_s \boldsymbol{U}]^\dagger \boldsymbol{y}$

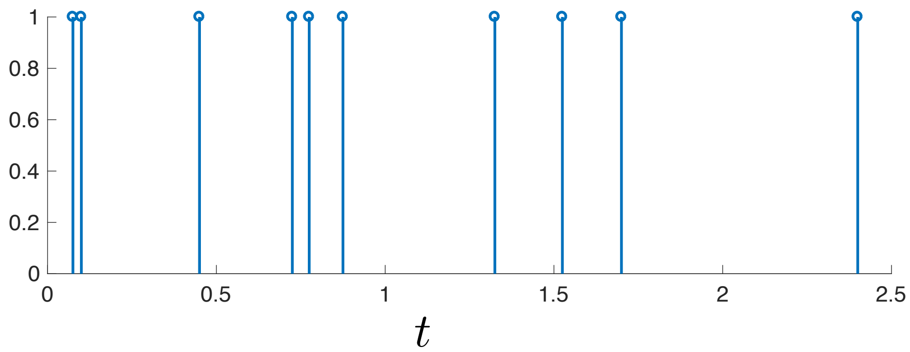
$$\widehat{\boldsymbol{x}}(0) = \boldsymbol{U}\widehat{\boldsymbol{\theta}}$$

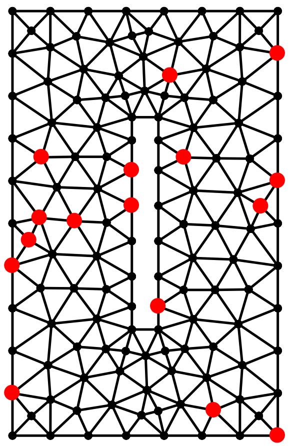
Remark: $\boldsymbol{\theta}$ is not sparse in general, as $\boldsymbol{x}(0)$ is sparse

Bandlimiting constraint is not required

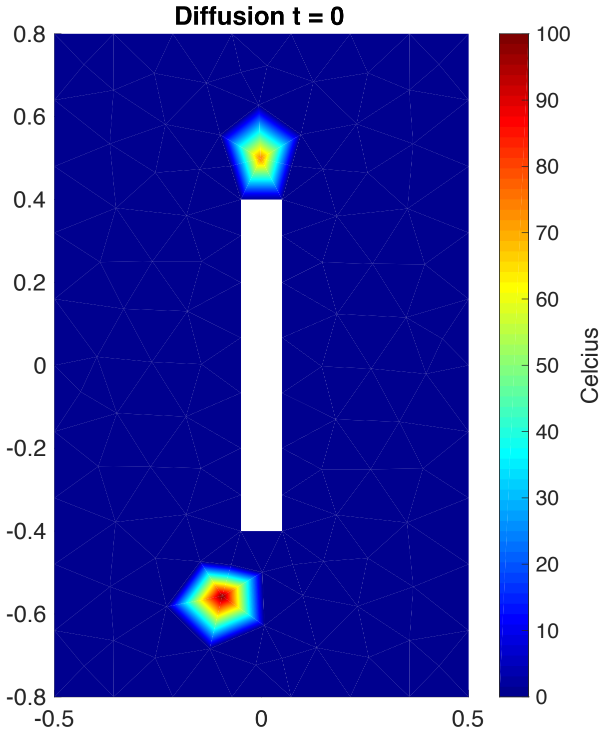
Experiments

sampled at 10 non-uniform time instances

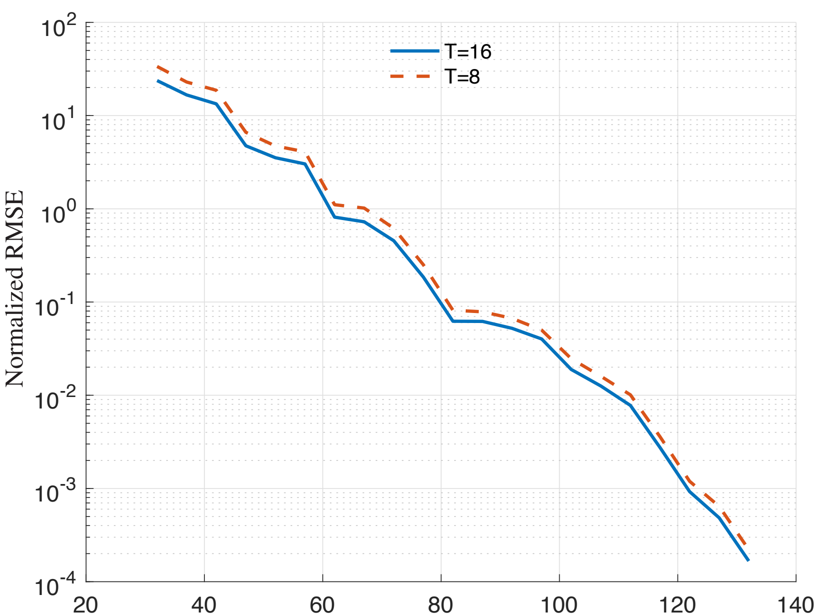




16 out 134 mesh points observed

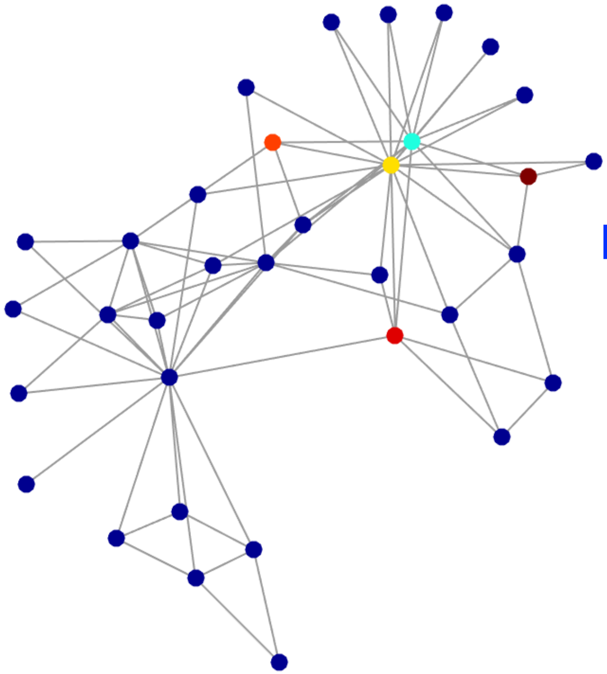


estimated initial condition

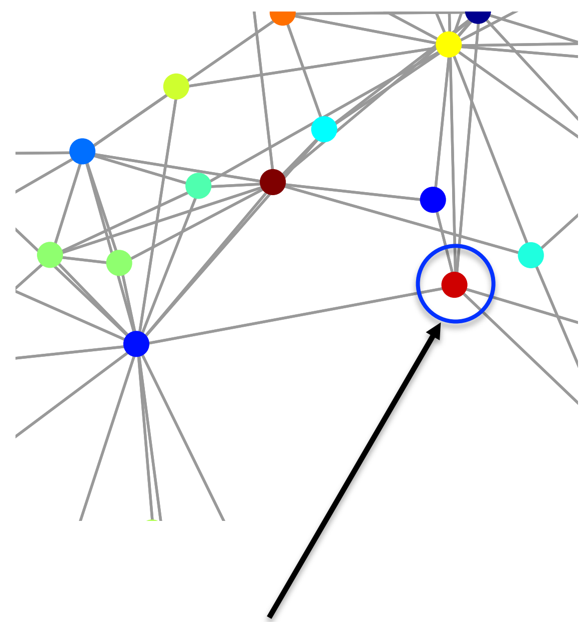


observed mesh points

Linear dynamics over networks



Linear dynamics over networks



Can we reconstruct a graph signal from observations at a single node?

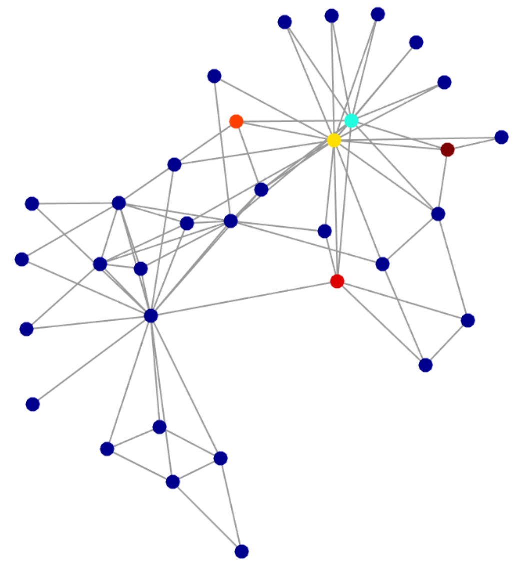
Linear dynamics on networks

- Information flow to a node from its neighbors

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{S}\boldsymbol{x}_{k-1} + \boldsymbol{x}u_{k-1} \\ y_k &= \boldsymbol{e}_i^T \boldsymbol{x}_k \end{aligned}$$

sample *node i*

Linear network dynamics



$\boldsymbol{x}_{-1} = 0$ and $\boldsymbol{x}_0 = \boldsymbol{x}$

$u_{k-1} = \delta[k]$ (Kronecker delta)

$\boldsymbol{e}_i$ is the ith column of the identity matrix

- Given observations $\boldsymbol{y} = \{y_0, \ldots, y_{K-1}\}$ estimate $\boldsymbol{x}$

K is the number of shifts applied

Linear dynamics on networks

- At the observed node

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{S} \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{S}^{K-1} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^H \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{U}\boldsymbol{\Lambda}^{K-1}\boldsymbol{U}^H \end{bmatrix} \boldsymbol{x}$$

$$= \boldsymbol{V}\text{diag}[\underline{\boldsymbol{u}}]\boldsymbol{U}^H\boldsymbol{x} = \boldsymbol{V}\text{diag}[\underline{\boldsymbol{u}}]\boldsymbol{x}_f$$

Spectral response

$\underline{\boldsymbol{u}} = \boldsymbol{e}_i^T \boldsymbol{U}$ and $[\boldsymbol{V}]_{i,j} = \lambda_j^{i-1}$ (Vandermonde)

- Aggregation sampling is natural while observing time domain signals

Linear dynamics on networks

Recall bandlimitedness:

- Suppose the support of the sparse $\boldsymbol{x}_f$ is known

$$\boldsymbol{x} = \boldsymbol{U}\boldsymbol{x}_f = \left[\ \boldsymbol{U}_{\mathsf{BL}} \ \middle|\ \star\ \right] \left[\begin{array}{c} \tilde{\boldsymbol{x}}_f \\ \hline \boldsymbol{0} \end{array}\right] \quad \Leftrightarrow \quad \boldsymbol{x} = \boldsymbol{U}_{\mathsf{BL}}\tilde{\boldsymbol{x}}_f$$

- The observations at *node i* will then be

$$\boldsymbol{y} = \boldsymbol{V}\text{diag}[\underline{\boldsymbol{u}}]\boldsymbol{x}_f = \boldsymbol{V}\text{diag}[\underline{\boldsymbol{u}}]\boldsymbol{E}_L\tilde{\boldsymbol{x}}_f = \boldsymbol{V}_{\mathsf{BL}}\tilde{\boldsymbol{x}}_f$$

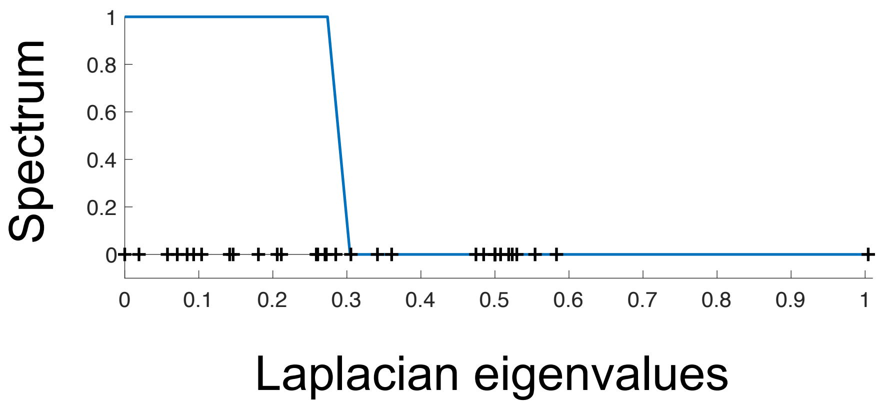
$$\boldsymbol{E}_L = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_L]$$

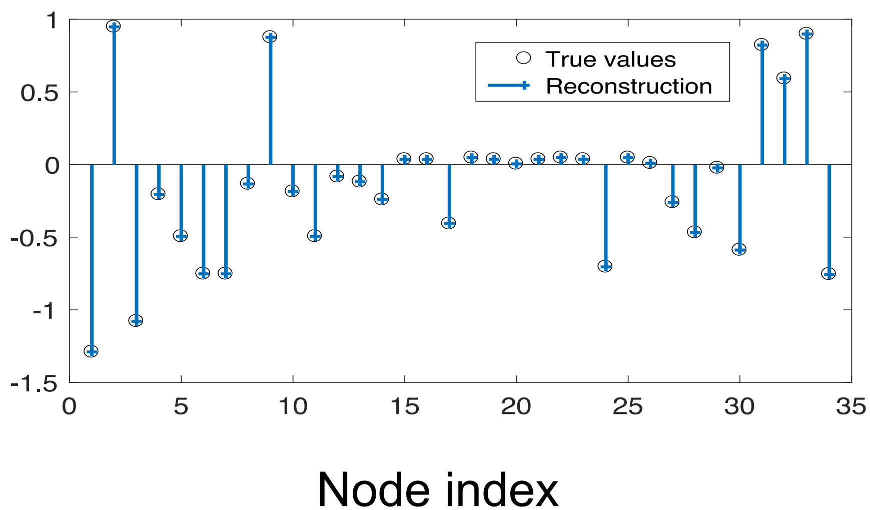
\# of shifts

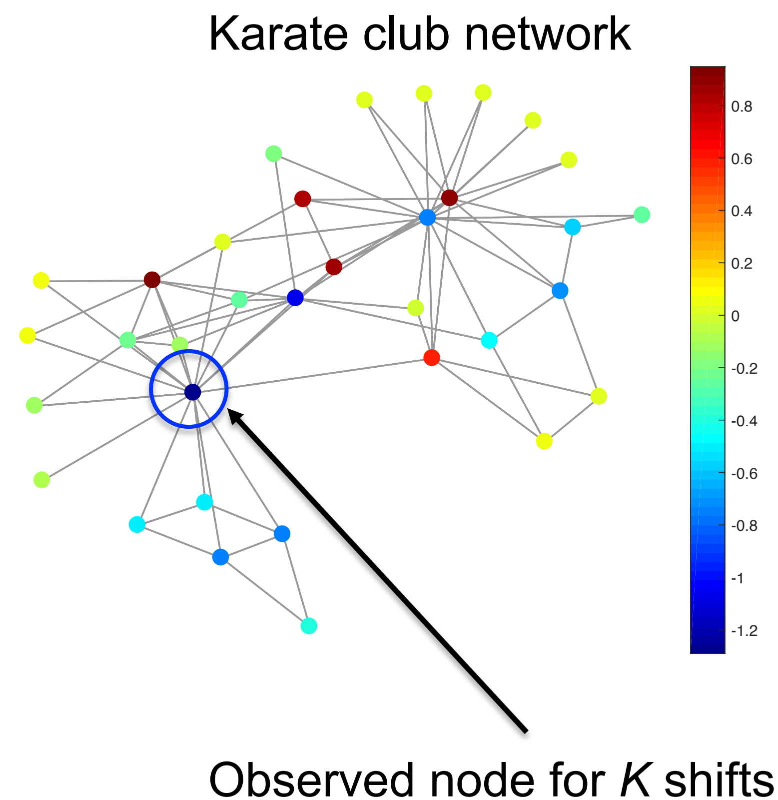
- If the matrix $\boldsymbol{V}_{\mathsf{BL}}$ has full column rank, which requires $K \geq L$:

Least squares solution: $\widehat{\tilde{\boldsymbol{x}}}_f = \boldsymbol{V}_{\mathsf{BL}}^{\dagger}\boldsymbol{y}$

Numerical experiments





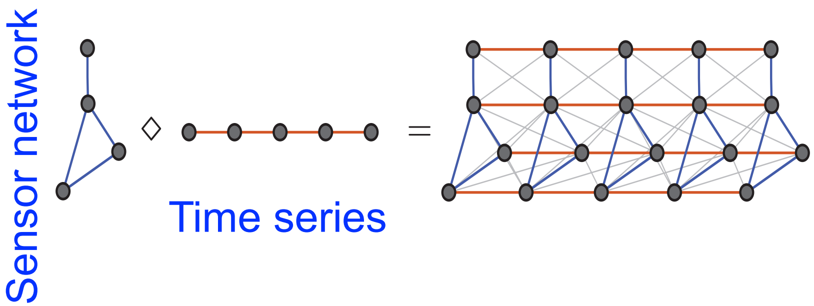


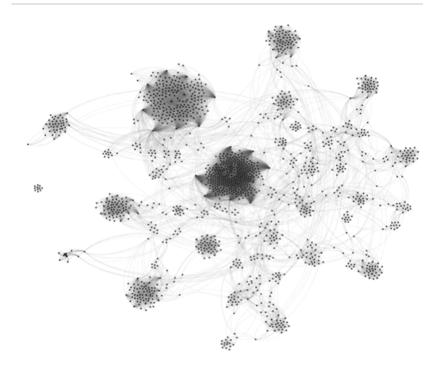
- ➢ Although reconstruction possible by observing a single node, system gets quickly ill conditioned (very sensitive to noise).
- ➢ Combining observations from a few more nodes might improve conditioning

Product Graph Sampling

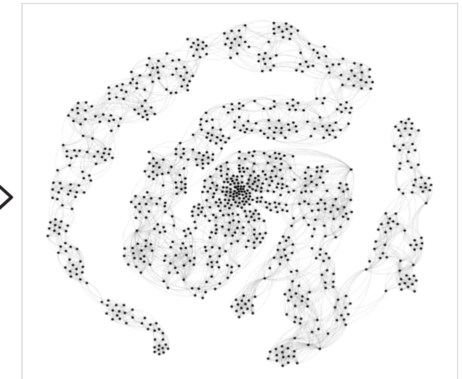
- G. Ortiz-Jiménez, M. Coutino, S.P. Chepuri, and G. Leus. Sampling and Reconstruction of Signals on Product Graphs. *GlobalSIP 2018*, Anaheim, USA..
- G. Ortiz-Jiménez, M. Coutino, S.P. Chepuri, and G. Leus. Sparse Sampling for Inverse Problems with Tensors. *IEEE TSP*, Feb 2019.

Sparse sampling on multigraph domains

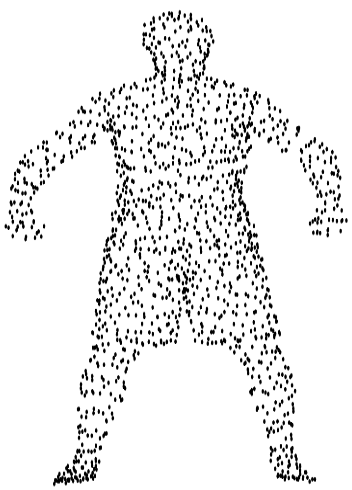




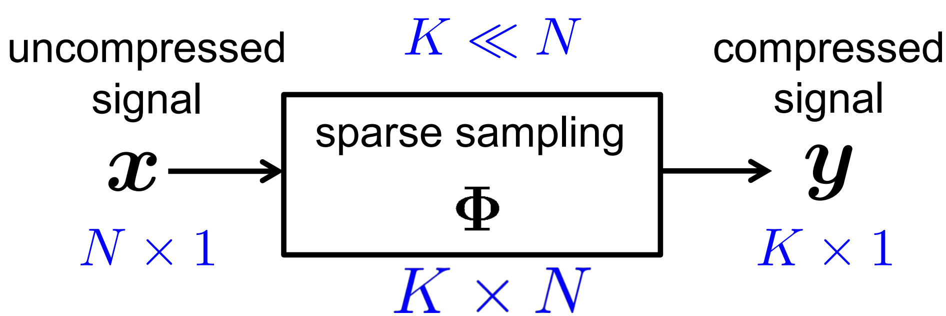
Movie graph



Social network

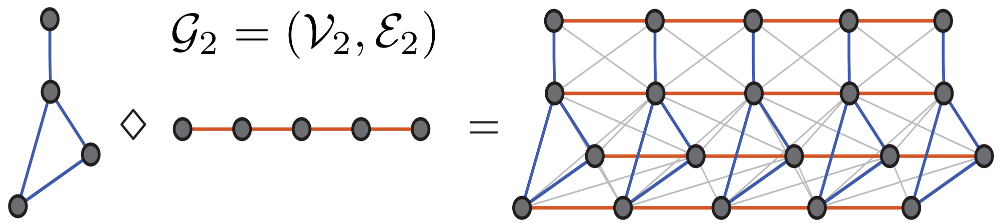


Dynamic 3D point cloud



Given y estimate x

Product graphs



N_2 nodes

$\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$

$\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$

N_1 nodes

$\mathcal{G}_\diamond = (\mathcal{V}_1 \times \mathcal{V}_2, \mathcal{E}_\diamond)$

$N = N_1 N_2$ nodes

- Cartesian product (colored edges)
- Kronecker product (gray edges)
- Strong product (all edges)

➢ Let us represent $\mathcal{G}_1$ and $\mathcal{G}_2$ with the graph-shift operators

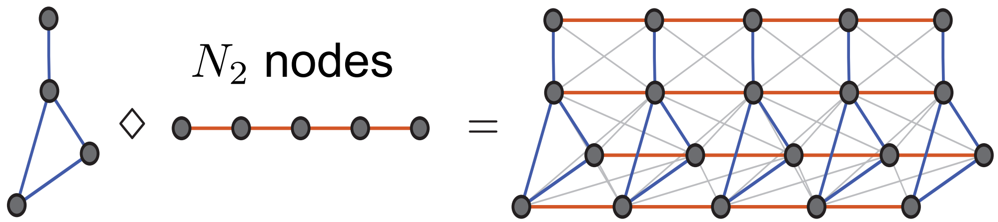
$$\boldsymbol{S}_1 = \boldsymbol{U}_1 \boldsymbol{\Lambda}_1 \boldsymbol{U}_1^H \in \mathbb{R}^{N_1 \times N_1} \quad \text{and} \quad \boldsymbol{S}_2 = \boldsymbol{U}_2 \boldsymbol{\Lambda}_2 \boldsymbol{U}_2^H \in \mathbb{R}^{N_2 \times N_2}$$

➢ The product graph $\mathcal{G}_\diamond$ has the graph-shift operator

$$\boldsymbol{S}_\diamond = (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2) \boldsymbol{\Lambda}_\diamond (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2)^H \in \mathbb{R}^{N \times N}$$

$\boldsymbol{\Lambda}_\diamond$ is a diagonal matrix that depends on $\mathcal{G}_1$ and $\mathcal{G}_2$, and the type of product

Product graph signals: The sampling problem

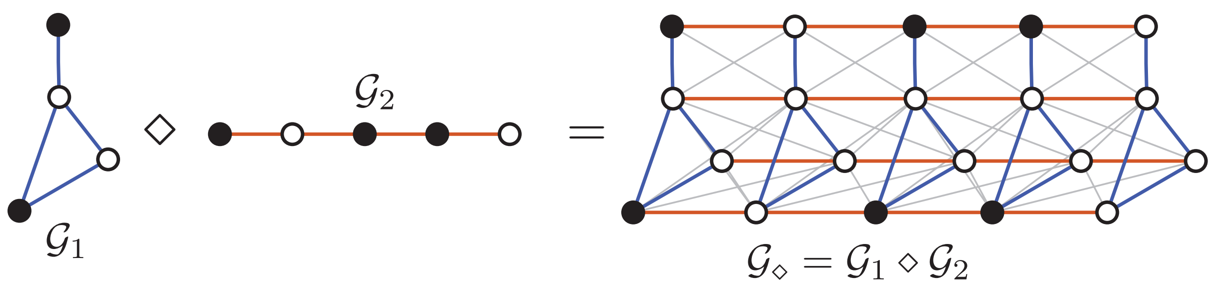


product graph signal

$\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2}$

Factor vertex selection

Observed product graph nodes



uncompressed signal

$\boldsymbol{X}$

$N_1 \times N_2$

sparse sampling

$\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$

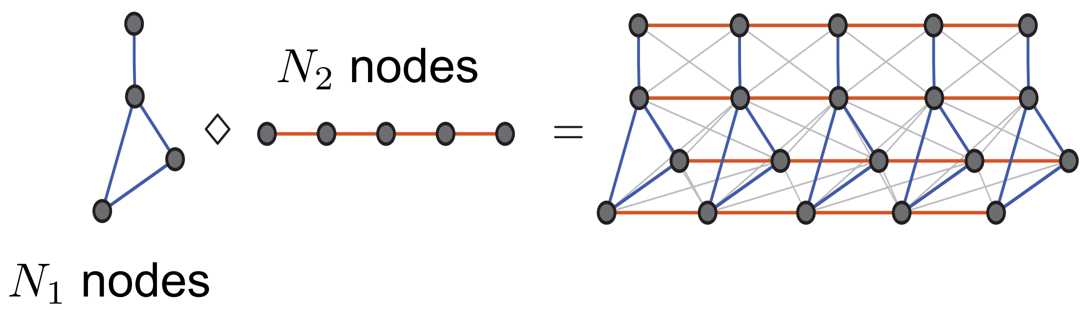
compressed signal

$\boldsymbol{Y} = \boldsymbol{\Phi}_1 \boldsymbol{X} \boldsymbol{\Phi}_2^T$

$K_1 \times K_2$

Given $\boldsymbol{Y}$ estimate $\boldsymbol{X}$

Product graph signal



product graph signal
$\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2}$

- Product graph signal $\boldsymbol{X}$ may be decomposed w.r.t. $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ as

$$\boldsymbol{X} = \boldsymbol{U}_1 \boldsymbol{X}_f \boldsymbol{U}_2^T \quad \Leftrightarrow \quad \boldsymbol{x} = (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2)\boldsymbol{x}_f$$

- More generally, for Rth-order product graph, we have a graph (tensor) signal

$$\mathcal{X} = \mathcal{X}_f \bullet_1 \boldsymbol{U}_1 \bullet_2 \boldsymbol{U}_2 \cdots \bullet \boldsymbol{U}_R \quad \Leftrightarrow \quad \boldsymbol{x} = (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2 \cdots \otimes \boldsymbol{U}_R)\boldsymbol{x}_f$$

$\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \cdots \times N_R}$

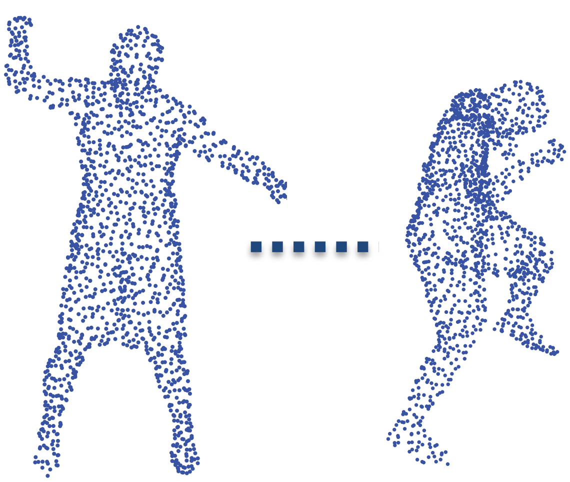
Bandlimited product graph signals

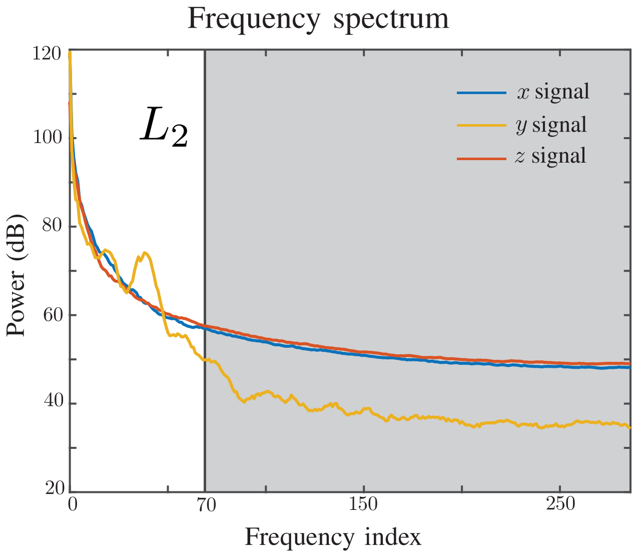
- Suppose the support of the sparse x_f is known

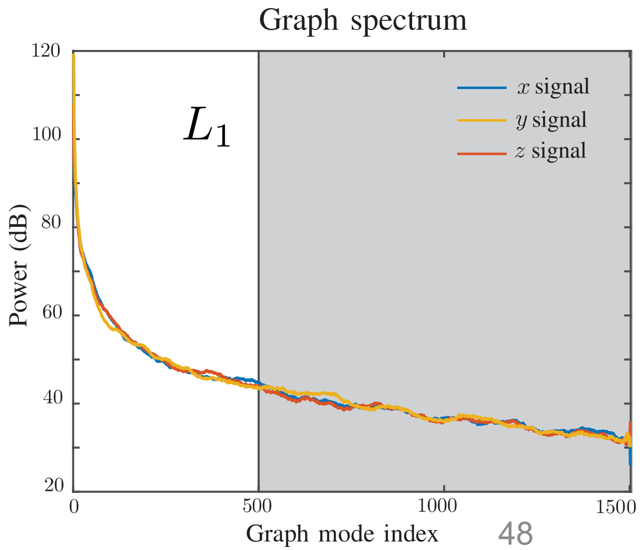
$$\boldsymbol{X} = \boldsymbol{U}_1 \boldsymbol{X}_f \boldsymbol{U}_2^T = \left[\, \underbrace{\tilde{\boldsymbol{U}}_1}_{N_1 \times L_1} \;\middle|\; \star \,\right] \left[\begin{array}{c|c} \tilde{\boldsymbol{X}}_f & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \underbrace{\tilde{\boldsymbol{U}}_2^T}_{L_2 \times N_2} \\ \hline \star \end{array}\right]$$

or

$$\boldsymbol{x} = (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2)\boldsymbol{x}_f = \left[\, (\tilde{\boldsymbol{U}}_1 \otimes \tilde{\boldsymbol{U}}_2) \;\middle|\; \star \,\right] \left[\begin{array}{c} \tilde{\boldsymbol{x}}_f \\ \hline \mathbf{0} \end{array}\right]$$







Bandlimited product graph signals

- Suppose the support of the sparse x_f is known

$$\boldsymbol{X} = \boldsymbol{U}_1 \boldsymbol{X}_f \boldsymbol{U}_2^T = \left[\; \underbrace{\tilde{\boldsymbol{U}}_1}_{N_1 \times L_1} \;\middle|\; \star \;\right] \left[\begin{array}{c|c} \tilde{\boldsymbol{X}}_f & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \underbrace{\tilde{\boldsymbol{U}}_2^T}_{L_2 \times N_2} \\ \hline \star \end{array}\right]$$

or

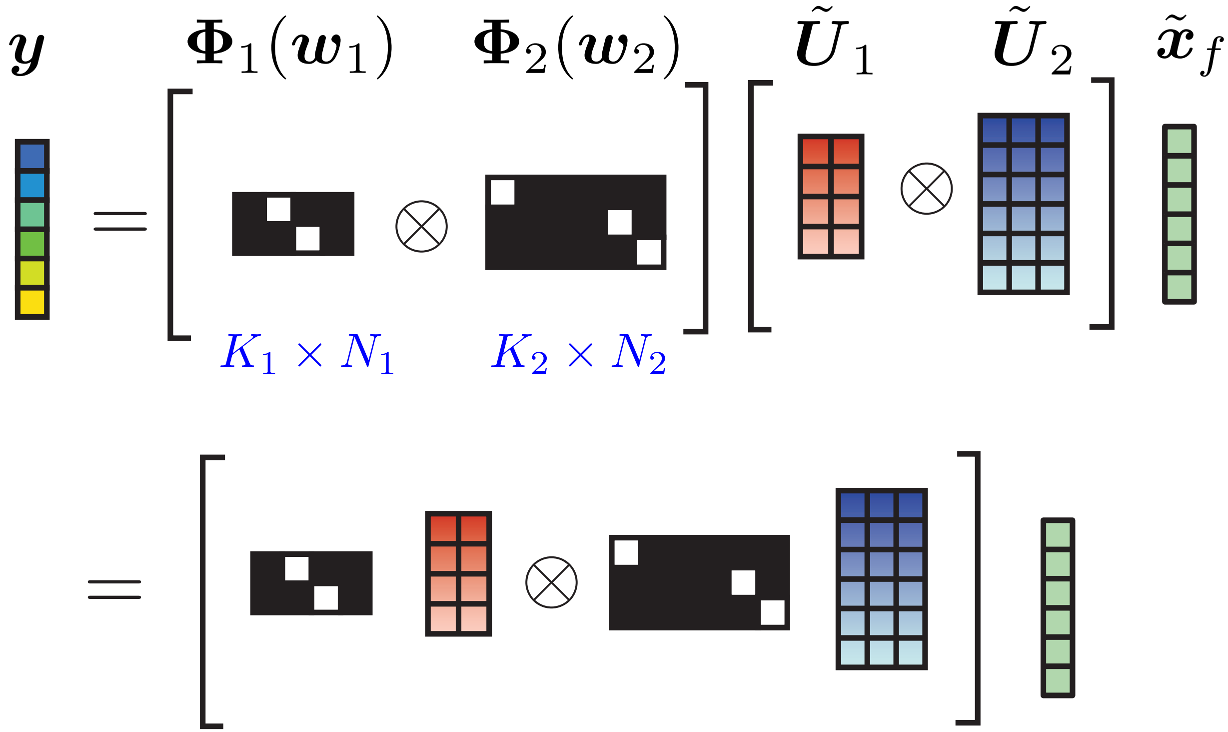
$$\boldsymbol{x} = (\boldsymbol{U}_1 \otimes \boldsymbol{U}_2)\boldsymbol{x}_f = \left[\; (\tilde{\boldsymbol{U}}_1 \otimes \tilde{\boldsymbol{U}}_2) \;\middle|\; \star \;\right] \left[\begin{array}{c} \tilde{\boldsymbol{x}}_f \\ \hline \mathbf{0} \end{array}\right]$$

- We can reconstruct the product graph signal from subsampled observations since

$$N_1 N_2 \gg L_1 L_2 \text{ and } \mathrm{rank}(\tilde{\boldsymbol{U}}_1 \otimes \tilde{\boldsymbol{U}}_2) = \mathrm{rank}(\tilde{\boldsymbol{U}}_1)\mathrm{rank}(\tilde{\boldsymbol{U}}_2)$$

Reconstruction with subspace prior

With sparse sampling, we get $K_1 K_2$ equations in $L_1 L_2$ unknowns



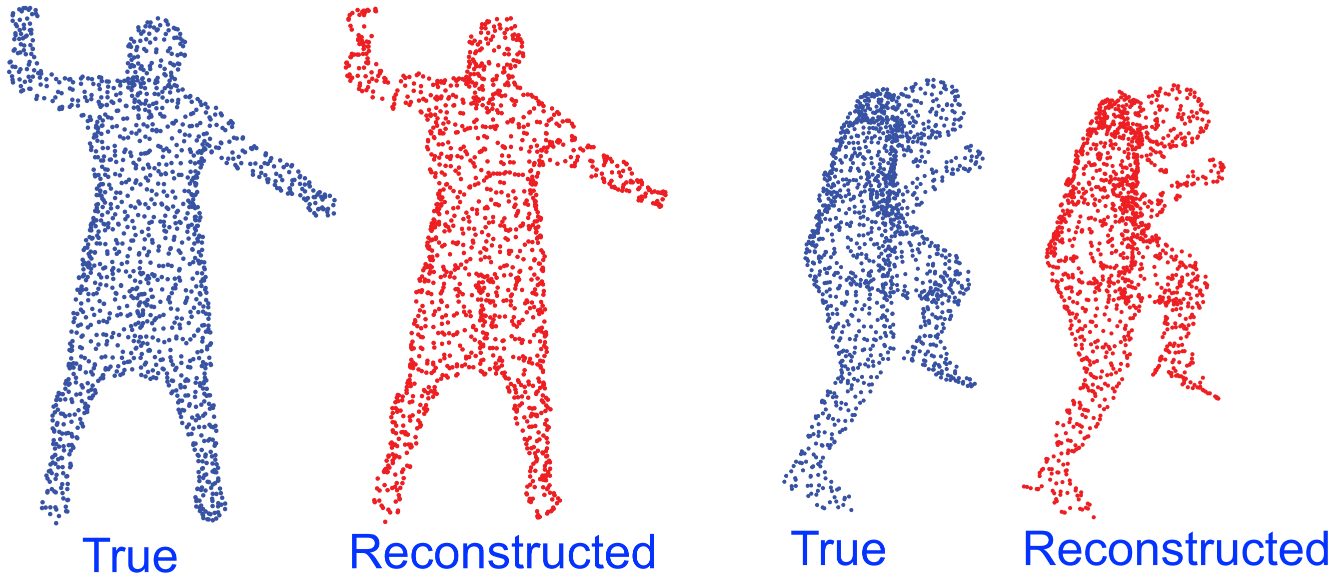
For unique reconstruction, we require $K_1 \geq L_1$ and $K_2 \geq L_2$

Least squares solution: $\hat{\tilde{\boldsymbol{x}}}_f = [(\boldsymbol{\Phi}_1 \boldsymbol{U}_1)^\dagger \otimes (\boldsymbol{\Phi}_2 \boldsymbol{U}_2)^\dagger]\boldsymbol{y}$

Design of $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ is crucial for the least-squares solution to be unique

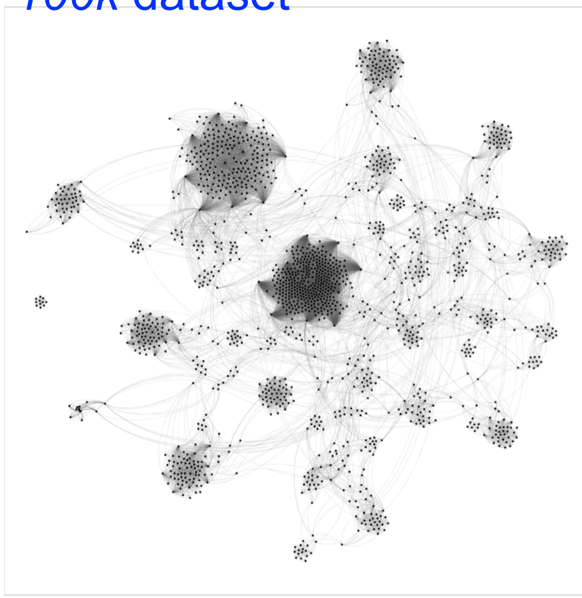
Numerical experiments – dynamic 3D point cloud

- 1502 markers, 573 frames. Product graph has 850000 vertices
- We sample 500 spatial points, and 70 time frames



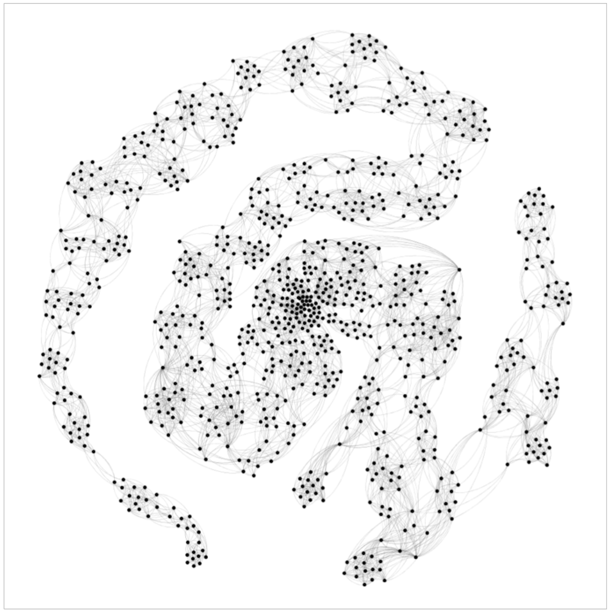
Numerical experiments – recommender system

MovieLens 100k dataset



Movie graph (1682 movies)

◇



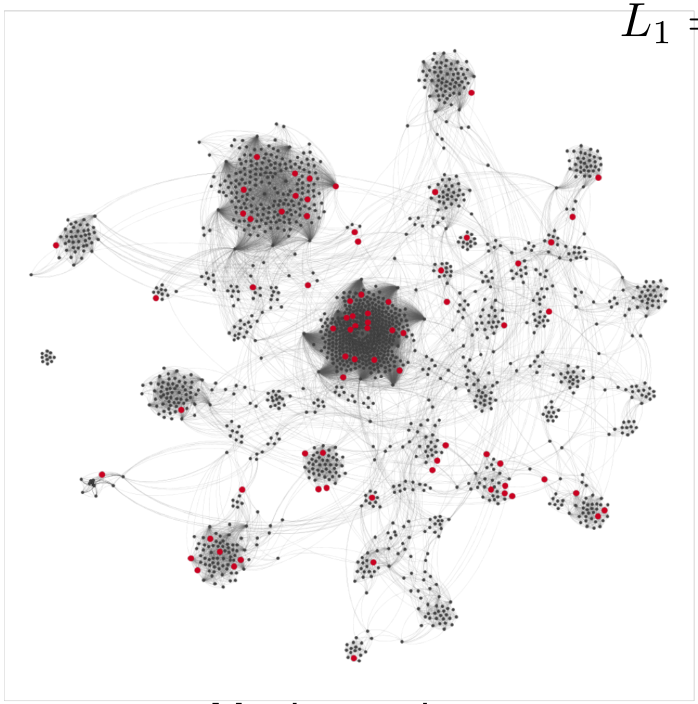
User graph (942 users)

- Product graph has about 1.6 million nodes
- Features used to build both the graphs (available with the dataset)
- Standard problem: Complete rating matrix using graph prior.
- *Active learning: Which users to probe for which movies?*

Numerical experiments – recommender system

MovieLens 100k dataset

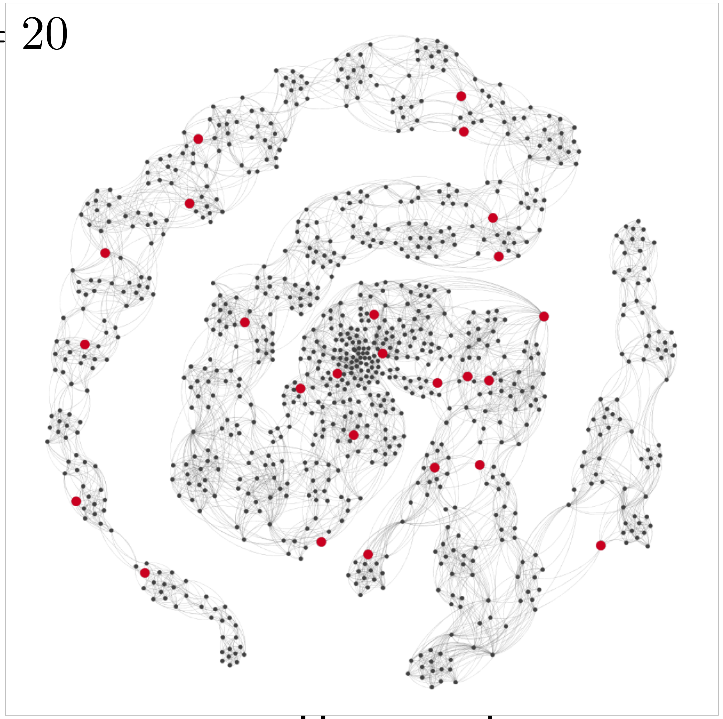
$L_1 = L_2 = 20$



Movie graph

75 movies sampled out of **1682 movies**

$\diamond$



User graph

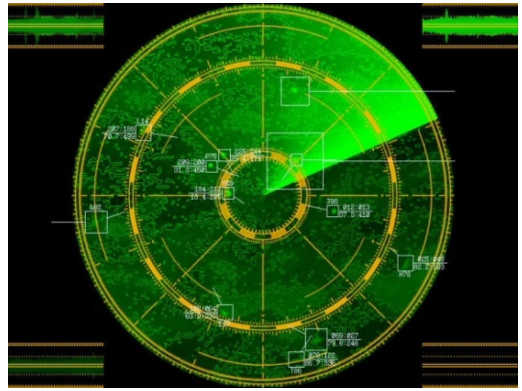
25 users sampled out of **942 users**

State-of-the-art matrix completion methods

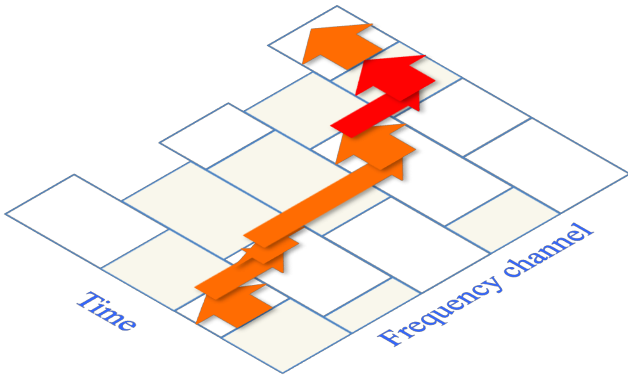
Method	Number of samples	RMSE
GMC [26]	80,000	0.996
GRALS [27]	80,000	0.945
sRGCNN [29]	80,000	0.929
GC-MC [30]	80,000	**0.905**
Our method	**1,875**	0.9347

Graph Covariance Sampling

- S.P. Chepuri and G. Leus. Graph Sampling for Covariance Estimation. *IEEE Journ. on Sel. Topics in Sig. Proc. and IEEE Trans. on Sig. and Info. Proc. over Networks, joint special issue on Graph Signal Processing, July 2017.*

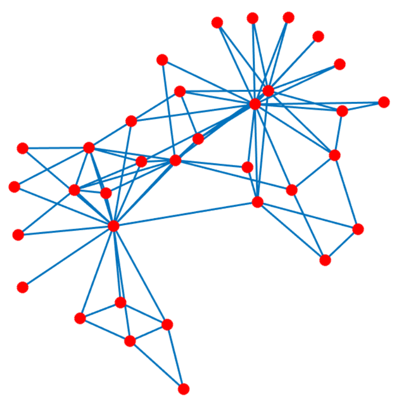


Radar
Doppler + angular spectra



Cognitive radio
frequency spectrum

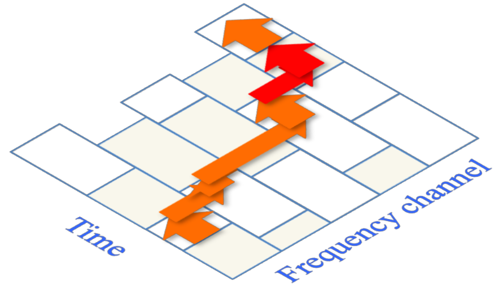
Radio astronomy
spatial spectrum



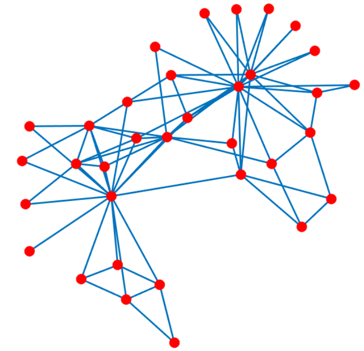
Graph-based inference
graph spectrum

Design sparse samplers taking into account the data structure

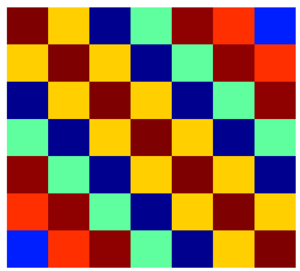
spatial spectrum



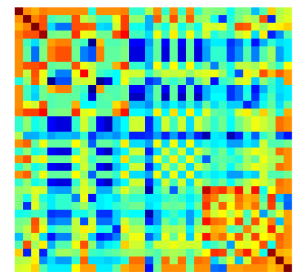
frequency spectrum



graph spectrum



structured (Toeplitz)



no apparent structure

uncompressed stationary signal $\boldsymbol{x}$ → compression $\boldsymbol{\Phi}$ → compressed signal $\boldsymbol{y}$

$K \ll N$

$\boldsymbol{R_x} = \mathrm{E}\left\{\boldsymbol{x}\boldsymbol{x}^H\right\}$ — $N \times N$

$\boldsymbol{\Phi}$ — $K \times N$

$\boldsymbol{R_y} = \mathrm{E}\left\{\boldsymbol{y}\boldsymbol{y}^H\right\}$ — $K \times K$

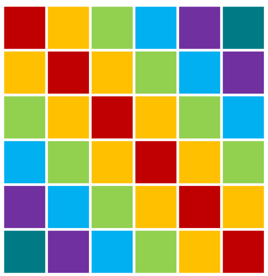
Given $\boldsymbol{R_y}$ or several realizations of $\boldsymbol{y}$ estimate $\boldsymbol{R_x}$

Compressive covariance sensing

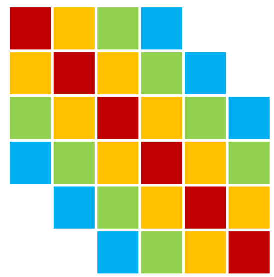
$$\boldsymbol{r}_{\boldsymbol{y}} = \mathrm{vec}(\boldsymbol{R}_{\boldsymbol{y}}) = \mathrm{vec}(\boldsymbol{\Phi}\boldsymbol{R}_{\boldsymbol{x}}\boldsymbol{\Phi}^T) = (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\mathrm{vec}(\boldsymbol{R}_{\boldsymbol{x}})$$

$K^2 \times 1$ $\qquad\qquad$ $N^2 \times 1$

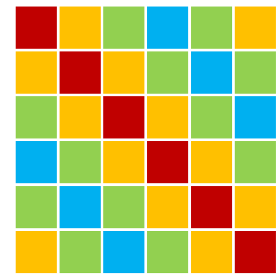
- Suppose the covariance matrix $\boldsymbol{R}_{\boldsymbol{x}}$ has a linear structure



Toeplitz



Banded



Circulant

$$\boldsymbol{R}_{\boldsymbol{x}}(\boldsymbol{\theta}) = \sum_{i=1}^{Q} \theta_i \boldsymbol{Q}_i \longrightarrow \boxed{\text{compression } \boldsymbol{\Phi}} \longrightarrow \boldsymbol{R}_{\boldsymbol{y}}(\boldsymbol{\theta}) = \sum_{i=1}^{Q} \theta_i \boldsymbol{\Phi}\boldsymbol{Q}_i\boldsymbol{\Phi}^T$$

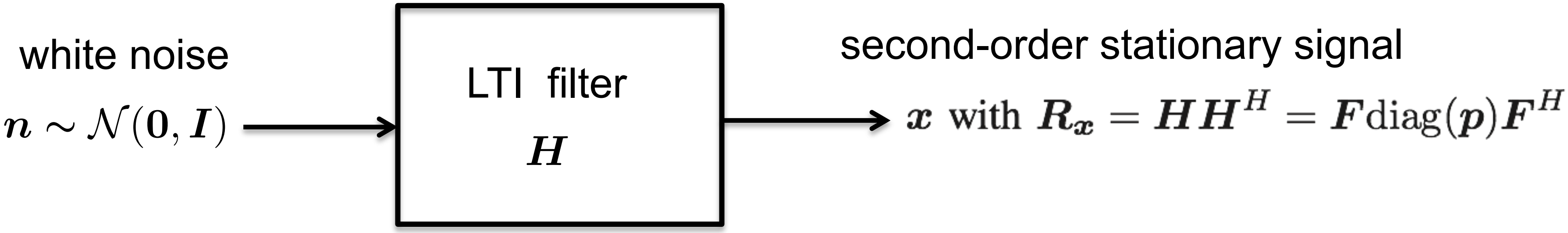
- If $K^2 > Q$: $\quad \boldsymbol{r}_{\boldsymbol{y}} = (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}\boldsymbol{\theta}$ $\xrightarrow{\text{least squares}}$ $\boldsymbol{\theta} = [(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}]^{\dagger}\boldsymbol{r}_{\boldsymbol{y}}$

Design of $\boldsymbol{\Phi}$ crucial for the solution to be unique

Second-order stationarity in time

Filtering white noise:

- Signal is the output of an LTI filter excited with white noise



- The covariance matrix is diagonalized by the Fourier matrix

$$\boldsymbol{R}_{\boldsymbol{x}} = \boldsymbol{F}\text{diag}(\boldsymbol{p})\boldsymbol{F}^H$$

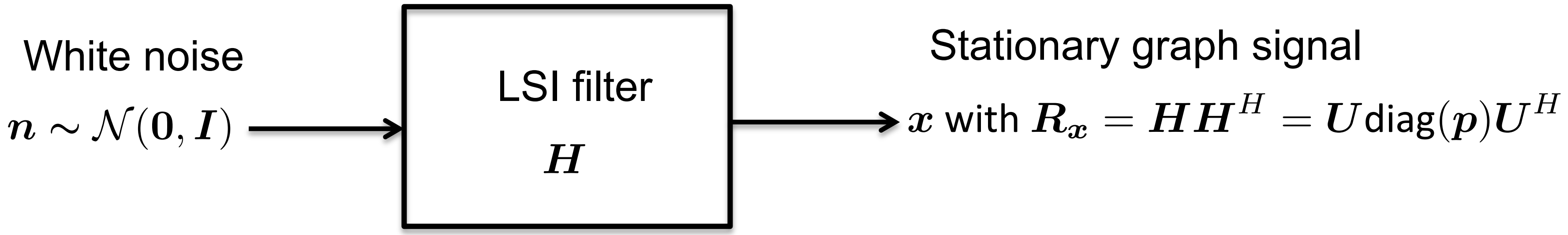
The process has power spectral density

$$\boldsymbol{p} = \text{diag}(\boldsymbol{F}^H \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{F})$$

Stationary graph signals

Filtering white noise:

- A random graph signal $\boldsymbol{x} \in \mathbb{R}^N$ is second-order stationary:



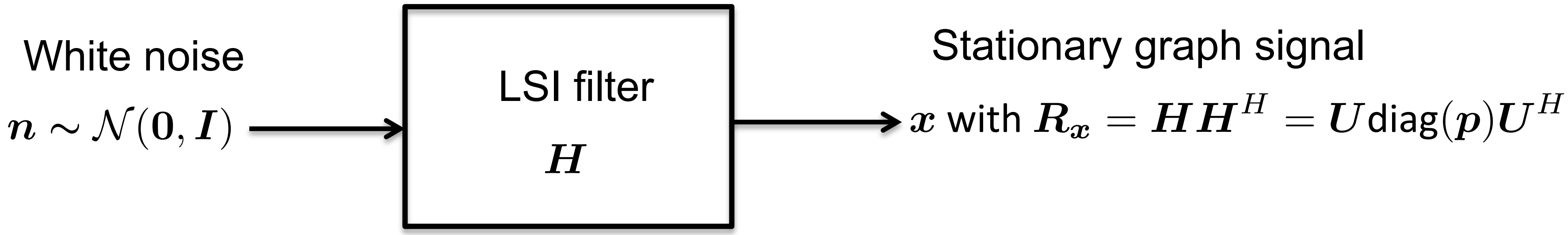
- The filter should be shift invariant $\boldsymbol{H}(\boldsymbol{S}\boldsymbol{x}) = \boldsymbol{S}(\boldsymbol{H}\boldsymbol{x}) \Leftrightarrow \boldsymbol{H} = \boldsymbol{U}\text{diag}(\boldsymbol{h}_f)\boldsymbol{U}^H$

- N. Perraudin and P. Vandergheynst, “Stationary signal processing on graphs,” *IEEE TSP, Jul. 2017.*
- A. Marques, S. Segarra, G. Leus, and A. Ribeiro, “Stationary graph processes and spectral estimation,” *IEEE TSP, Nov. 2017.*

Stationary graph signals

Filtering white noise:

- A random graph signal $\boldsymbol{x} \in \mathbb{R}^N$ is second-order stationary:



Simultaneous diagonalization:

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^H \qquad \boldsymbol{R_x} = \boldsymbol{U}\text{diag}(\boldsymbol{p})\boldsymbol{U}^H$$

- The process has power spectral density

$$\boldsymbol{p} = \text{diag}(\boldsymbol{U}^H \boldsymbol{R_x} \boldsymbol{U})$$

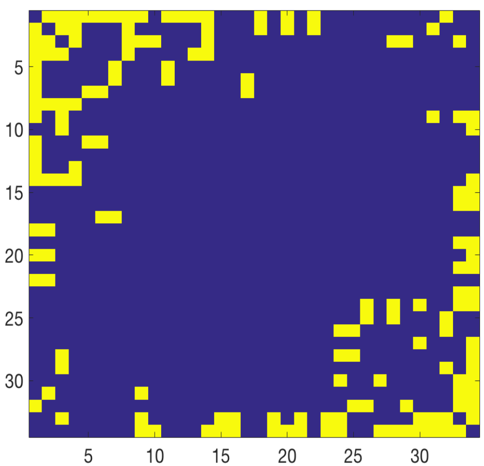
Remark (second-order stationarity in time):

$\boldsymbol{R_x}$ is a circulant matrix, which can be diagonalized by the DFT matrix

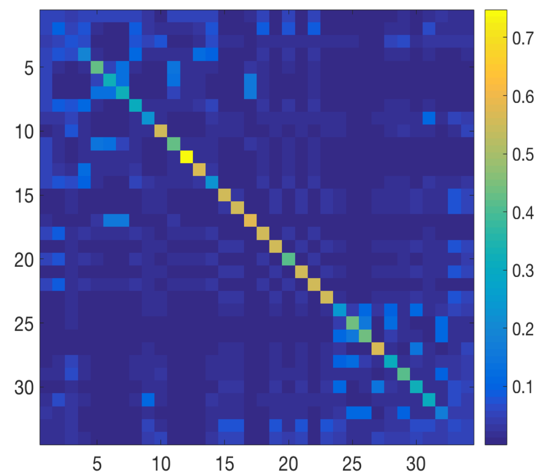
- N. Perraudin and P. Vandergheynst, “Stationary signal processing on graphs,” *IEEE TSP, Jul. 2017.*
- A. Marques, S. Segarra, G. Leus, and A. Ribeiro, “Stationary graph processes and spectral estimation,” *IEEE TSP, Nov. 2017.*

Stationary graph signals

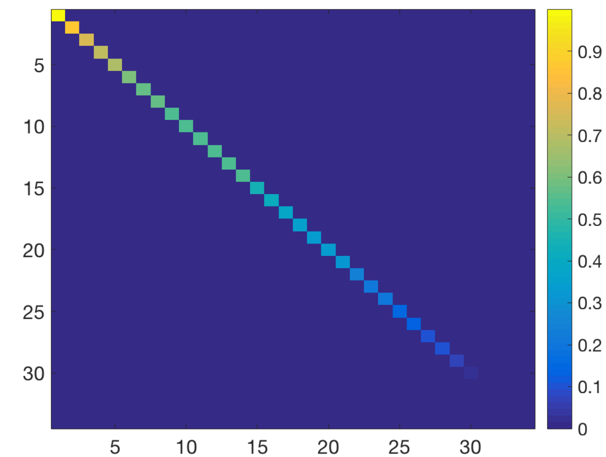
- Stationary process $\boldsymbol{x} \in \mathbb{R}^N$ on a graph shift $\boldsymbol{S}$



Adjacency matrix
(Karate club network)



Covariance matrix



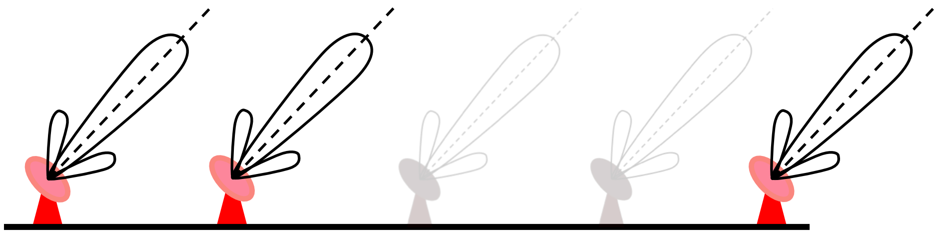
Spectral domain
$\boldsymbol{U}^H \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{U}$

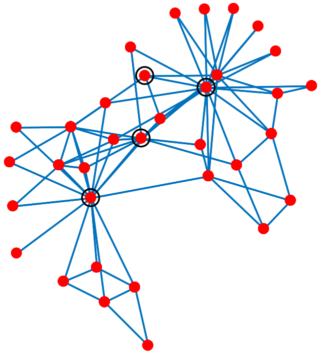
Power spectrum estimation is crucial for statistical inference
smoothing, prediction, deconvolution

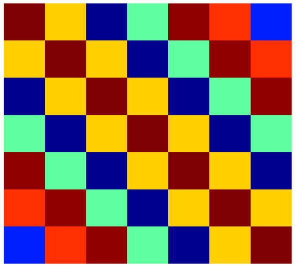
Power spectrum estimation

Estimate the power spectrum

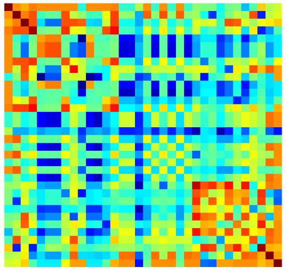
a. by observing a reduced subset of nodes/sensors (i.e., subsample)
b. without using spectral priors (e.g., sparsity, bandlimited with known support)







structured (Toeplitz)



no apparent structure

Non-parametric method

- The covariance again admits a linear structure

$$\boldsymbol{R}_{\boldsymbol{x}} = \boldsymbol{U}\,\mathrm{diag}(\boldsymbol{p})\boldsymbol{U}^H \qquad \boldsymbol{R}_{\boldsymbol{x}} = \sum_{i=1}^{N} p_i \boldsymbol{u}_i \boldsymbol{u}_i^H = \sum_{i=1}^{N} p_i \boldsymbol{Q}_i$$

- After compression:

$$\boldsymbol{R}_{\boldsymbol{x}} = \sum_{i=1}^{N} p_i \boldsymbol{Q}_i \longrightarrow \boxed{\text{compression } \boldsymbol{\Phi}} \longrightarrow \boldsymbol{R}_{\boldsymbol{y}} = \sum_{k=i}^{N} p_i \boldsymbol{\Phi}\boldsymbol{Q}_i\boldsymbol{\Phi}^T$$

- We have K^2 equations in N unknowns

vec$(\boldsymbol{A}\mathrm{diag}(\boldsymbol{d})\boldsymbol{B}) = (\boldsymbol{B}^T \circ \boldsymbol{A})\boldsymbol{d}$

$$\begin{aligned}\boldsymbol{r}_y = \mathrm{vec}(\boldsymbol{R}_y) &= (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\mathrm{vec}(\boldsymbol{R}_x) \\ &= (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})(\boldsymbol{U} \circ \boldsymbol{U})\boldsymbol{p} \\ &= (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}_{\mathrm{NP}}\boldsymbol{p}\end{aligned}$$

- If the matrix $(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}_{\mathrm{NP}}$ has full column rank, which requires $K^2 \geq N$

$$\hat{\boldsymbol{p}} = [(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}_{\mathrm{NP}}]^{\dagger}\boldsymbol{r}_{\boldsymbol{y}}$$

Parametric method (moving average)

- Graph signal is a moving average graph process of order $L-1$

$$\boldsymbol{x} = \boldsymbol{H}(\boldsymbol{h})\boldsymbol{n} = \sum_{l=0}^{L-1} h_l \boldsymbol{S}^l \boldsymbol{n} = \boldsymbol{U}\left(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l\right)\boldsymbol{U}^H \boldsymbol{n}$$

with covariance matrix

$$\boldsymbol{R}_{\boldsymbol{x}} = \boldsymbol{H}(\boldsymbol{h})\boldsymbol{H}^H(\boldsymbol{h}) = \boldsymbol{U}\left(\sum_{l=0}^{L-1} h_l \boldsymbol{\Lambda}^l\right)^2 \boldsymbol{U}^H$$

- We can express $\boldsymbol{R}_{\boldsymbol{x}}$ as a *matrix polynomial* of the *graph-shift* operator

$$\boldsymbol{R}_{\boldsymbol{x}}(\boldsymbol{b}) = \sum_{k=0}^{Q-1} b_k \boldsymbol{S}^k$$

Covariance matching (*basis expansion*): $Q = \underbrace{\min\{2L-1, N\}}$

degree of minimal polynomial of the *graph-shift*

For, $L=2$, $\boldsymbol{R}_{\boldsymbol{x}} = h_0^2\mathbf{I} + 2h_0h_1\boldsymbol{S} + h_1^2\boldsymbol{S}^2$

Parametric method (moving average)

- For a moving average graph process on an undirected graph we have

$$\boldsymbol{R_x} = \textstyle\sum_{k=0}^{Q-1} b_k \boldsymbol{S}^k \qquad\qquad Q = \min\{2L-1, N\}$$

- After compression:

$$\boldsymbol{R_x} = \textstyle\sum_{k=0}^{Q-1} b_k \boldsymbol{S}^k \longrightarrow \boxed{\text{compression } \boldsymbol{\Phi}} \longrightarrow \boldsymbol{R_y} = \textstyle\sum_{k=0}^{Q-1} b_k \boldsymbol{\Phi} \boldsymbol{S}^k \boldsymbol{\Phi}^T$$

- We have K^2 equations in Q unknowns

$$\begin{aligned} \boldsymbol{r}_y = \mathrm{vec}(\boldsymbol{R}_y) &= (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\mathrm{vec}(\boldsymbol{R}_x) \\ &= (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})[\mathrm{vec}(\boldsymbol{S}^0), \ldots, \mathrm{vec}(\boldsymbol{S}^{Q-1})]\boldsymbol{b} \\ &= (\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}_{\mathrm{MA}}\boldsymbol{b} \end{aligned}$$

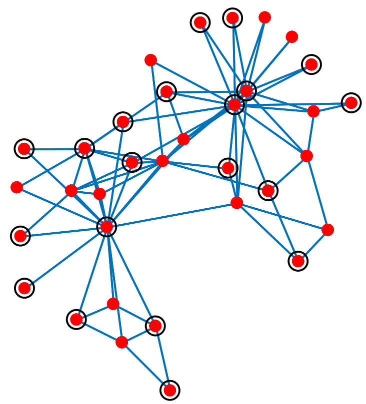
- If the matrix $(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}_{\mathrm{MA}}$ has full column rank, which requires $K^2 \geq Q$

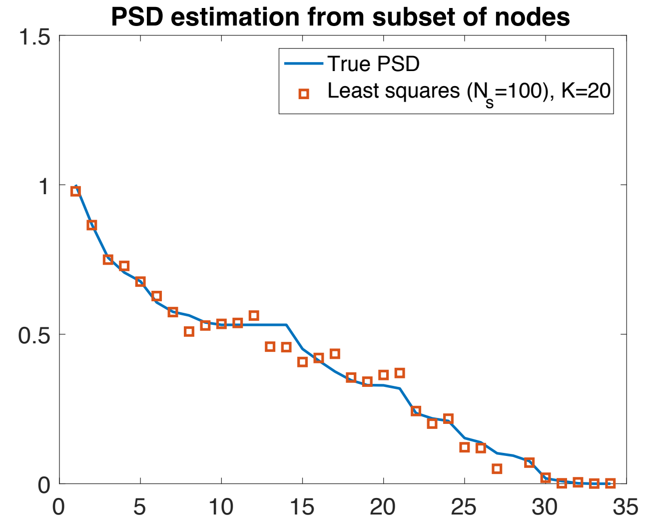
$$\hat{\boldsymbol{b}} = [(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}_{\mathrm{MA}}]^{\dagger}\boldsymbol{r_y}$$

Illustration – Karate club network

Non-parametric approach

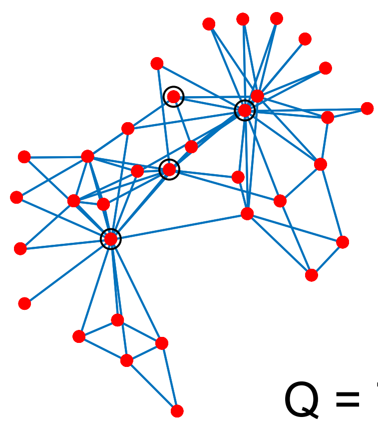
Sample 20 out of 34 nodes



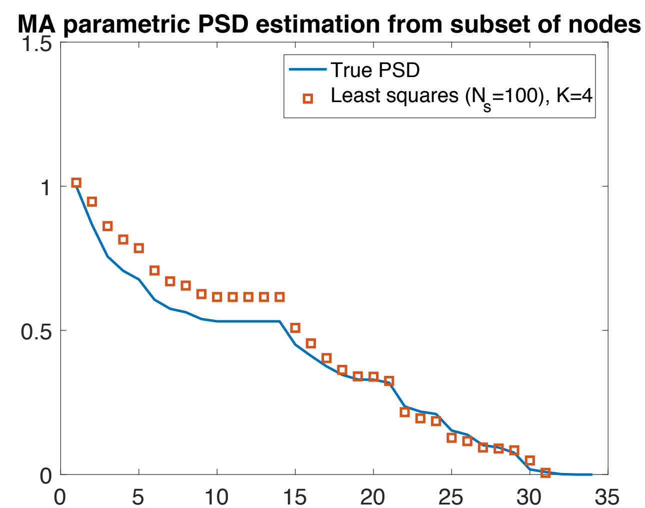


Parametric approach

Sample 4 out of 34 nodes

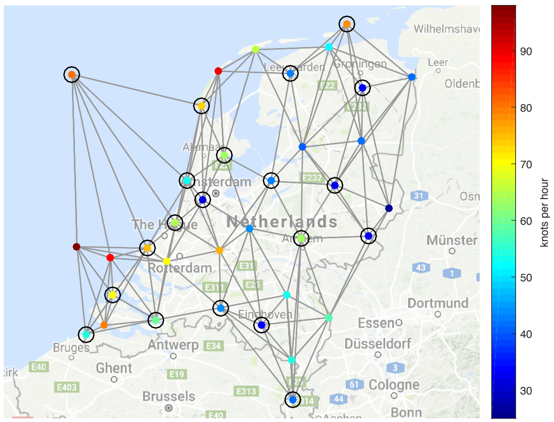


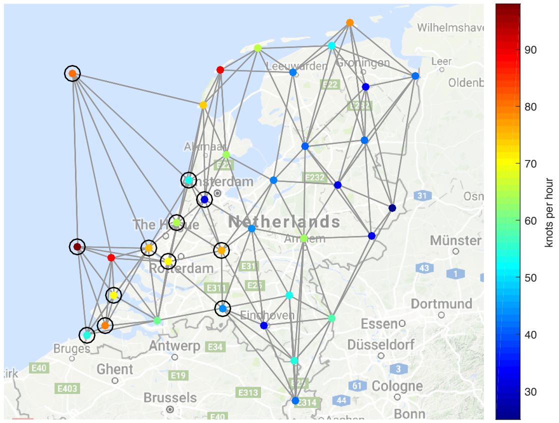
Q = 7

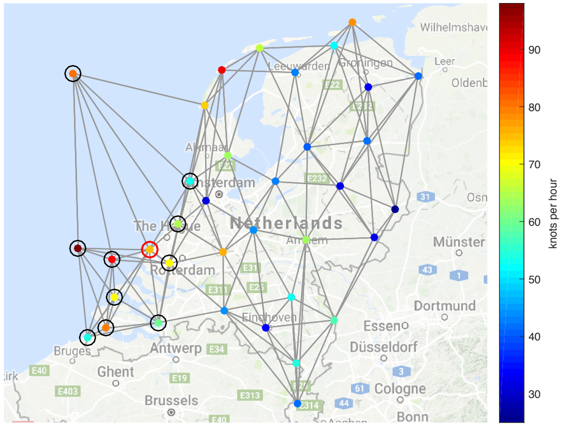


Wind speed dataset

Non-parametric approach	Moving average approach	Autoregressive approach



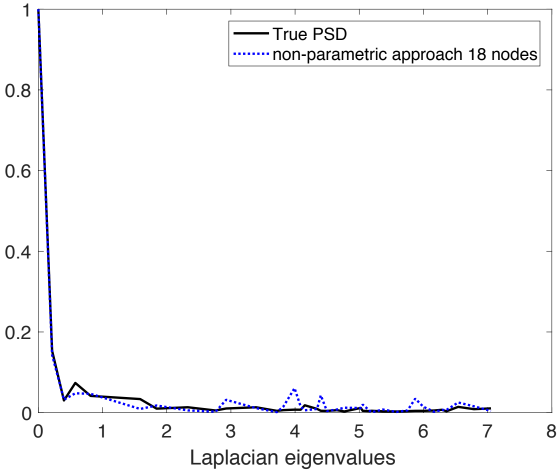


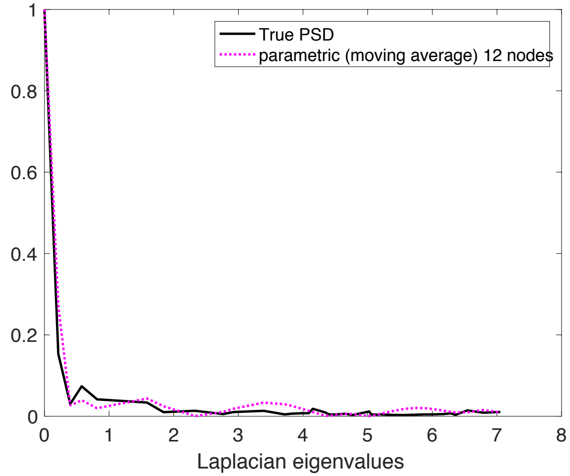


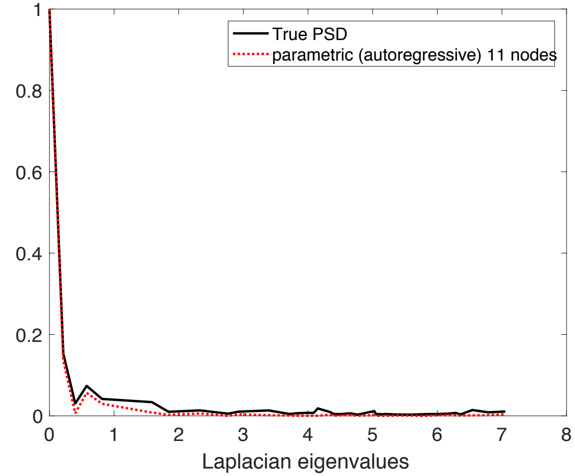
Sample 18 out of 36 stations

12 out of 36 stations

11 out of 36 stations





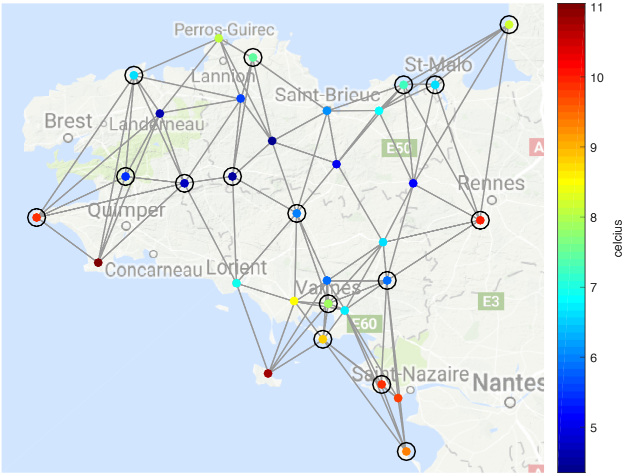


L=6 => Q = 11

P = 1

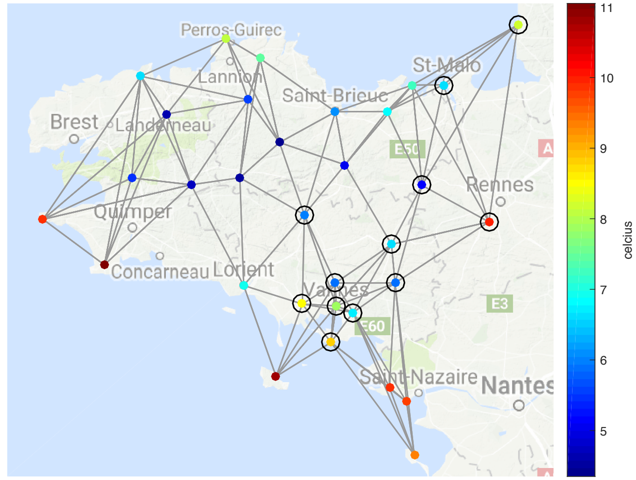
Temperature dataset

Non-parametric approach



Sample 16 out of 32 nodes

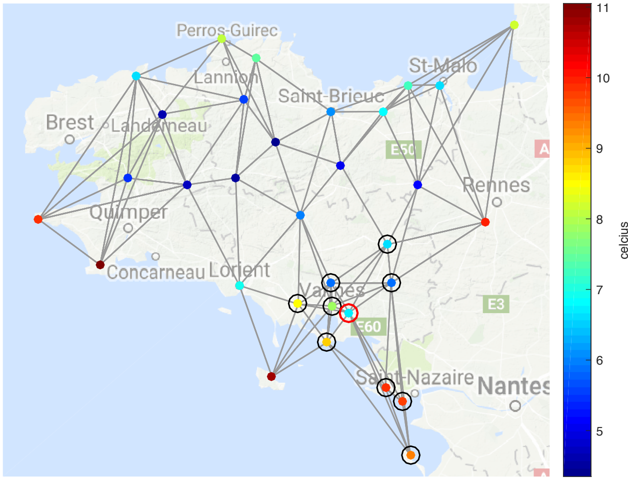
Moving average approach



12 out of 32 nodes

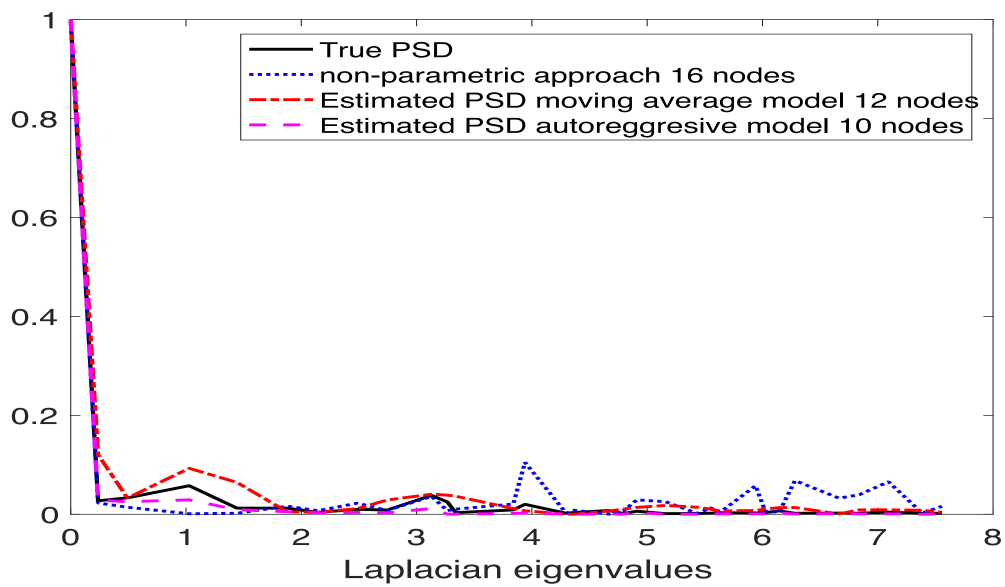
Q = 11

Autoregressive approach



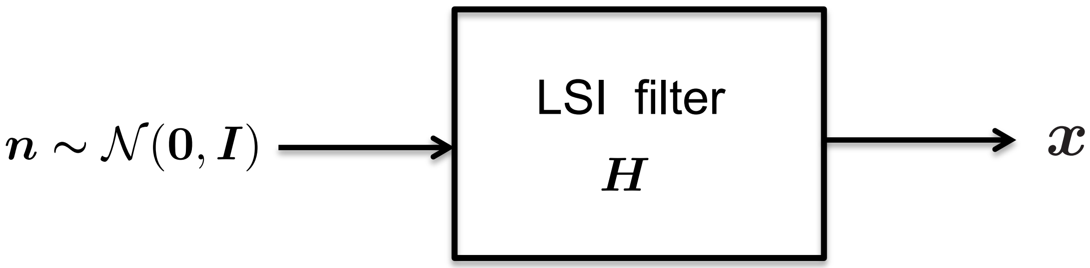
10 out of 32 nodes

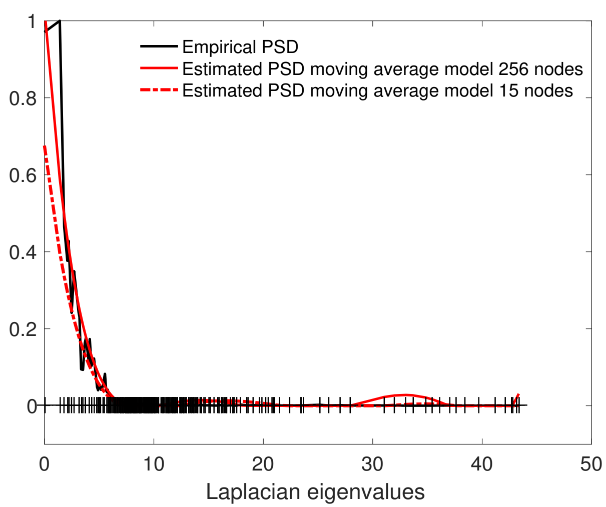
P = 1

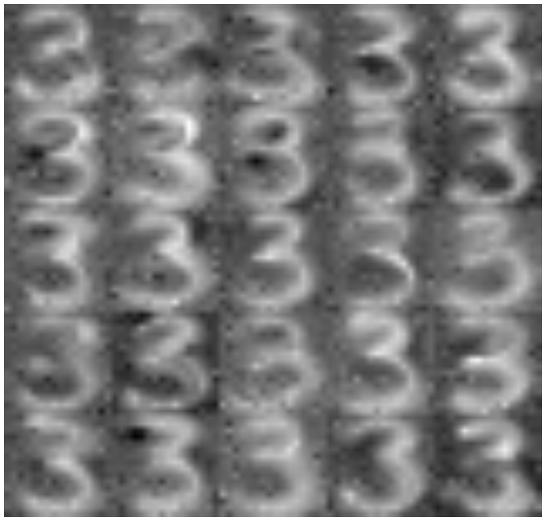


Generate digits

- Nearest neighbor graph built using digit 3 (16 x 16 pixels) from the USPS dataset.
- Graph signal (pixel intensity) is of length 256

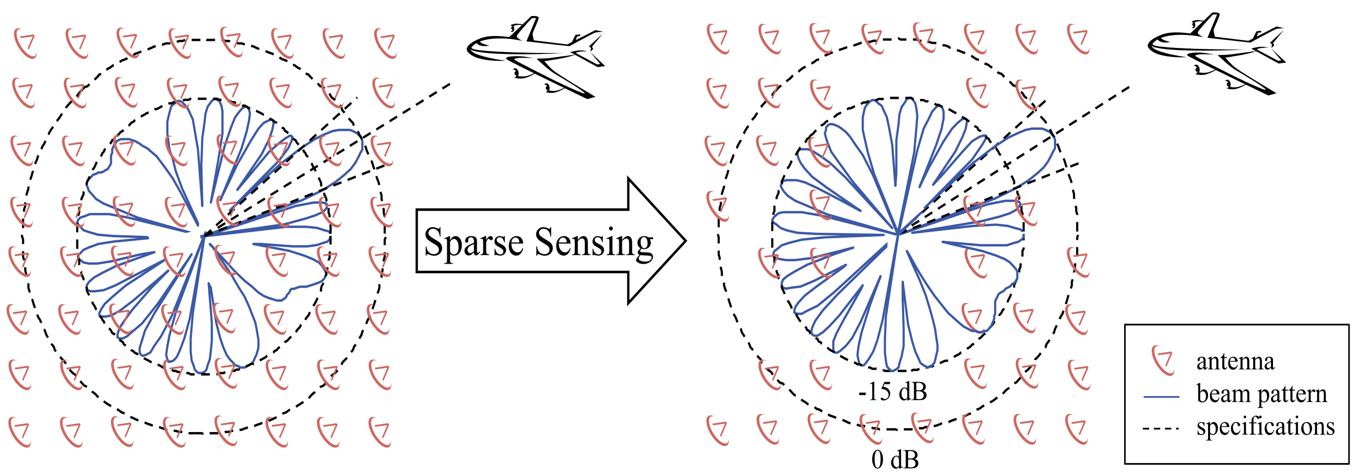






25 realizations

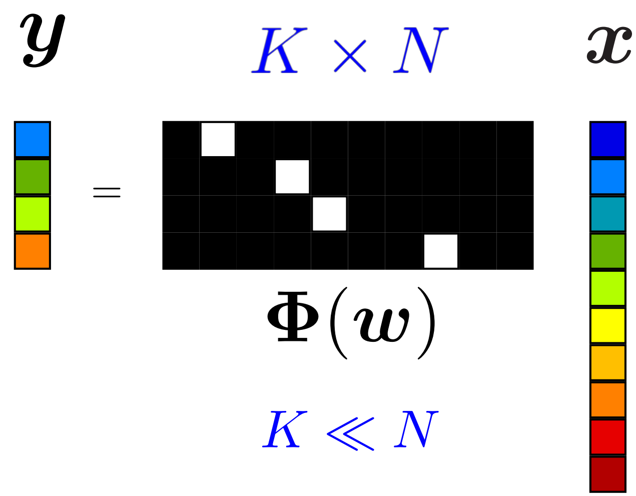
Sparse Sampler Design



S.P. Chepuri and G. Leus. Sparse Sensing for Statistical Inference. *Foundations and Trends in Signal Processing, Vol. 9: No. 3–4, pp 233-368, Dec. 2016.*

Sparse sensing models

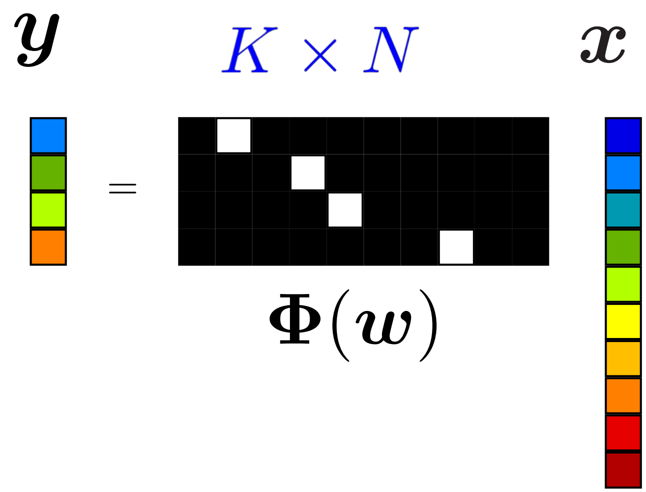
Sparsely sensed signals



Least squares solution: $[\boldsymbol{\Phi} \boldsymbol{U}_{\text{BL}}]^{\dagger} \boldsymbol{y}$

Sparse sensing models

Sparsely sensed statistics

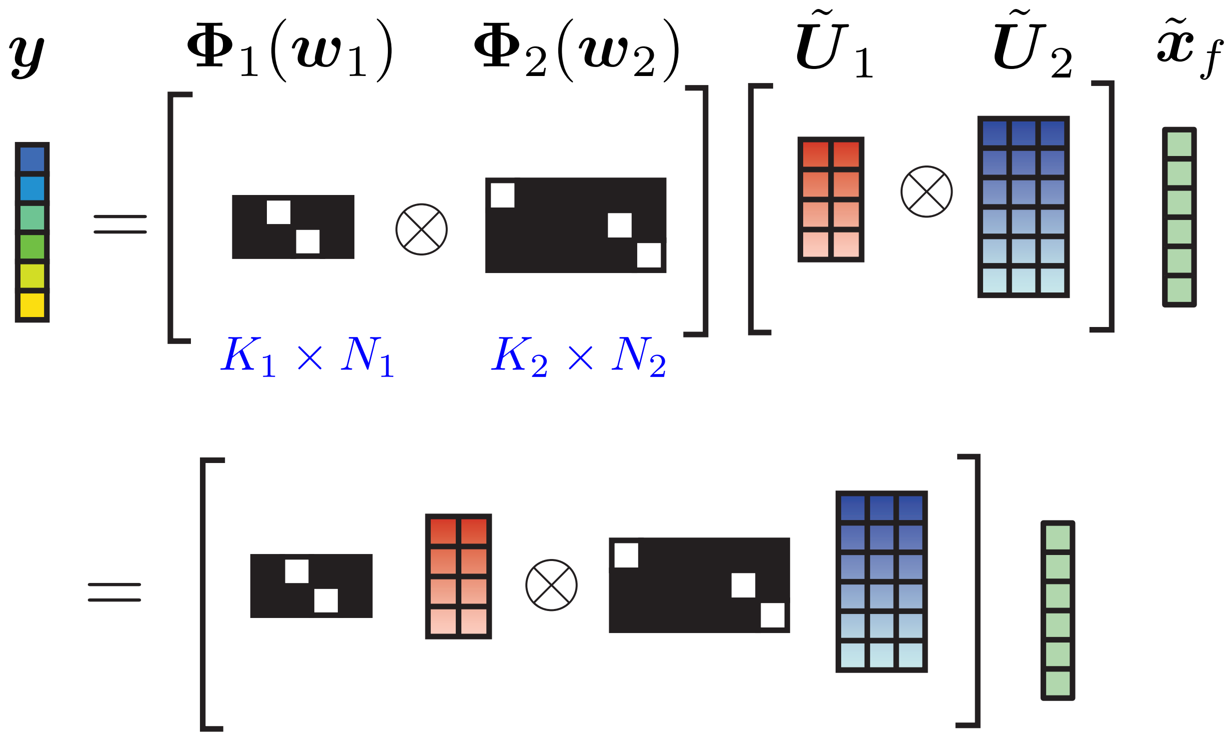


$$\boldsymbol{R}_{\boldsymbol{y}} = \mathrm{E}\left\{\boldsymbol{y}\boldsymbol{y}^H\right\} \qquad \boldsymbol{R}_{\boldsymbol{x}} = \mathrm{E}\left\{\boldsymbol{x}\boldsymbol{x}^H\right\}$$

Least squares solution: $[(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}]^{\dagger}\boldsymbol{r}_{\boldsymbol{y}}$

Sparse sensing models

Sparsely sensed multidomain signals



Least squares solution: $[(\boldsymbol{\Phi}_1\boldsymbol{U}_1)^\dagger \otimes (\boldsymbol{\Phi}_2\boldsymbol{U}_2)^\dagger]\boldsymbol{y}$

What is sparse sampling?

$$\boldsymbol{\Phi}(\boldsymbol{w}) \in \{0,1\}^{K \times N}$$

$\boldsymbol{y}$

$\boldsymbol{R_y} = \mathrm{E}\left\{\boldsymbol{y}\boldsymbol{y}^H\right\}$

$\boldsymbol{x}$

$\boldsymbol{R_x} = \mathrm{E}\left\{\boldsymbol{x}\boldsymbol{x}^H\right\}$

- Sampling matrix is determined by the sampling vector/set

$\boldsymbol{w} = [w_1, w_2, \ldots, w_N]^T \in \{0,1\}^N$ or $\mathcal{S} = \{n | w_n = 1, n = 1, 2, \ldots, N\}$

$w_m = (0)1$ sample or vertex is (not) selected

- Sparse sampling structure
  - only one nonzero entry per row
  - many zero columns

Design problem

Select the "best" subset of vertices out of the candidate vertices that guarantee a certain desired reconstruction accuracy.

$$\underset{\boldsymbol{w}}{\text{optimize}} \; f(\boldsymbol{w}) \quad \text{s.to} \quad \text{card}(\boldsymbol{w}) = K, \quad \boldsymbol{w} \in \{0,1\}^N$$

or

$$\underset{\mathcal{S} \subset \mathcal{N}}{\text{optimize}} \; f(\mathcal{S}) \quad \text{s.to} \quad |\mathcal{S}| = K$$

$f(\boldsymbol{w})$ reconstruction performance metric

K sample size

$\boldsymbol{w} = [w_1, w_2, \ldots, w_N]^T \in \{0,1\}^N$

$\mathcal{S} = \{n | w_n = 1, n = 1, 2, \ldots, N\}$

$w_m = (0)1$ sample or vertex is (not) selected

Design problem

Select the "best" subset of vertices out of the candidate vertices that guarantee a certain desired reconstruction accuracy.

$$\underset{\boldsymbol{w}}{\text{optimize}} \ f(\boldsymbol{w})$$
$$\text{s.to} \quad \text{card}(\boldsymbol{w}) = K$$
$$\boldsymbol{w} \in \{0, 1\}^N$$

or

$$\underset{\mathcal{S} \subset \mathcal{N}}{\text{optimize}} \ f(\mathcal{S})$$
$$\text{s.to} \quad |\mathcal{S}| = K$$

Nonconvex Boolean problem

Solutions to the combinatorial problem

Exact solutions:

- ➢ Exhaustive search over
  - ❑ $\binom{M}{K}$ possible candidates
- ➢ Branch-and-bound methods

  [Lawler-Wood-1966], [Nguyen-Miller-1992]
  - ❑ long runtimes even for a modest sized problem

- E. L. Lawler and D. E. Wood, “Branch-and-bound methods: A survey,” *Oper. Res.*, vol. 14, pp. 699–719, 1966.
- N. Nguyen and A. Miller, “A review of some exchange algorithms for constructing discrete D-optimal designs,” *Comput. Statist. Data Anal.*, vol. 14, pp. 489–498, 1992

Solutions to the combinatorial problem

Suboptimal solutions:

- **Convex** optimization (polynomial time)

  [Joshi-Boyd-2009], [Chepuri-Leus-2015]

  - convex relaxation for $\{0, 1\}, f(\boldsymbol{w})$
  - thresholding, randomization to get back a Boolean solution
  - Semidefinite program (typically)

- S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Trans. Signal Process*., vol. 57, no. 2, pp. 451–462, Feb. 2009
- S.P. Chepuri and G. Leus. "Sparsity-Promoting Sensor Selection for Non-linear Measurement Models," *IEEE Trans. on Signal Processing*, vol. 63, no. 3, pp. 684-698, Feb. 2015.

Solutions to the combinatorial problem

Suboptimal solutions:

- **Submodular** optimization (linear search time)

  [Krause-Singh-Guestrin-2008], [Ranieri-Chebira-Vetterli-2014]

  - Submodularity of $f(\mathcal{S})$
  - greedy search
  - solution is near optimal

- A. Krause, A. Singh, and C. Guestrin, “Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies,” *J. Machine Learn. Res.*, vol. 9, pp. 235–284, Feb. 2008.
- J. Ranieri, A. Chebira, and M. Vetterli, “Near-optimal sensor placement for linear inverse problems,” *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1135–1146, Mar. 2014

Submodular optimization

Requires $f(\cdot)$ to be submodular function of its arguments

- Define the sampling set:

$$\mathcal{X} := \mathcal{S} = \{n|w_n = 1, n = 1, 2, \ldots, N\}$$

or

$$\mathcal{X} := \mathcal{N} \setminus \mathcal{S} = \{n|w_n = 0, n = 1, 2, \ldots, N\}$$

- Set function $f(\mathcal{X})$ is submodular, if $\forall \mathcal{X} \subseteq \mathcal{Y} \subset N$, $s \in \mathcal{N} \setminus \mathcal{Y}$

$$f(\mathcal{X} \cup \{s\}) - f(\mathcal{X}) \geq f(\mathcal{Y} \cup \{s\}) - f(\mathcal{Y})$$

- Set function $f(\mathcal{X})$ is monotone non-decreasing, if

$$f(\mathcal{X} \cup \{s\}) \geq f(\mathcal{X})$$

Design problem

Select the "best" subset of vertices out of the candidate vertices that guarantee a certain desired reconstruction accuracy.

$$\underset{\mathcal{X}}{\text{maximize}} \; f(\mathcal{X})$$
$$\text{s.to} \quad |\mathcal{X}| = L$$

$$L = K \text{ or } L = N - K$$

Nonconvex Boolean problem

Submodular optimization

If $f(\cdot)$ is **submodular** and **monotonic**

Linear sweep time

Algorithm 1 Greedy algorithm

Require $\mathcal{X} = \emptyset, L$.
for $k = 1$ to L
$$s^* = \arg\max_{s \notin \mathcal{X}} f(\mathcal{X} \cup \{s\})$$
$$\mathcal{X} \leftarrow \mathcal{X} \cup \{s^*\}$$
end
Return $\mathcal{X}$

$$L = K \text{ or } L = N - K$$

Then, greedy algorithm is near-optimal

$$f(\mathcal{X}) \geq \underbrace{(1 - 1/e)}_{63\%} \max_{|\mathcal{Y}|=L} f(\mathcal{Y})$$

[Nemhauser-Wolsey-Fisher-1978]

- G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions— I,” Mathematical Programming, vol. 14, no. 1, pp. 265–294, 1978.

Design problem

Select the “best” subset of vertices out of the candidate vertices that guarantee a certain desired reconstruction accuracy.

$$\underset{\mathcal{X}}{\text{maximize}} \; f(\mathcal{X})$$
$$\text{s.to} \quad |\mathcal{X}| = L$$

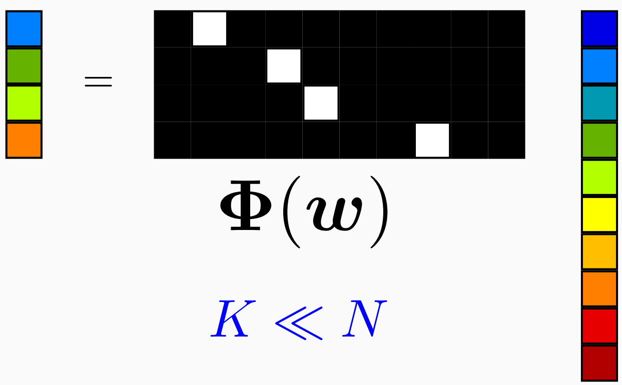
$$L = K \text{ or } L = N - K$$

What is a suitable submodular function $f(\mathcal{X})$ for sparse sampling?

Sparse sensing models

Sparsely sensed signals

$\boldsymbol{y}$ $K \times N$ $\boldsymbol{x}$



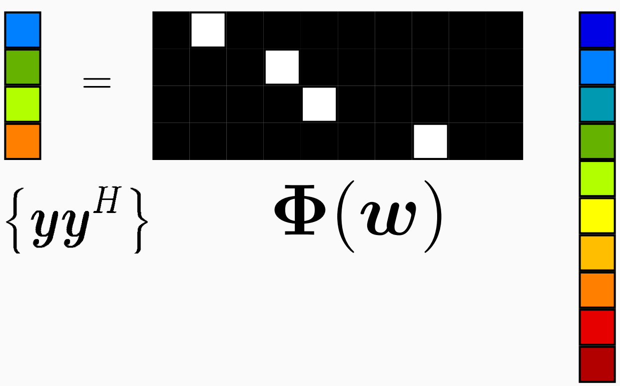
$\boldsymbol{\Phi}(\boldsymbol{w})$

$K \ll N$

Least squares solution: $[\boldsymbol{\Phi} \boldsymbol{U}_{\text{BL}}]^{\dagger} \boldsymbol{y}$

Sparsely sensed statistics

$\boldsymbol{y}$ $K \times N$ $\boldsymbol{x}$



$\boldsymbol{R}_{\boldsymbol{y}} = \mathrm{E}\left\{\boldsymbol{y}\boldsymbol{y}^H\right\}$ $\boldsymbol{\Phi}(\boldsymbol{w})$

$\boldsymbol{R}_{\boldsymbol{x}} = \mathrm{E}\left\{\boldsymbol{x}\boldsymbol{x}^H\right\}$

Least squares solution: $[(\boldsymbol{\Phi} \otimes \boldsymbol{\Phi})\boldsymbol{\Psi}]^{\dagger} \boldsymbol{r}_{\boldsymbol{y}}$

How do design the subsampler?

- Quality of the least squares solution

$$[\boldsymbol{\Phi}\boldsymbol{U}_{\text{BL}}]^{\dagger}\boldsymbol{y} \quad \text{or} \quad [(\boldsymbol{\Phi}\otimes\boldsymbol{\Phi})\boldsymbol{\Psi}]^{\dagger}\boldsymbol{r}_b$$

depends on the spectrum (eigenvalues) of

$$\boldsymbol{T}(\boldsymbol{w}) = [\boldsymbol{\Phi}\boldsymbol{U}_{\text{BL}}]^H[\boldsymbol{\Phi}\boldsymbol{U}_{\text{BL}}] = \boldsymbol{U}_{\text{BL}}^H\text{diag}(\boldsymbol{w})\boldsymbol{U}_{\text{BL}}$$

or

$$\boldsymbol{T}(\boldsymbol{w}) = [(\boldsymbol{\Phi}\otimes\boldsymbol{\Phi})\boldsymbol{\Psi}]^H[(\boldsymbol{\Phi}\otimes\boldsymbol{\Phi})\boldsymbol{\Psi}] = \boldsymbol{\Psi}^H[\text{diag}(\boldsymbol{w})\otimes\text{diag}(\boldsymbol{w})]\boldsymbol{\Psi}$$

- We try to balance the spectrum:

$$\arg\max_{\boldsymbol{w}\in\{0,1\}^N} \quad \log\det\{\boldsymbol{T}(\boldsymbol{w})\} \quad \text{s.to} \quad \|\boldsymbol{w}\|_0 = K$$

Scalar measure of the error covariance matrix

How to design the subsampler?

$$\arg \max_{\boldsymbol{w}\in\{0,1\}^N} \log\det\{\boldsymbol{T}(\boldsymbol{w})\} \quad \text{s.to} \quad \|\boldsymbol{w}\|_0 = K$$

- Using set notation

$$\mathcal{X} = \{m | w_m = 1, m = 1, 2, \ldots, M\}$$

- Set function

$$f(\mathcal{X}) = \log\det\left\{\sum_{i\in\mathcal{X}} \boldsymbol{u}_{\mathrm{BL},i}\boldsymbol{u}_{\mathrm{BL},i}^H\right\} \quad \text{or} \quad f(\mathcal{X}) = \log\det\left\{\sum_{(i,j)\in\mathcal{X}\times\mathcal{X}} \boldsymbol{\psi}_{i,j}\boldsymbol{\psi}_{i,j}^H\right\}$$

$$\boldsymbol{U}_{\mathrm{BL}} = [\boldsymbol{u}_{\mathrm{BL},1}, \cdots, \boldsymbol{u}_{\mathrm{BL},N}]^T \qquad \boldsymbol{\Psi} = [\boldsymbol{\psi}_{1,1}, \boldsymbol{\psi}_{1,2}, \cdots, \boldsymbol{\psi}_{N,N}]^H$$

Set function is submodular and monotone non-decreasing

How to design the subsampler?

$$\arg \max_{\boldsymbol{w} \in \{0,1\}^N} \log\det\{T(\boldsymbol{w})\} \quad \text{s.to} \quad \|\boldsymbol{w}\|_0 = K$$

- This combinatorial optimization can be near optimally solved using a low-complexity greedy algorithm

$$f(\mathcal{X}) \geq \underbrace{(1 - 1/e)}_{63\%} \max_{|\mathcal{Y}|=K} f(\mathcal{Y})$$

[Nemhauser-Wolsey-Fisher-1978]

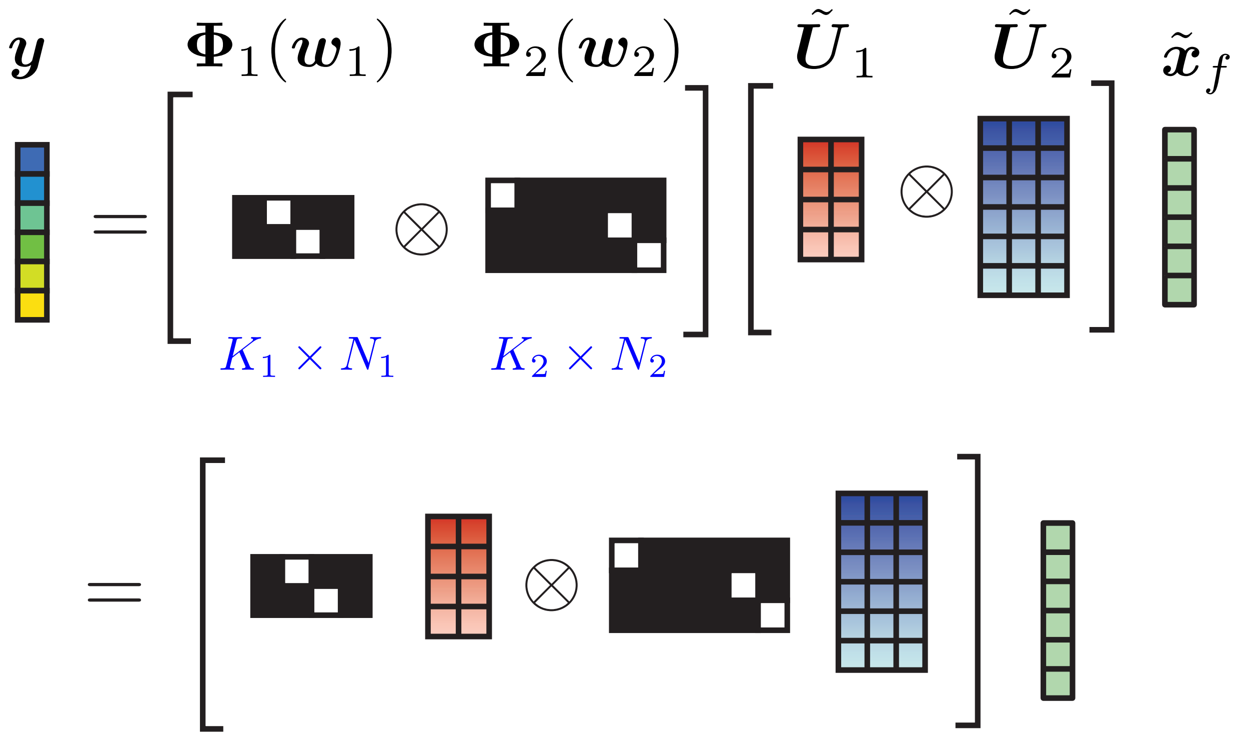
1. **Require** $\mathcal{X} = \emptyset, K$.
2. **for** $k = 1$ to K
3. $\quad s^* = \arg\max_{s \notin \mathcal{X}} f(\mathcal{X} \cup \{s\})$
4. $\quad \mathcal{X} \leftarrow \mathcal{X} \cup \{s^*\}$
5. **end**
6. **Return** $\mathcal{X}$

✓ Leverages submodularity
✓ Linear sweep time

- G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions— I,” Mathematical Programming, vol. 14, no. 1, pp. 265–294, 1978.

Sparse sensing models

Sparsely sensed multidomain signals



Least squares solution: $[(\boldsymbol{\Phi}_1\boldsymbol{U}_1)^\dagger \otimes (\boldsymbol{\Phi}_2\boldsymbol{U}_2)^\dagger]\boldsymbol{y}$

Design of $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ is crucial for the least-squares solution to be unique

How to design the subsampler?

- Quality of the least squares solution

$$[(\boldsymbol{\Phi}_1\boldsymbol{U}_1)^\dagger \otimes (\boldsymbol{\Phi}_2\boldsymbol{U}_2)^\dagger]\boldsymbol{y}$$

depends on the error covariance matrix

$$\begin{aligned}\boldsymbol{T}(\mathcal{X}) &= \left(\boldsymbol{\Phi}_1\tilde{\boldsymbol{U}}_1 \otimes \boldsymbol{\Phi}_2\tilde{\boldsymbol{U}}_2\right)^H \left(\boldsymbol{\Phi}_1\tilde{\boldsymbol{U}}_1 \otimes \boldsymbol{\Phi}_2\tilde{\boldsymbol{U}}_2\right)\\ &= (\boldsymbol{\Phi}_1\tilde{\boldsymbol{U}}_1)^H(\boldsymbol{\Phi}_1\tilde{\boldsymbol{U}}_1) \otimes (\boldsymbol{\Phi}_2\tilde{\boldsymbol{U}}_2)^H(\boldsymbol{\Phi}_2\tilde{\boldsymbol{U}}_2)\\ &= \boldsymbol{T}_1(\mathcal{X}_1) \otimes \boldsymbol{T}_2(\mathcal{X}_2)\end{aligned}$$

$\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$

- Since rank$(\boldsymbol{A} \otimes \boldsymbol{B}) =$ rank$(\boldsymbol{A})$rank$(\boldsymbol{B})$, we require (additional constraints)

$$|\mathcal{X}_1| \geq L_1 \text{ and } |\mathcal{X}_2| \geq L_2$$

How to design the subsampler?

- As before, we optimize a scalar function of the error covariance matrix

$$\underset{\mathcal{X}}{\text{maximize}} \; f(\boldsymbol{T}(\mathcal{X}))$$

$$\text{s.to} \quad |\mathcal{X}| = K, \; \mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$$

$$|\mathcal{X}| \geq L_1 \quad |\mathcal{X}_2| \geq L_2$$

- In particular, we minimize the so-called *frame potential* (related to the mean squared error)

$$F(\mathcal{X}) := \text{trace}\{\boldsymbol{T}^H\boldsymbol{T}\} = \text{trace}\{\boldsymbol{T}_1^H\boldsymbol{T}_1 \otimes \boldsymbol{T}_2^H\boldsymbol{T}_2\} := F_1(\mathcal{X}_1)F_2(\mathcal{X}_2)$$

- Or, maximize the set function with change of variable $\mathcal{S} = \mathcal{N} \setminus \mathcal{X}$

$$G(\mathcal{S}) = F(\mathcal{N}) - F(\mathcal{N} \setminus \mathcal{S}) \qquad \mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$$

Set function is submodular and monotone non-decreasing

How to design the subsampler?

- Therefore, we have to solve

Truncated partition matroid

$$\underset{\mathcal{S}\subseteq\mathcal{N}}{\text{maximize}}\ G(\mathcal{S})$$

$$\text{s.to}\quad \mathcal{S}\in\mathcal{I}_u\cap\mathcal{I}_u,$$
$$\mathcal{I}_u=\{\mathcal{S}\subseteq\mathcal{N}:\mathcal{S}\leq N-K\}$$
$$\mathcal{I}_p=\{\mathcal{S}\subseteq\mathcal{N}:|\mathcal{S}\cap\mathcal{N}_i|\leq N_i-L_i, i=1,2\}$$

[Ortiz-Jiménez et al.-2018]

1. **Require** $\mathcal{X}=\emptyset, K, \mathcal{I}_u, \mathcal{I}_p$.
2. **for** $k=1$ to $N-K$
3. $s^*=\underset{s\notin\mathcal{X}}{\arg\max}\{f(\mathcal{X}\cup\{s\}):\mathcal{X}\in\mathcal{I}_u\cap\mathcal{I}_p\}$
4. $\mathcal{X}\leftarrow\mathcal{X}\cup\{s^*\}$
5. **end**
6. **Return** $\mathcal{X}$

- Near optimality guarantees

$$G(\mathcal{S}_{\text{greedy}})\geq\frac{1}{2}G(\mathcal{S}^\star)$$

[Nemhauser-Wolsey-Fisher-1978]

- Linear sweep time

- G. Ortiz-Jiménez, M. Coutino, S.P. Chepuri, and G. Leus. Sparse Sampling for Inverse Problems with Tensors. *IEEE TSP (under review)*, June 2018. (available as arXiv:1806.10976).
- G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions— I," Mathematical Programming, vol. 14, no. 1, pp. 265–294, 1978.

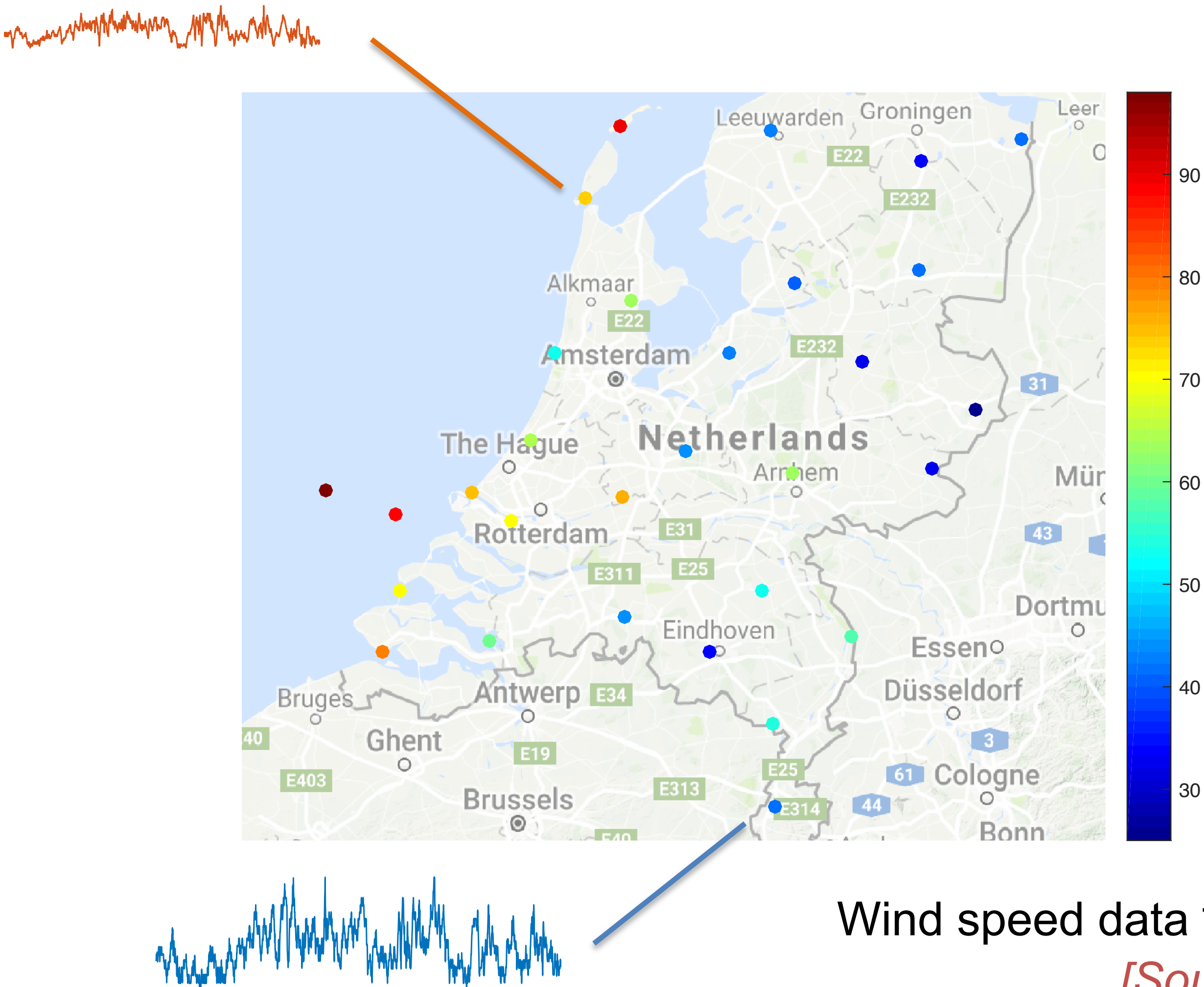
Graph Learning or Topology Inference

- S.P. Chepuri, S. Liu, G. Leus, and A. Hero. Learning Sparse Graphs Under Smoothness Prior. *ICASSP 2017*, New Orleans, USA.

- S.K. Kadambari and S.P. Chepuri. Learning Product Graphs from Multidomain Signals. ICASSP 2020, Barcelona, Spain.

- V. Kalofolias, "How to learn a graph from smooth signals," in Proc. of the 19th International Conference on Artificial Intelligence and Statistics, 2016.

- X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE TSP, vol. 64, no. 23, Dec. 2016*.



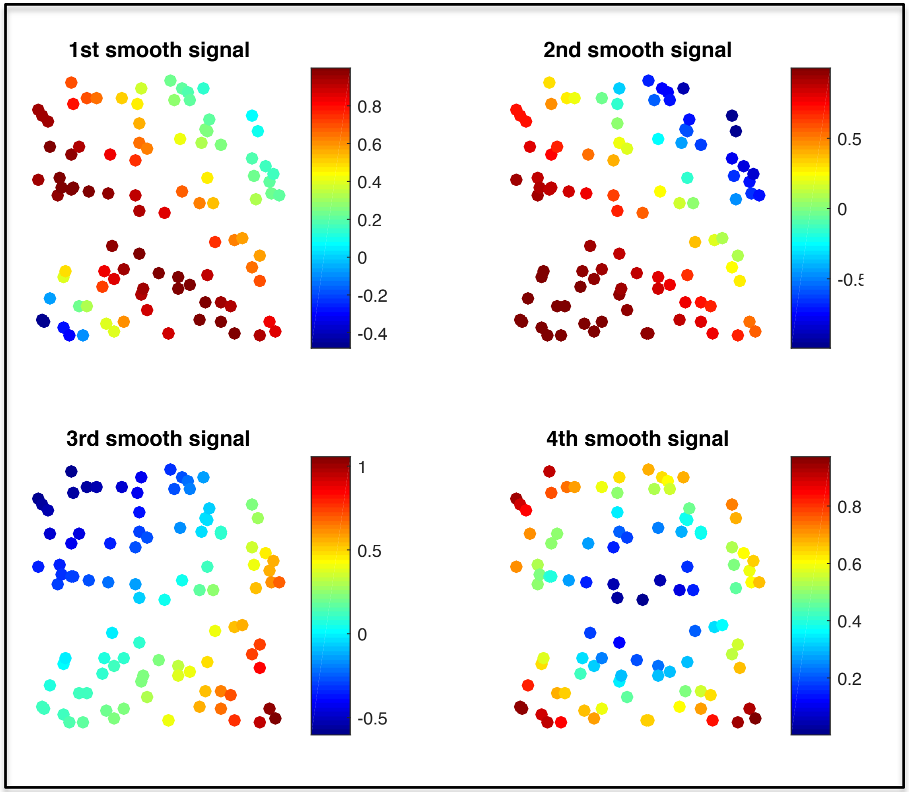
Wind speed data from 30 stations

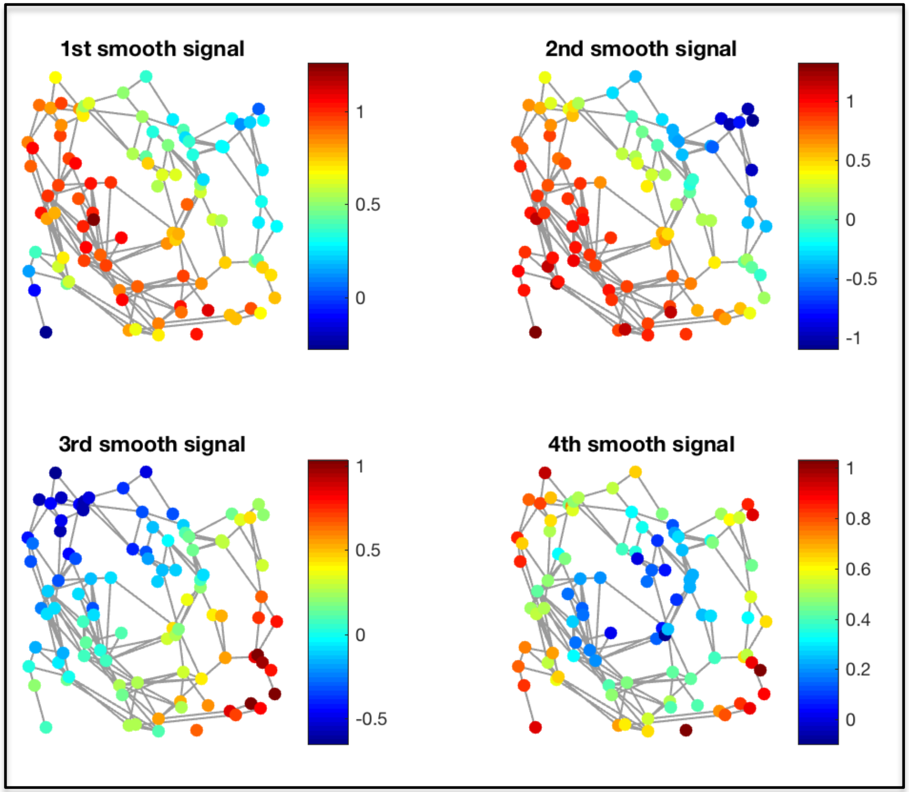
[Source: KNMI, Netherlands]

"Learn a sparse graph that sufficiently explains the data"

Sparse graph learning problem

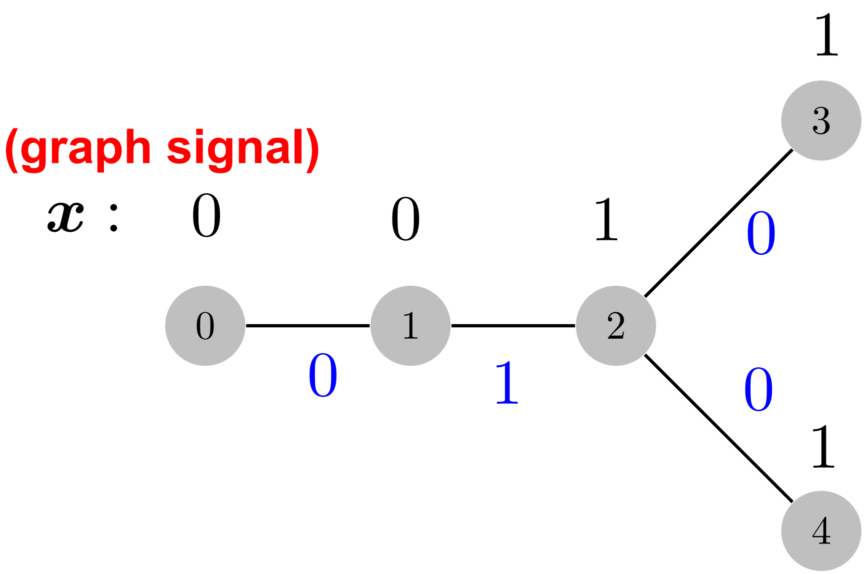
Learn a "**sparse graph**" (or estimate the graph Laplacian matrix) from smooth data





Learnt graph with K = 175 edges using 4 snapshots

Graph Laplacian – quadratic form



$$x^T L x = \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2$$

$$= 1$$

Sum of squares of differences across edges

- Quantifies **smoothness** of x with respect to the underlying graph

- When multiple snapshots x_i for $i = 1, 2, \ldots, T$ are available, then the quadratic form will be

$$\sum_{i=1}^{T} x_i^T L x_i = \mathrm{tr}(X^T L X)$$

- Small values of $\mathrm{tr}(\mathbf{X}^T \mathbf{L}_N \mathbf{X})$ implies that $\mathbf{X}$ is smooth on the graph

Graph Learning from smooth data

- Given training graph data $\boldsymbol{X} : N \times T$, or its noisy or incomplete version, $\boldsymbol{Y}$, estimate the graph Laplacian matrix

- This is an ill-posed problem, but we know the set of all the valid Laplacian matrices

$$\mathcal{L}_N := \{\mathbf{L} \in \mathbb{R}^{N \times N} | \mathbf{L1} = \mathbf{0}, \mathrm{tr}(\mathbf{L}) = N, L_{ij} = L_{ji} \leq 0, i \neq j\}$$

- The graph learning problem reduces to

$$\underset{\mathbf{L}_N \in \mathcal{L}_N, \mathbf{X}}{\text{minimize}} \quad f(\mathbf{X}, \mathbf{Y}) + \alpha \mathrm{tr}(\mathbf{X}^T \mathbf{L}_N \mathbf{X}) + \beta \|\mathbf{L}_N\|_F^2$$

$\|.\|_F^2$ controls the distribution the edge weights of the learned graph

α and β are two positive regularization parameters

Graph Learning from smooth data

- The graph learning problem is then solved using alternating minimization:

Step 1 (convex optimization): Fix $\mathbf{X}$

$$\underset{\mathbf{L}\in\mathcal{L}_{\mathcal{N}}}{\text{minimize}} \quad \alpha\,\text{tr}\{\mathbf{X}^T\mathbf{L}\mathbf{X}\} + \beta\,\|\mathbf{L}\|_F^2$$

✓ Since the Laplacian matrix is symmetric for undirected graphs, we need to estimate only its upper or lower triangular elements.

Step 2: Fix $\mathbf{L}$

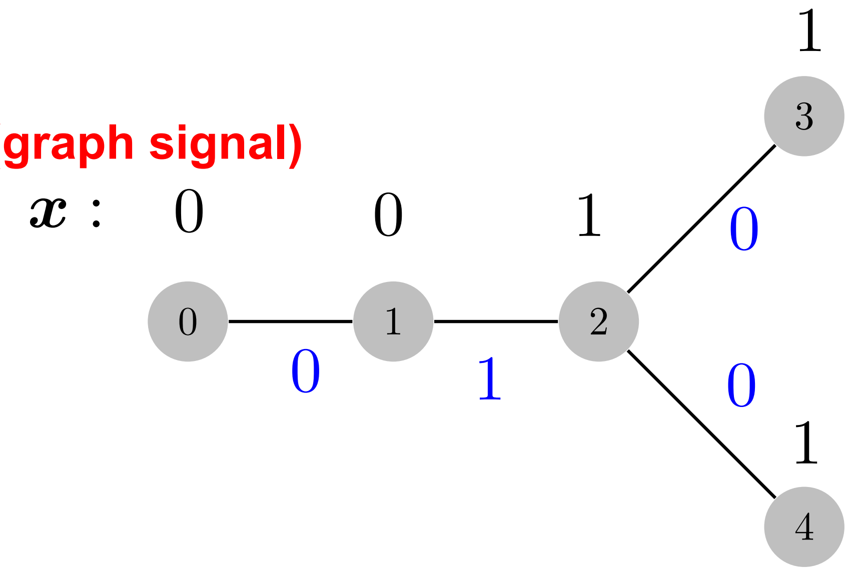
$$\underset{\mathbf{X}}{\text{minimize}} \quad f(\mathbf{X},\mathbf{Y}) + \alpha\text{tr}\{\mathbf{X}^T\mathbf{L}\mathbf{X}\}$$

✓ Depending the observation model, often the above problem can be relaxed to a convex optimization problem.

Requires parameter tuning

Graph Laplacian – quadratic form

(graph signal)



$$\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2$$

$$= 1$$

Sum of squares of differences across edges

➢ Laplacian matrix can be written as a outer product of "incidence" vectors

$$\boldsymbol{L} = \boldsymbol{A}\boldsymbol{A}^T = \sum_{m=1}^{M} \boldsymbol{a}_m \boldsymbol{a}_m^T \quad \text{(quadratic form)}$$

$[\boldsymbol{a}_m]_i = 1$

$[\boldsymbol{a}_m]_j = -1$

zeros elsewhere

For an edge "m" connecting node "i " and "j "

Graph learning as a sampling problem

- Denote the subgraph of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ or K-sparse graph

$$\mathcal{G}_s(\mathcal{V}, \mathcal{E}_s) \text{ with the edge set } \mathcal{E}_s \subset \mathcal{E} \text{ such that } |\mathcal{E}_s| = K \ll M$$

- Introduce an "edge sampling" vector

$$\boldsymbol{w} = [w_1, w_2, \cdots, w_M]^T \in \{0, 1\}^M$$

$w_m = 1$ if an edge belongs to the edge subset $\mathcal{E}_s$

No. of edges of:
- Complete graph
- Given graph

- Graph Laplacian of the K-sparse graph

$$\boldsymbol{L}_s(\boldsymbol{w}) = \sum_{m=1}^{M} w_m \boldsymbol{a}_m \boldsymbol{a}_m^T$$

(Recall the outer product decomposition of the Laplacian)

Sparse edge selection

- Given L "noiseless" graph signals $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L]$

- *K-sparse* graph learning will be

$$\arg\min_{\boldsymbol{w} \in \mathcal{W}} \quad \frac{1}{L} \sum_{k=1}^{L} \boldsymbol{x}_k^T \boldsymbol{L}_s(\boldsymbol{w}) \boldsymbol{x}_k = \frac{1}{L} \mathrm{tr}\{\boldsymbol{X}^T \boldsymbol{L}_s(\boldsymbol{w}) \boldsymbol{X}\}$$

$$\mathcal{W} = \{\boldsymbol{w} \in \{0,1\}^M \mid \|\boldsymbol{w}\|_0 = K\}$$

Non-convex (Boolean optimization problem)

Sparse edge selection

- Given L "noiseless" graph signals $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L]$
- *K-sparse* graph learning will be

$$\arg\min_{\boldsymbol{w}\in\mathcal{W}} \quad \frac{1}{L}\sum_{k=1}^{L} \boldsymbol{x}_k^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}_k = \frac{1}{L}\mathrm{tr}\{\boldsymbol{X}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\}$$

$$\mathcal{W} = \{\boldsymbol{w} \in \{0,1\}^M \mid \|\boldsymbol{w}\|_0 = K\}$$

- Cost function (modular):

$$\frac{1}{L}\mathrm{tr}\left\{\boldsymbol{X}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\right\} = \sum_{m=1}^{M} w_m \mathrm{tr}\left\{\boldsymbol{X}^T (\boldsymbol{a}_m \boldsymbol{a}_m{}^T)\boldsymbol{X}\right\}$$

- **Solution: rank ordering!**

  ✓ Computational complexity O(K log K), or O(K) with parallel implementation

Sparse edge selection

- Given L "noiseless" graph signals, K-sparse graph learning

$$\arg\min_{\boldsymbol{w}\in\mathcal{W}} \quad \frac{1}{L}\sum_{k=1}^{L}\boldsymbol{x}_k^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}_k = \frac{1}{L}\mathrm{tr}\{\boldsymbol{X}^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\}$$

$$\mathcal{W} = \{\boldsymbol{w}\in\{0,1\}^M \mid \|\boldsymbol{w}\|_0 = K\}$$

Example: Suppose covariance matrix of $\boldsymbol{x}$ is $\boldsymbol{R}_{\boldsymbol{x}}$, then

$$L^{-1}\mathrm{tr}\{\boldsymbol{X}^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\} = \sum_{m=1}^{M} w_m(\boldsymbol{a}_m{}^T\widehat{\boldsymbol{R}}_{\boldsymbol{x}}\boldsymbol{a}_m)$$

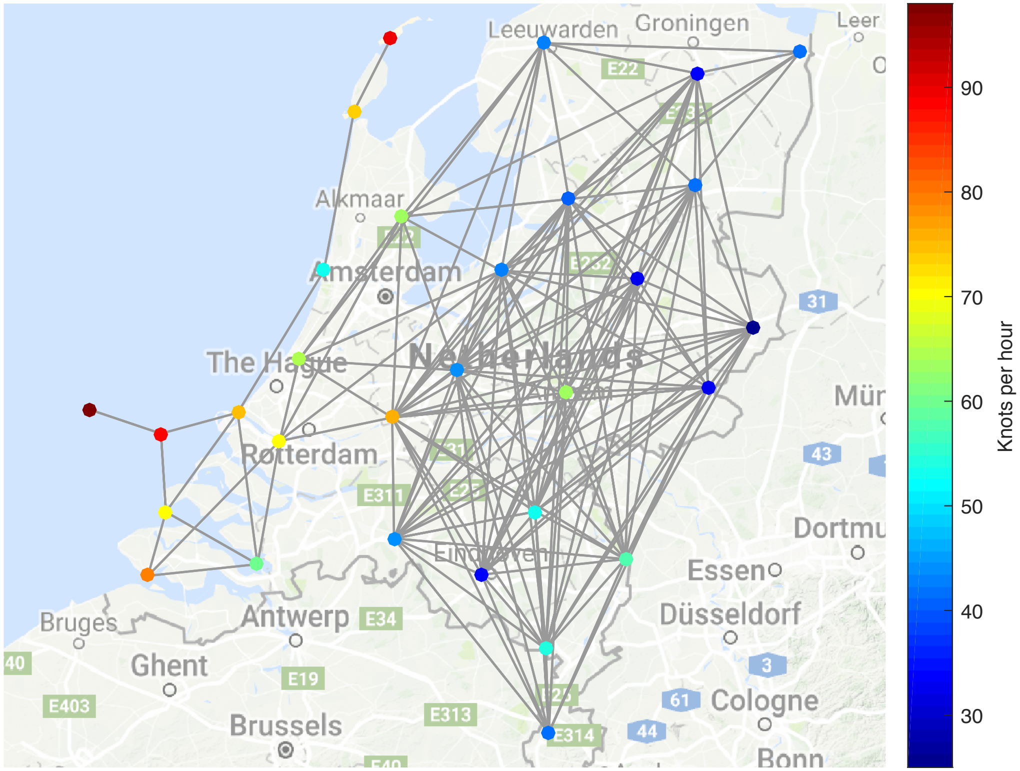
Solution: select K edges between those nodes having highest cross-correlation as

$$\boldsymbol{a}_m{}^T\widehat{\boldsymbol{R}}_{\boldsymbol{x}}\boldsymbol{a}_m = [\widehat{\boldsymbol{R}}_{\boldsymbol{x}}]_{i,i} + [\widehat{\boldsymbol{R}}_{\boldsymbol{x}}]_{j,j} - 2[\widehat{\boldsymbol{R}}_{\boldsymbol{x}}]_{i,j}$$

(Special case: GMRF model with $\boldsymbol{R}_{\boldsymbol{x}} := \boldsymbol{L}^{\dagger} + \sigma^2\mathbf{I}$)

Numerical experiments – windspeed data

K=125

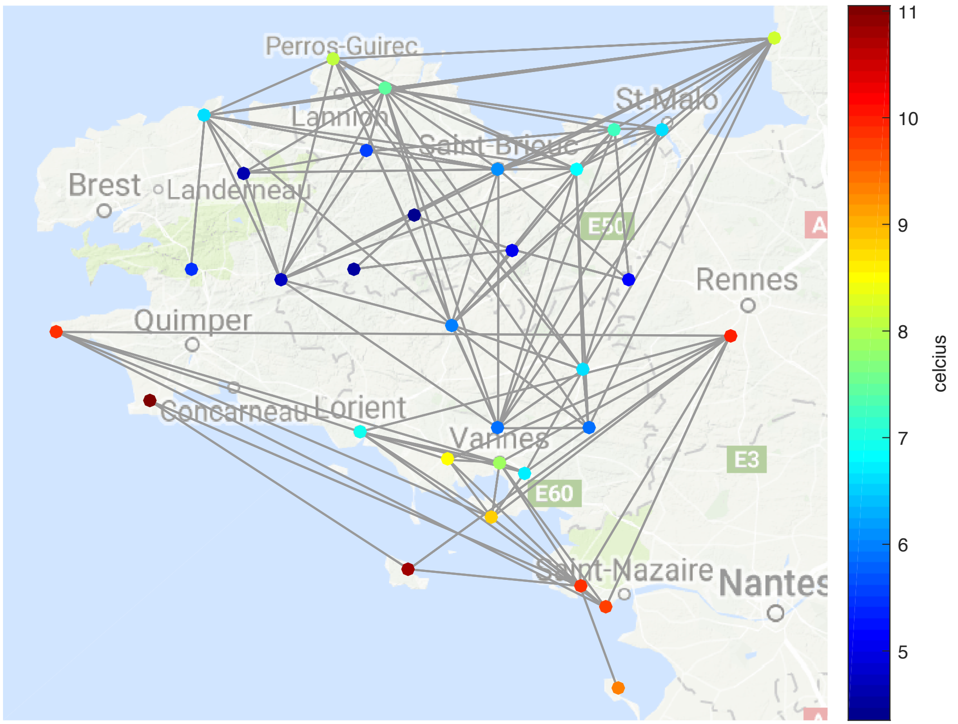


Wind speed data of year 2002 from 30 stations

[Source: KNMI, Netherlands]

Numerical experiments – French temp. data

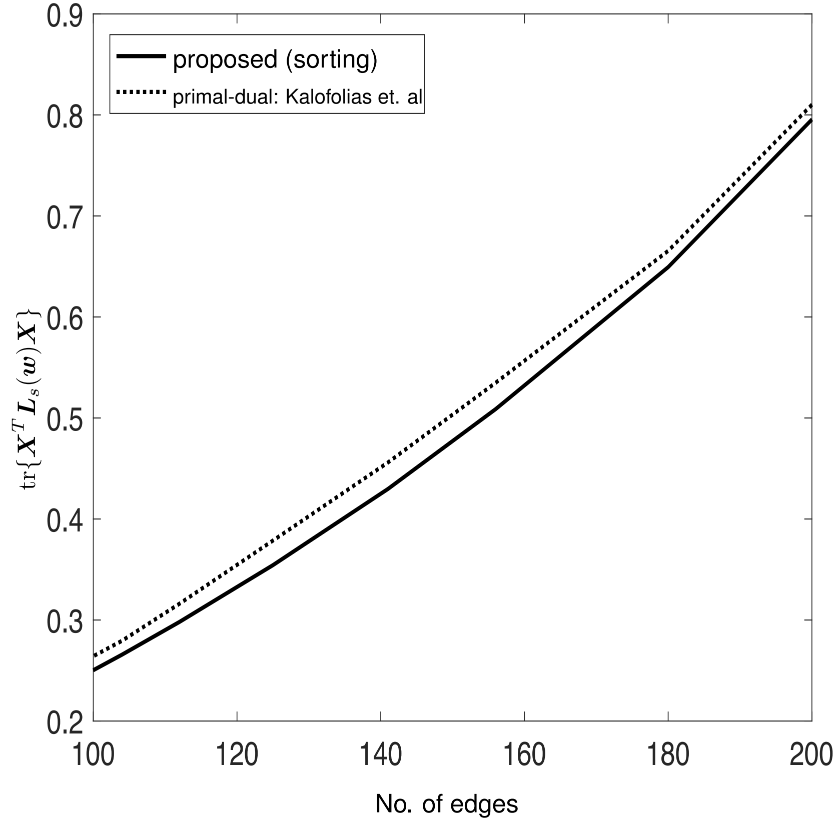
K=110



Temperature data of Brittany, France from 32 stations

Thanks to N. Perraudin and P. Vandergheynst for the dataset.

Numerical experiments - performance



Kalofolias: $$\text{minimize}_{\boldsymbol{L}\in\mathcal{L}} \sum_{k=1}^{L} \boldsymbol{x}_k^T \boldsymbol{L} \boldsymbol{x}_k + \lambda \text{card}(\mathbf{L})$$

$$\mathcal{L} = \{\boldsymbol{L} \succeq 0, L_{i,j} = L_{j,i} \leq 0, \boldsymbol{L}\mathbf{1} = \mathbf{0}\}$$

• V. Kalofolias, “How to learn a graph from smooth signals,” in Proc. of the 19th International Conference on Artificial Intelligence and Statistics, 2016, pp. 920–929.

Sparse edge selection with "denoising"

- Given "L" noisy signals: $\boldsymbol{y}_k = \boldsymbol{x}_k + \boldsymbol{n}_k$,

$$\arg\min_{\{\boldsymbol{x}_k\}_{k=1}^{L}, \boldsymbol{w} \in \mathcal{W}} \frac{1}{L} \sum_{k=1}^{L} (\|\boldsymbol{y}_k - \boldsymbol{x}_k\|_2^2 + \gamma\, \boldsymbol{x}_k^T \boldsymbol{L}_s(\boldsymbol{w}) \boldsymbol{x}_k)$$

- Alternating minimization

Fixed $\boldsymbol{w}$: $\boldsymbol{X}_{\min}(\boldsymbol{w}) = [\mathbf{I} + \gamma \boldsymbol{L}_s(\boldsymbol{w})]^{-1} \boldsymbol{Y}$ (denoising)

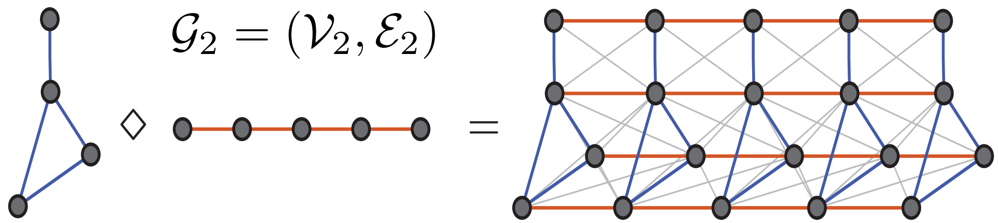
Fixed $\boldsymbol{X}$: $\boldsymbol{w}_{\min}(\boldsymbol{X})$ sorting, as before (edge selection)

- ✓ Converges to a stationary point
- ✓ Suffers from the choice of the initial estimate

Product graph learning

- S.K. Kadambari and S.P. Chepuri. Learning Product Graphs from Multidomain Signals. ICASSP 2020, Barcelona, Spain.

Product graph learning



- Cartesian product (colored edges)
- Kronecker product (gray edges)
- Strong product (all edges)

Given $\mathbf{L}_N$ the graph factors $\mathbf{L}_P$ and $\mathbf{L}_Q$ can be obtained by solving

$$\underset{\mathbf{L}_P \in \mathcal{L}_P, \mathbf{L}_Q \in \mathcal{L}_Q}{\text{minimize}} \; \|\mathbf{L}_N - \mathbf{L}_P \oplus \mathbf{L}_Q\|_F^2$$

➢ This is a twostep approach

✓ computing a size $-N$ Laplacian matrix

✓ factorizing the Laplacian matrix into $\mathbf{L}_P$ and $\mathbf{L}_Q$

One-step approach

- When Laplacian matrix has a Cartesian product structure

$$\mathbf{L}_N = \mathbf{L}_P \oplus \mathbf{L}_Q = \mathbf{I}_Q \otimes \mathbf{L}_P + \mathbf{L}_Q \otimes \mathbf{I}_P$$

- Product graph learning problem reduces to

$$\underset{\mathbf{L}_P \in \mathcal{L}_\mathcal{P}, \mathbf{L}_Q \in \mathcal{L}_\mathcal{Q}}{\text{minimize}} \quad \alpha \, \text{tr}\{\mathbf{X}^T (\mathbf{L}_P \oplus \mathbf{L}_Q) \mathbf{X}\} + \beta_1 \|\mathbf{L}_P\|_F^2 + \beta_2 \|\mathbf{L}_Q\|_F^2$$

  - ✓ The optimization problem is convex
  - ✓ we need to solve for only the upper or lower triangular elements
  - ✓ The problem is equivalent to

$$\underset{\mathbf{z} \in \mathbb{R}^K}{\text{minimize}} \quad \tfrac{1}{2}\mathbf{z}^T \mathbf{P}\mathbf{z} + \mathbf{q}^T \mathbf{z}, \quad \text{subject to} \quad \mathbf{C}\mathbf{z} = \mathbf{d}, \mathbf{z} \geq \mathbf{0}$$

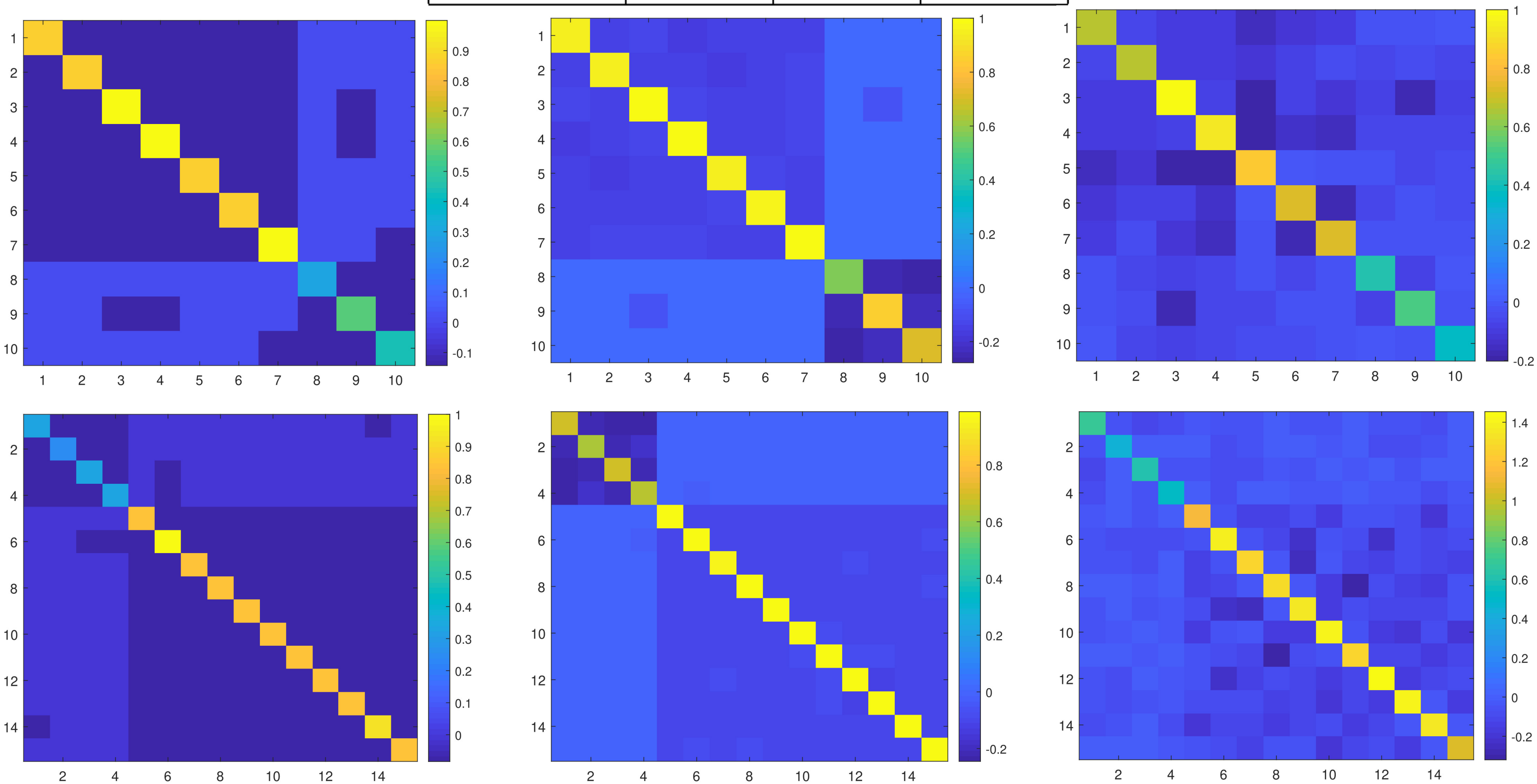
  - ✓ has an explicit water-filling solution

- S.K. Kadambari and S.P. Chepuri," Learning product graphs from multidomain signals," ICASSP 2020, Barcelona, Spain.

Numerical experiments

F-measure:

Method	L_P	L_Q	L_N
Solver 1	0.9615	0.9841	0.9755
Solver 2	0.7556	0.7842	0.7612



Ground Truth

One-step method

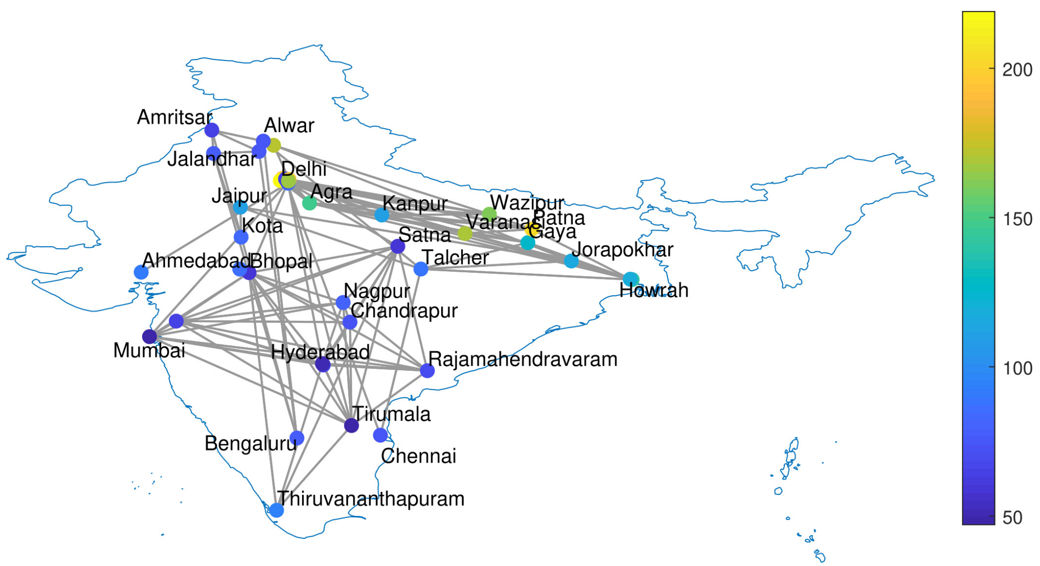
Two-step method

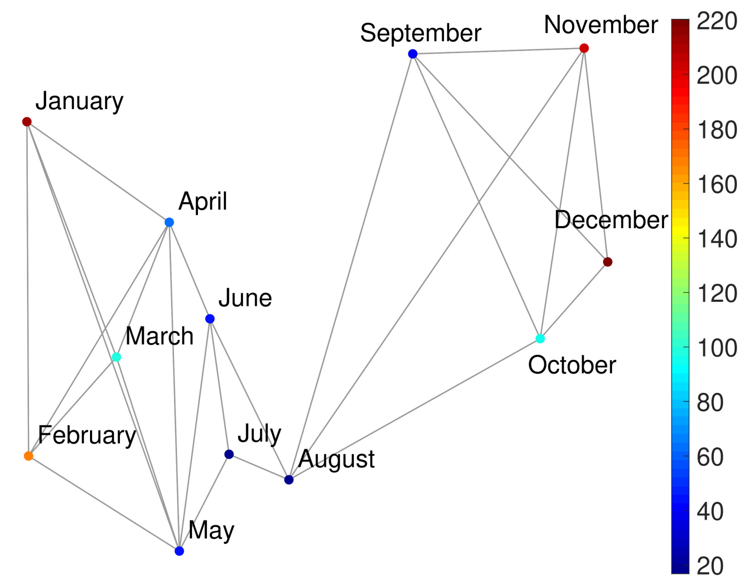
Air quality data

- PM 2.5 data collected over 40 air quality monitoring stations in different locations in India for each day of the year 2018

- The dataset has missing entries, which are imputed using a graph Laplacian regularized nuclear norm minimization

$$f(\mathbf{X},\mathbf{Y}) := \sum_{i=1}^{T} \|\mathcal{A}(\mathbf{X}_i - \mathbf{Y}_i)\|_F^2 + \gamma \|\mathbf{X}_i\|_*$$

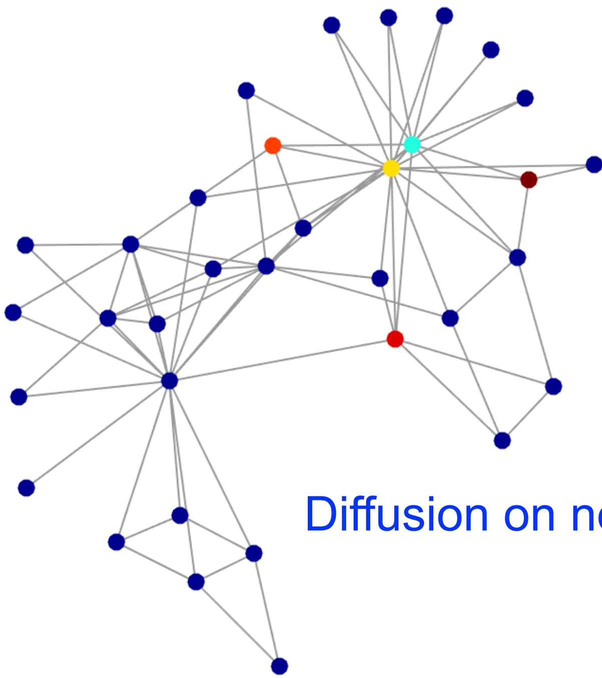




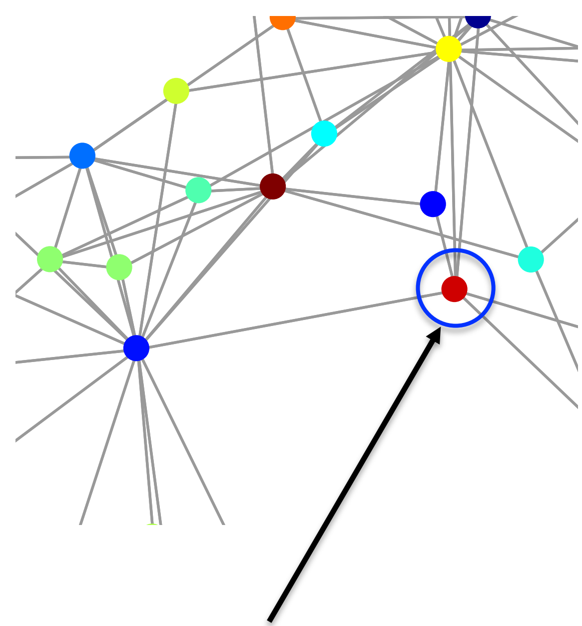
Topology inference from partial observations

- S.P. Chepuri, M. Coutino, A. Marques, and G. Leus. Disitributed Analytical Graph Identification, ICASSP 2018, Vancouver, Cannada.

Distributed computation of eigenmodes of a network



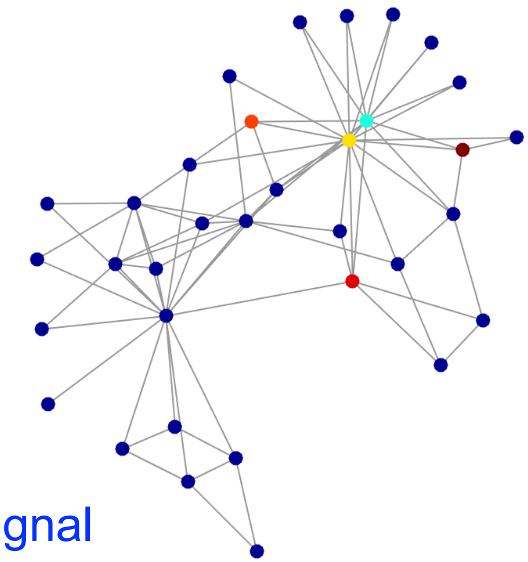
Diffusion on networks



Can we infer the graph topology using observations at a single node?

Linear dynamics on networks

Linear network dynamics



➢ Information flow to a node from its neighbors

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{b}u_{k-1}$$
$$y_k = \boldsymbol{e}_i^T \boldsymbol{x}_k$$

Observation at *node i*

Known excitation signal

$\boldsymbol{x}_{-1} = 0$ and $\boldsymbol{x}_0 = \boldsymbol{b}$

$\boldsymbol{e}_i$ is the ith column of the identity matrix

➢ Given observations $\boldsymbol{y} = \{y_0, \ldots, y_{K-1}\}$ and $\boldsymbol{b}$ compute $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$

Each node will have an overview of the network

Computing eigenfrequencies

- Information flow to a node from its neighbors

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{b}u_{k-1} \\ y_k &= \boldsymbol{e}_i^T \boldsymbol{x}_k \end{aligned}$$

$\boldsymbol{x}_{-1} = 0$ and $\boldsymbol{x}_0 = \boldsymbol{b}$

- At *node i*, we *aggregate measurements*

[Marques et al.-2016]

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{A} \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{A}^{K-1} \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{U}\boldsymbol{\Lambda}^{K-1}\boldsymbol{U}^T \end{bmatrix} \boldsymbol{b}$$

- A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," TSP 2016.

Computing eigenfrequencies

- At the observed node

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{A} \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{A}^{K-1} \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{U}\boldsymbol{\Lambda}^{K-1}\boldsymbol{U}^T \end{bmatrix} \boldsymbol{b}$$

$$= \boldsymbol{V}\text{diag}[\underline{\boldsymbol{u}}]\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{V}\boldsymbol{\theta}$$

$$\underline{\boldsymbol{u}} = \boldsymbol{e}_i^T \boldsymbol{U}$$

Remark: $\boldsymbol{U}^T\boldsymbol{b}$ should not be sparse to excite all modes

$$\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N] = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_N \\ \lambda_1^2 & \lambda_2^2 & \cdots & \lambda_N^2 \\ \vdots & \vdots & & \vdots \\ \lambda_1^{K-1} & \lambda_2^{K-1} & \cdots & \lambda_N^{K-1} \end{bmatrix}$$

Roots of this Vandermonde matrix are the eigenfrequencies

Computing eigenfrequencies

- Arrange data in each node as

$$\boldsymbol{Y}_0 = \begin{bmatrix} y_N & y_{N-1} & \cdots & y_1 \\ y_{N+1} & y_N & \cdots & y_2 \\ \vdots & \vdots & & \vdots \\ y_{K-2} & y_{K-3} & \cdots & y_{N-K-1} \end{bmatrix} \qquad \boldsymbol{Y}_1 = \begin{bmatrix} y_{N-1} & y_{N-2} & \cdots & y_0 \\ y_N & y_{N-1} & \cdots & y_1 \\ \vdots & \vdots & & \vdots \\ y_{K-1} & y_{K-2} & \cdots & y_{N-K} \end{bmatrix}$$

- To form the data matrices, we require *2N aggregations*

- Roots of the pencil of matrices $\boldsymbol{Y}_0 - \lambda \boldsymbol{Y}_1$ produce the roots of $\boldsymbol{V}$

- Eigenfrequencies are the generalized eigenvalues

$$\boldsymbol{\Lambda} = \texttt{geig}(\boldsymbol{Y}_0, \boldsymbol{Y}_1) = \texttt{eig}(\boldsymbol{Y}_1^{-1} \boldsymbol{Y}_0)$$

Computing eigenfrequencies

- Arrange data in each node as

$$\boldsymbol{Y}_0 = \begin{bmatrix} y_N & y_{N-1} & \cdots & y_1 \\ y_{N+1} & y_N & \cdots & y_2 \\ \vdots & \vdots & & \vdots \\ y_{K-2} & y_{K-3} & \cdots & y_{N-K-1} \end{bmatrix} \qquad \boldsymbol{Y}_1 = \begin{bmatrix} y_{N-1} & y_{N-2} & \cdots & y_0 \\ y_N & y_{N-1} & \cdots & y_1 \\ \vdots & \vdots & & \vdots \\ y_{K-1} & y_{K-2} & \cdots & y_{N-K} \end{bmatrix}$$

- When some $\{\lambda_i\}$ are very close, $\boldsymbol{Y}_1$ is ill-conditioned

Generalized Schur decomposition: $\boldsymbol{Y}_0 = \boldsymbol{Q}\boldsymbol{S}\boldsymbol{Z}^H$ and $\boldsymbol{Y}_1 = \boldsymbol{Q}\boldsymbol{T}\boldsymbol{Z}^H$

$\boldsymbol{Q}$ is a unitary matrix

$\boldsymbol{S}$ and $\boldsymbol{T}$ are upper triangular matrices

$$\lambda(\boldsymbol{Y}_0, \boldsymbol{Y}_1) = \{[\boldsymbol{S}]_{nn}/[\boldsymbol{T}]_{nn} \, : \, [\boldsymbol{T}]_{nn} > \epsilon\}$$

Computing the eigenmodes

- To compute the eigenvectors, we require multiple snapshots of the data.
- Suppose $M \geq K$ snapshots of the input signal are available

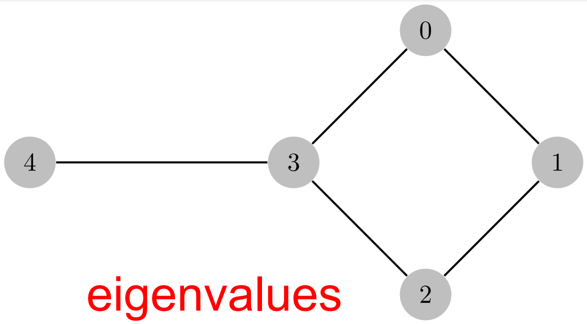
$$
\begin{aligned}
[\boldsymbol{y}_1 \cdots \boldsymbol{y}_M] &= \begin{bmatrix} \boldsymbol{e}_i^T \\ \boldsymbol{e}_i^T \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T \\ \vdots \\ \boldsymbol{e}_i^T \boldsymbol{U} \boldsymbol{\Lambda}^{K-1} \boldsymbol{U}^T \end{bmatrix} [\boldsymbol{b}_1 \cdots \boldsymbol{b}_M] \\
\boldsymbol{Y} &= \boldsymbol{V} \text{diag}[\underline{\boldsymbol{u}}] \boldsymbol{U}^T \boldsymbol{B}
\end{aligned}
$$

- Inverting $\boldsymbol{V}$ and $\boldsymbol{B}$

$$
\boldsymbol{H} = \boldsymbol{V}^\dagger \boldsymbol{Y} \boldsymbol{B}^\dagger = \text{diag}[\underline{\boldsymbol{u}}] \boldsymbol{U}^T \Rightarrow \boldsymbol{G} = \boldsymbol{H}^T \boldsymbol{H} = \boldsymbol{U} \text{diag}^2[\underline{\boldsymbol{u}}] \boldsymbol{U}^T
$$

Eigenmodes of the graph are the eigenvectors of G

Numerical experiments



Laplacian matrix

eigenvalues

eigenvectors

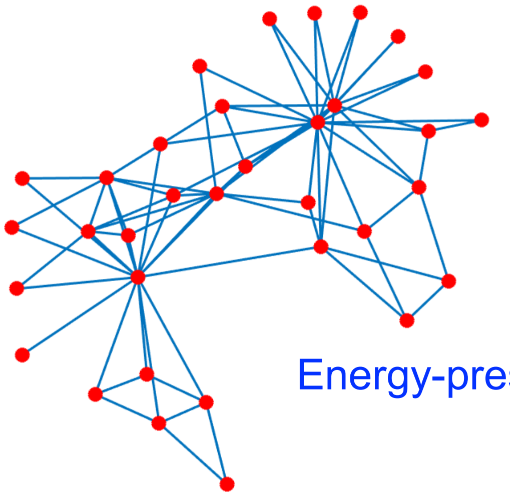
$$\boldsymbol{\lambda} = \begin{bmatrix} 0 \\ 0.8299 \\ 2 \\ 2.6889 \\ 4.4812 \end{bmatrix} \qquad \boldsymbol{U} = \begin{bmatrix} -0.4472 & -0.2560 & 0.7071 & 0.2422 & -0.4193 \\ -0.4472 & -0.4375 & 0 & -0.7031 & 0.3380 \\ -0.4472 & -0.2560 & -0.7071 & 0.2422 & -0.4193 \\ -0.4472 & 0.1380 & 0 & 0.5362 & 0.7024 \\ -0.4472 & 0.8115 & 0 & -0.3175 & -0.2018 \end{bmatrix}$$

$$\hat{\boldsymbol{U}} = \begin{bmatrix} -0.7071 & -0.4472 & 0.4193 & -0.2560 & -0.2422 \\ 0 & -0.4472 & -0.3380 & -0.4375 & 0.7031 \\ 0.7071 & -0.4472 & 0.4193 & -0.2560 & -0.2422 \\ 0 & -0.4472 & -0.7024 & 0.1380 & -0.5362 \\ 0 & -0.4472 & 0.2018 & 0.8115 & 0.3175 \end{bmatrix}$$

✓ Eigenvectors are recovered up to a sign flip and column permutation

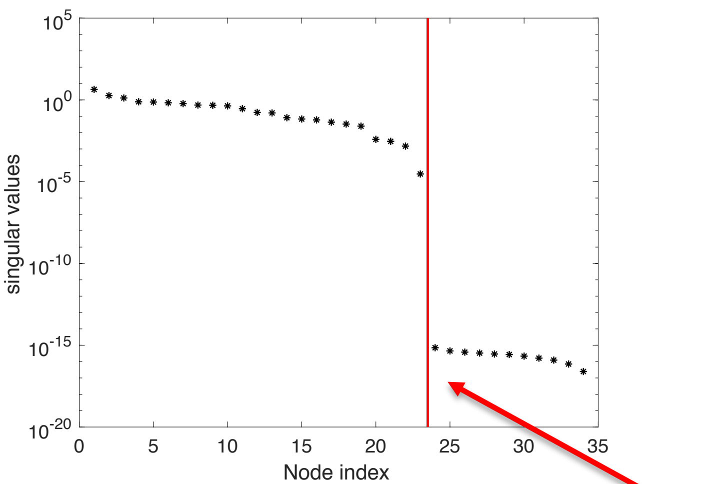
✓ Frequency interpretation of the eigenvectors are retained

Numerical experiments

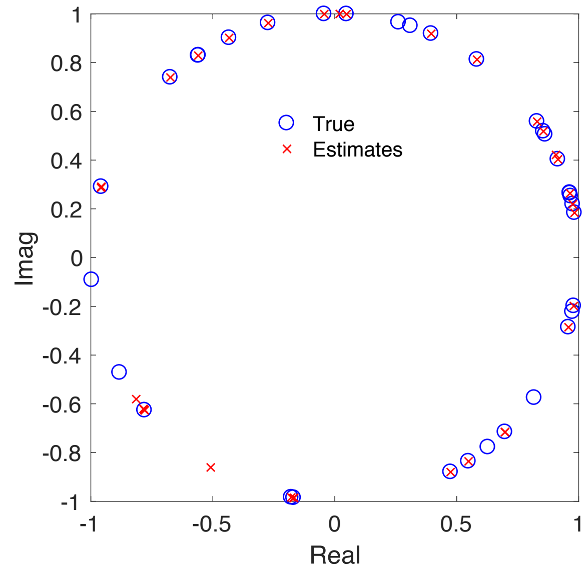


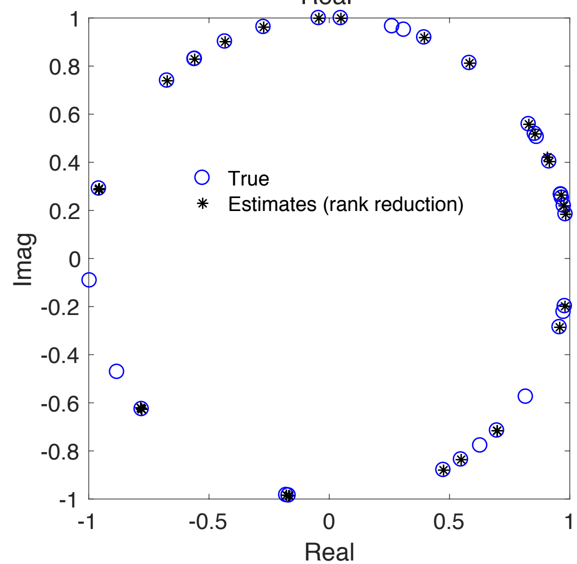
Energy-preserving shift

Spectrum of the Toeplitz data matrix $\boldsymbol{Y}_1$



$$\lambda(\boldsymbol{Y}_0, \boldsymbol{Y}_1) = \{[\boldsymbol{S}]_{nn}/[\boldsymbol{T}]_{nn} \,:\, [\boldsymbol{T}]_{nn} > \epsilon\}$$



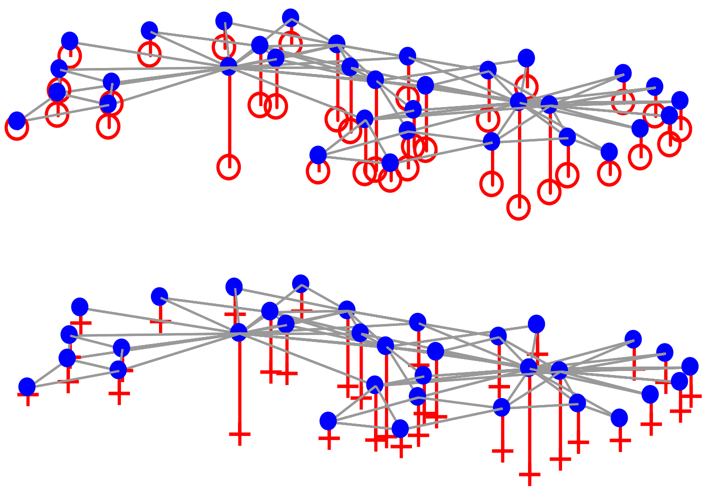


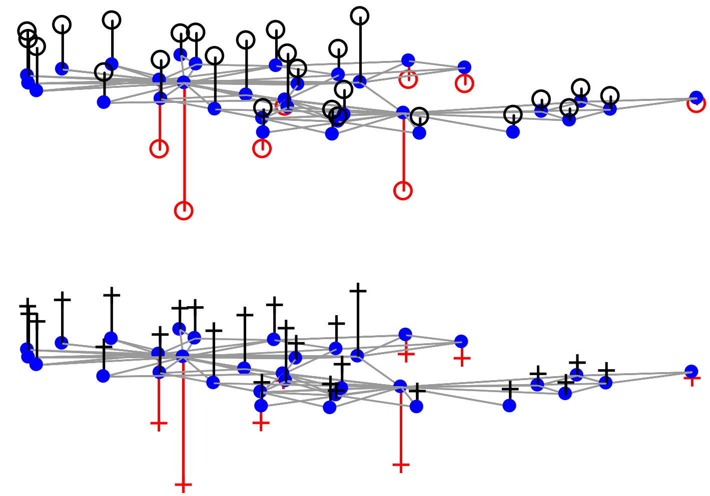
- A. Gavili and X.-P. Zhang, “On the shift operator, graph frequency and optimal filtering in graph signal processing,” TSP 2017.

Numerical experiments

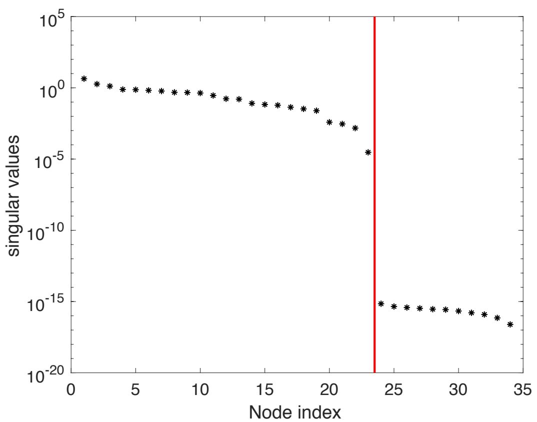
$\boldsymbol{u}_1$ $\boldsymbol{u}_N$

$\hat{\boldsymbol{u}}_1$ $\hat{\boldsymbol{u}}_N$

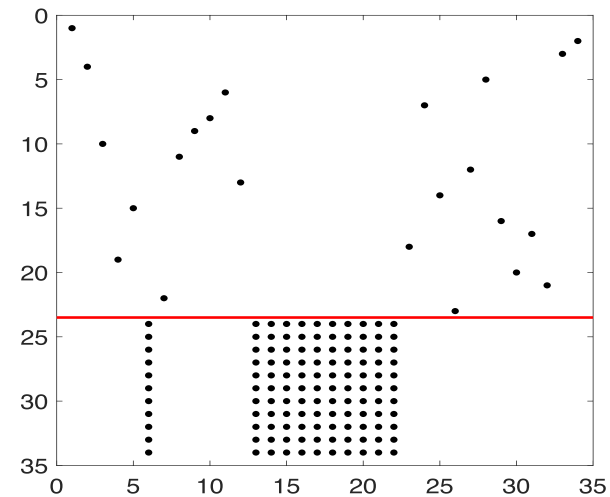




Spectrum of the Toeplitz data matrix $\boldsymbol{Y}_1$



$\hat{\boldsymbol{U}}^H \boldsymbol{U}$

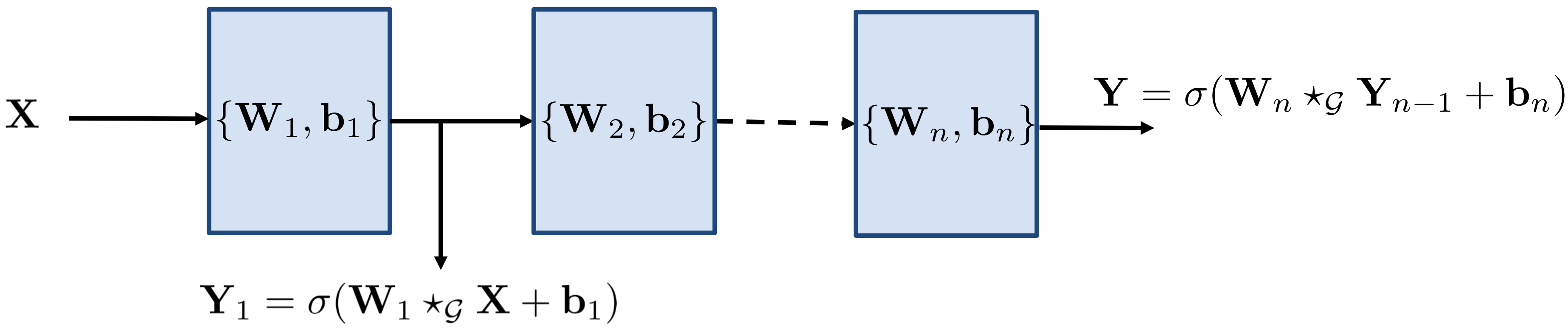


Well represented modes are recovered up to a column permutation

Geometric deep learning

- http://geometricdeeplearning.com/
- M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond Euclidean data, IEEE Signal Processing Magazine 2017 (Review paper)
- S.K. Kadambari and S.P. Chepuri, Fast Graph Convolutional Recurrent Neural Networks. Asilomar 2019, Pacific Grove, USA
- A. Madapu, S. Segarra, S.P. Chepuri, and A. Marques, Generative Adversarial Networks for Graph Data Imputation from Signed Observations. ICASSP 2020, Barcelona, Spain

Graph neural nets (GCNs)



Chebyshev polynomial

$$\mathbf{W} \star_{\mathcal{G}} \mathbf{X} = \sum_{k=0}^{K} w_k \mathbf{T}_k(\mathbf{L})$$

$$\mathbf{T}_k(x) = x\mathbf{T}_{k-1}(x) - \mathbf{T}_{k-2}(x)$$

$$\mathbf{T}_0 = 1 \qquad \mathbf{T}_1 = x$$

[Defferrard et al. 2016]

First-order (fast) variant

$$\mathbf{W} \star_{\mathcal{G}} \mathbf{X} = \mathbf{W}\mathbf{L}\mathbf{X}$$

[Kipf et al. 2016]

Henceforth, we focus on this variant

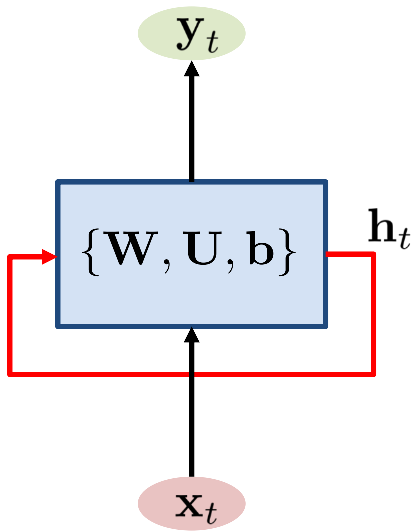
- Michaël Defferrard et al. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems* 2016.
- Thomas N. Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." *International Conference on Learning Representations 2017*.

Recurrent neural nets (RNNs) and variants

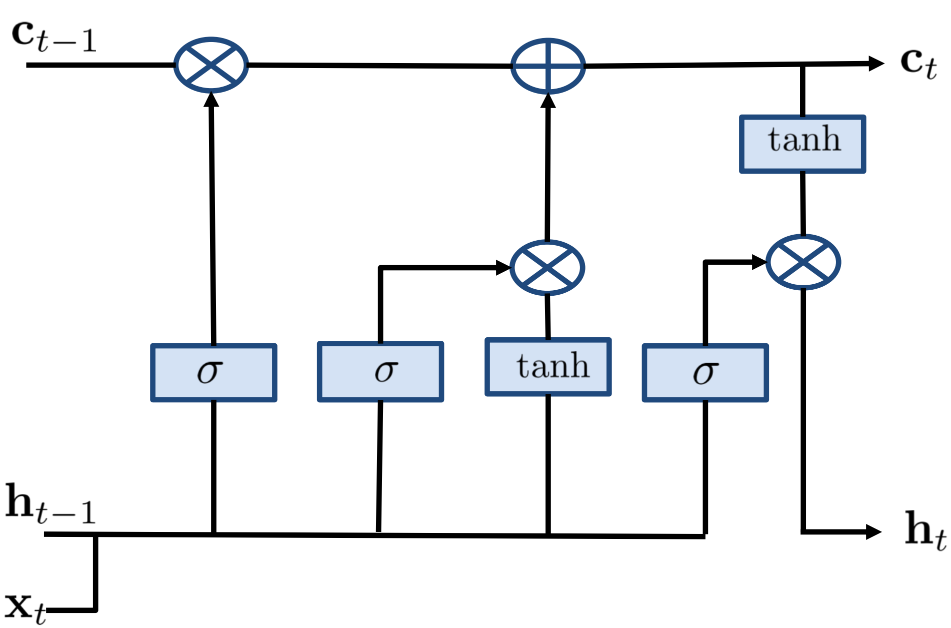
- Standard RNN

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$$

$$\mathbf{y}_t = \sigma(\mathbf{V}\mathbf{h}_t + \mathbf{z})$$



- Long short term memory (LSTM)



$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$

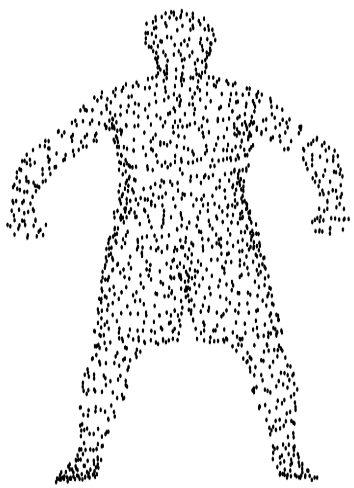
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

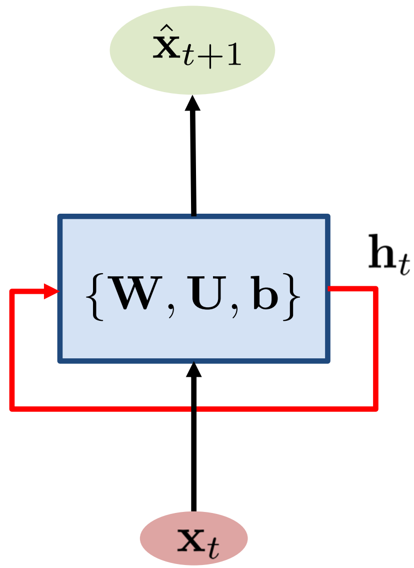
$$\mathbf{h}_t = \mathbf{o}_t \odot \sigma(\mathbf{c}_t)$$

Graph recurrent neural nets (GCRN)

- When the data is defined on a graph, the multiplications in standard RNN are replaced with graph convolutions.



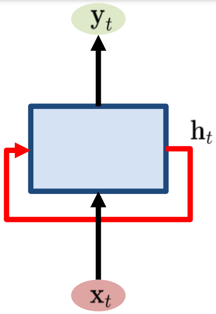
Dynamic 3D point cloud



$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{L}\mathbf{x}_t + \mathbf{U}\mathbf{L}\mathbf{h}_{t-1} + \mathbf{b})$$
$$\hat{\mathbf{x}}_{t+1} = \sigma(\mathbf{V}\mathbf{L}\mathbf{h}_t + \mathbf{z})$$

- At each time step, the prediction loss function is given by $J_t(\mathbf{x}_t, \hat{\mathbf{x}}_{t+1}, \boldsymbol{\theta})$
  - $\boldsymbol{\theta}$ is the set of all trainable parameters
- Loss after T time steps is given by $J(\mathbf{X}, \boldsymbol{\theta}) = \sum_{t=1}^{T} J_t(\mathbf{x}_t, \hat{\mathbf{x}}_{t+1}, \boldsymbol{\theta})$

Gradient issues with standard GCRNN



$$\mathbf{h}_t = \sigma(w\mathbf{L}\mathbf{x}_t + u\mathbf{L}\mathbf{h}_{t-1} + \mathbf{b})$$

- The gradient of the loss function J w.r.t. the tuning parameters

$$\frac{\partial J}{\partial w} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial \mathbf{h}_t} \prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \frac{\partial \mathbf{h}_1}{\partial w} \qquad \frac{\partial J}{\partial u} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial \mathbf{h}_t} \prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \frac{\partial \mathbf{h}_1}{\partial u} \qquad \frac{\partial J}{\partial \mathbf{b}} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial \mathbf{h}_t} \prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \frac{\partial \mathbf{h}_1}{\partial \mathbf{b}}$$

$$\prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \prod_{t=2}^{T} (u\mathbf{D}_t\mathbf{L})$$

$$\mathbf{D}_t = \mathrm{diag}(\sigma'(w\mathbf{L}\mathbf{x}_t + u\mathbf{L}\mathbf{h}_{t-1} + \mathbf{b}))$$

When we choose $\sigma(\cdot) = \texttt{relu}$, then $\mathbf{D}_t$ is an Identity matrix

$$\prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = (u\mathbf{L})^{T-2}$$

- If the largest eigenvalue of $(u\mathbf{L})$ is sufficiently small (i.e., < 1) the gradient will shrink exponentially

- If it is large, the gradient will explode

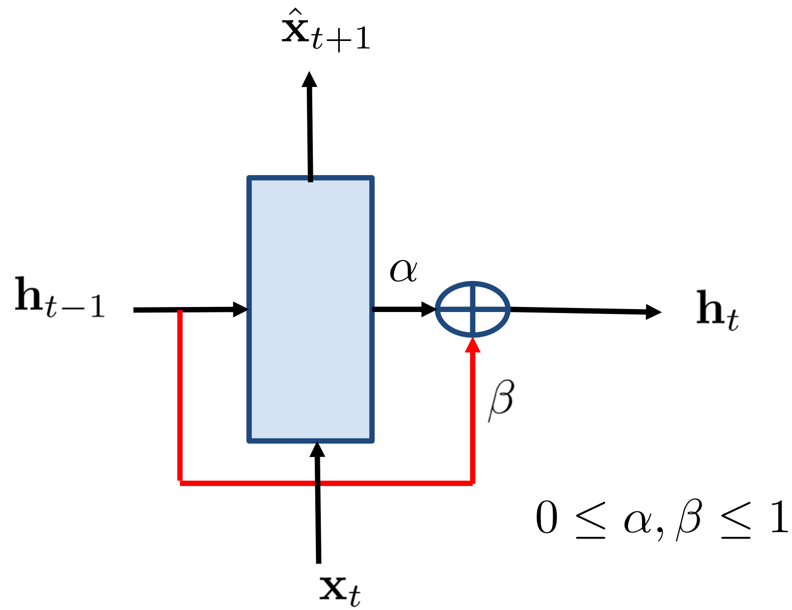
Residual connection

- To stabilize the gradients, a simple *weighted residual connection* maybe introduced.
  - A bit similar to *leaky LMS*

$$\tilde{\mathbf{h}}_t = \sigma(w\mathbf{L}\mathbf{x}_t + u\mathbf{L}\mathbf{h}_t + \mathbf{b}_1)$$

$$\mathbf{h}_t = \alpha\tilde{\mathbf{h}}_t + \beta\mathbf{h}_{t-1}$$

$$\hat{\mathbf{x}}_{t+1} = \sigma(\mathbf{V}\mathbf{L}\mathbf{h}_t + \mathbf{z})$$



- α and β are the two additional trainable parameters
- $\alpha = 1$ and $\beta = 0$ corresponds to the standard GRNN

• A. Kusupati et al., "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in Proc. of the Advances in Neural Information Processing Systems (NIPS), Alberta, Canada, Dec. 2018

Gradients with the residual connection

- The gradient of the loss function w.r.t. w is determined by

$$\prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \prod_{t=2}^{T} (\alpha u \mathbf{D}_t \mathbf{L} + \beta \mathbf{I}) =: \mathbf{M}$$

- The stability of the gradient depends on $\alpha \mathbf{D}_t u \mathbf{L} + \beta \mathbf{I}$, whose condition number is bounded by

$$\mathrm{cond}(\mathbf{M}) \leq \frac{(1 + \frac{\alpha}{\beta} \max_t \|u \mathbf{D}_t \mathbf{L}\|)^{T-2}}{(1 - \frac{\alpha}{\beta} \max_t \|u \mathbf{D}_t \mathbf{L}\|)^{T-2}}$$

- If $\alpha = 0$ and $\beta = 1$, this number is 1, which implies that the gradient is stable, but ignores data/training.

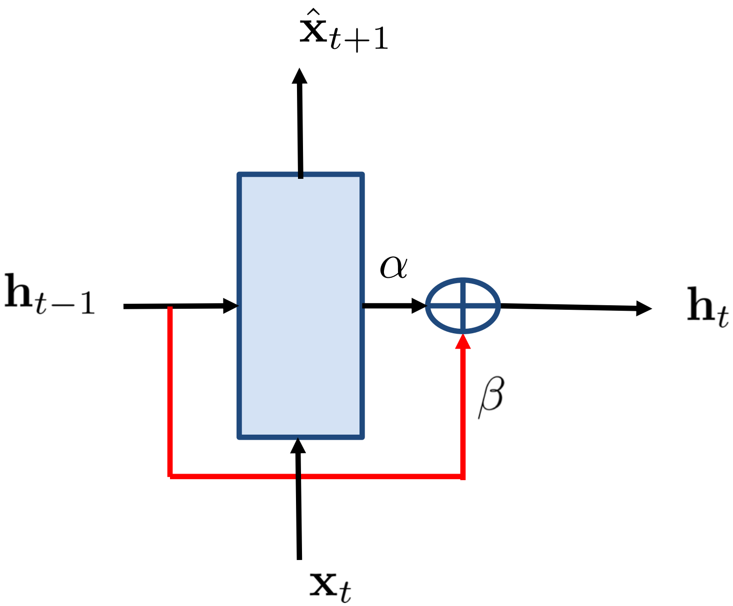
- A. Kusupati et al., “Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network,” in Proc. of the Advances in Neural Information Processing Systems (NIPS), Alberta, Canada, Dec. 2018

Fast graph recurrent nets

$$\tilde{\mathbf{h}}_t = \sigma(\mathbf{WLx}_t + \mathbf{ULh}_{t-1} + \mathbf{b})$$

$$\mathbf{h}_t = \alpha\tilde{\mathbf{h}}_t + \beta\mathbf{h}_{t-1}$$

$$\hat{\mathbf{x}}_{t+1} = \sigma(\mathbf{Vh}_t + \mathbf{z})$$



Gradient and condition number:

$$\prod_{t=2}^{T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \prod_{t=2}^{T} (\alpha \mathbf{UD}_t\mathbf{L} + \beta\mathbf{I}) =: \mathbf{M}$$

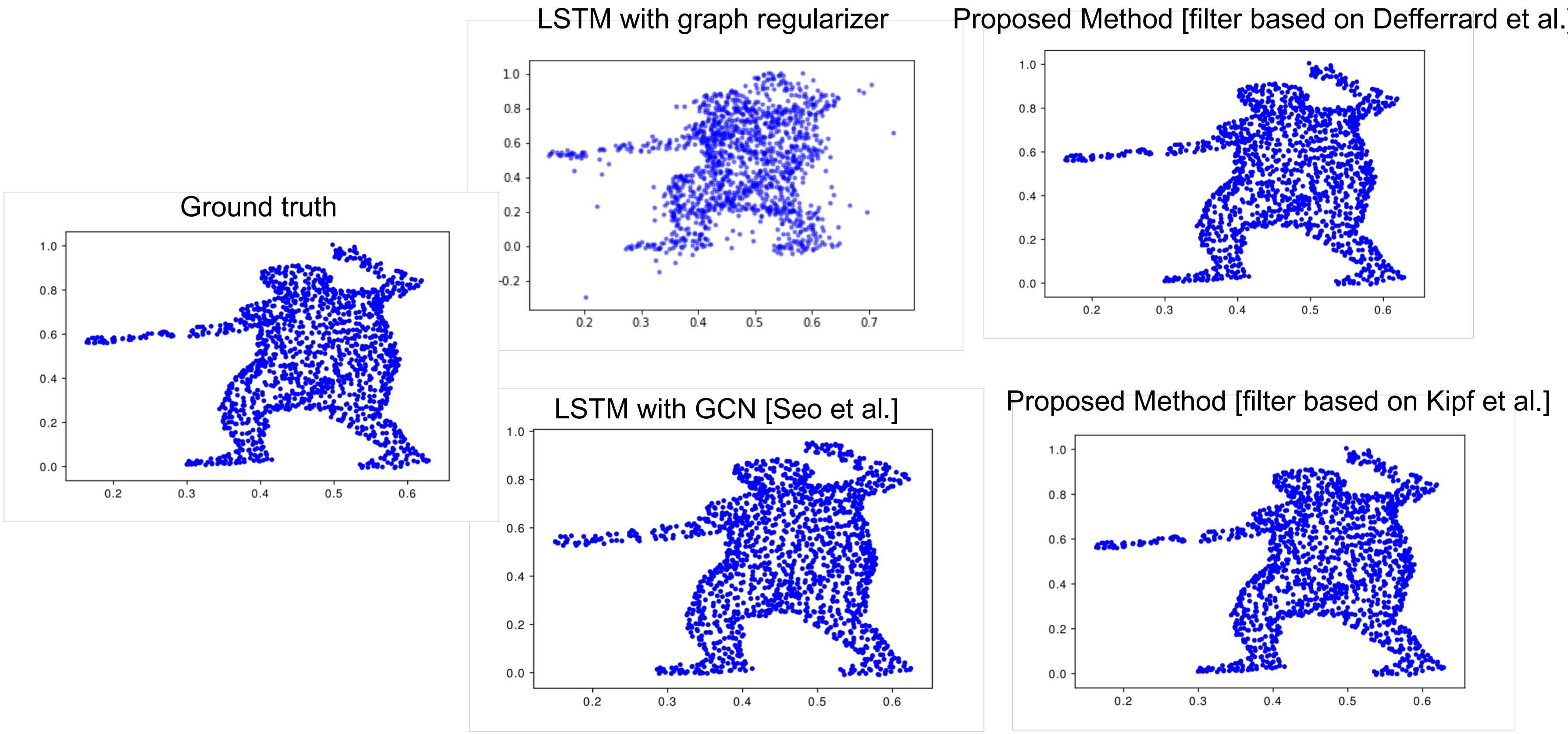
$$\mathrm{cond}(\mathbf{M}) \leq \frac{(1+\frac{\alpha}{\beta}\max_t \|\mathbf{D}_t\mathbf{UL}\|)^{T-2}}{(1-\frac{\alpha}{\beta}\max_t \|\mathbf{D}_t\mathbf{UL}\|)^{T-2}}$$

Remark: For non-graph cases, one may also train for unitary weights (*unitary RNN*)

Numerical results - setup

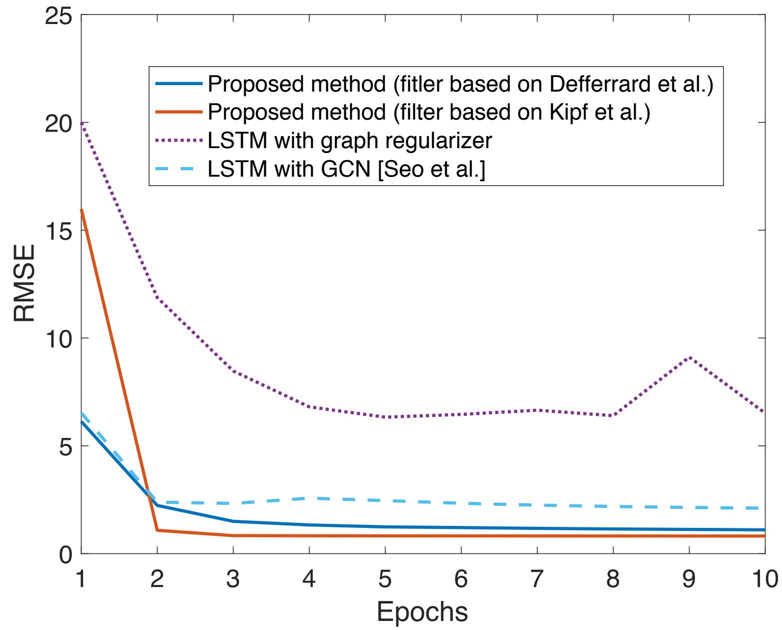
- To evaluate the performance of the proposed method, we use a dynamic 3-D point cloud dataset (a human pose)
- Given a 3-D point cloud frame at a time step T , the task is to predict the next 3-D point cloud frame
- The data has 1502 3D points and 573 time frames
- We use 80% of data available to train the model and 20% to test the model
- Training data is used to construct a nearest neighbour graph
- The learning rate is initialized to 10^{-2} and we use ADAM optimizer for training

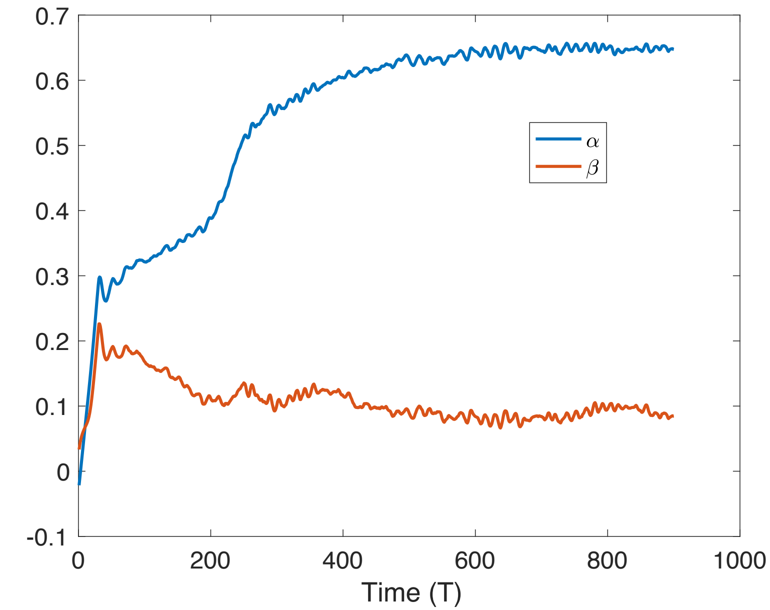
Numerical results



- Thomas N. Kipf, and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
- Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems*. 2016.
- Youngjoo Seo et al. "Structured sequence modeling with graph convolutional recurrent networks." *International Conference on Neural Information Processing*. Springer, Cham, 2018.

Numerical results

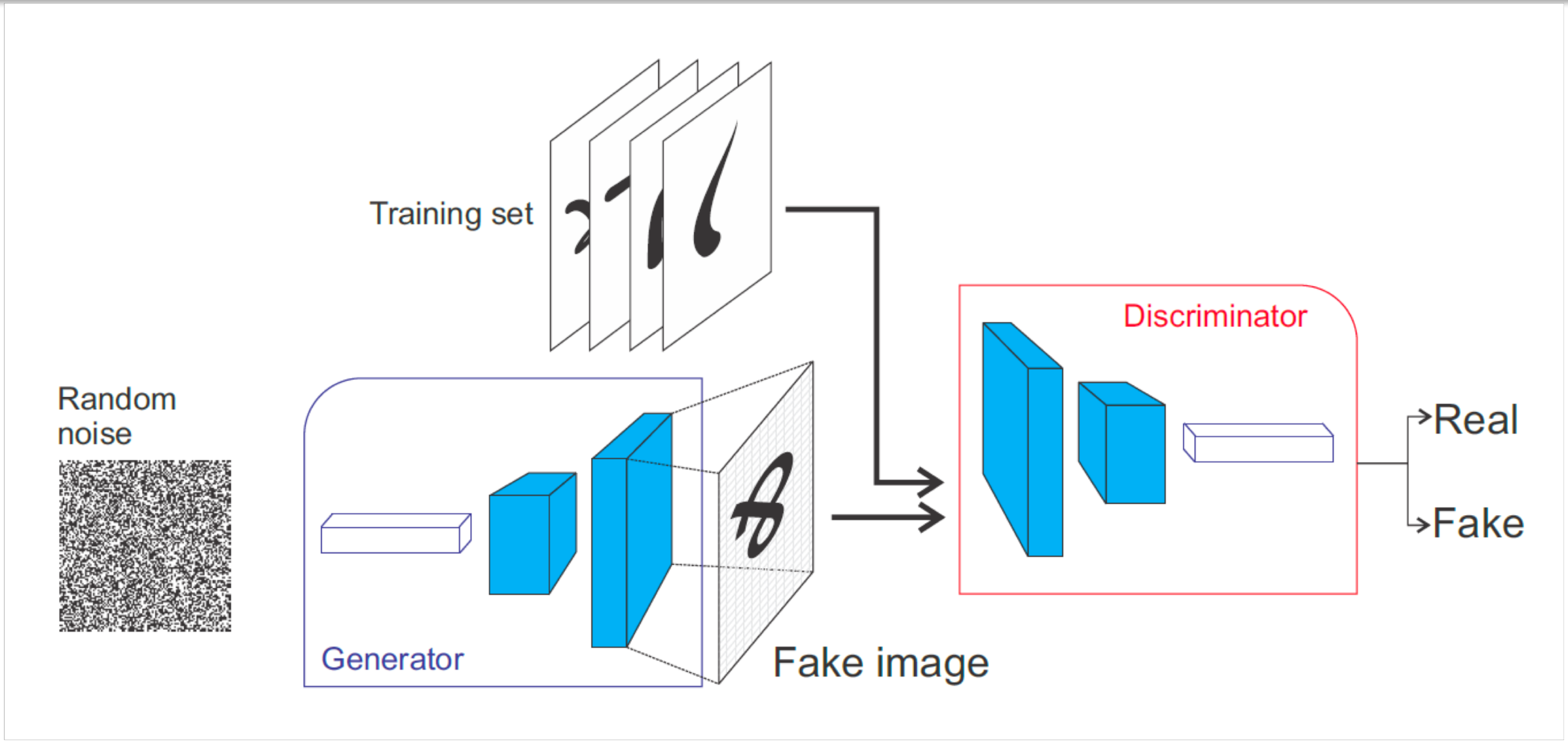




	# parameters	3D point cloud
LSTM with GCN	$4Fp + 4p^2 + 4n$	6080
Proposed	$2Fp + p^2 + 2n + 2$	**3003**

- Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
- Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems*. 2016.
- Youngjoo Seo et al. "Structured sequence modeling with graph convolutional recurrent networks." *International Conference on Neural Information Processing*. Springer, 2018.

GANs

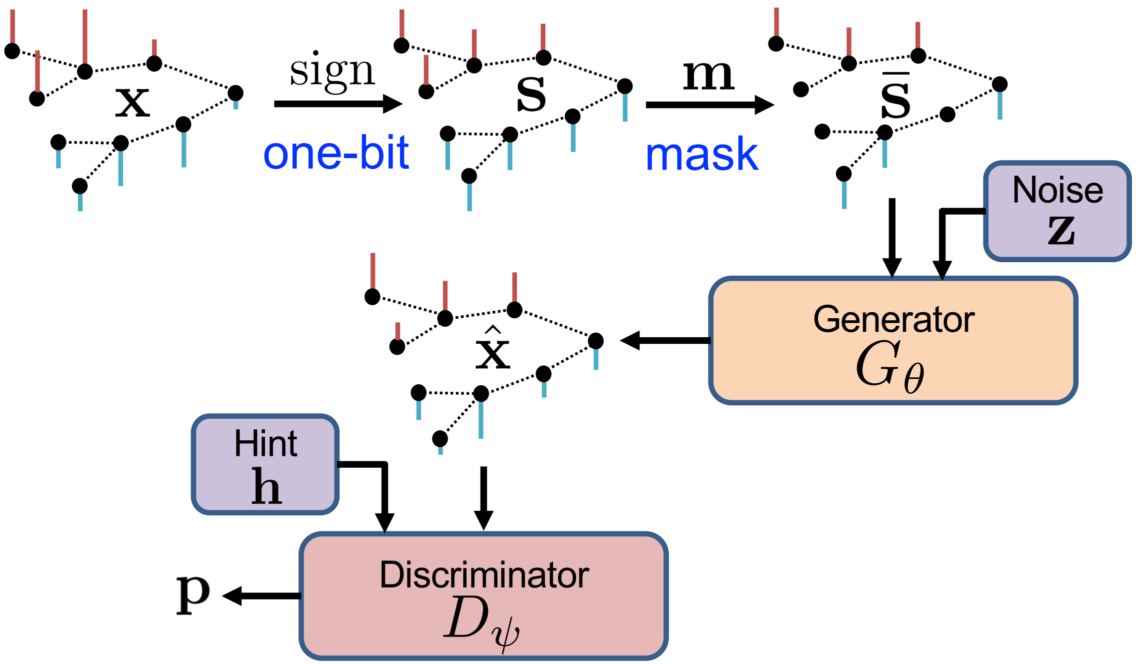


Generative adversarial nets

https://thispersondoesnotexist.com

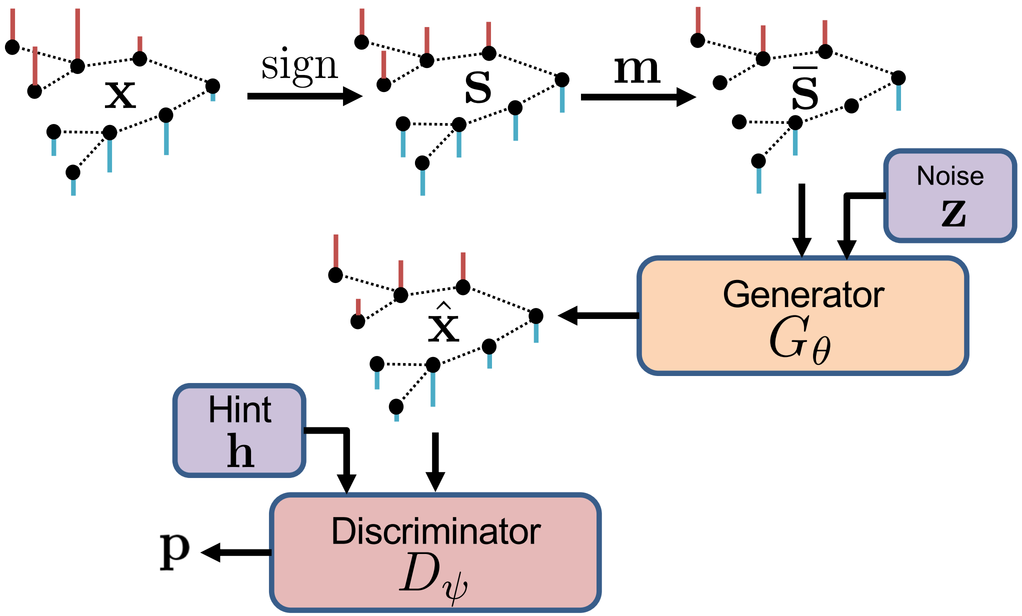
Graph GANs

- Given one-bit quantized data we want to reconstruct the original signal
- This is also referred to PU learning, where we observe only positive labels (in this case, we use signed measurements)



J. Yoon, J. Jordon, and M. Schaar, “Gain: Missing data imputation using generative adversarial nets,” in *International Conference on Machine Learning*, 2018, pp. 5675–5684.

Graph GANs



Generator: $\hat{\mathbf{x}} = G_\theta(\bar{\mathbf{s}}, (\mathbf{1} - \mathbf{m}) \odot \mathbf{z})$

Discriminator: $p_n = D_\psi(\text{sign}(\hat{\text{x}}), \text{h})$

(estimates the mask matrix)

θ and ψ are network parameters

Discriminator loss: $\mathcal{L}^{D}_{\theta,\psi}(p_n, m_n) = -[m_n \log(p_n) + (1 - m_n)\log(1 - p_n)]$

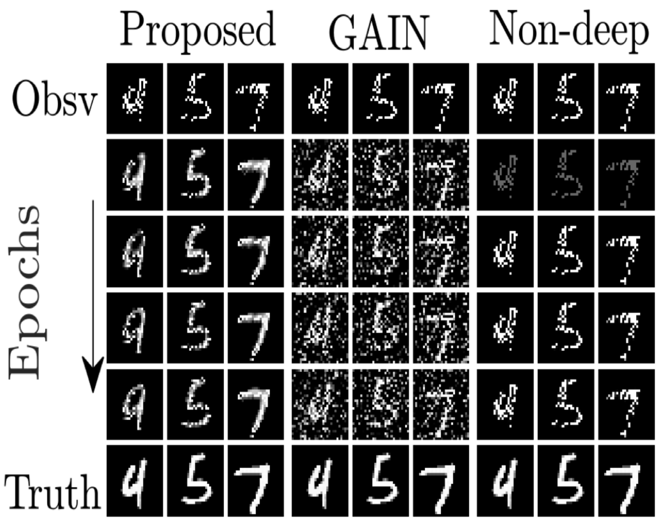
Generator loss: $\mathcal{L}^{G_1}_{\theta,\psi}(p_n, m_n) = -(1 - m_n)\log(p_n)$ min. when D is deceived

$$\mathcal{L}^{G_2}_{\theta}(\bar{\mathbf{s}}, \hat{\mathbf{x}}) = \sum_{i=1}^{N} m_i \left(\bar{s}_i - \text{sign}(\hat{x}_i)\right)^2$$ Consistency with observations

$$\mathcal{L}^{G_3}_{\theta}(\hat{\mathbf{x}}) = \text{TV}^{\ell_2}_{\mathcal{G}}(\hat{\mathbf{x}})$$ Smooth over graph

Solve the min-max problem
$$\min_{\theta}\max_{\psi} \; -\mathcal{L}^{D}_{\theta,\psi}(p_n, m_n) + \mathcal{L}^{G_1}_{\theta,\psi}(p_n, m_n) + \alpha\mathcal{L}^{G_2}_{\theta}(\bar{\mathbf{s}}, \hat{\mathbf{x}}) + \beta\mathcal{L}^{G_3}_{\theta}(\hat{\mathbf{x}}),$$

Graph GANs

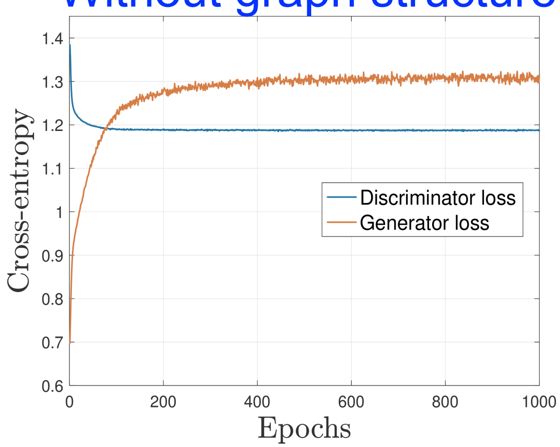


Method	Error
Proposed	**0.36**
GAIN	1.12
Iterative gradient descent	0.49

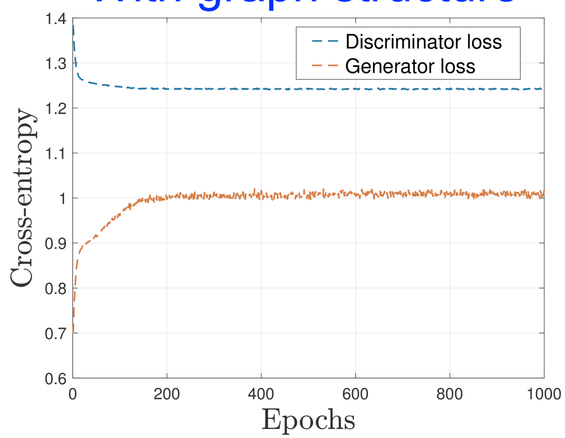
Handwritten MNIST data set
Image size: 28 x 28
Batch size: 384
No. of samples for training: 54192
No. of samples tested: 9984

Non-deep method: Laplacian regularization and gradient descent with sgn(.) approximated with tanh(.)

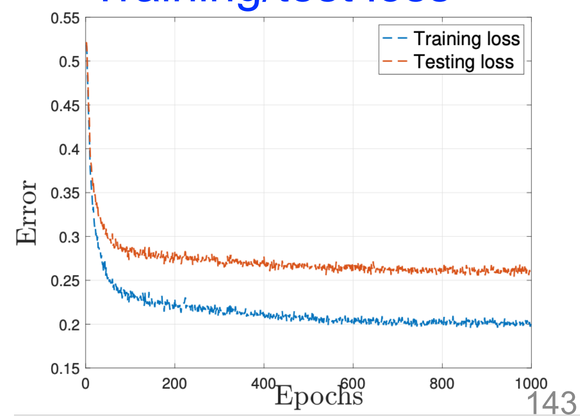
Without graph structure



With graph structure



Training/test loss



Summary

- Introduction to graph signal processing
- Active learning or sampling and recovery of graph signals
- Graph learning or topology inference
- Geometric deep learning (GNNs, RNNs and GANs)

Thank You!

https://ece.iisc.ac.in/~spchepuri/

Kernel-based reconstruction

- Popular within machine learning for nonlinear function estimation

- Kernel methods seek an estimation of a function in a reproducing kernel Hilbert space (RKHS)

$$\mathcal{H} = \left\{ x : x(v) = \sum_{n=1}^{N} \alpha_n k(v, v_n),\ \alpha_n \in \mathbb{R} \right\}$$

basis functions

Kernel map $k : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$

$k(v_n, v_m)$ measures similarity between signal values at v_n and v_m

- Any graph signal can be assumed to be in RKHS

$$\boldsymbol{x} = \boldsymbol{K}\boldsymbol{\alpha}$$

$$[\boldsymbol{K}]_{n,m} = k(v_n, v_m)$$

Kernel-based reconstruction

RKHS inner product of $x(v) = \sum_{n=1}^{N} \alpha_n k(v, v_n)$ and $x'(v) = \sum_{n=1}^{N} \alpha'_n k(v, v_n)$

$$\langle x, x' \rangle_{\mathcal{H}} = \sum_{n=1}^{N} \sum_{n=1}^{N} \alpha_n \alpha'_n k(v_n, v'_n) = \boldsymbol{\alpha}'^T \boldsymbol{K} \boldsymbol{\alpha}$$

RKHS-based function estimator can be used to reconstruct signals

$$\hat{\boldsymbol{x}} = \boldsymbol{K}\boldsymbol{\alpha}$$

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \mathcal{L}(\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{K}\boldsymbol{\alpha}) + \mu \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}$$

promotes smoothness

Or, equivalently

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{H}} \mathcal{L}(\boldsymbol{y}, \boldsymbol{\Phi}\boldsymbol{x}) + \mu \boldsymbol{x}^T \boldsymbol{K}^{\dagger} \boldsymbol{x}$$

$$\boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} = \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{K}^{\dagger} \boldsymbol{K} \boldsymbol{\alpha}$$

$\mathcal{L}(\cdot)$ is a loss function

Kernel-based reconstruction – ridge regression

- Parameterization via *representer theorem*

$$\hat{\boldsymbol{x}} = \boldsymbol{K}\boldsymbol{\alpha} = \boldsymbol{K}\boldsymbol{\Phi}^T\bar{\boldsymbol{\alpha}} \qquad \bar{\boldsymbol{\alpha}} \in \mathbb{R}^K$$

Terms corresponding to unobserved vertices play no role in kernel expansion

$$\hat{\bar{\boldsymbol{\alpha}}} = \arg\min_{\bar{\boldsymbol{\alpha}} \in \mathbb{R}^K} \mathcal{L}(\boldsymbol{y}, \bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}}) + \mu\bar{\boldsymbol{\alpha}}^T\bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}} \qquad \bar{\boldsymbol{K}} = \boldsymbol{\Phi}\boldsymbol{K}\boldsymbol{\Phi}^T$$

- Kernel ridge regression

$$\begin{aligned}\hat{\bar{\boldsymbol{\alpha}}} &= \arg\min_{\bar{\boldsymbol{\alpha}} \in \mathbb{R}^K} \frac{1}{K}\|\boldsymbol{y} - \bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}}\|^2 + \mu\bar{\boldsymbol{\alpha}}^T\bar{\boldsymbol{K}}\bar{\boldsymbol{\alpha}} \\ &= (\bar{\boldsymbol{K}} + \mu K \mathbf{I})^{-1}\boldsymbol{y}\end{aligned}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{K}\boldsymbol{\Phi}^T(\bar{\boldsymbol{K}} + \mu K \mathbf{I})^{-1}\boldsymbol{y}$$

Kernel-based reconstruction

Choice of kernels

- Graph bandlimited kernels

$$\boldsymbol{x} = \boldsymbol{U}_{\text{BL}} \tilde{\boldsymbol{x}}_f$$

- Other topology-based kernel (promotes smooth signal estimates)

$$\boldsymbol{K} = r^{\dagger}(\boldsymbol{L}) = \boldsymbol{U} r^{\dagger}(\boldsymbol{\Lambda}) \boldsymbol{U}^T$$

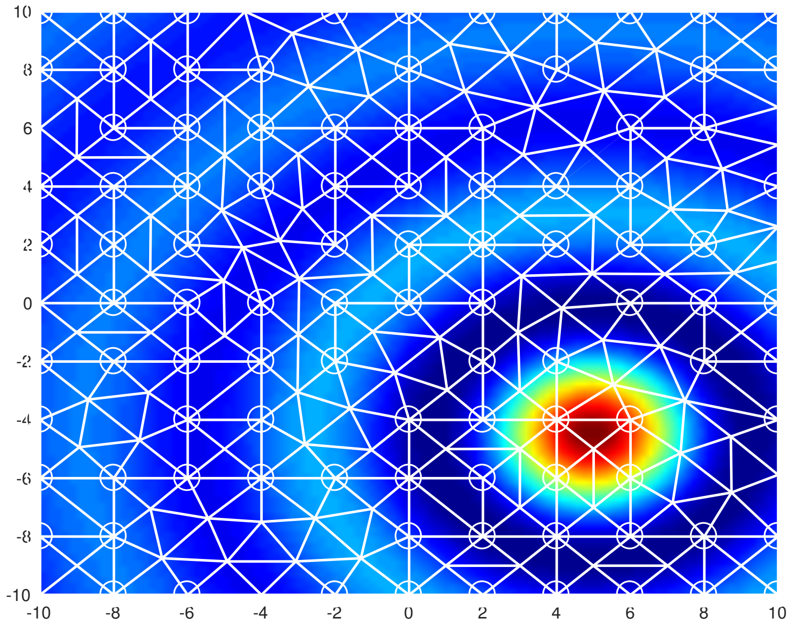
$r : \mathbb{R} \rightarrow \mathbb{R}_+$

Diffusion kernel: $r(\lambda) = exp\{\sigma^2 \lambda / 2\}$

p-step random walk kernel: $r(\lambda) = (a - \lambda)^{-p}, a \geq 2$

Laplacian (regularization) kernel: $r(\lambda) = 1 + \sigma^2 \lambda$

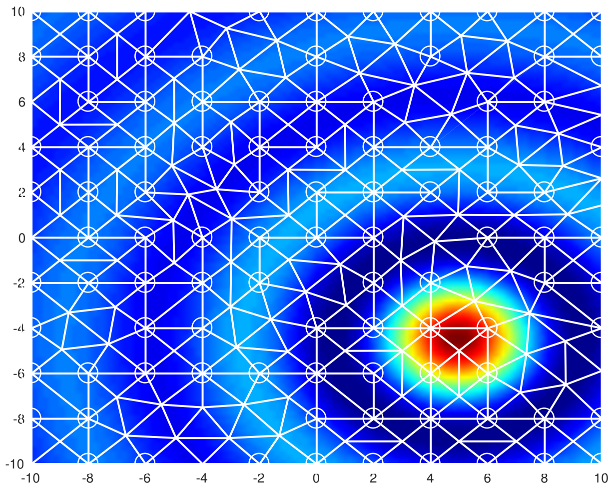
Numerical experiments



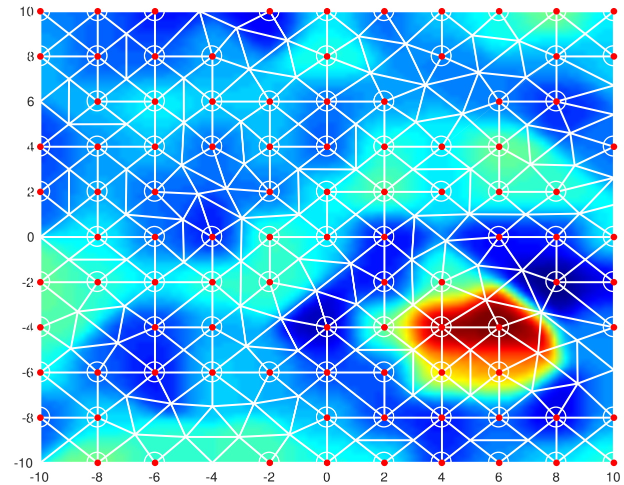
Wave field

- 2-D field estimation
- Rectangular domain of 10×10m
- Source located at coordinates $(x, y) = (5, -4.5)$
- Noise covariance $\boldsymbol{\Sigma} = \text{Toeplitz}\{1, \rho, \ldots, \rho^{N-1}\}$.
- Gaussian radial basis kernel with $\sigma = 0.8$.

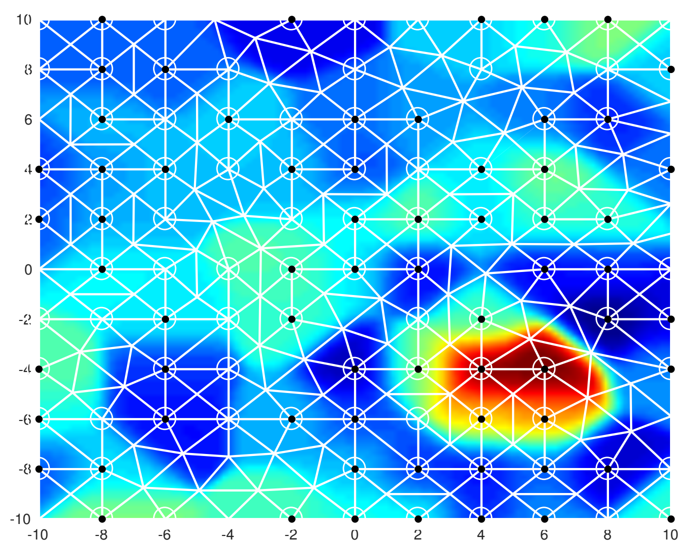
Numerical experiments



Ground truth

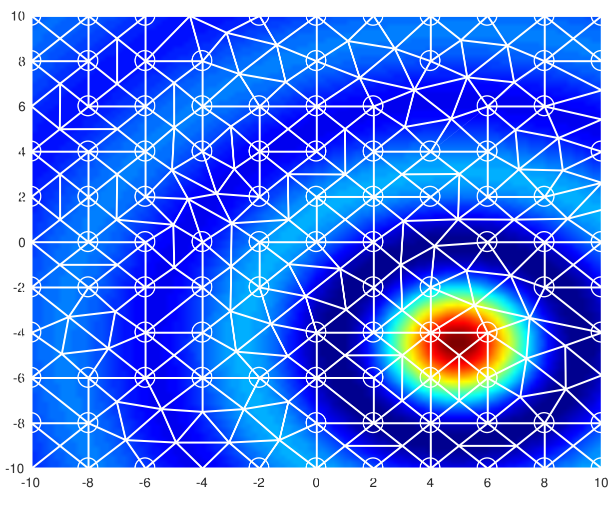


No subsampling (N=97)

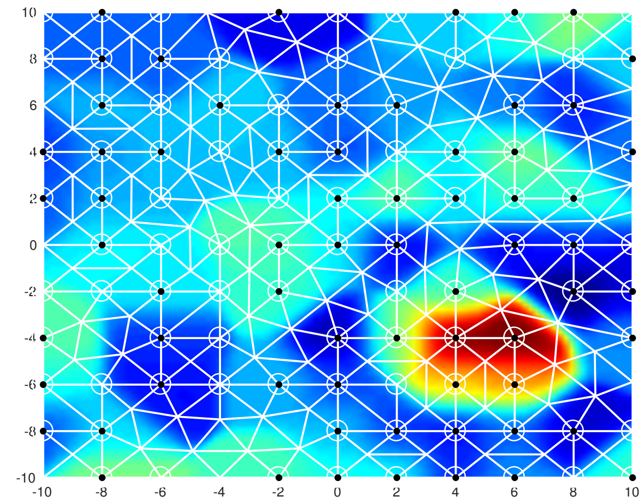


Measured 67 out of 97 mesh points

Sampler design for kernel-based method



Ground truth



Measured 67 out of 97 mesh points

Design of sampling sets for kernel methods

- Submodular optimization
- Convex optimization

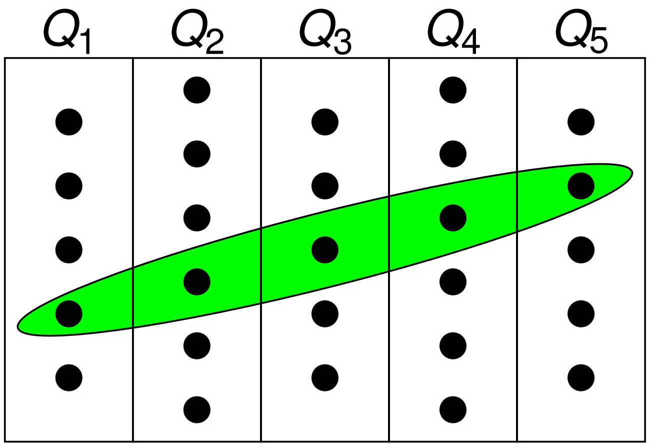
[Coutino-Chepuri-Leus-2018]

- M. Coutino, S.P. Chepuri and G. Leus. Subset Selection for Kernel-based Reconstruction. In Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2018), Calgary, Canada, April 2018.

Matroids

A finite matroid $\mathcal{M}$ is a pair $(\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is a finite set (also called the ground set) and $\mathcal{I}$ is a family of subsets of $\mathcal{N}$ (called the independent sets) that satisfies the following properties:

1. The empty set is independent, i.e., $\emptyset \in \mathcal{I}$.
2. For every $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{N}$, if $\mathcal{Y} \in \mathcal{I}$, then $\mathcal{X} \in \mathcal{I}$.
3. For every $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{N}$ such that $|\mathcal{Y}| > |\mathcal{X}|$ and $\mathcal{X}, \mathcal{Y} \in \mathcal{I}$ there exists one $x \in \mathcal{Y} \setminus \mathcal{X}$ such that $\mathcal{X} \cup \{x\} \in \mathcal{I}$.

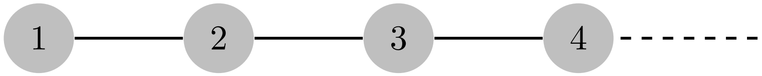


Example: *partition matroid*

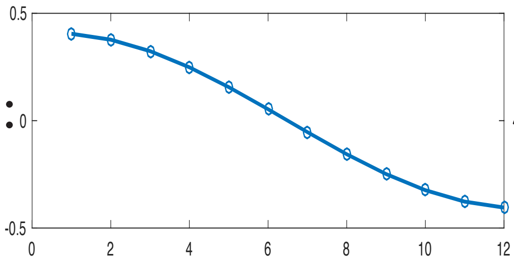
S is independent, if $|S \cap Q_i| \leq 1$ for each Q_i.

Fourier-like basis

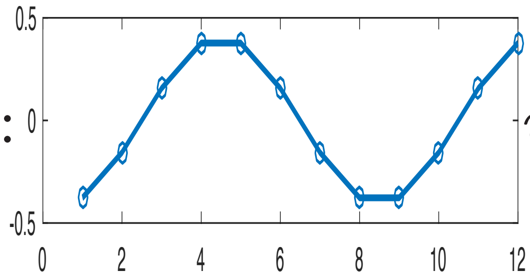
Path graph with 12 nodes



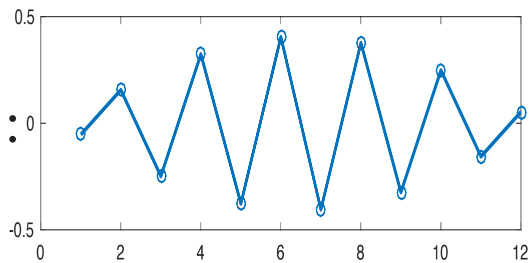
u_2 :



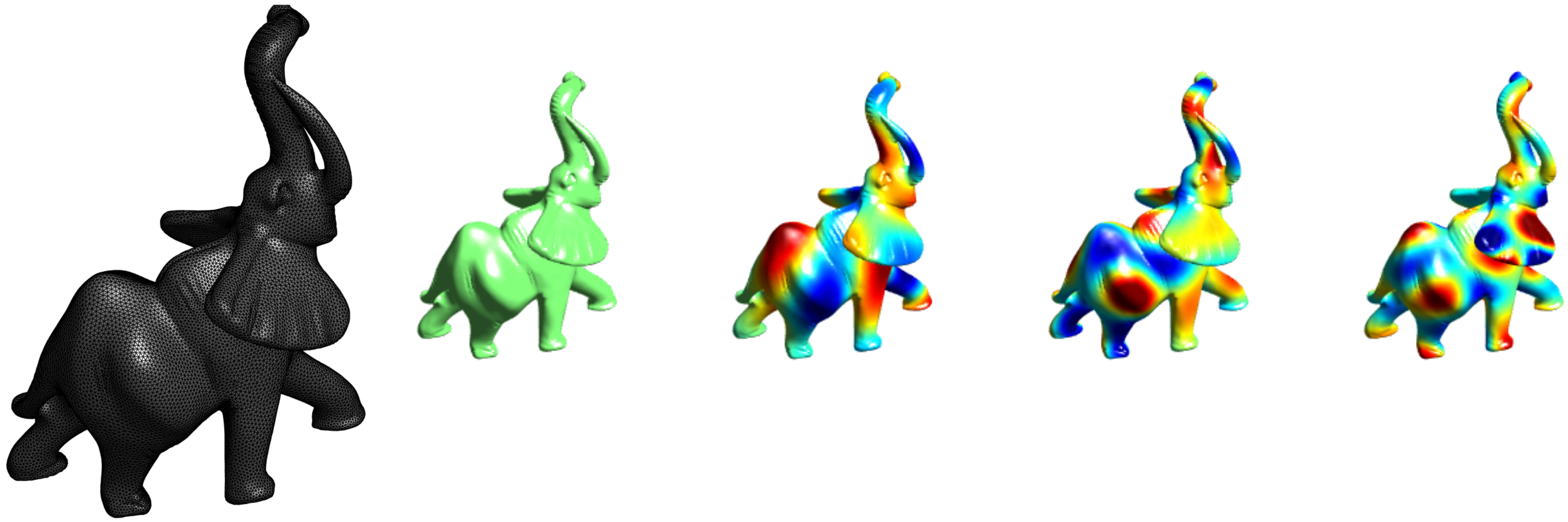
u_4 :



u_{12} :

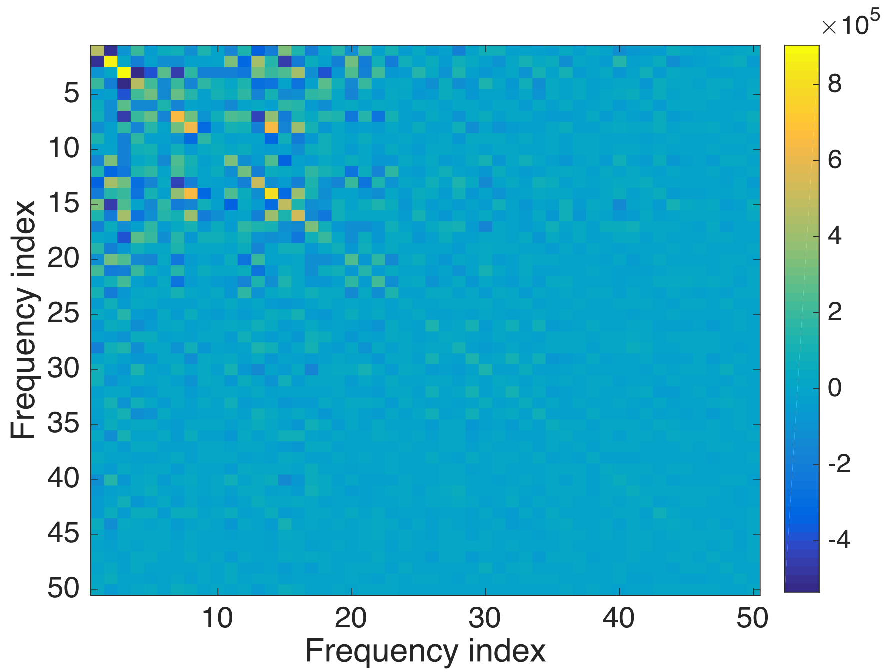


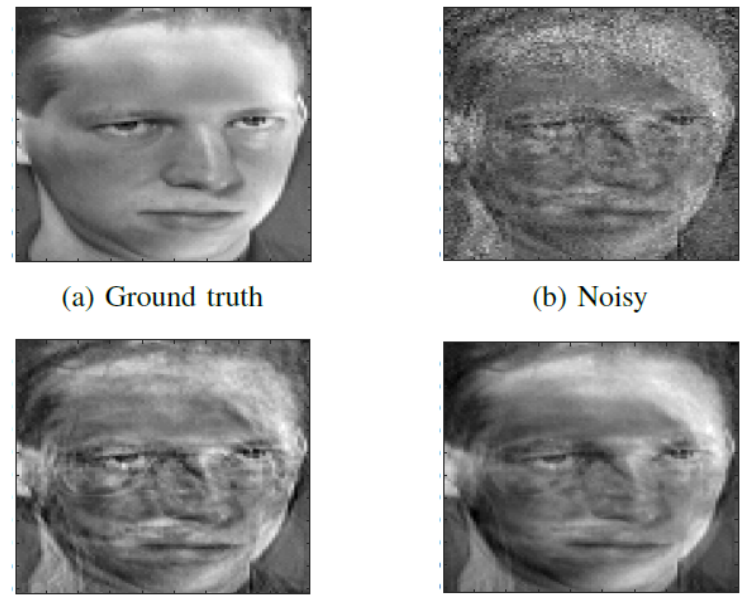
fundamental modes of vibration of a string with free ends



PSD of face images

PSD estimation for spectral signatures of faces of different people





(a) Ground truth
(b) Noisy
(c) Low-pass filter
(d) Wiener filter

- Graph process corresponding to a single individual is stationary in the covariance matrix graph related to multiple individuals

- Estimated PSD can be used for Wiener filtering