AIMS CDT - Signal Processing

Michaelmas Term 2023

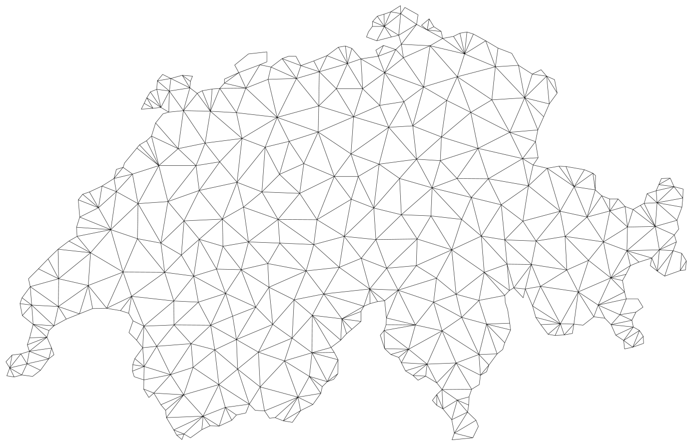
Xiaowen Dong

Department of Engineering Science

UNIVERSITY OF OXFORD

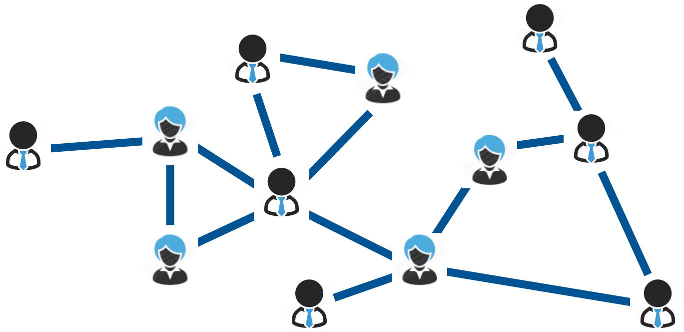
Introduction to
Graphs Signal Processing

Networks are pervasive



geographical network

traffic network

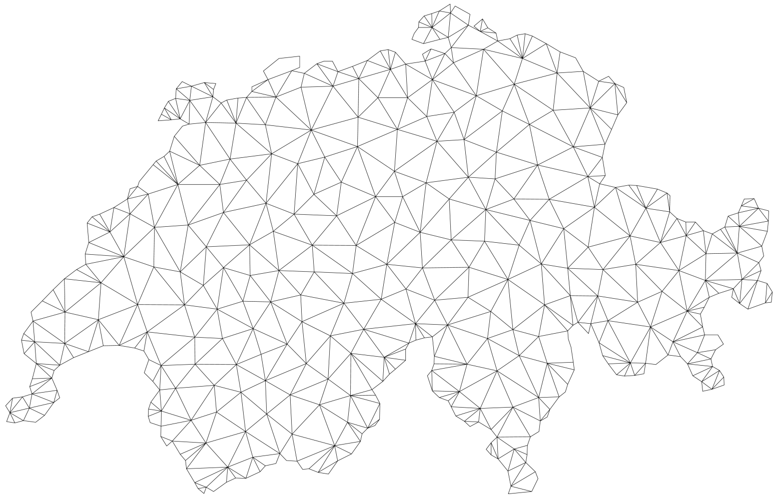


social network

brain network

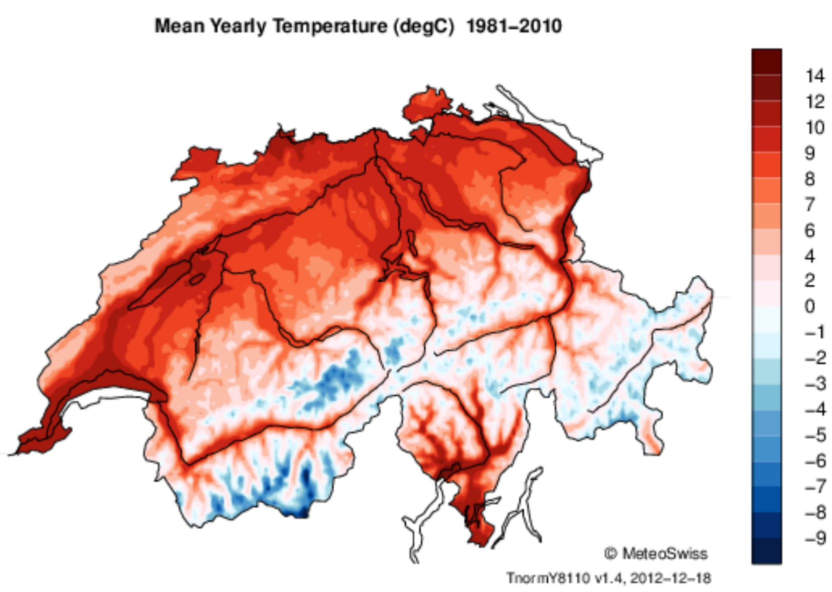
graphs provide mathematical representation of networks

Graph-structured data are pervasive



- vertices
  - geographical regions
- edges
  - geographical proximity between regions

Graph-structured data are pervasive

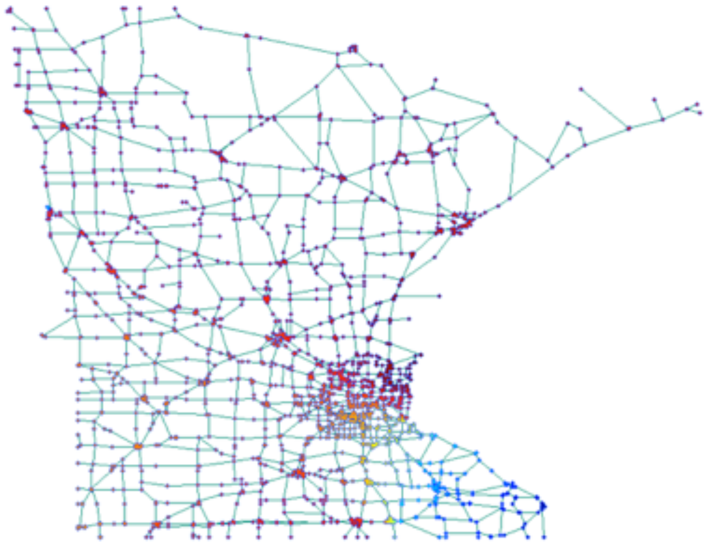


- vertices
  - geographical regions
- edges
  - geographical proximity between regions
- signal
  - temperature records in these regions

Graph-structured data are pervasive

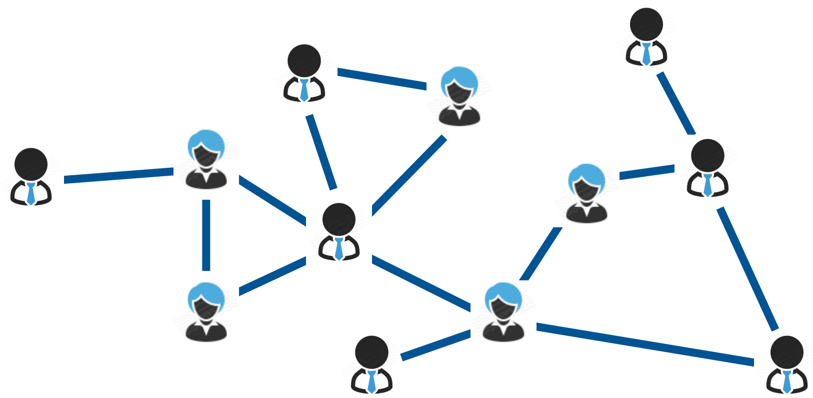
- vertices
  - road junctions
- edges
  - road connections

Graph-structured data are pervasive



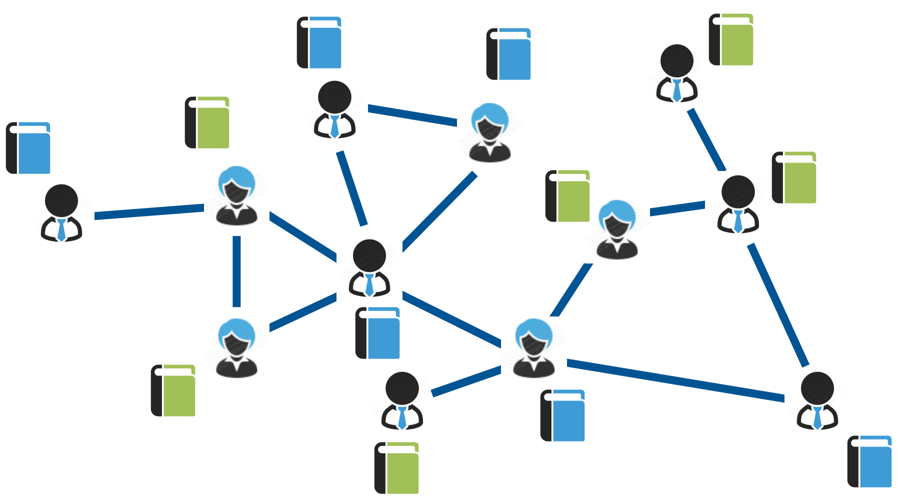
- vertices
  - road junctions
- edges
  - road connections
- signal
  - traffic congestion at junctions

Graph-structured data are pervasive



- vertices
  - individuals
- edges
  - friendship between individuals

Graph-structured data are pervasive

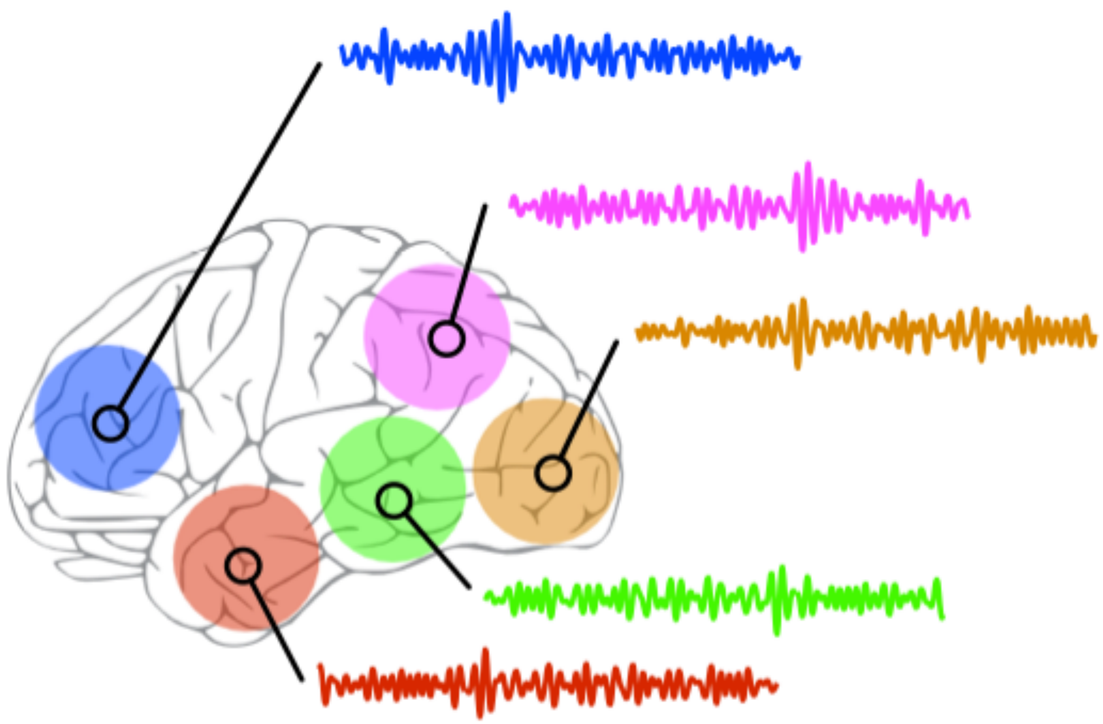


- vertices
  - individuals
- edges
  - friendship between individuals
- signal
  - personal interest

Graph-structured data are pervasive

- vertices
  - brain regions
- edges
  - structural connectivity between brain regions

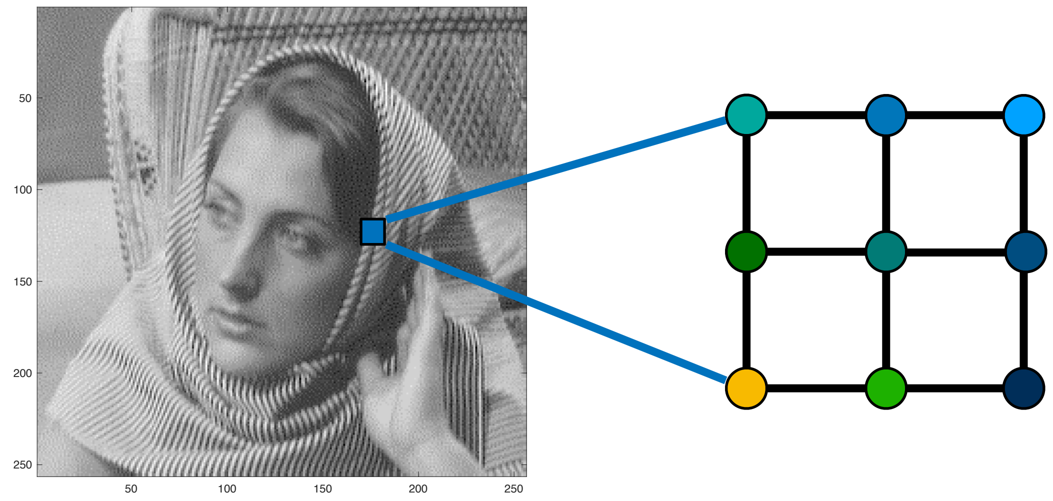
Graph-structured data are pervasive



- vertices
  - brain regions
- edges
  - structural connectivity between brain regions
- signal
  - blood-oxygen-level-dependent (BOLD) time series

Richiardi et al., "Machine learning with brain graphs," IEEE SPM, 2013.

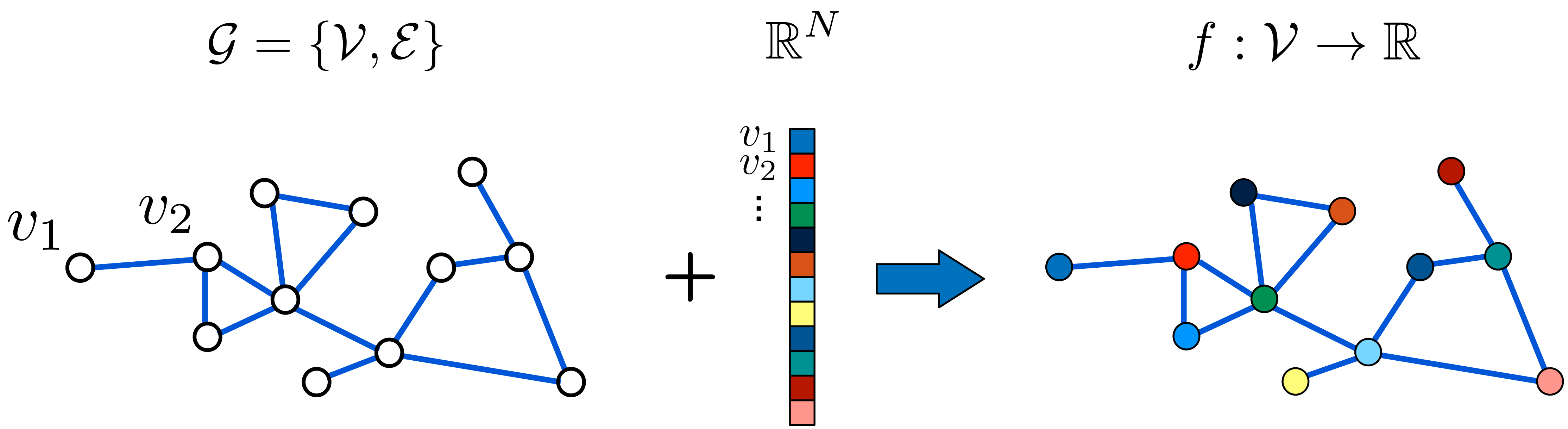
Graph-structured data are everywhere



- nodes
  - pixels
- edges
  - spatial proximity between pixels
- signal
  - pixel values

Graph signal processing

- Graph-structured data can be represented by signals defined on graphs or **graph signals**

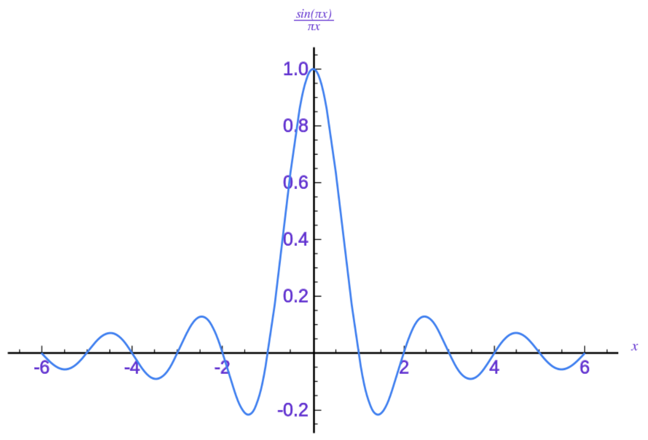


takes into account both structure (edges) and data (values at nodes)

Graph signal processing

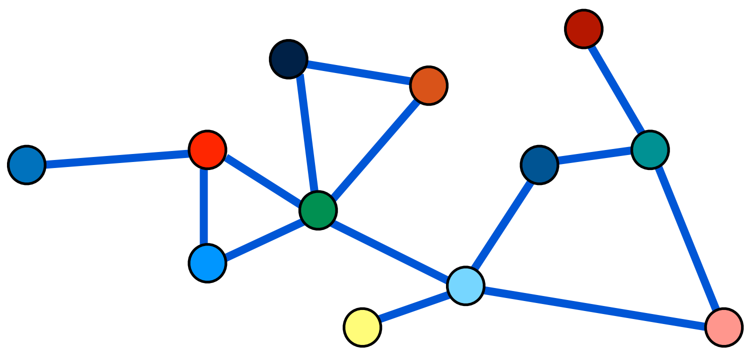
- Graph-structured data can be represented by signals defined on graphs or **graph signals**

1D signal



2D signal

$f : \mathcal{V} \rightarrow \mathbb{R}$



how to generalise classical signal processing tools (e.g. convolution) on irregular domains such as graphs?

Graph signal processing

- Graph signals provide a nice compact format to encode structure within data
- Generalisation of classical signal processing tools can greatly benefit analysis of such data
- Numerous applications: Transportation, biomedical, social, economic network analysis
- An increasingly rich literature
  - classical signal processing
  - algebraic and spectral graph theory
  - computational harmonic analysis
  - machine learning

Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

Two paradigms

- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

Two paradigms

- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

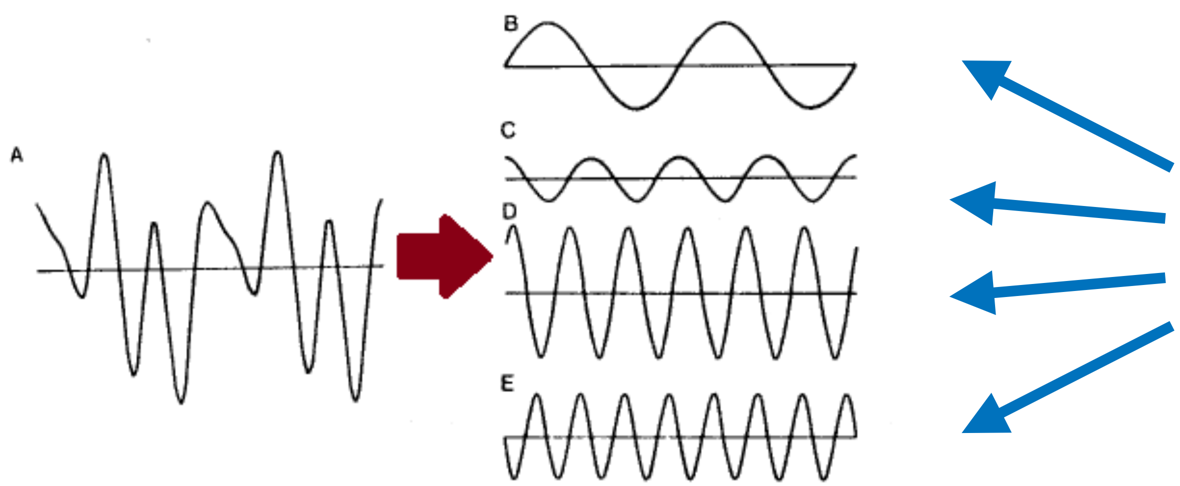
important for signal analysis

Two paradigms

- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

important for signal analysis

- Classical Fourier transform provides frequency domain representation of signals



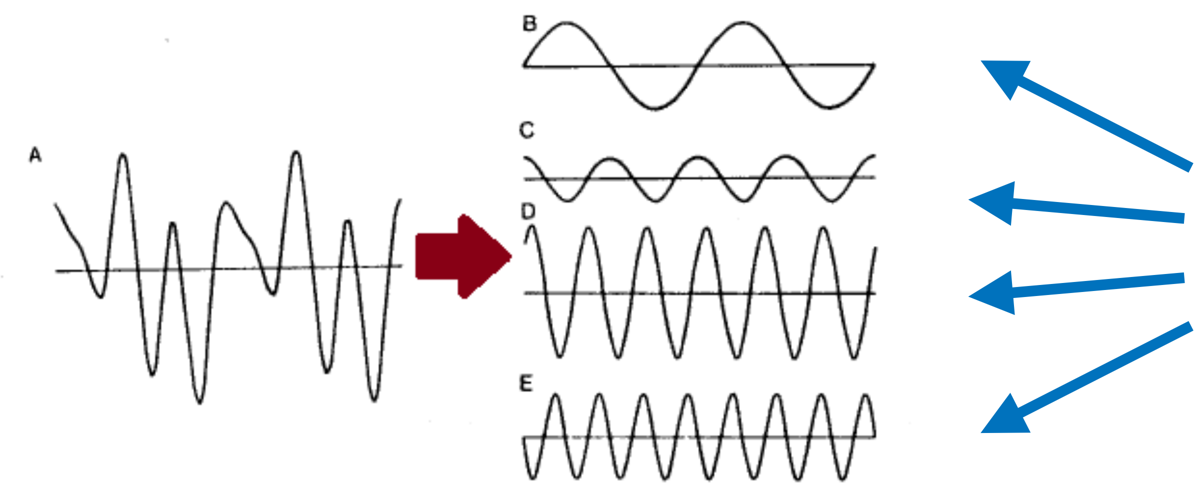
- “building blocks” of signal
- different frequency (oscillation)

Two paradigms

- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

important for signal analysis

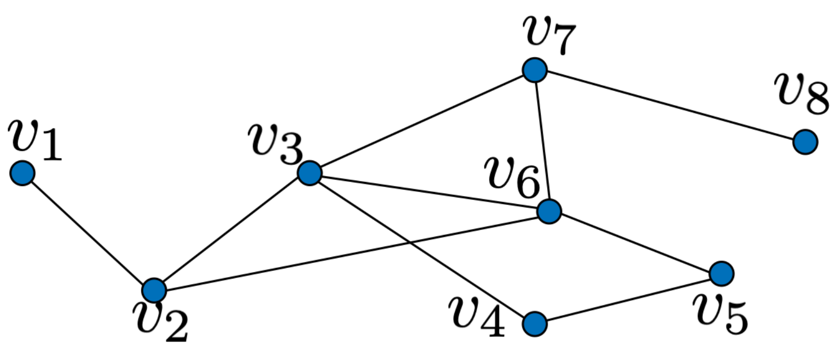
- Classical Fourier transform provides frequency domain representation of signals



- "building blocks" of signal
- different frequency (oscillation)

- What about a notion of frequency for graph signals?

Graph Laplacian



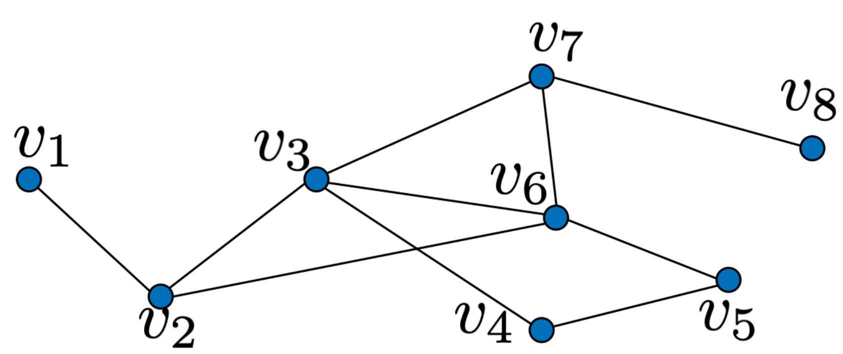
weighted and undirected graph:

$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$

Graph Laplacian

weighted and undirected graph:

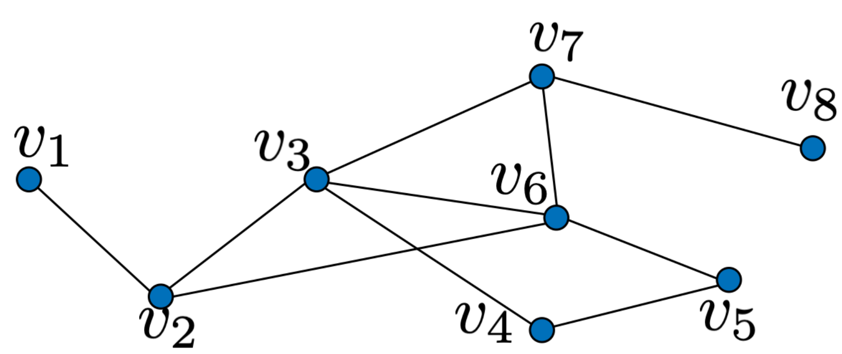
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$W$$

Graph Laplacian



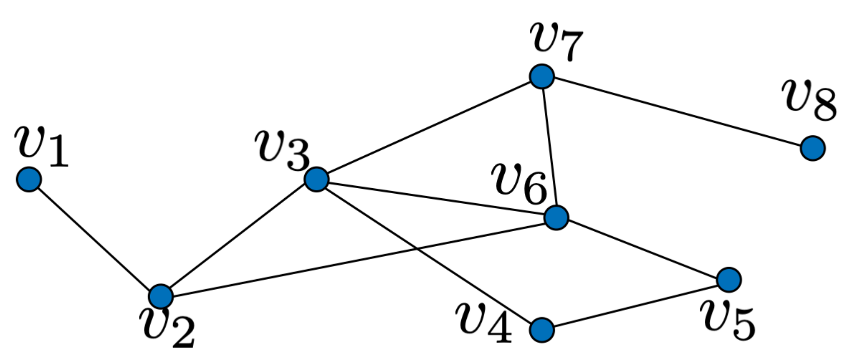
weighted and undirected graph:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

$$D = \mathrm{diag}(d(v_1), \cdots, d(v_N))$$

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad W = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Graph Laplacian



weighted and undirected graph:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

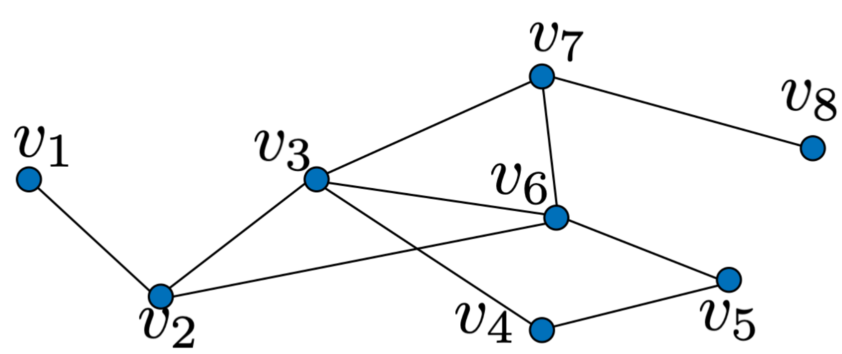
$$D = \mathrm{diag}(d(v_1), \cdots, d(v_N))$$

$$L = D - W$$ **equivalent to W!**

$$\underbrace{\begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&3&0&0&0&0&0&0\\ 0&0&4&0&0&0&0&0\\ 0&0&0&2&0&0&0&0\\ 0&0&0&0&2&0&0&0\\ 0&0&0&0&0&4&0&0\\ 0&0&0&0&0&0&3&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0&1&0&0&0&0&0&0\\ 1&0&1&0&0&1&0&0\\ 0&1&0&1&0&1&1&0\\ 0&0&1&0&1&0&0&0\\ 0&0&0&1&0&1&0&0\\ 0&1&1&0&1&0&1&0\\ 0&0&1&0&0&1&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}}_{W} = \underbrace{\begin{pmatrix} 1&-1&0&0&0&0&0&0\\ -1&3&-1&0&0&-1&0&0\\ 0&-1&4&-1&0&-1&-1&0\\ 0&0&-1&2&-1&0&0&0\\ 0&0&0&-1&2&-1&0&0\\ 0&-1&-1&0&-1&4&-1&0\\ 0&0&-1&0&0&-1&3&-1\\ 0&0&0&0&0&0&-1&1 \end{pmatrix}}_{L}$$

- symmetric
- off-diagonal entries non-positive
- rows sum up to zero

Graph Laplacian



weighted and undirected graph:

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

$$D = \mathrm{diag}(d(v_1), \cdots, d(v_N))$$

$$L = D - W$$ **equivalent to W!**

$$L_{\mathrm{norm}} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$$

$$\begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&3&0&0&0&0&0&0\\ 0&0&4&0&0&0&0&0\\ 0&0&0&2&0&0&0&0\\ 0&0&0&0&2&0&0&0\\ 0&0&0&0&0&4&0&0\\ 0&0&0&0&0&0&3&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} - \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 1&0&1&0&0&1&0&0\\ 0&1&0&1&0&1&1&0\\ 0&0&1&0&1&0&0&0\\ 0&0&0&1&0&1&0&0\\ 0&1&1&0&1&0&1&0\\ 0&0&1&0&0&1&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} = \begin{pmatrix} 1&-1&0&0&0&0&0&0\\ -1&3&-1&0&0&-1&0&0\\ 0&-1&4&-1&0&-1&-1&0\\ 0&0&-1&2&-1&0&0&0\\ 0&0&0&-1&2&-1&0&0\\ 0&-1&-1&0&-1&4&-1&0\\ 0&0&-1&0&0&-1&3&-1\\ 0&0&0&0&0&0&-1&1 \end{pmatrix}$$

D $\qquad$ W $\qquad$ L

- symmetric
- off-diagonal entries non-positive
- rows sum up to zero

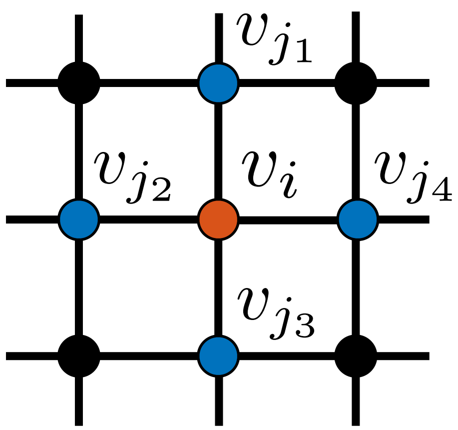
Graph Laplacian

Why graph Laplacian?

Graph Laplacian

Why graph Laplacian?

- provides an approximation of the Laplace operator



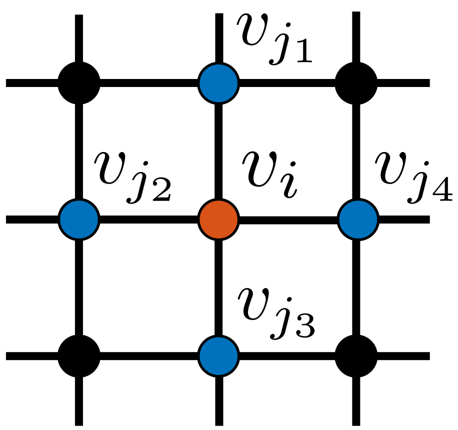
$$(Lf)(i) = \big(4f(i) - f(j_1) - f(j_2) - f(j_3) - f(j_4)\big)/(\delta x)^2$$

standard 5-point stencil for approximating $-\nabla^2 f$

Graph Laplacian

Why graph Laplacian?

- provides an approximation of the Laplace operator

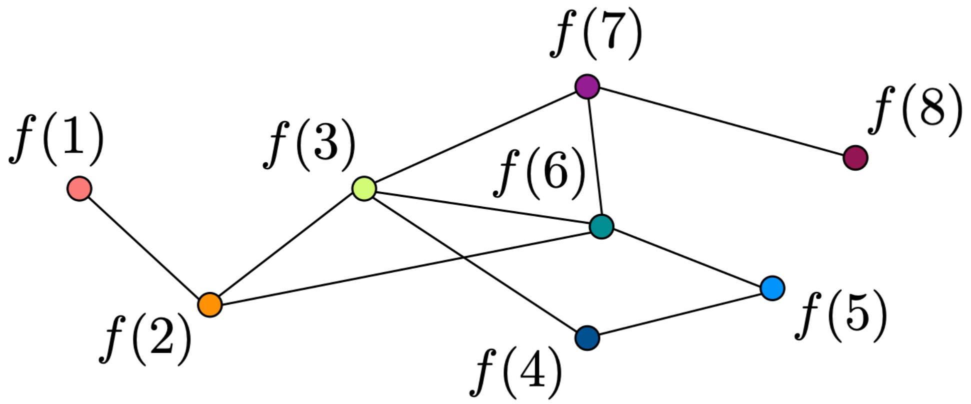


$$(Lf)(i) = \big(4f(i) - f(j_1) - f(j_2) - f(j_3) - f(j_4)\big)/(\delta x)^2$$

standard 5-point stencil for approximating $-\nabla^2 f$

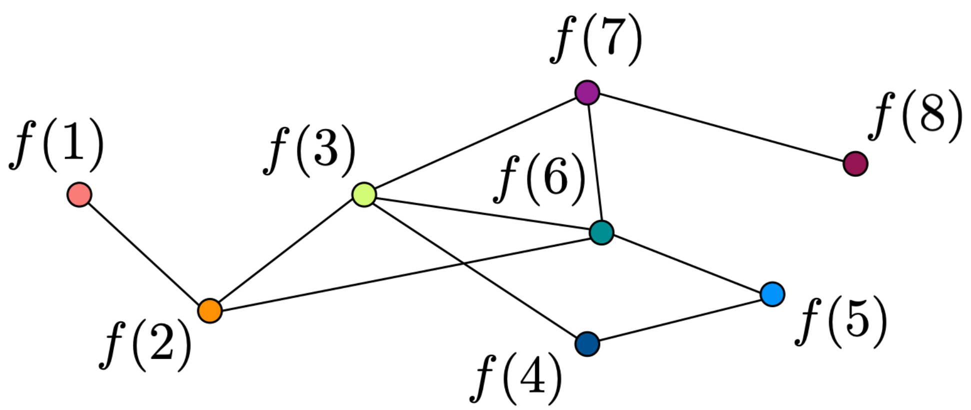
- converges to the Laplace-Beltrami operator (given certain conditions)
- provides a notion of "frequency" on graphs

Graph Laplacian



graph signal $f : \mathcal{V} \to \mathbb{R}$

Graph Laplacian



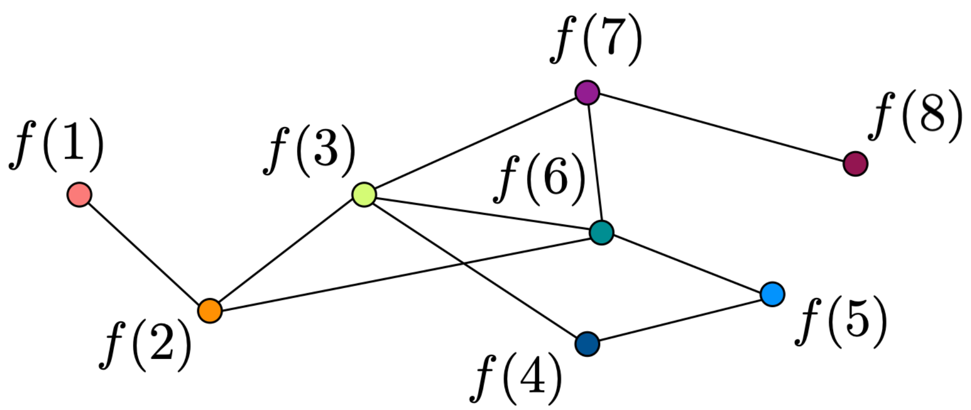
graph signal $f : \mathcal{V} \to \mathbb{R}$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}$$

$$Lf(i) = \sum_{j=1}^{N} W_{ij}(f(i) - f(j))$$

Zhou and Schölkopf, "A regularization framework for learning from graph data," ICML Workshop, 2004.

Graph Laplacian



graph signal $f : \mathcal{V} \to \mathbb{R}$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}$$

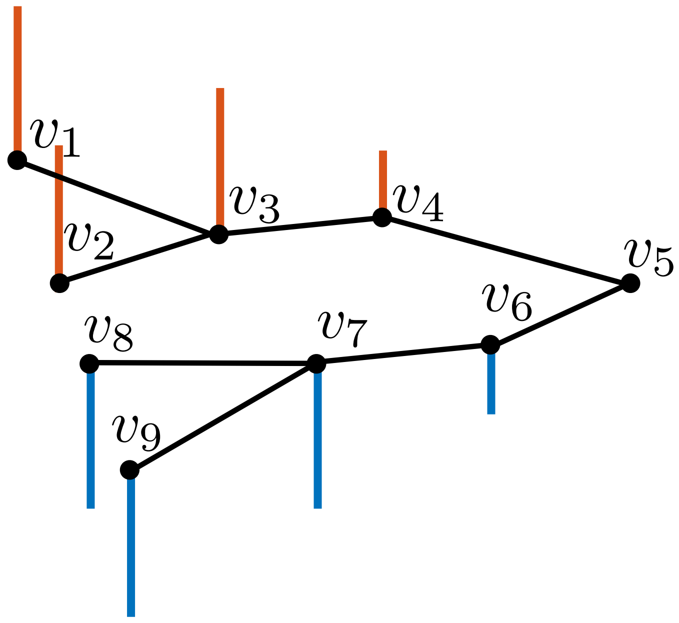
$$Lf(i) = \sum_{j=1}^{N} W_{ij}(f(i) - f(j))$$

$$\begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}^T \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}$$

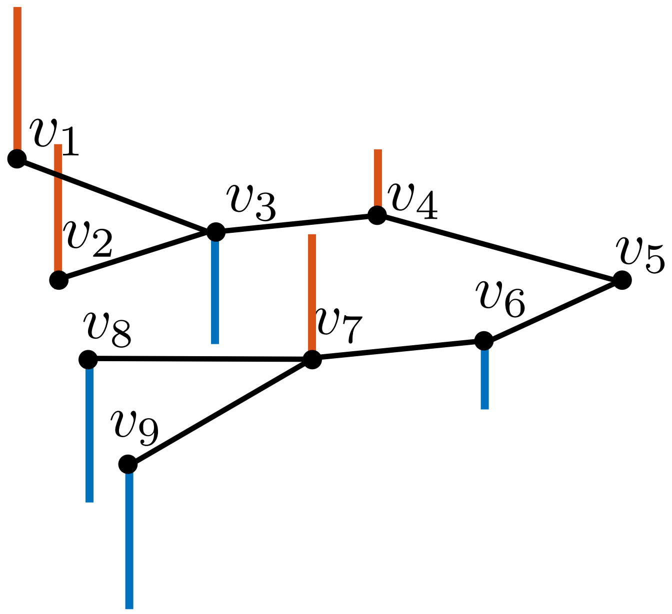
$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{N} W_{ij} \left(f(i) - f(j)\right)^2$$

a measure of "smoothness"

Graph Laplacian



$f^T L f = 1$



$f^T L f = 21$

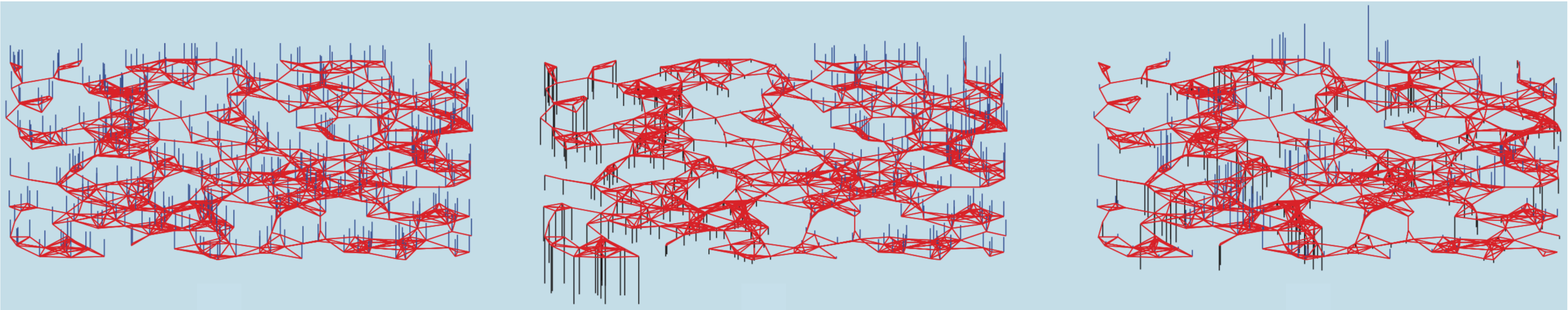
Graph Laplacian

- L has a complete set of orthonormal eigenvectors: $L = \chi \Lambda \chi^T$

$$L = \underbrace{\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}}_{\chi} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} \text{---} & \chi_0^T & \text{---} \\ & \cdots & \\ \text{---} & \chi_{N\text{-}1}^T & \text{---} \end{bmatrix}}_{\chi^T}$$

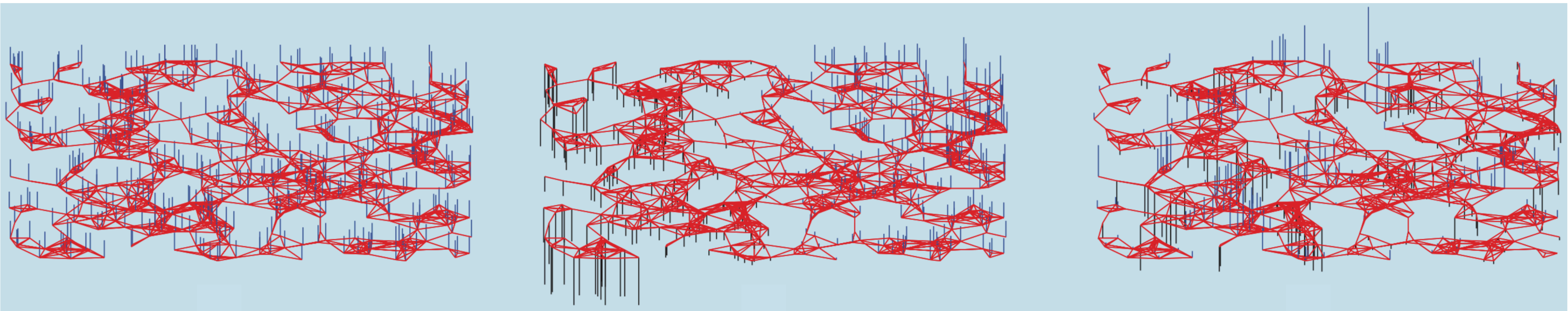
- Eigenvalues are usually sorted increasingly: $0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{N-1}$

Graph Fourier transform



χ_0 χ_1 χ_{50}

Graph Fourier transform



χ_0 χ_1 χ_{50}

low frequency → high frequency

$L = \chi \Lambda \chi^T$

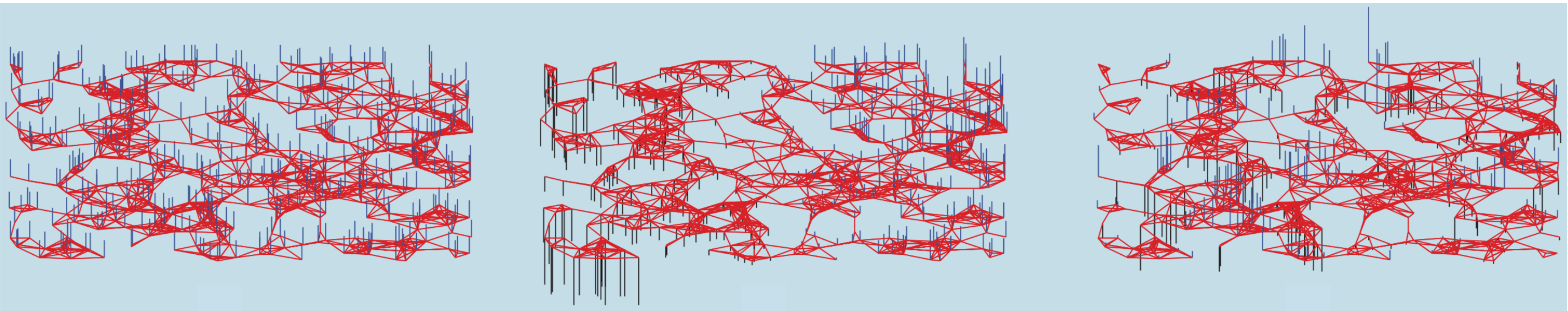
$\chi_0^T L \chi_0 = \lambda_0 = 0$

$\chi_{50}^T L \chi_{50} = \lambda_{50}$

- Eigenvectors associated with smaller eigenvalues have values that vary less rapidly along the edges

Shuman et al., "The emerging field of signal processing on graphs," IEEE SPM, 2013.

Graph Fourier transform



low frequency → high frequency

$L = \chi \Lambda \chi^T$

$\chi_0^T L \chi_0 = \lambda_0 = 0$

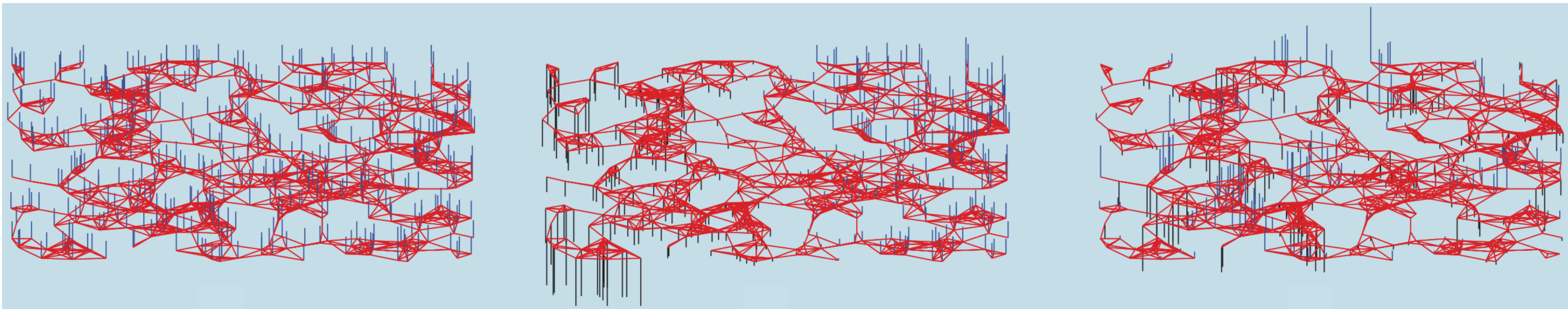
$\chi_{50}^T L \chi_{50} = \lambda_{50}$

graph Fourier transform:

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \quad \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}^T \begin{matrix} | \\ f \\ | \end{matrix}$$

Shuman et al., "The emerging field of signal processing on graphs," IEEE SPM, 2013.

Graph Fourier transform



χ_0 χ_1 χ_{50}

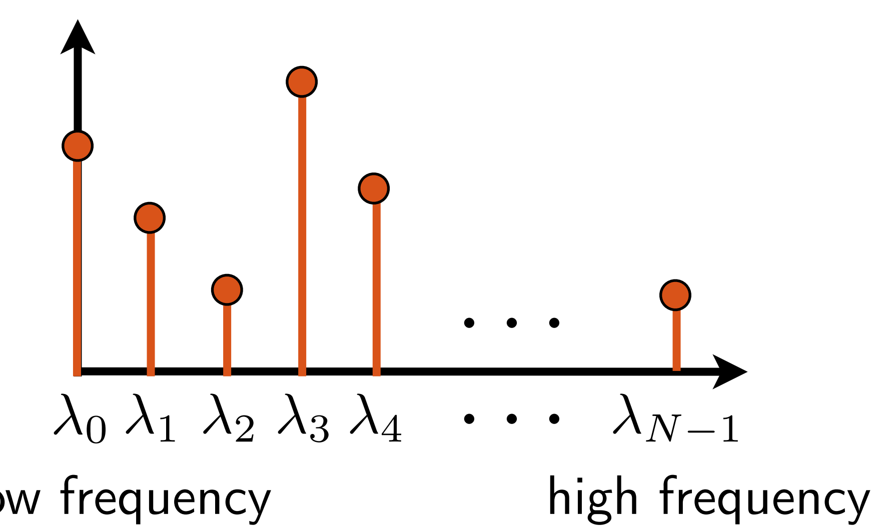
low frequency → high frequency

$$L = \chi \Lambda \chi^T$$

$\chi_0^T L \chi_0 = \lambda_0 = 0$ $\chi_{50}^T L \chi_{50} = \lambda_{50}$

graph Fourier transform:

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \quad \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}^T \begin{matrix} | \\ f \\ | \end{matrix}$$



Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

one-dimensional Laplace operator: $-\nabla^2$

eigenfunctions: $e^{j\omega x}$

Classical FT: $$\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$$

$$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$$

Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

one-dimensional Laplace operator: $-\nabla^2$

eigenfunctions: $e^{j\omega x}$

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$

$$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$$

graph Laplacian: L

eigenvectors: χ_ℓ

$f : V \to \mathbb{R}^N$

Graph FT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$

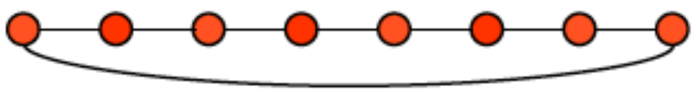
$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

Graph Fourier transform

- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

one-dimensional Laplace operator: $-\nabla^2$	graph Laplacian: L
eigenfunctions: $e^{j\omega x}$	eigenvectors: χ_ℓ
	$f : V \to \mathbb{R}^N$
Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$	Graph FT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$
$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$	$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

Two special cases



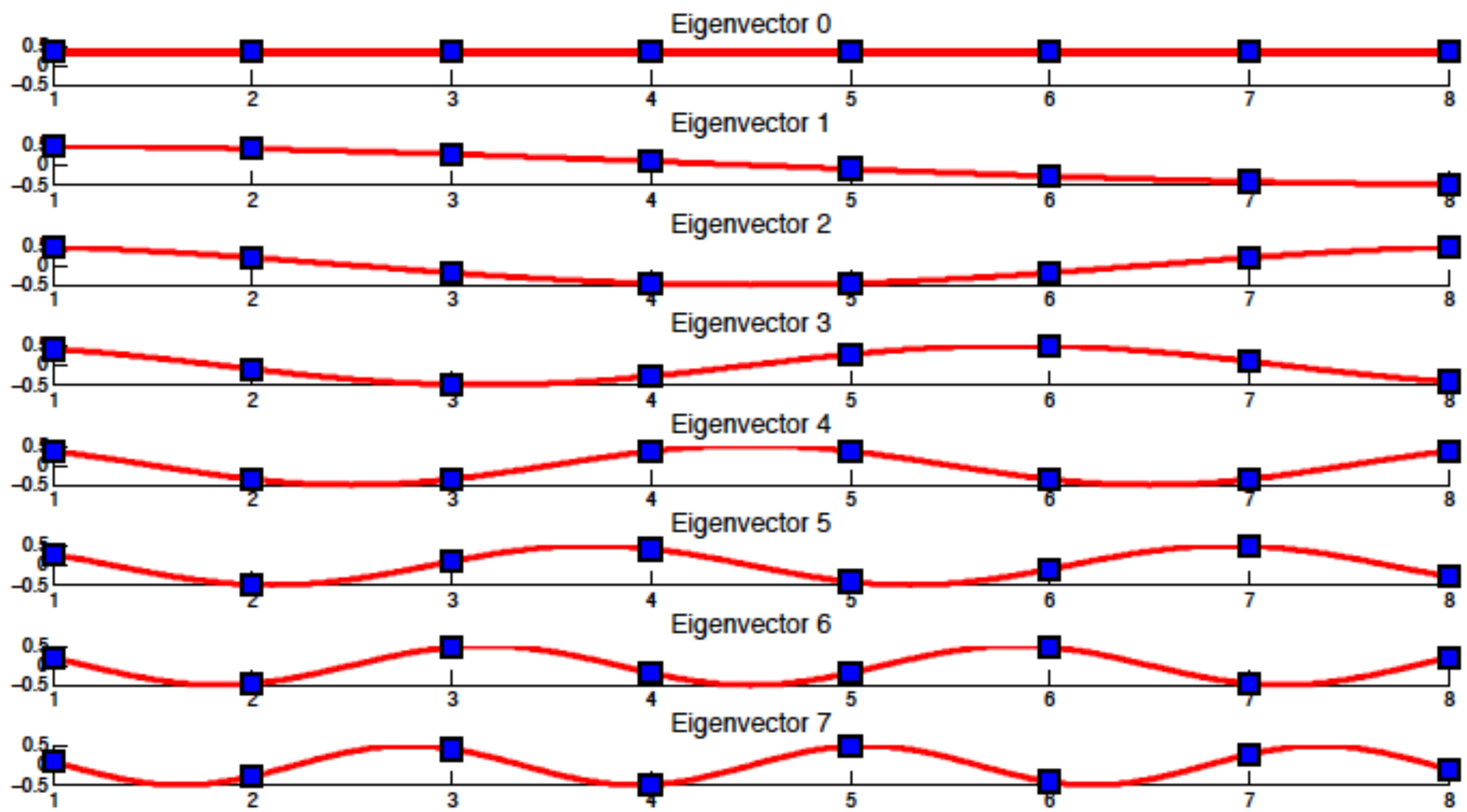
- (Unordered) Laplacian eigenvalues: $\lambda_\ell = 2 - 2\cos\left(\frac{2\ell\pi}{N}\right)$
- One possible choice of orthogonal Laplacian eigenvectors:

$$\chi_\ell = \left[1, \omega^\ell, \omega^{2\ell}, \ldots, \omega^{(N-1)\ell}\right], \text{ where } \omega = e^{\frac{2\pi j}{N}}$$

- $\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$ is the Discrete Fourier Transform (DFT) matrix

Two special cases

- $\lambda_\ell = 2 - 2\cos\left(\frac{\pi\ell}{N}\right)$
- $\chi_0(i) = \frac{1}{\sqrt{N}}$, $\chi_\ell(i) = \sqrt{\frac{2}{N}}\cos\left(\frac{\pi\ell(i-0.5)}{N}\right)$, $\ell = 1, 2, \ldots, N-1$



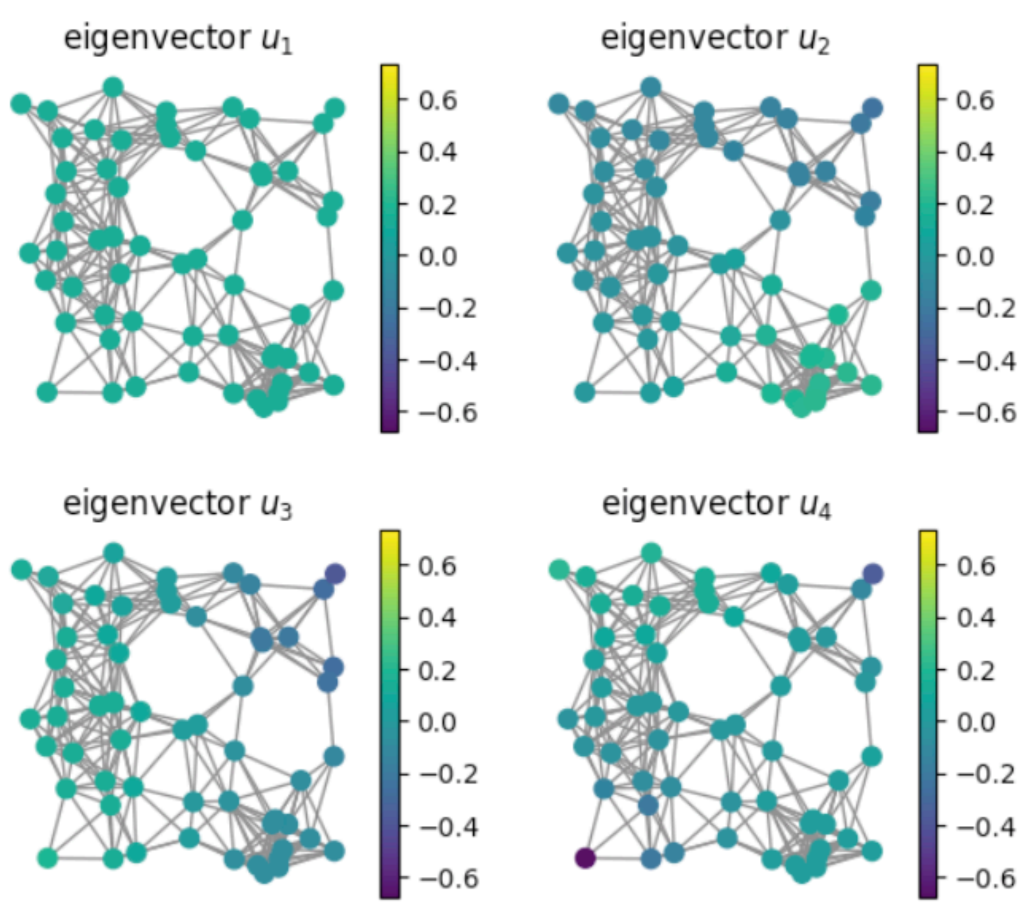
$$\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$$ is the Discrete Cosine Transform matrix (DCT-II, Strang, 1999), which is used in JPEG image compression

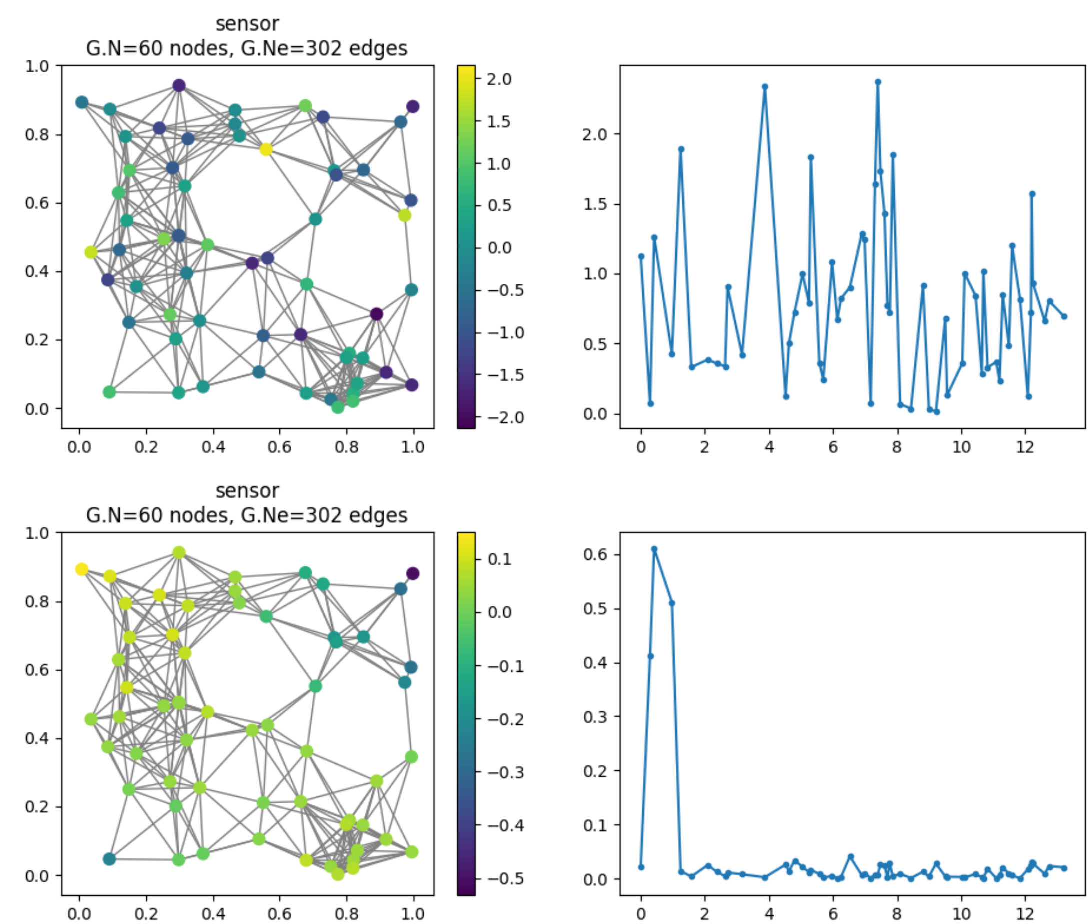
Vandergheynst and Shuman, "Wavelets on graphs, an introduction," Université de Provence, 2011.

Example on a general graph

GFT:

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}^T \begin{matrix} | \\ f \\ | \end{matrix}$$





Outline

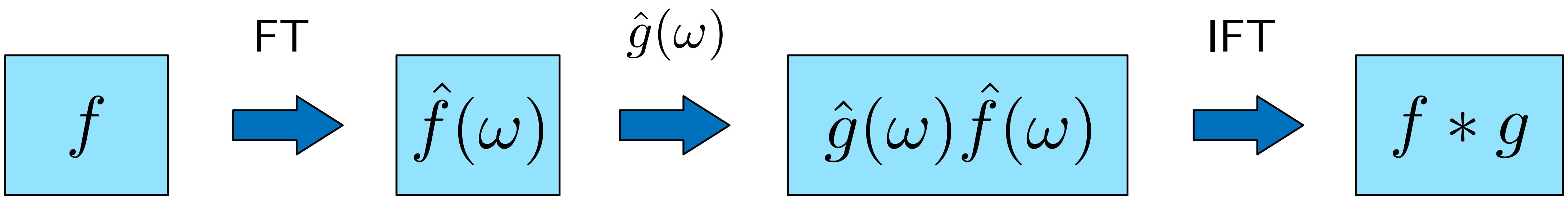
- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

Classical frequency filtering

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx \qquad f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$

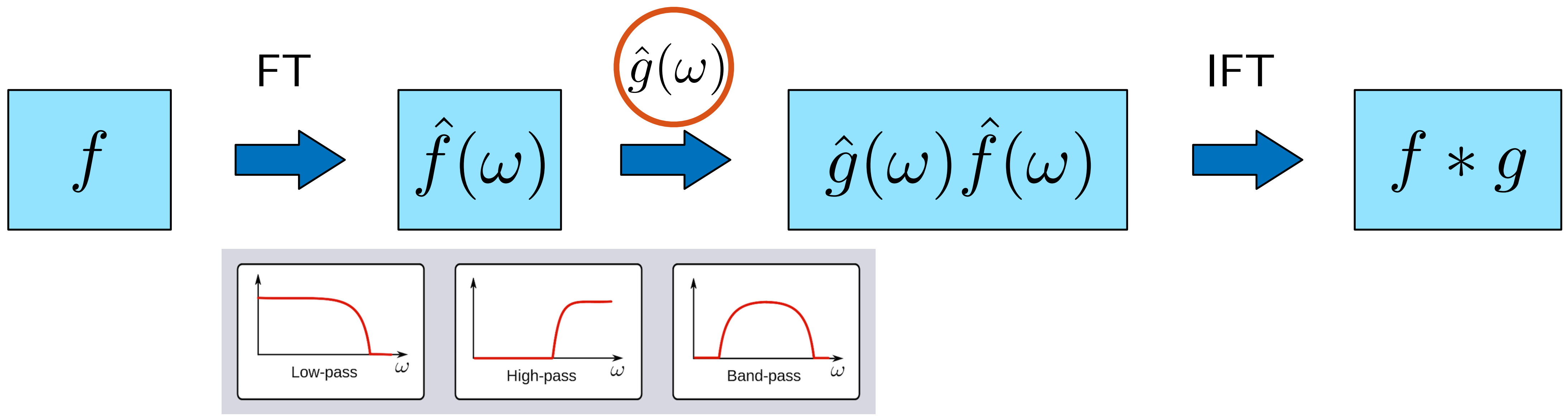
Classical frequency filtering

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx \qquad f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$



Classical frequency filtering

Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x)dx \qquad f(x) = \frac{1}{2\pi}\int \hat{f}(\omega)e^{j\omega x}d\omega$



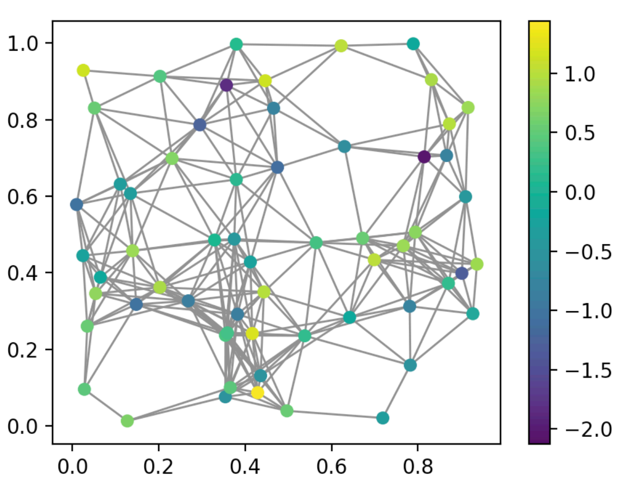
Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

Graph spectral filtering

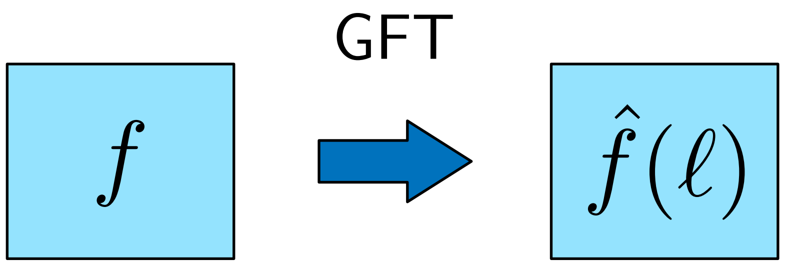
$$\text{GFT:}\quad \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

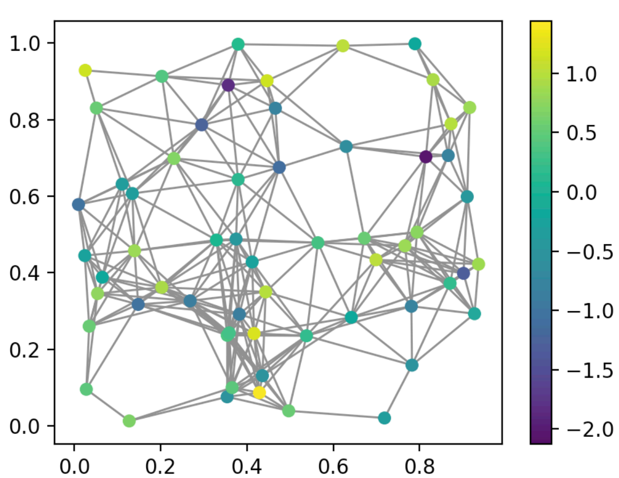
f

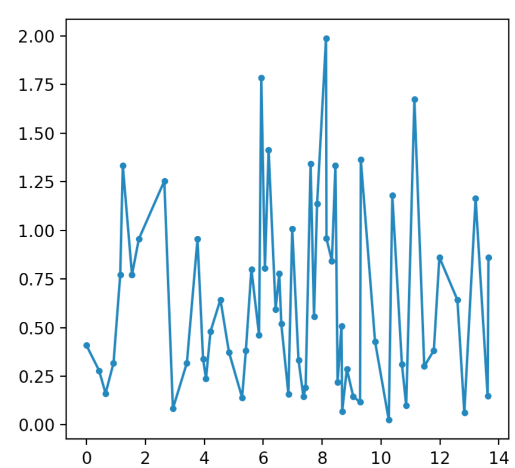


Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

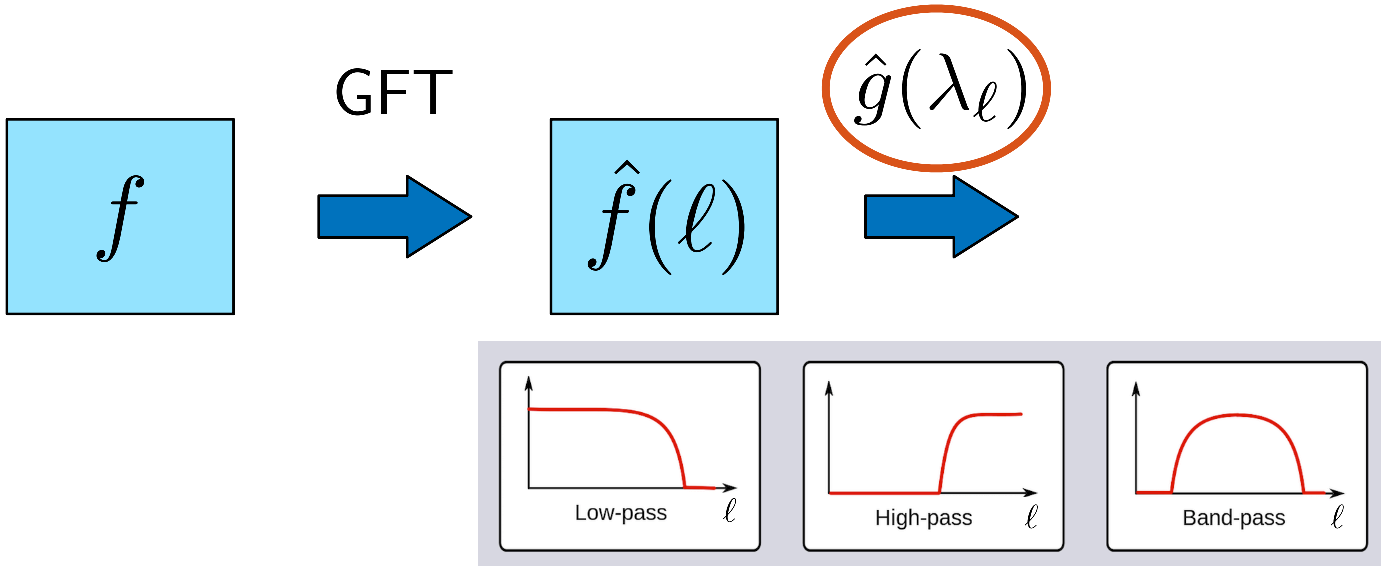


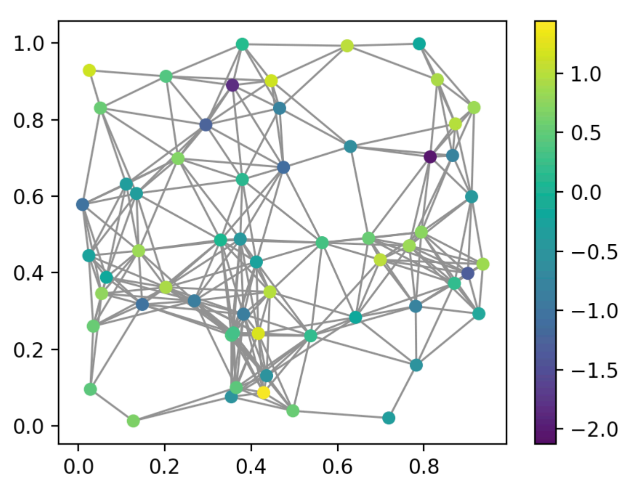


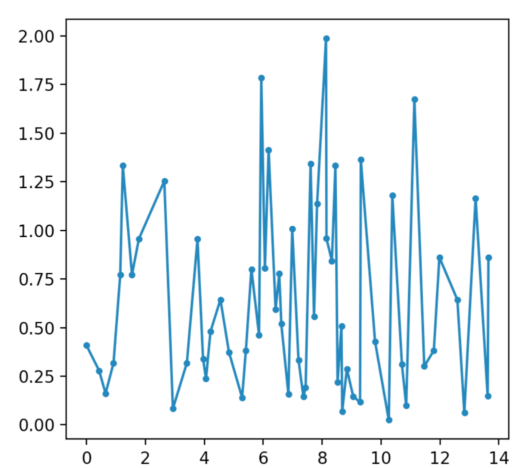


Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

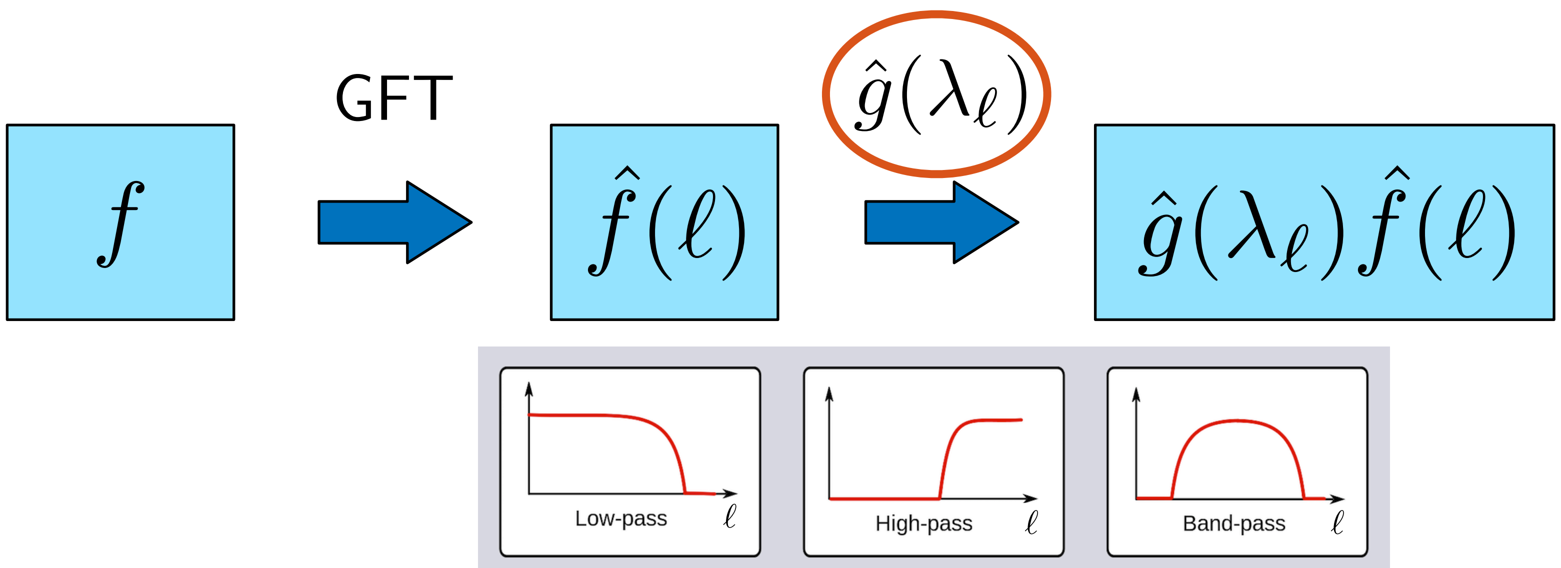


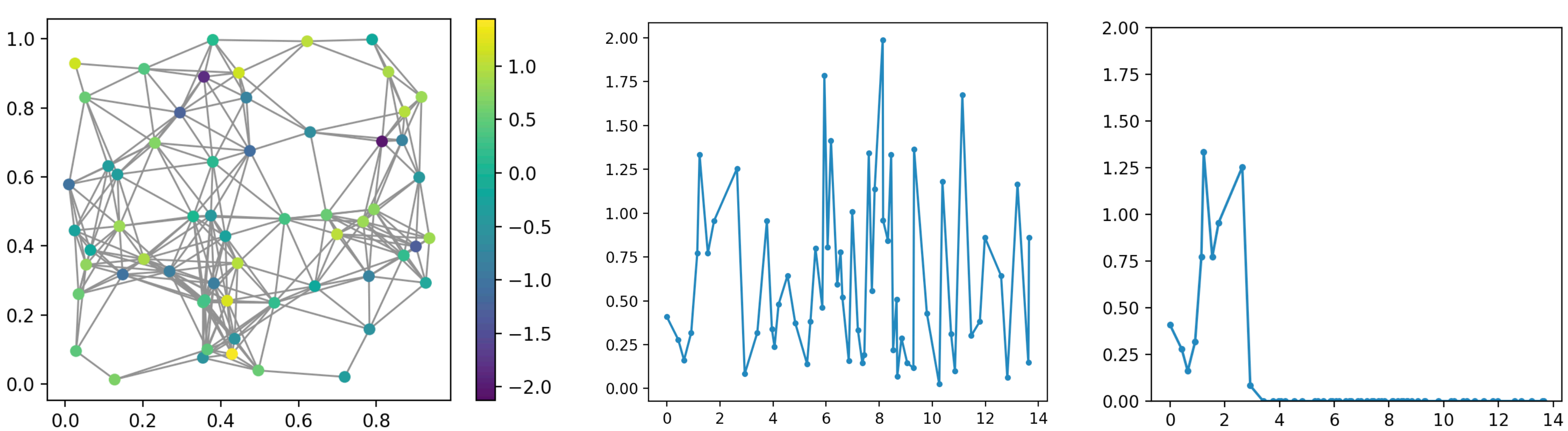




Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

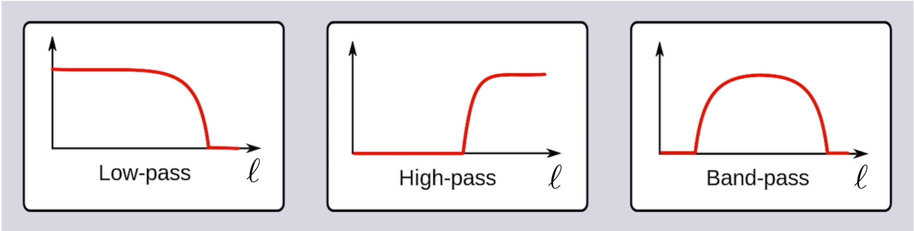


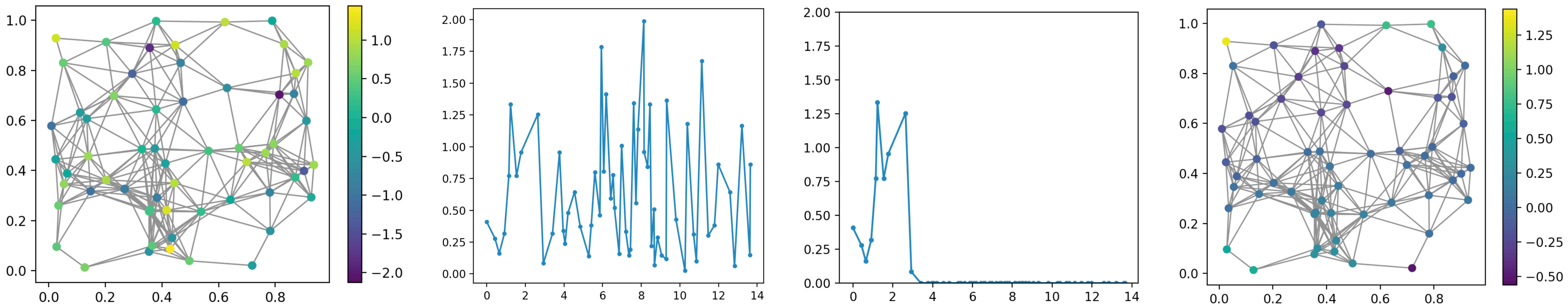


Graph spectral filtering

GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

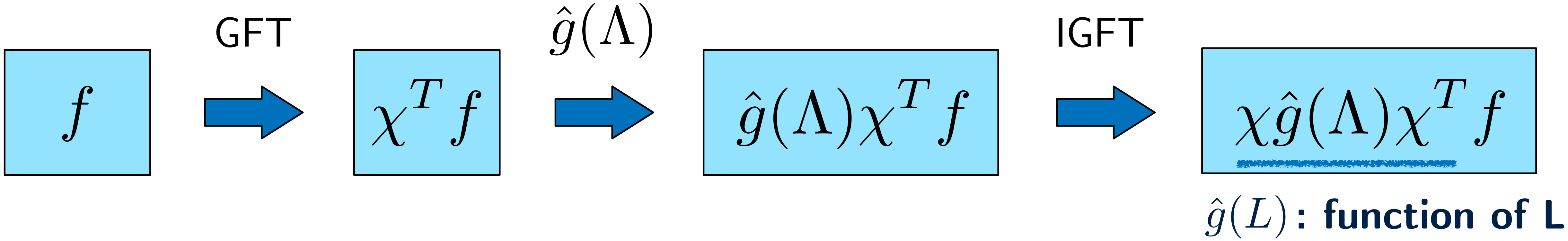
f → GFT → $\hat{f}(\ell)$ → $\hat{g}(\lambda_\ell)$ → $\hat{g}(\lambda_\ell)\hat{f}(\ell)$ → IGFT → $f(i) = \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell)\hat{f}(\ell)\chi_\ell(i)$





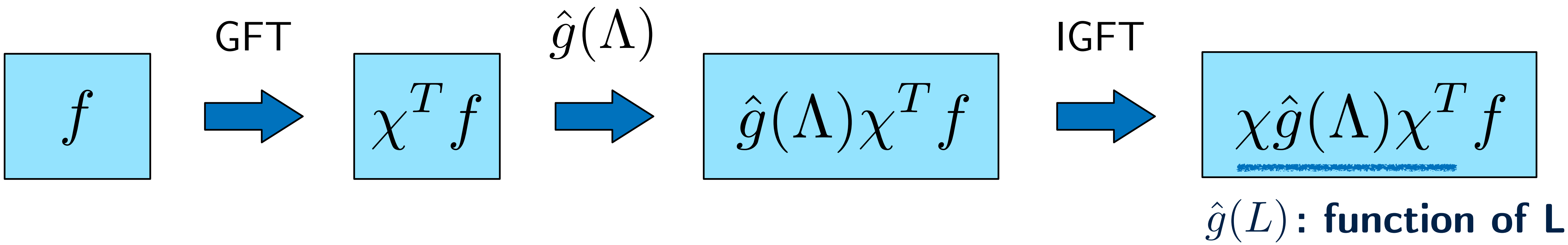
Graph transform/dictionary design

- Transforms and dictionaries can be designed through graph spectral filtering: Functions of graph Laplacian!



Graph transform/dictionary design

- Transforms and dictionaries can be designed through graph spectral filtering: Functions of graph Laplacian!

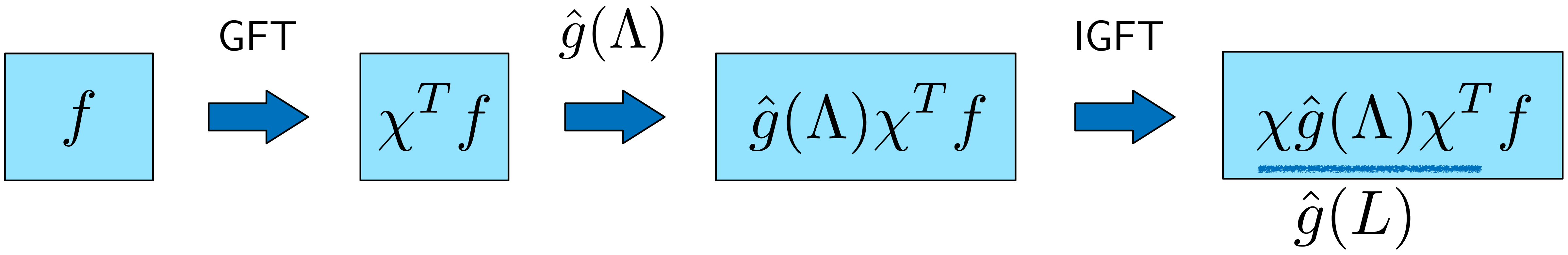


- Important properties can be achieved by properly defining $\hat{g}(L)$, such as localisation of atoms

- Closely related to kernels and regularisation on graphs

Smola and Kondor, "Kernels and regularization on graphs", COLT, 2003.

A practical example



A practical example

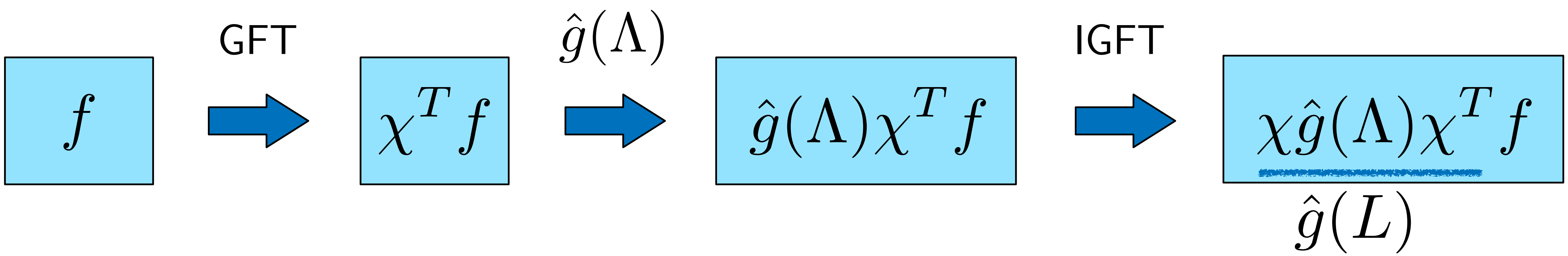
f $\xrightarrow{\text{GFT}}$ $\chi^T f$ $\xrightarrow{\hat{g}(\Lambda)}$ $\hat{g}(\Lambda)\chi^T f$ $\xrightarrow{\text{IGFT}}$ $\chi\hat{g}(\Lambda)\chi^T f$

$\hat{g}(L)$

problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

A practical example



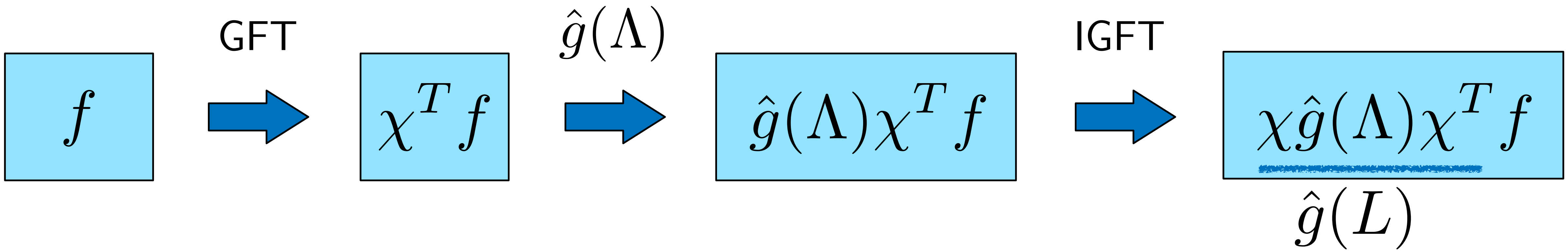
problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

data fitting term

"smoothness" assumption

A practical example



problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

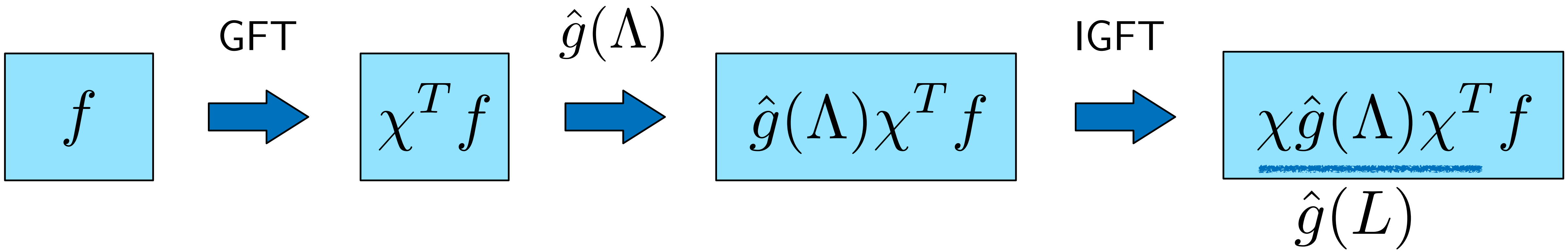
$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

data fitting term

"smoothness" assumption

$$y^* = \underbrace{(I + \gamma L)^{-1}}_{\hat{g}(L)} f$$

A practical example



problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg\min_y \{||y - f||_2^2 + \gamma y^T L y\}$$

data fitting term

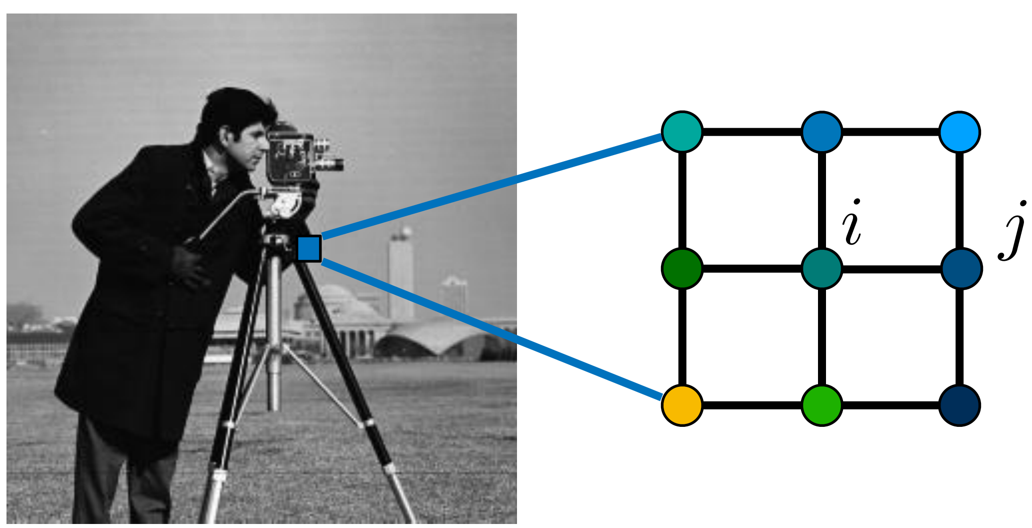
"smoothness" assumption

$$y^* = \underbrace{(I + \gamma L)^{-1}}_{\hat{g}(L)} f = \chi(1 + \gamma\Lambda)^{-1}\chi^T f$$

remove noise by low-pass filtering in graph spectral domain!

A practical example

- noisy image as observed noisy graph signal
- regular grid graph (weights inversely proportional to pixel value difference)

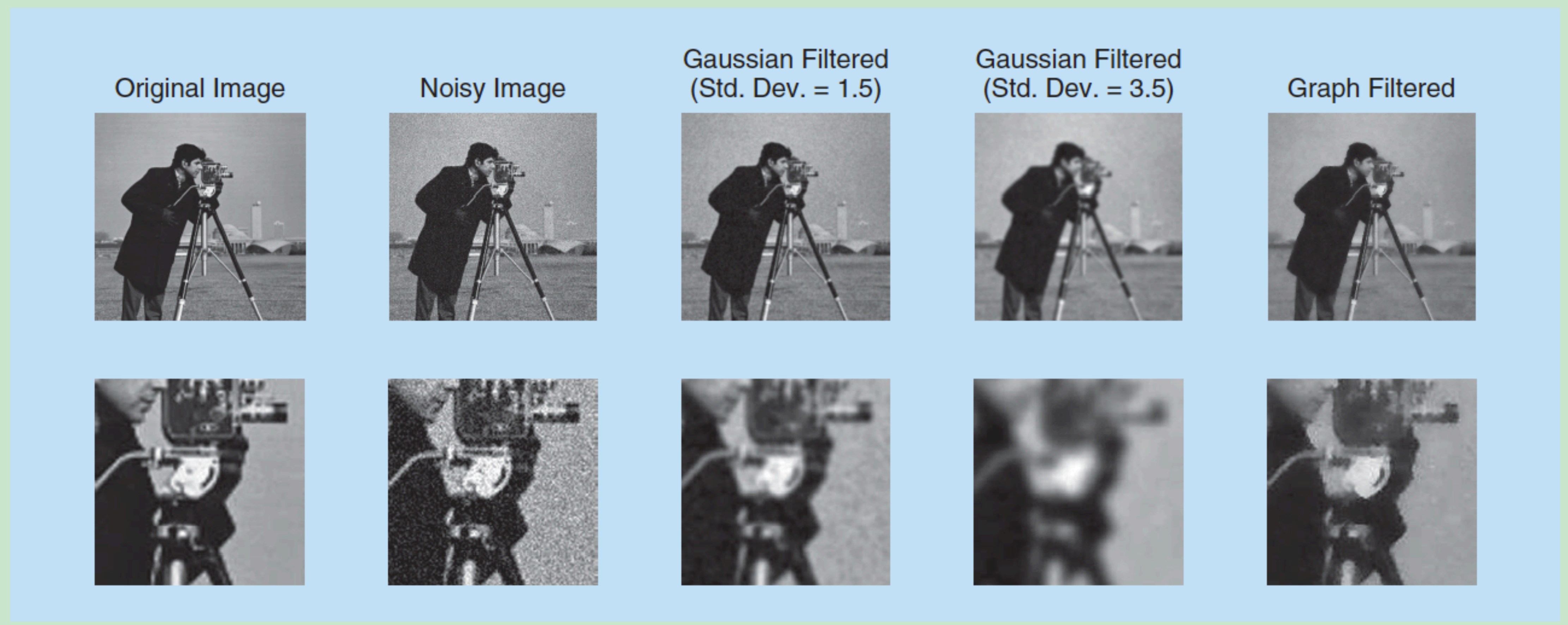


$$w_{ij} = \frac{1}{|f(i) - f(i)|}$$

Shuman et al., "The emerging field of signal processing on graphs," IEEE SPM, 2013.

A practical example

- noisy image as observed noisy graph signal
- regular grid graph (weights inversely proportional to pixel value difference)



[FIGS2] **Image denoising via classical Gaussian filtering and graph spectral filtering.**

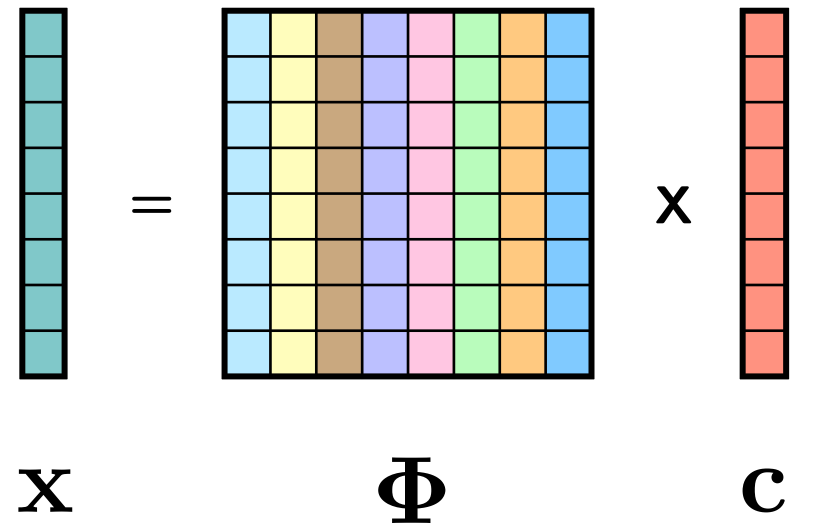
Shuman et al., "The emerging field of signal processing on graphs," IEEE SPM, 2013.

Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

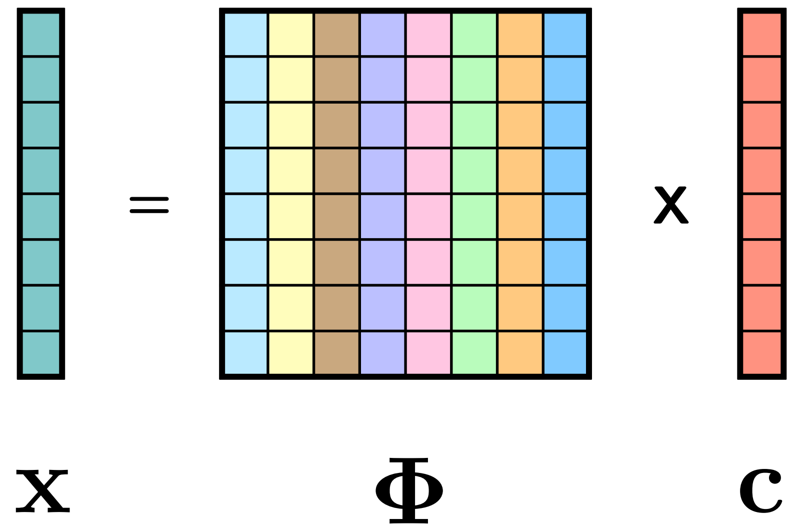
Classical vs. Graph dictionaries

classical signal

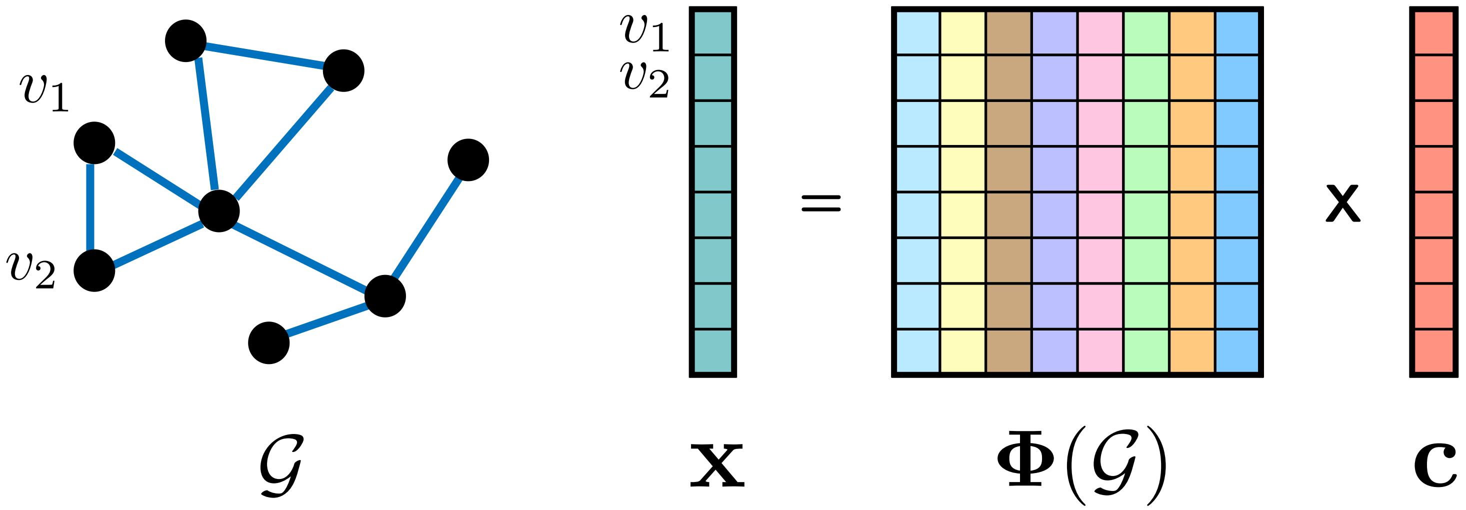


Classical vs. Graph dictionaries

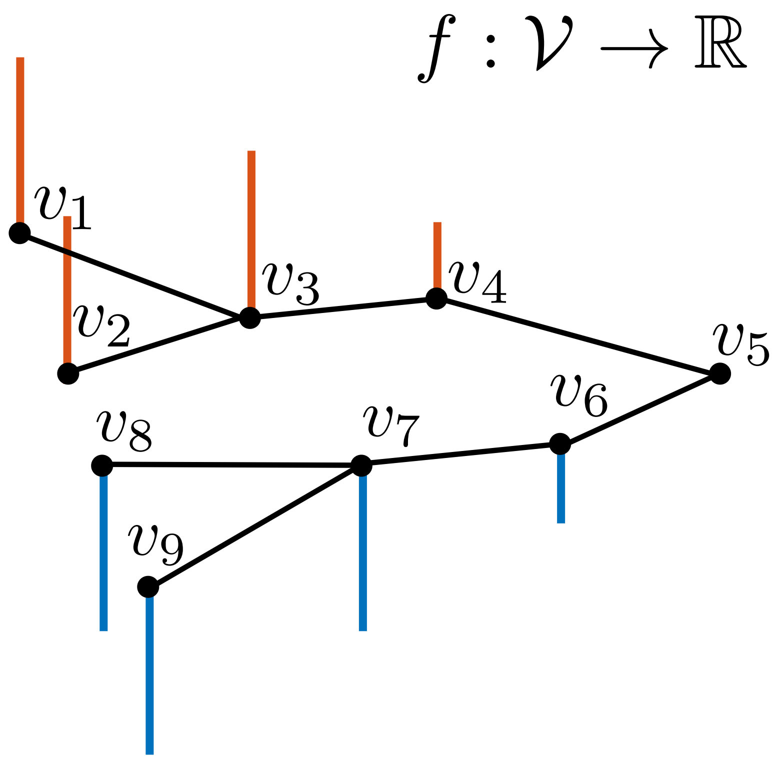
classical signal



graph signal

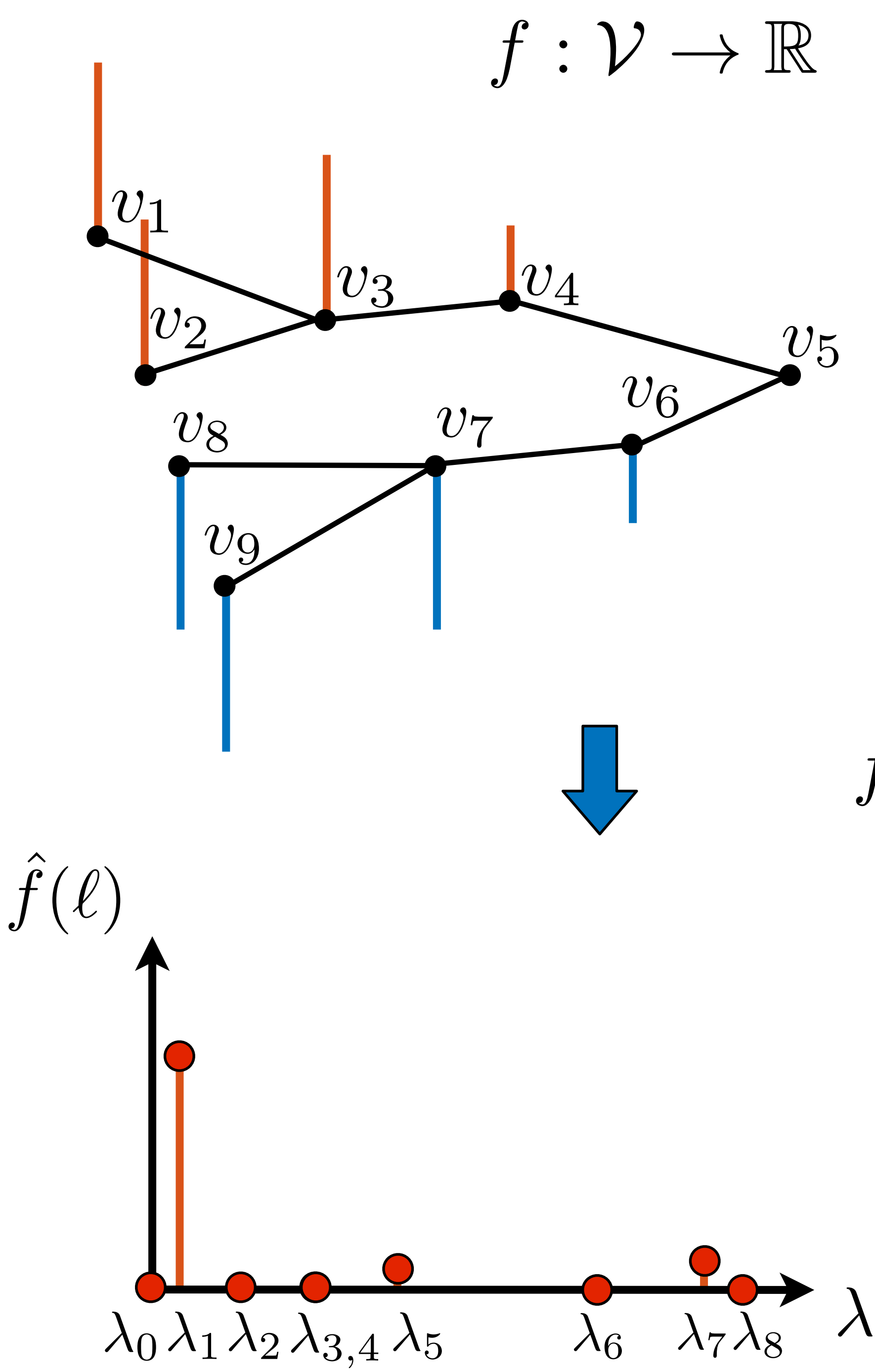


GFT provides a first graph dictionary



vertex domain

GFT provides a first graph dictionary

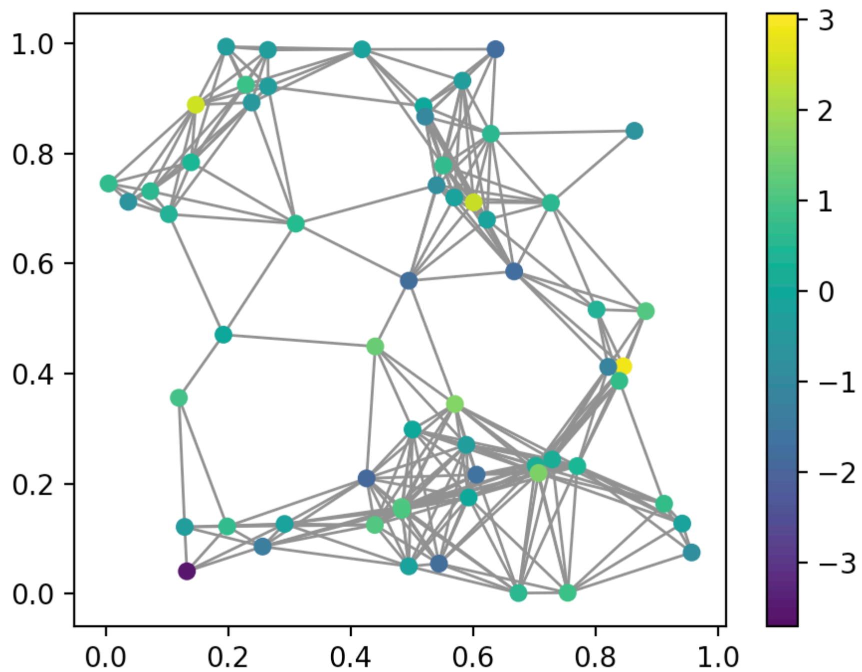


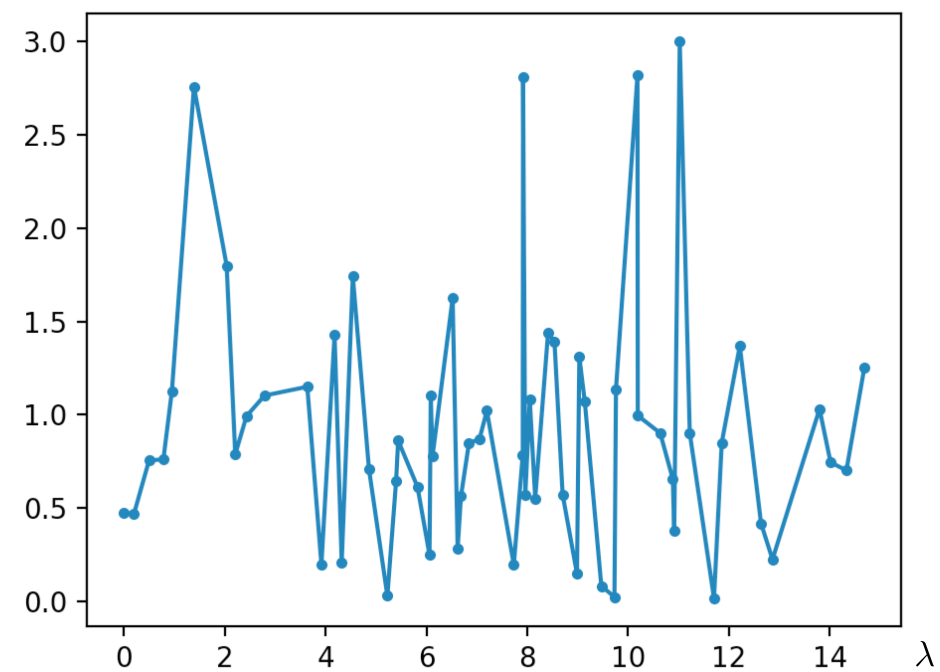
vertex domain

$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \quad \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}^T \begin{matrix} | \\ f \\ | \end{matrix}$$

frequency (graph spectral) domain

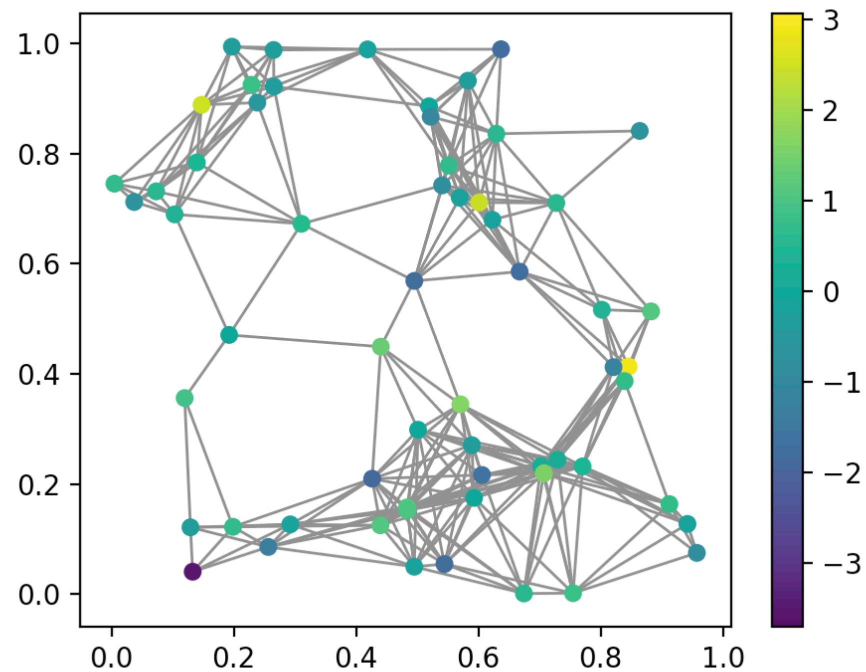
GFT provides a first graph dictionary



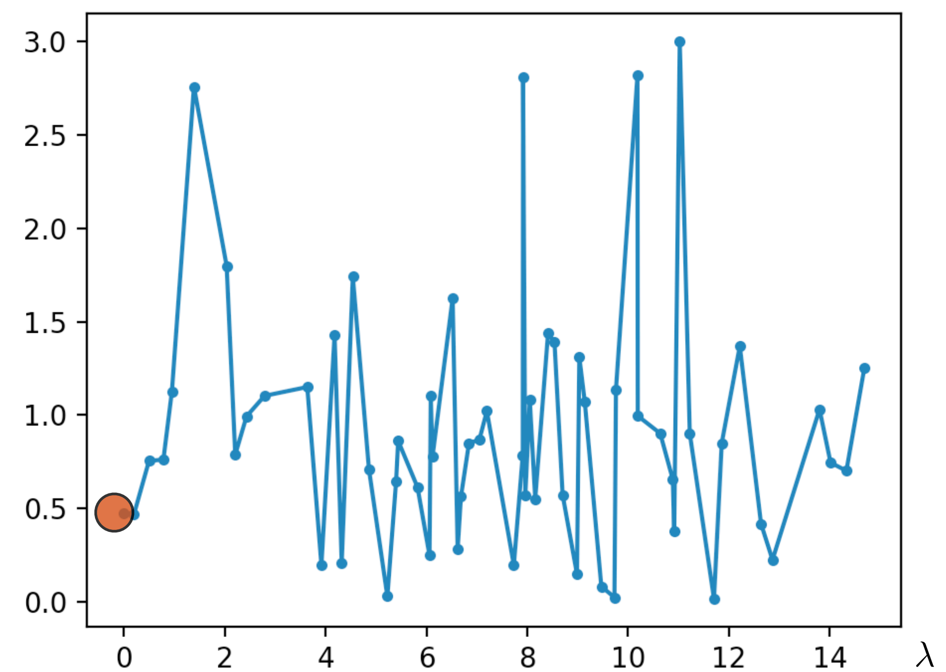


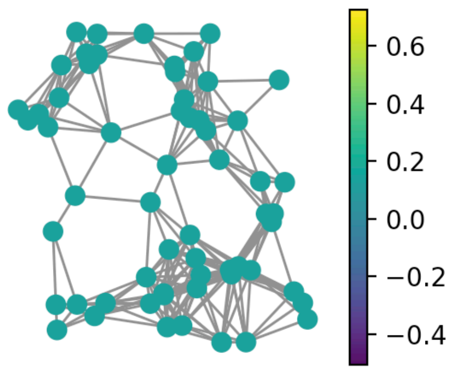
GFT provides a first graph dictionary

f



$\hat{f}_\ell$

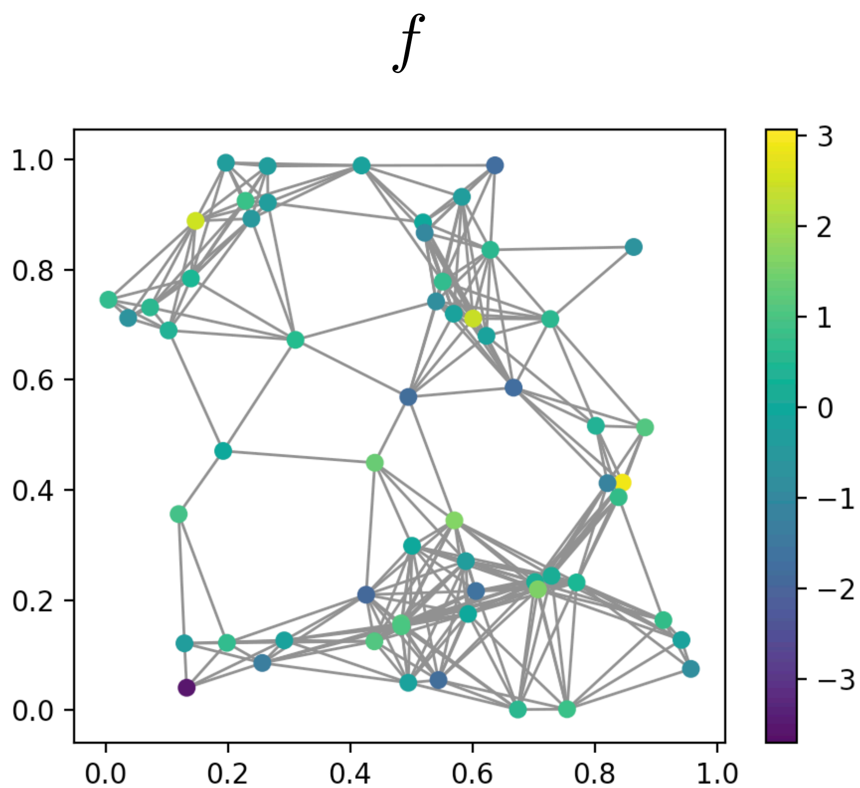


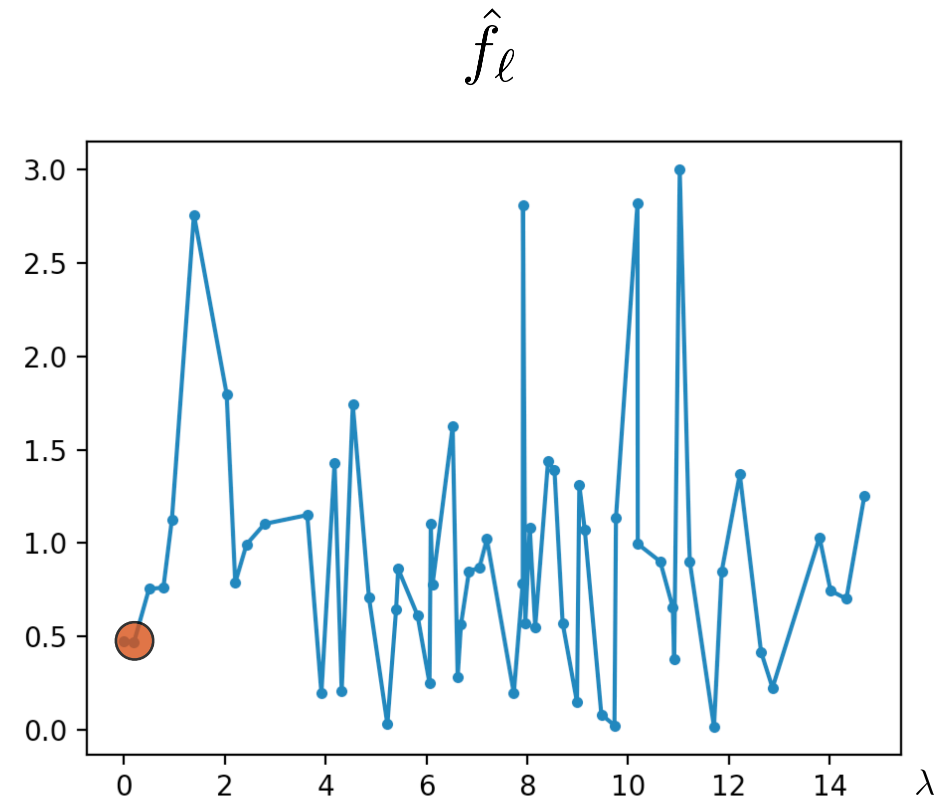


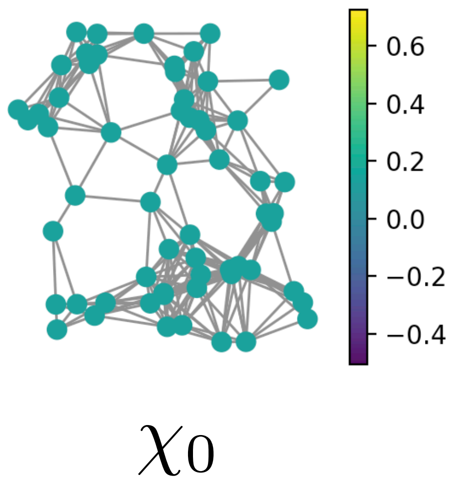
χ_0

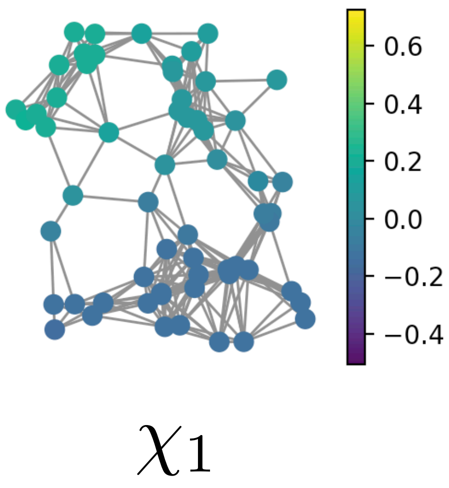
GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary



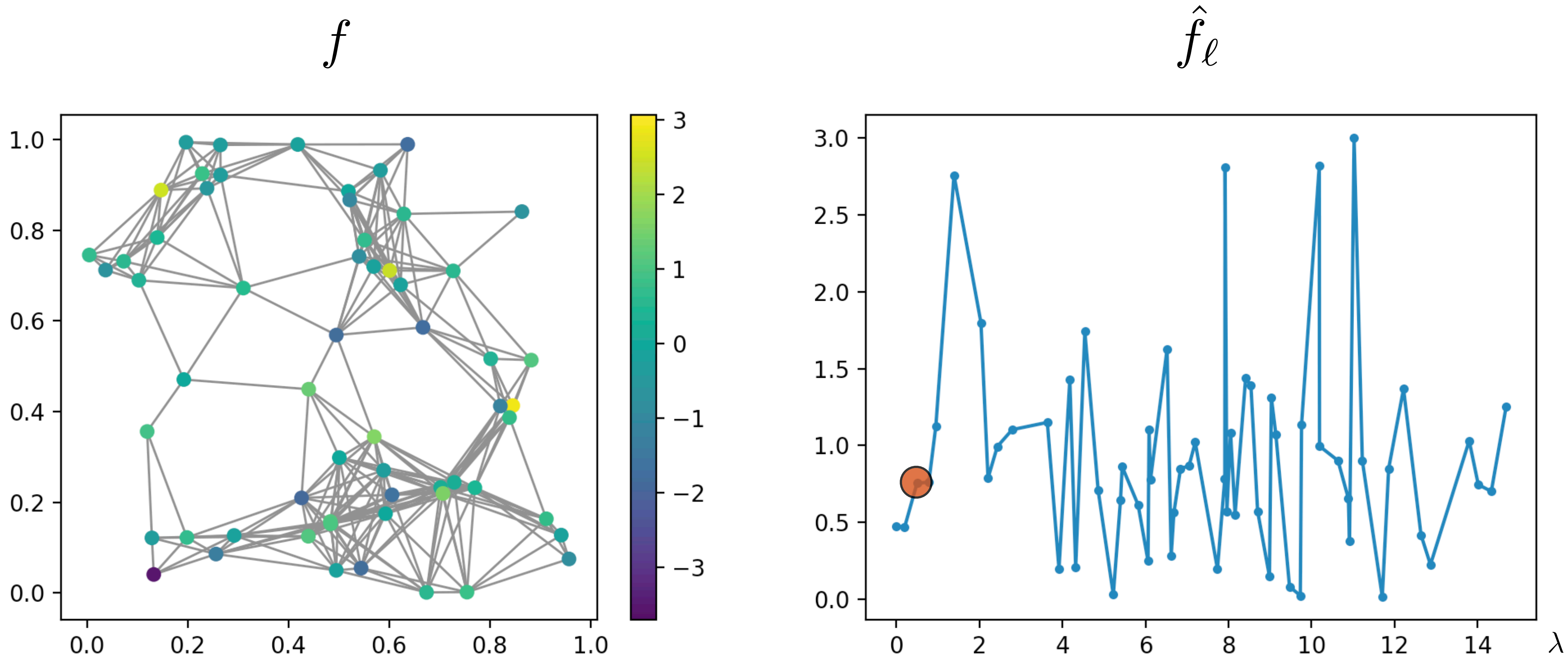


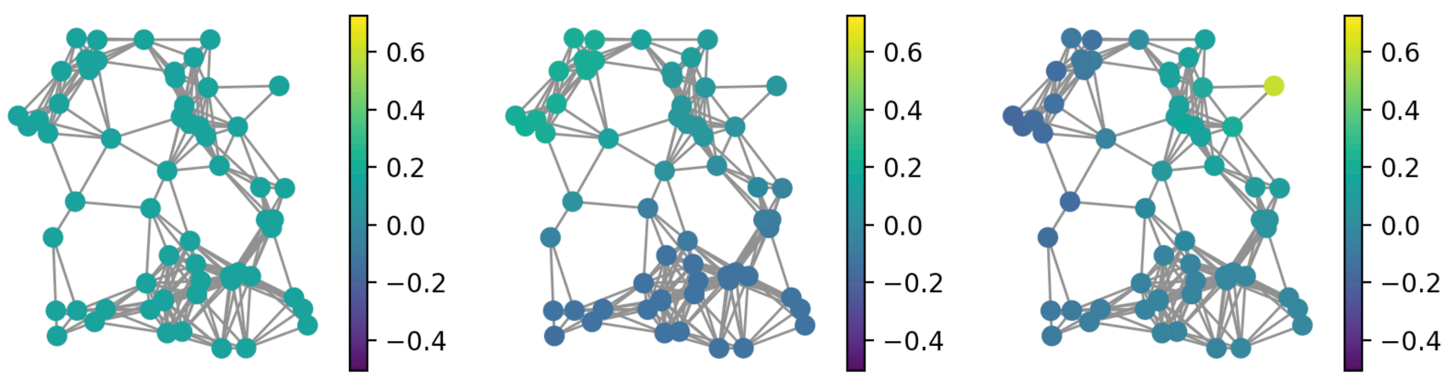




GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary

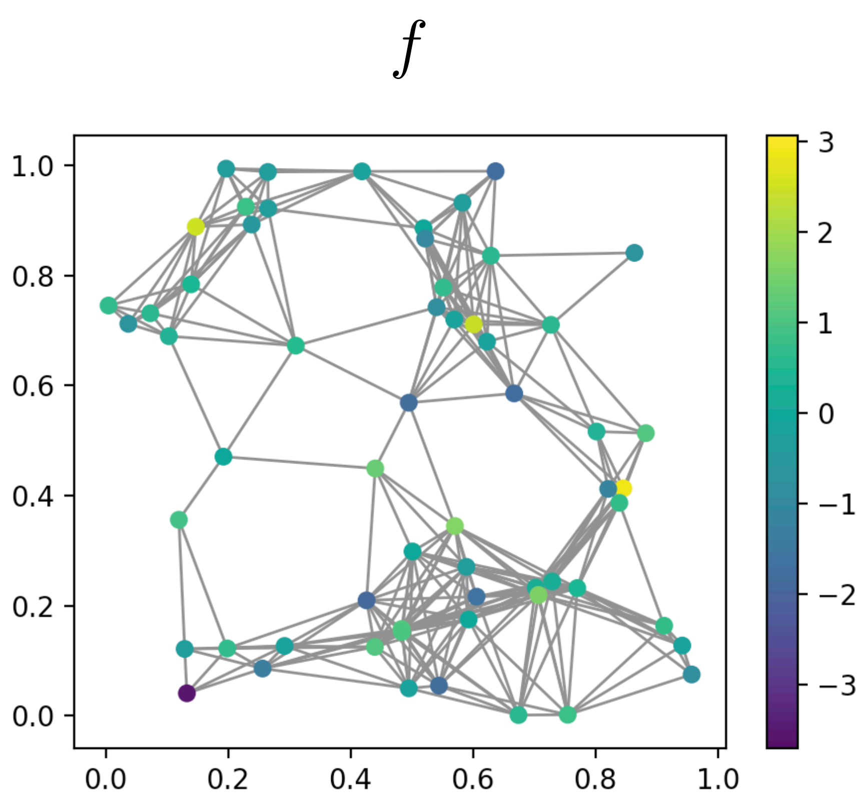


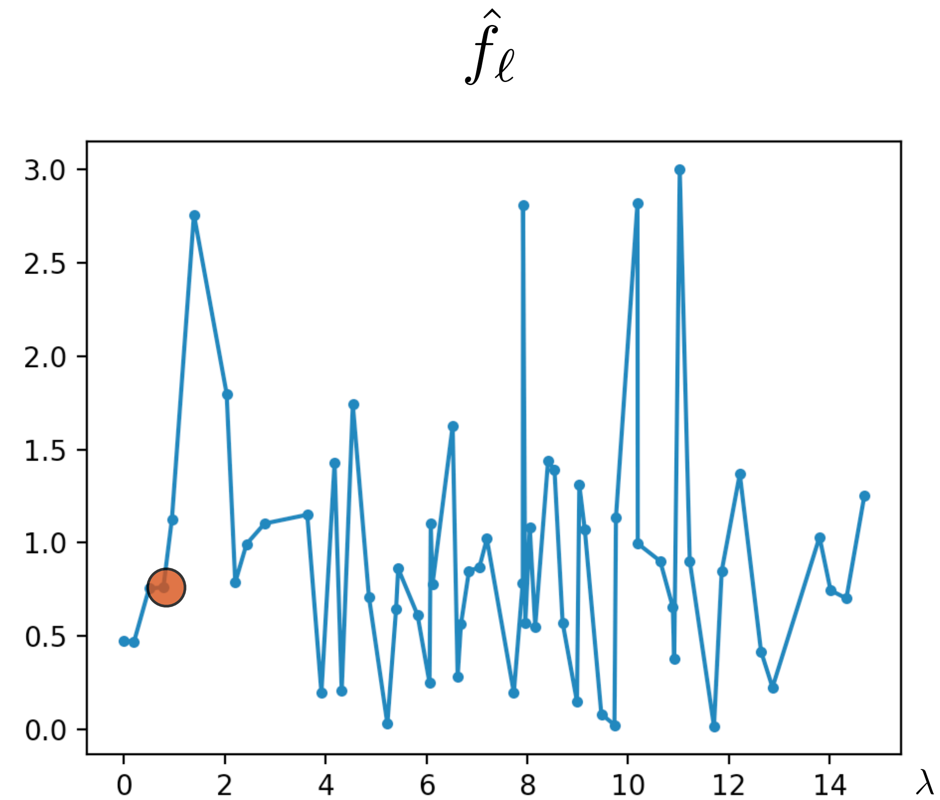


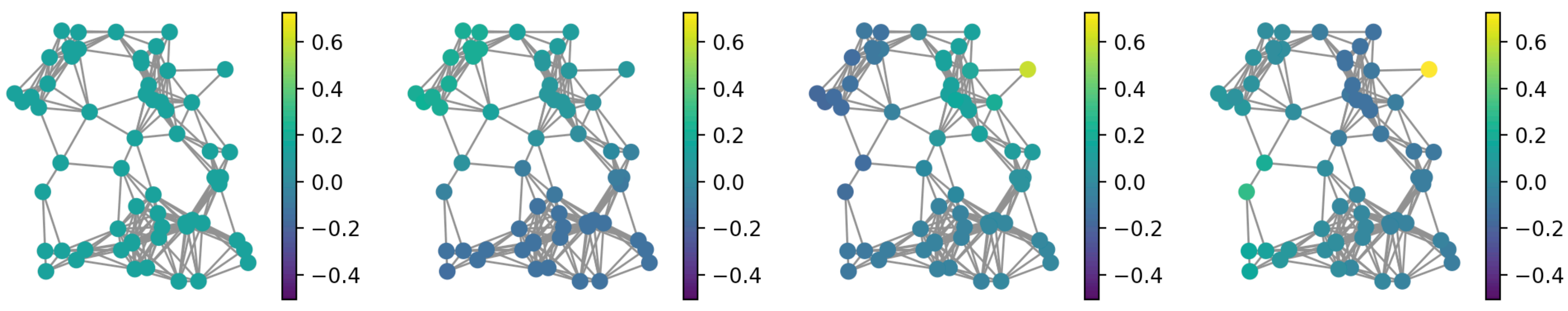
χ_0 χ_1 χ_2

GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary



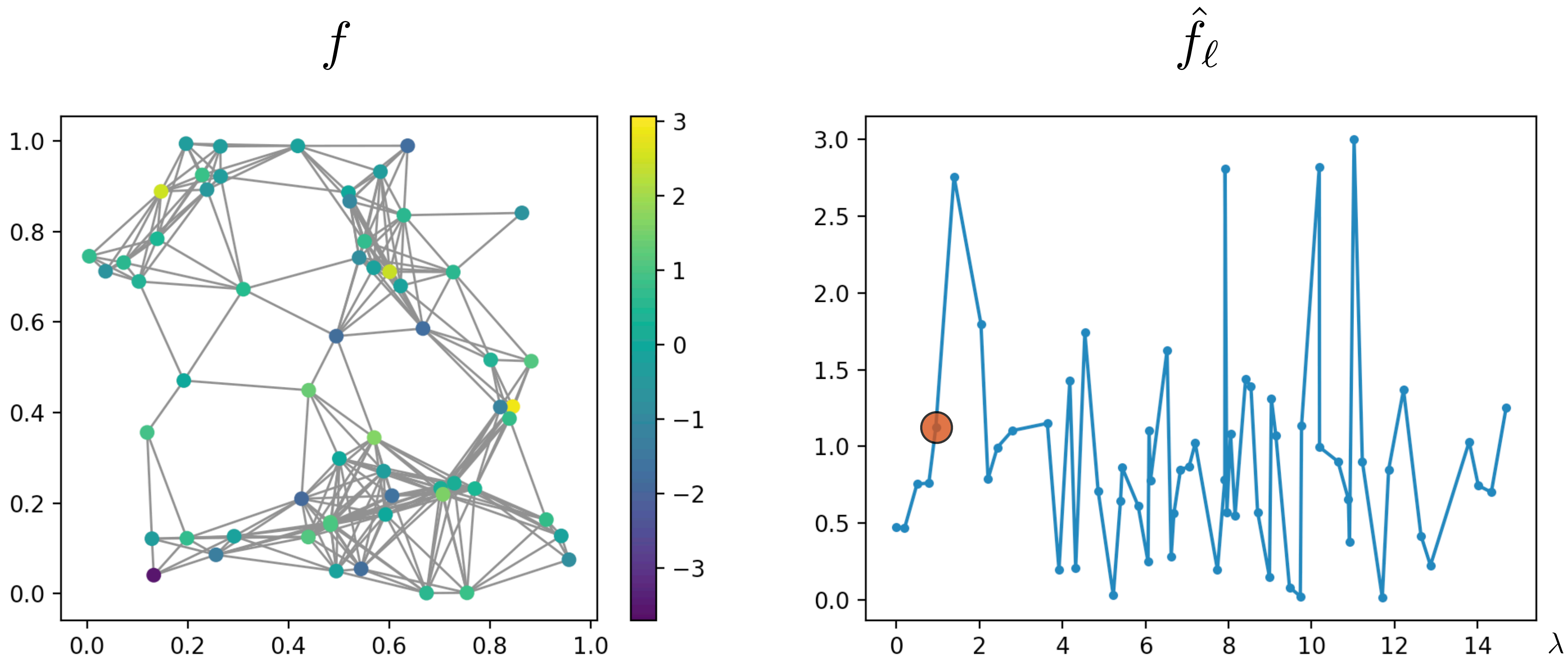


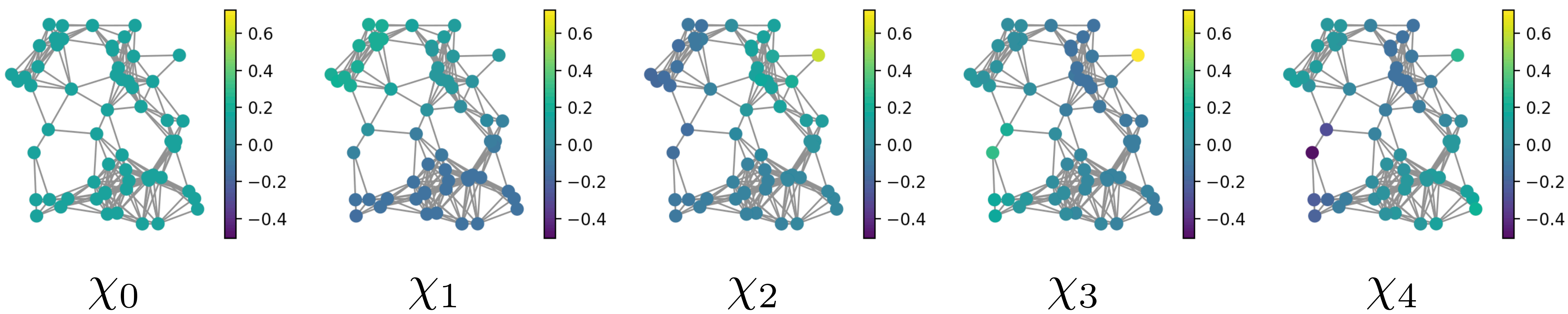


χ_0 χ_1 χ_2 χ_3

GFT atoms (corresponding to discrete frequencies)

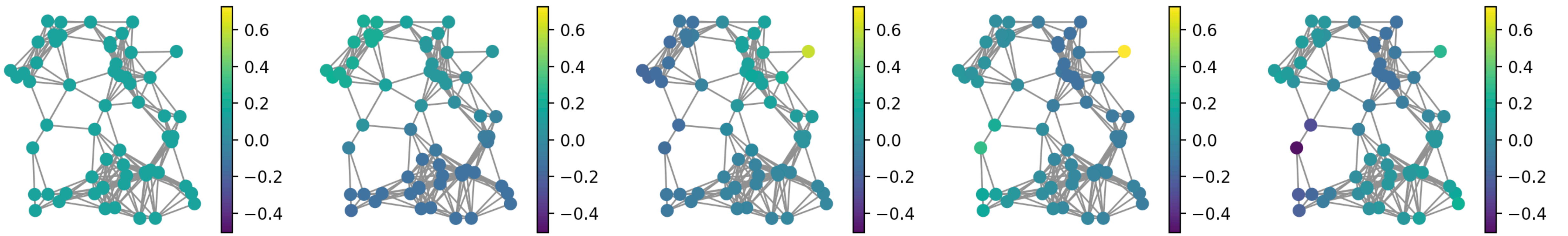
GFT provides a first graph dictionary





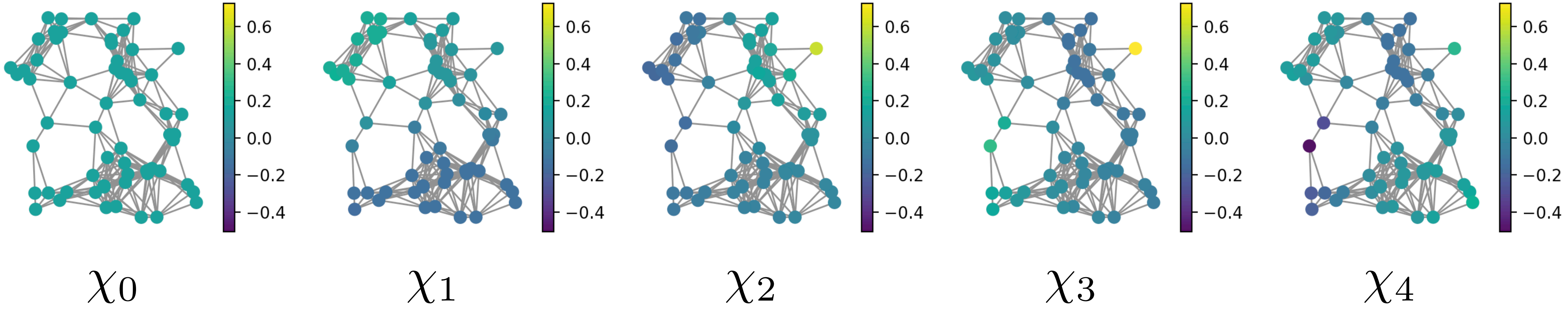
GFT atoms (corresponding to discrete frequencies)

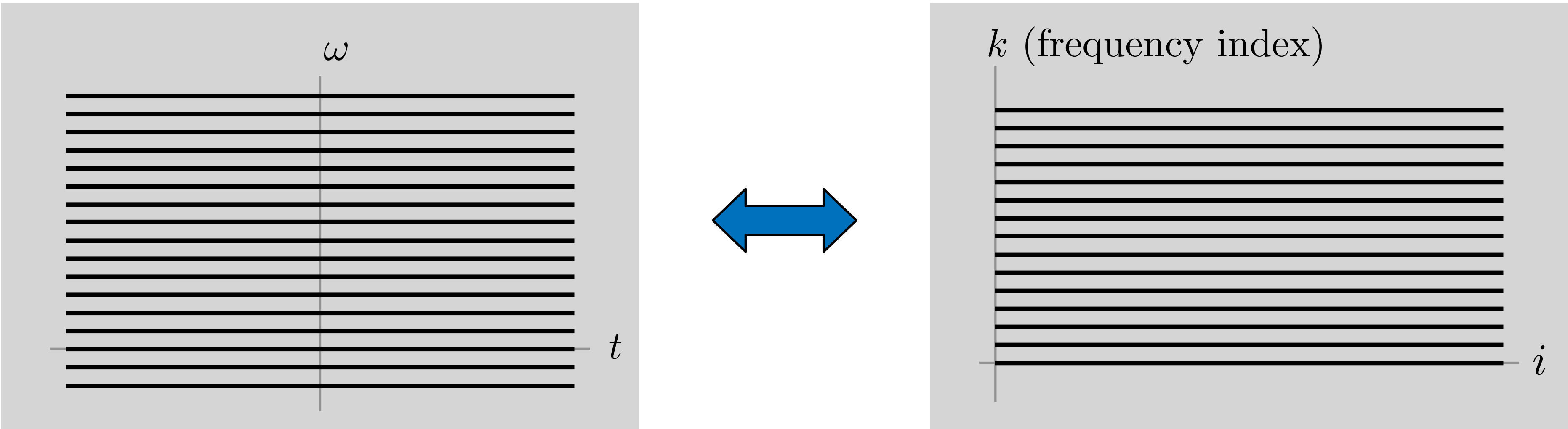
GFT provides a first graph dictionary



χ_0 χ_1 χ_2 χ_3 χ_4

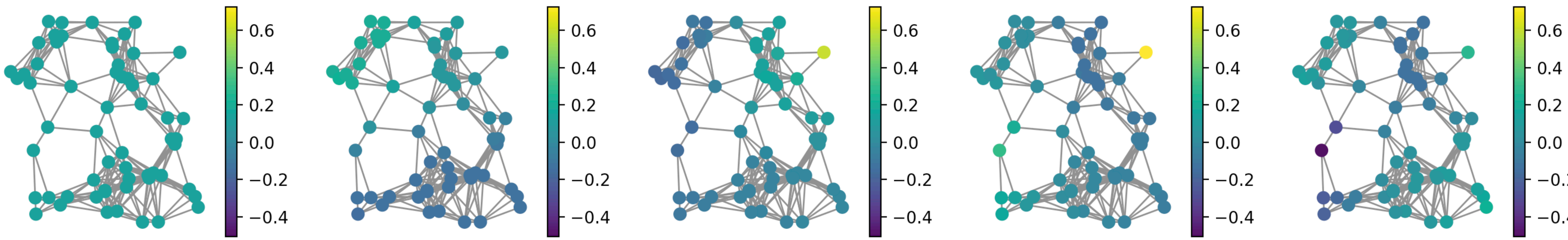
GFT provides a first graph dictionary



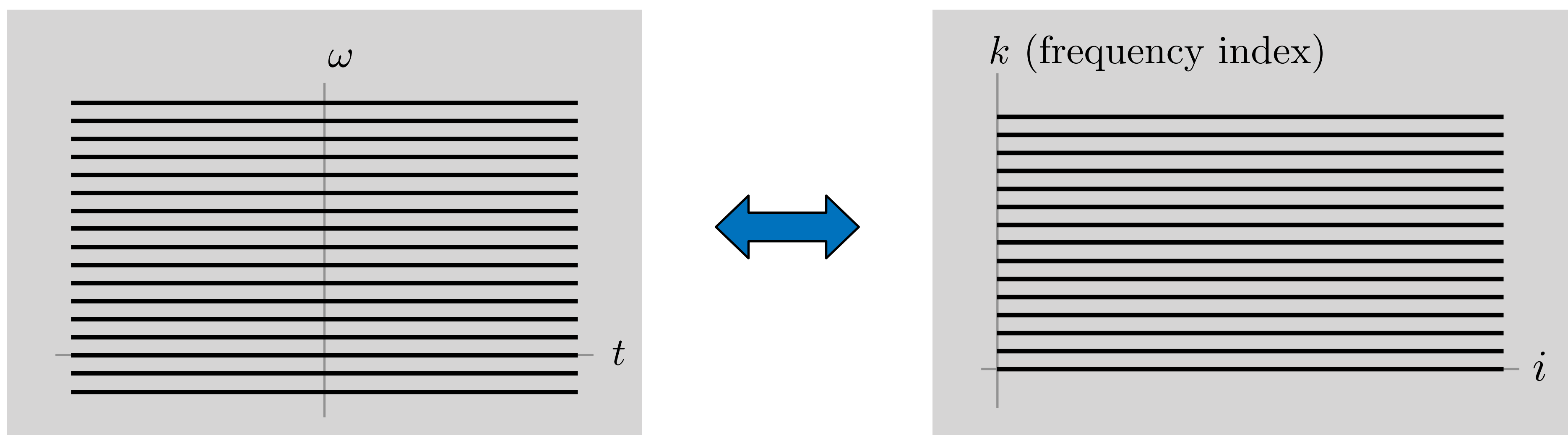


- like complex exponentials in classical FT, eigenvectors in GFT have **global** support

GFT provides a first graph dictionary



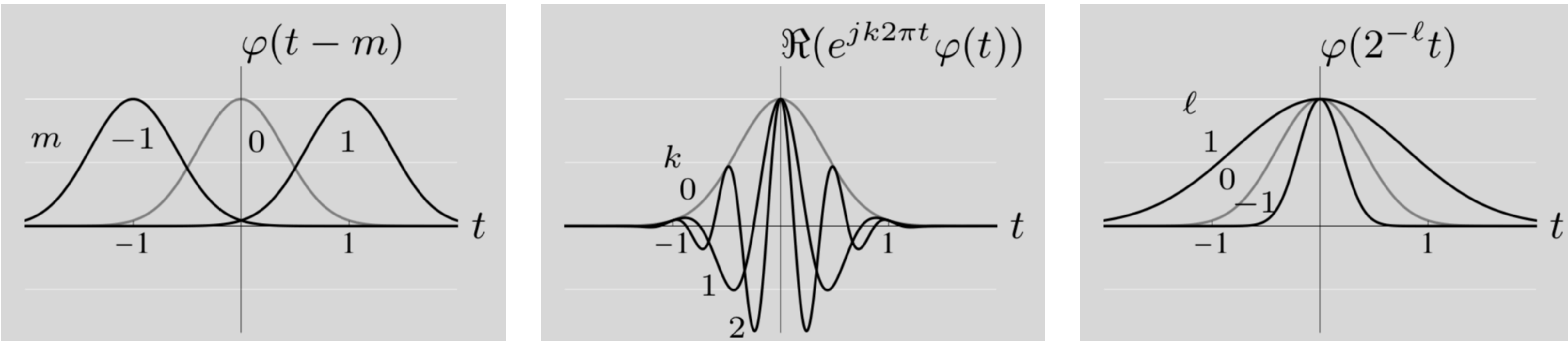
χ_0 χ_1 χ_2 χ_3 χ_4



- like complex exponentials in classical FT, eigenvectors in GFT have **global** support
- can we design **localised** atoms on graphs?

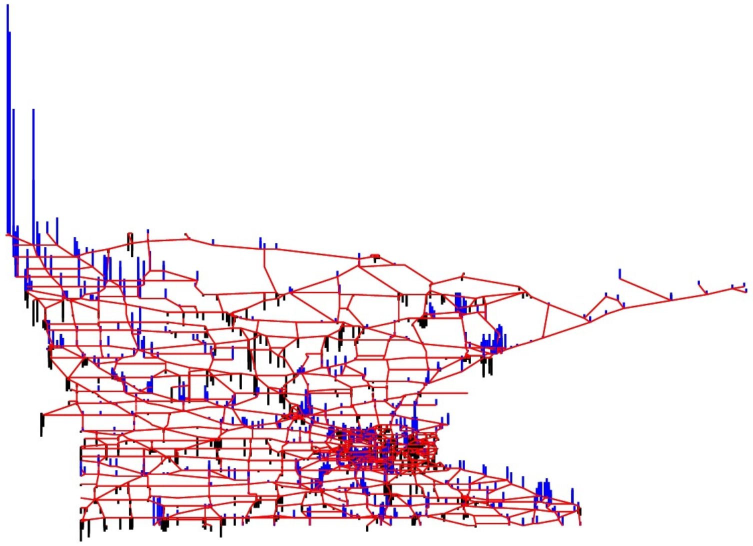
Basic operations for graph signals

basic operations in Euclidean domain



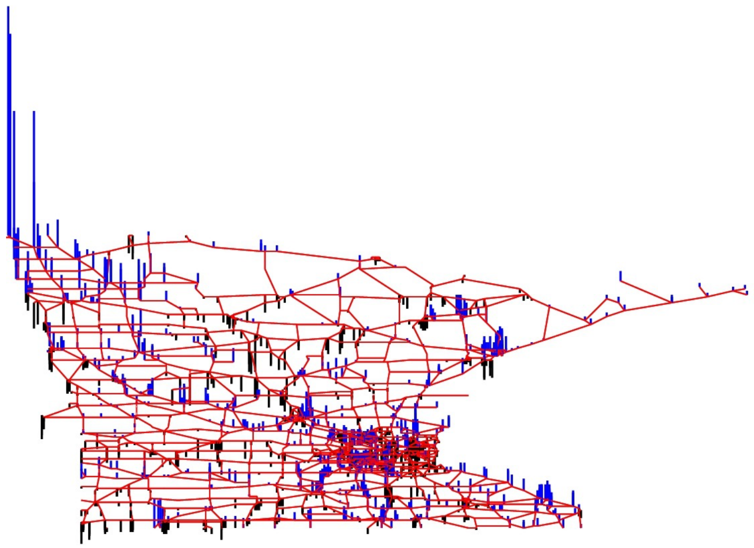
- recall that we used a set of structured functions (e.g., shifted and modulated) to produce localised items

Basic operations for graph signals



- recall that we used a set of structured functions (e.g., shifted and modulated) to produce localised items
- we need to define for graph signals the basic operations of **convolution**, **shift**, **modulation**

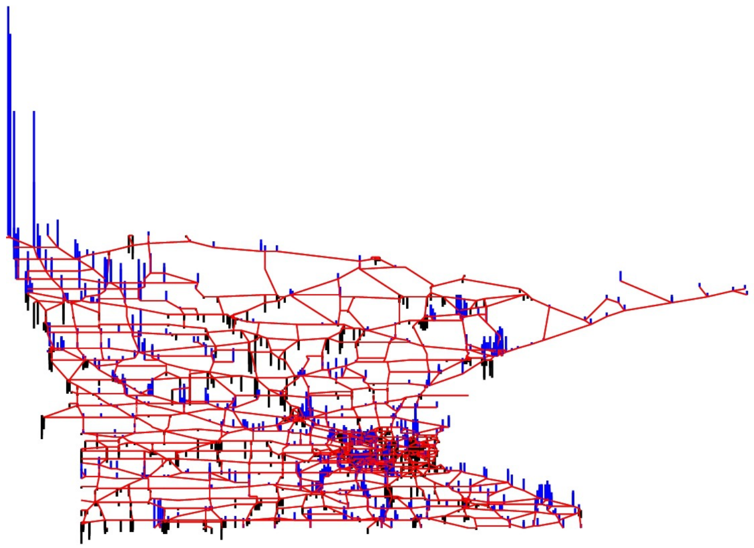
Convolution



classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

Convolution



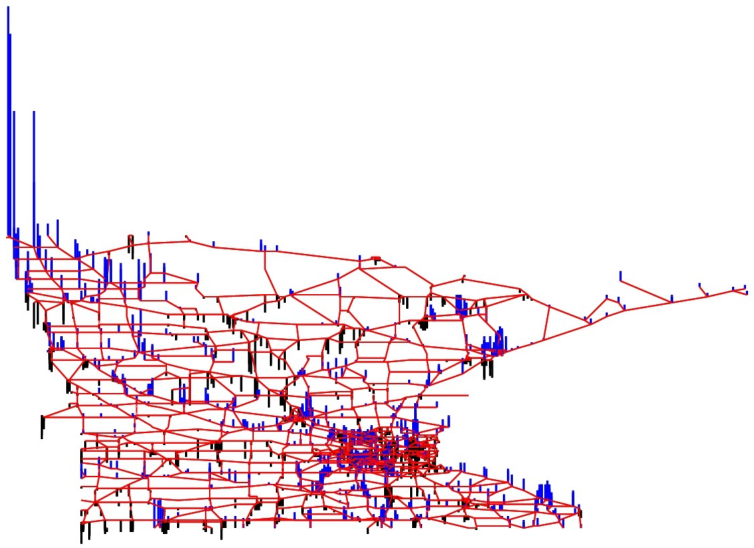
classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Convolution



classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

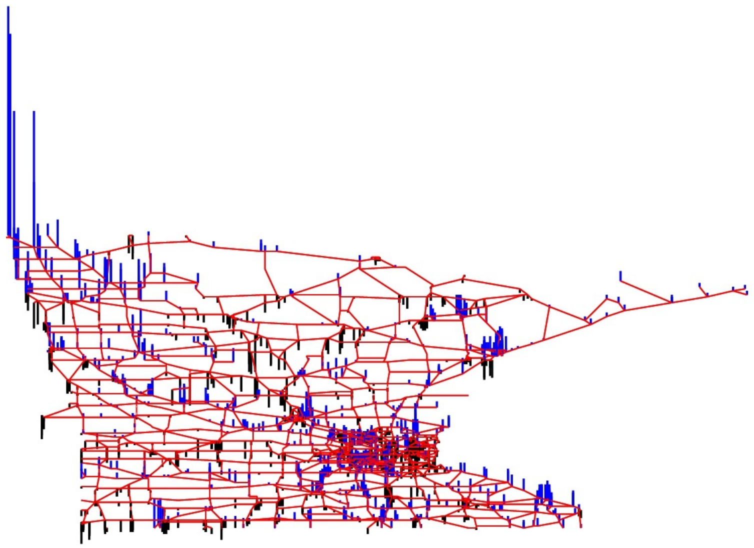
$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

graph convolution **multiplication in graph spectral domain**

$$\widehat{(f * g)}(\lambda) = (\hat{f} \circ \hat{g})(\lambda)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Convolution



classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

graph convolution

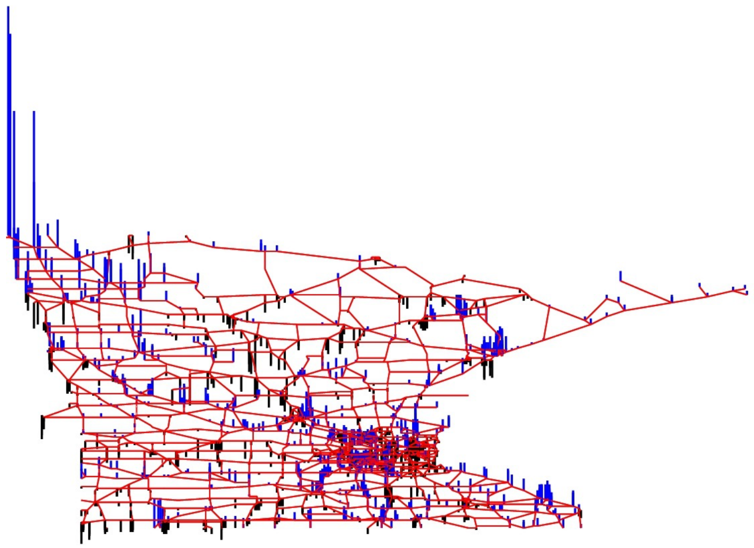
multiplication in graph spectral domain

$$\widehat{(f * g)}(\lambda) = (\hat{f} \circ \hat{g})(\lambda)$$

$$(f * g)(n) := \sum_{\ell=0}^{N-1} \hat{f}(\ell) \hat{g}(\ell) \chi_{\ell}(n)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

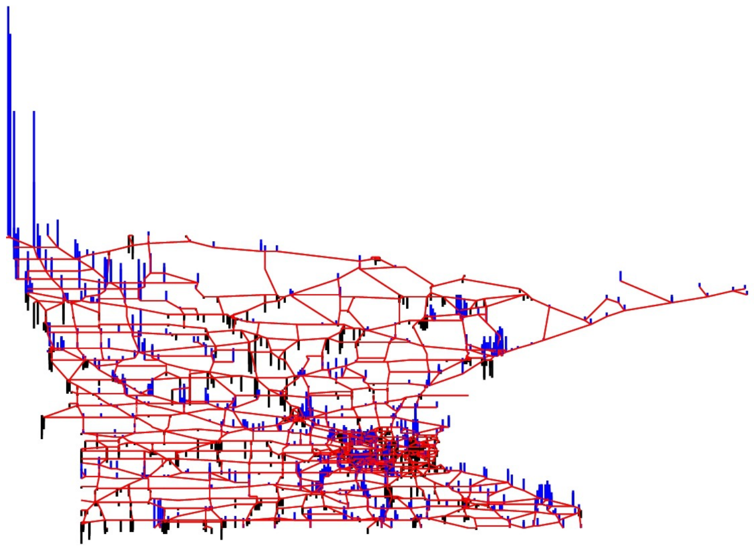
Vertex-domain shift



original signal

classical shift $(T_u f)(t) := f(t-u) = (f * \delta_u)(t)$

Vertex-domain shift



original signal

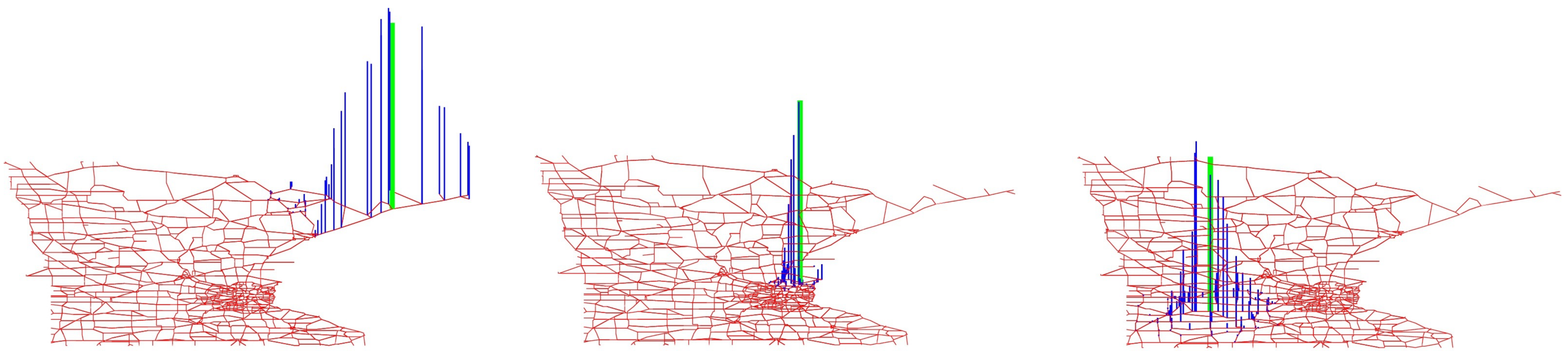
classical shift $$(T_u f)(t) := f(t-u) = (f * \delta_u)(t)$$

graph shift **convolution with a "delta" on graph**

$$(T_i f)(n) := \sqrt{N}(f * \delta_i)(n)$$
$$= \sqrt{N} \sum_{\ell=0}^{N-1} \hat{f}(\ell)\chi_\ell^*(i)\chi_\ell(n)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Vertex-domain shift



shifted version of the signal to different centring vertex (in green)

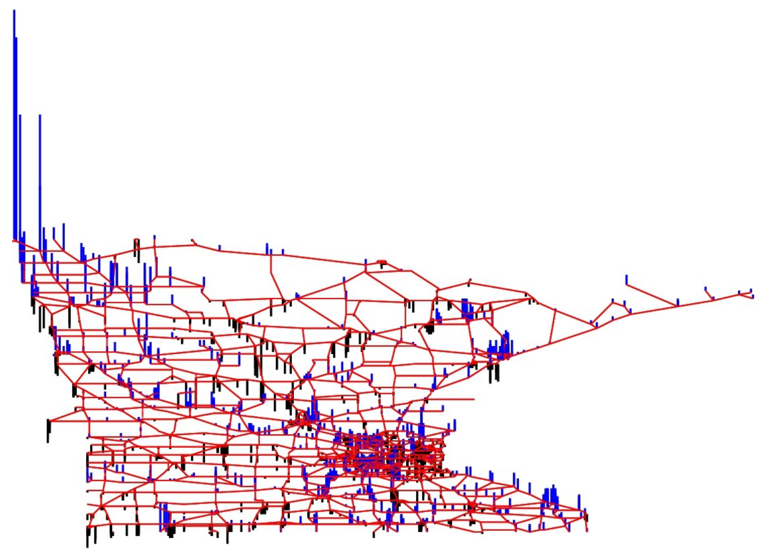
classical shift $$(T_u f)(t) := f(t-u) = (f * \delta_u)(t)$$

graph shift **convolution with a "delta" on graph**

$$(T_i f)(n) := \sqrt{N}(f * \delta_i)(n) = \sqrt{N} \sum_{\ell=0}^{N-1} \hat{f}(\ell)\chi_\ell^*(i)\chi_\ell(n)$$

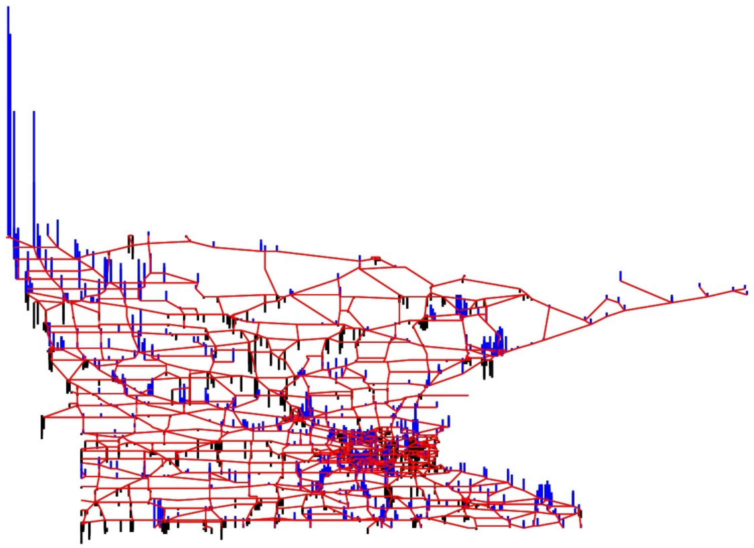
Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Modulation



classical modulation $(M_\xi f)(t) := e^{j2\pi\xi t} f(t)$

Modulation



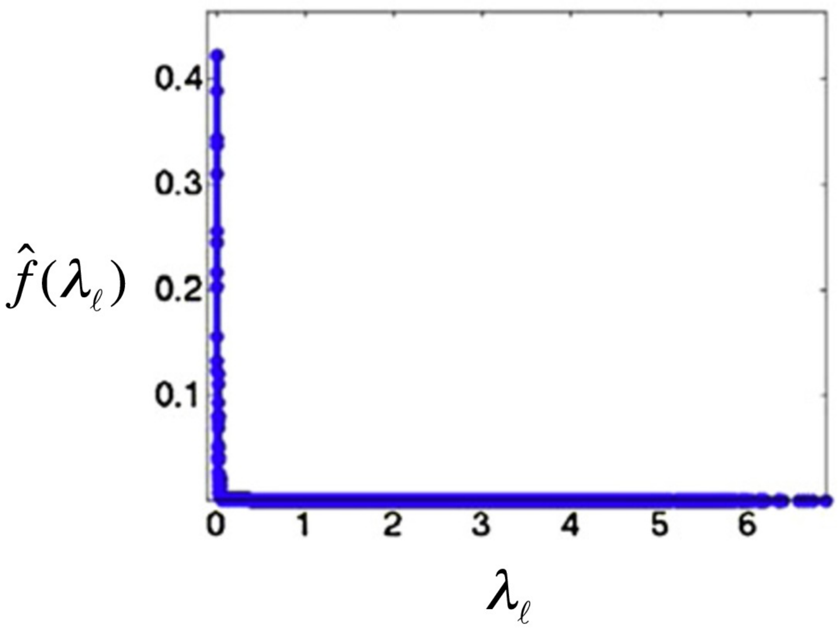
classical modulation $(M_\xi f)(t) := e^{j2\pi\xi t} f(t)$

graph modulation **multiply by a graph Laplacian eigenvector**

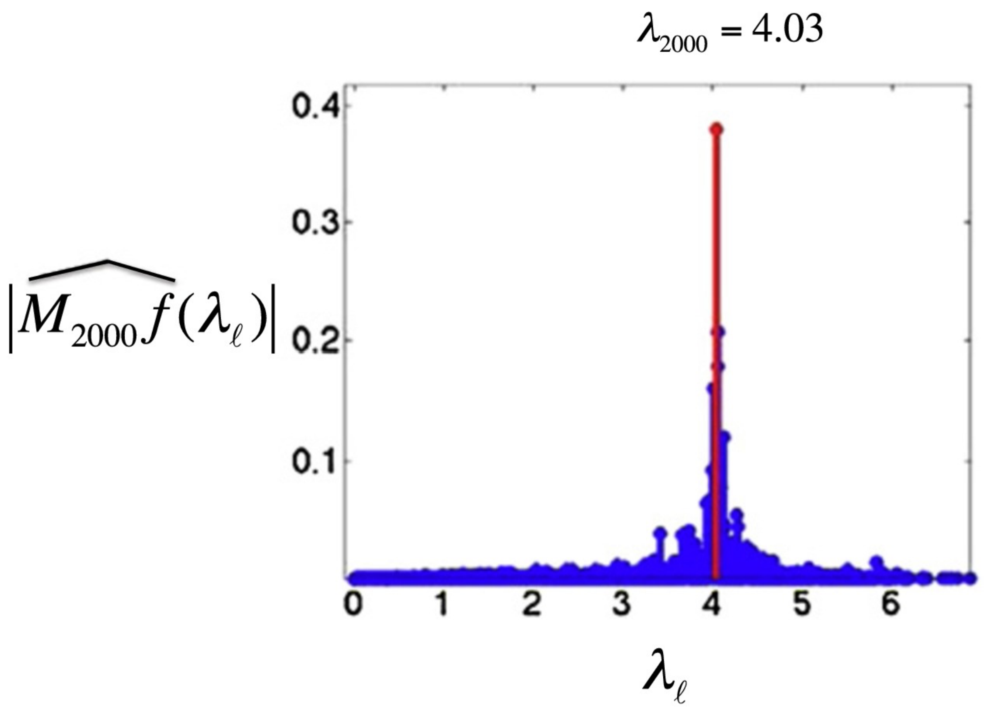
$$(M_k f)(n) := \sqrt{N} f(n) \chi_k(n)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Modulation



original signal



modulated signal

classical modulation $(M_\xi f)(t) := e^{j2\pi\xi t} f(t)$

graph modulation **multiply by a graph Laplacian eigenvector**

$$(M_k f)(n) := \sqrt{N} f(n) \chi_k(n)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Windowed graph Fourier transform

- With the shift and modulation operators for graph signals we can define a windowed graph Fourier transform (WGFT)

classical windowed Fourier atom

$$g_{u,\xi}(t) := (M_\xi T_u g)(t) = e^{j2\pi\xi t} g(t-u)$$

Windowed graph Fourier transform

- With the shift and modulation operators for graph signals we can define a windowed graph Fourier transform (WGFT)

classical windowed Fourier atom

$$g_{u,\xi}(t) := (M_\xi T_u g)(t) = e^{j2\pi\xi t} g(t-u)$$

windowed graph Fourier atom

$$\begin{aligned} g_{i,k}(n) &:= (M_k T_i g)(n) \\ &= N\chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell)\chi_\ell^*(i)\chi_\ell(n) \end{aligned}$$

Windowed graph Fourier transform

- With the shift and modulation operators for graph signals we can define a windowed graph Fourier transform (WGFT)

classical windowed Fourier atom

$$g_{u,\xi}(t) := (M_\xi T_u g)(t) = e^{j2\pi\xi t} g(t-u)$$

windowed graph Fourier atom

$$g_{i,k}(n) := (M_k T_i g)(n)$$

$$= N\chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell)\chi_\ell^*(i)\chi_\ell(n)$$

Windowed graph Fourier transform

- With the shift and modulation operators for graph signals we can define a windowed graph Fourier transform (WGFT)

classical windowed Fourier atom

$$g_{u,\xi}(t) := (M_\xi T_u g)(t) = e^{j2\pi\xi t} g(t-u)$$

windowed graph Fourier atom

$$g_{i,k}(n) := (M_k T_i g)(n)$$

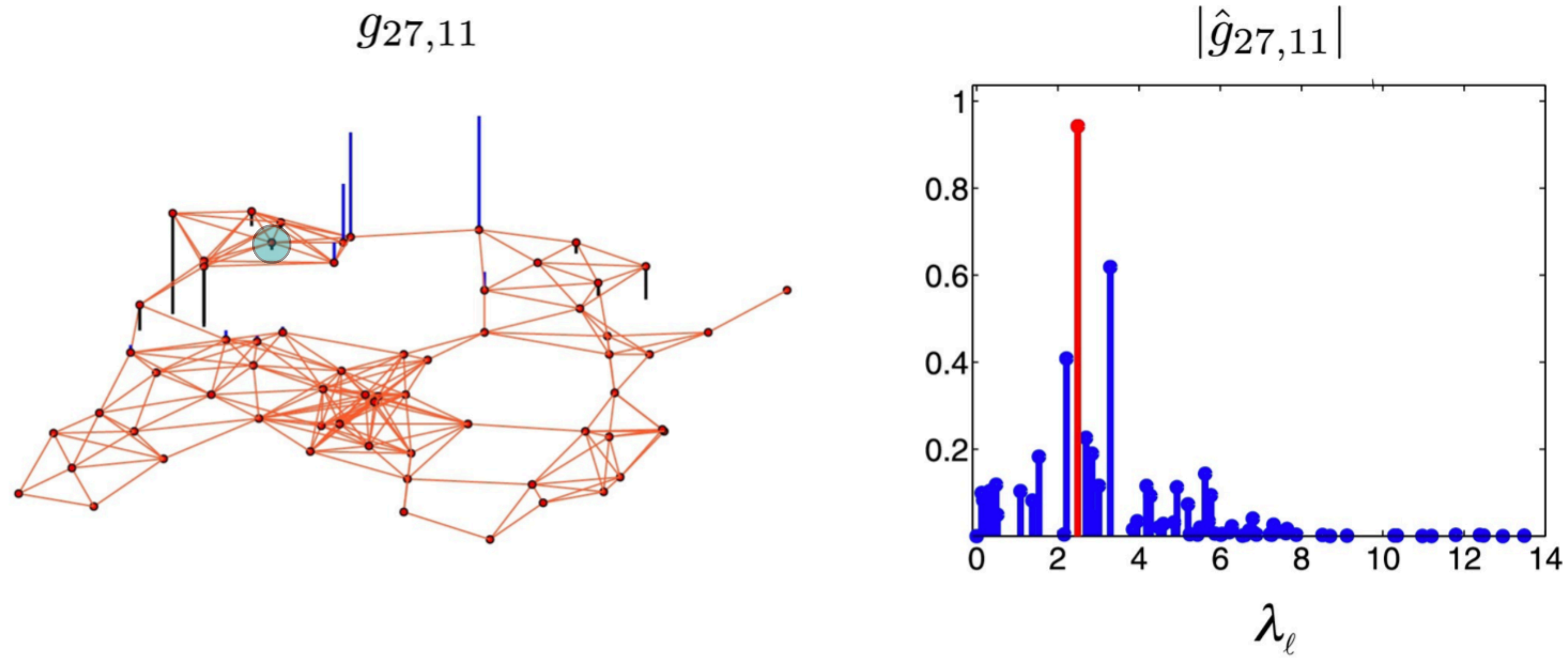
$$= N \chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$

windowed graph Fourier transform

$$Sf(i,k) := \langle f, g_{i,k} \rangle$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Windowed graph Fourier transform



$$\hat{g}(\lambda_\ell) = 1.45e^{-3\lambda_\ell}$$

Wavelets on graphs

- With the shift and scaling operators for graph signals we can define a spectral graph wavelet transform (SGWT)

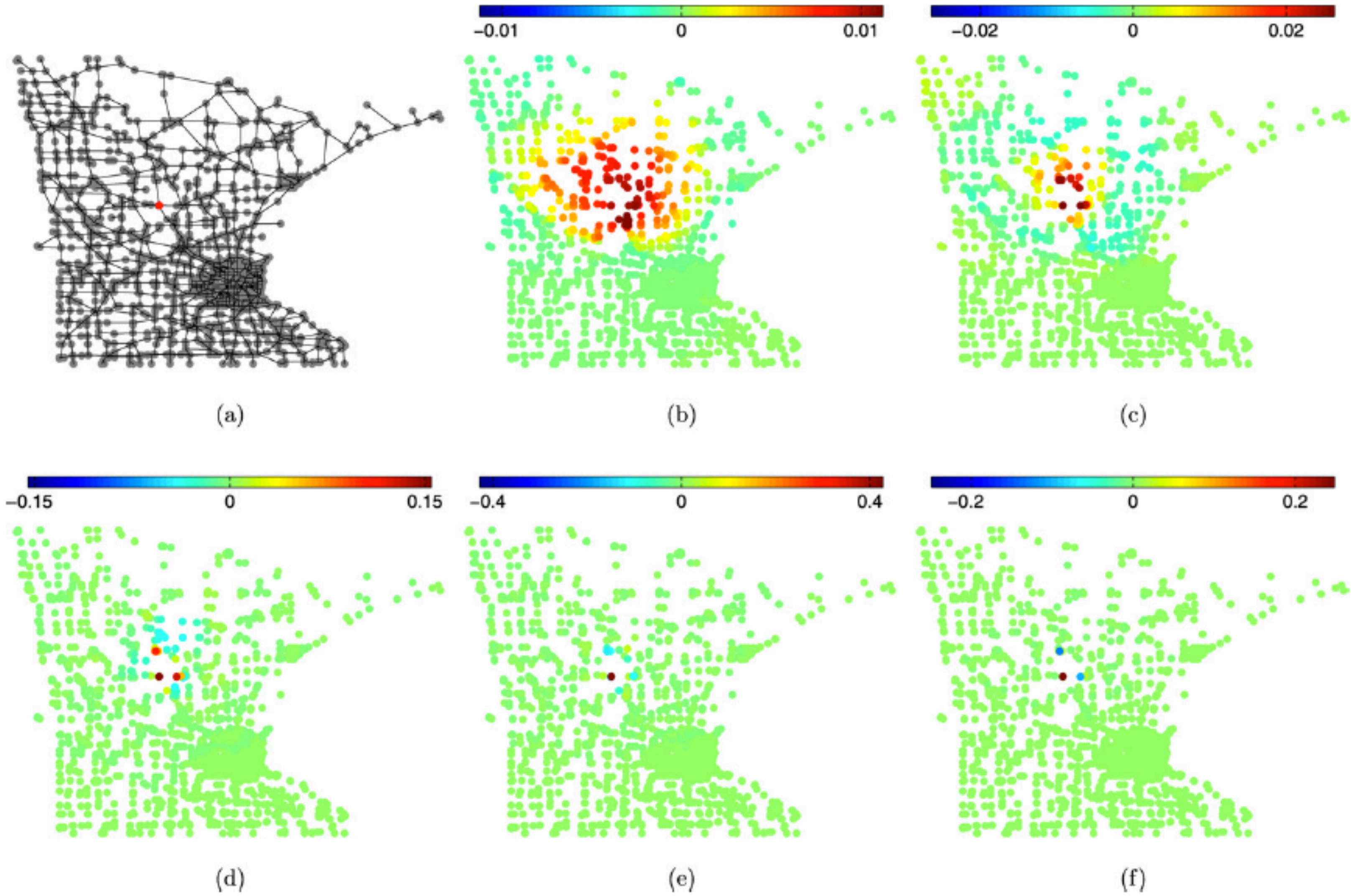


Fig. 4. Spectral graph wavelets on Minnesota road graph, with $K = 100$, $J = 4$ scales. (a) Vertex at which wavelets are centered, (b) scaling function, (c)–(f) wavelets, scales 1–4.

Hammond et al., "Wavelets on graphs via spectral graph theory," ACHA, 2011.

Wavelets on graphs

- With the shift and scaling operators for graph signals we can define a spectral graph wavelet transform (SGWT)

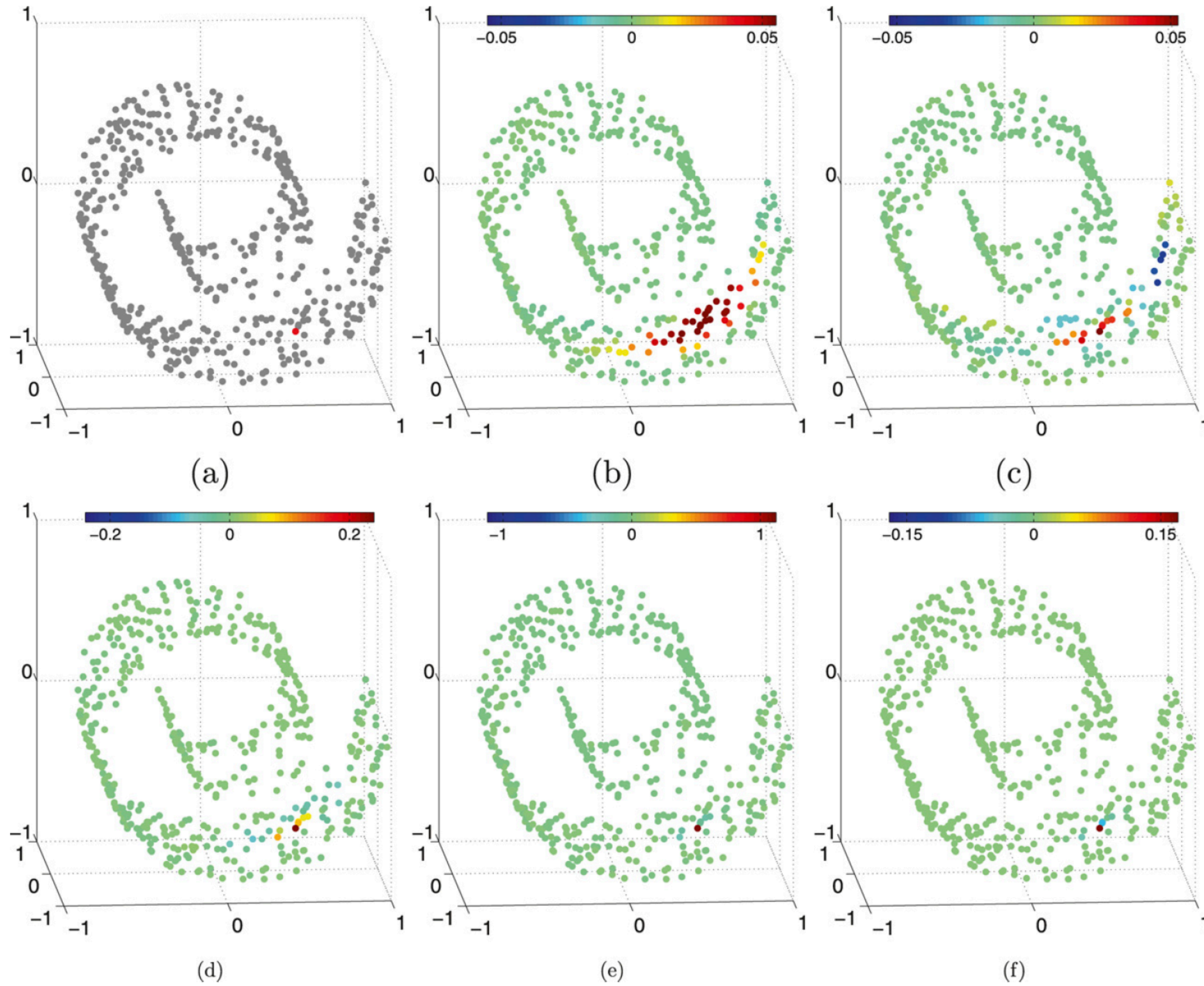


Fig. 3. Spectral graph wavelets on Swiss roll data cloud, with $J = 4$ wavelet scales. (a) Vertex at which wavelets are centered, (b) scaling function, (c)–(f) wavelets, scales 1–4.

Hammond et al., "Wavelets on graphs via spectral graph theory," ACHA, 2011.

WGFT vs SGWT atoms

WGFT atom

$$g_{i,k}(n) := (M_k T_i g)(n)$$

$$= N\chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell)\chi_\ell^*(i)\chi_\ell(n)$$

SGWT atom

$$\psi_{i,s}(n) := (T_i D_s g)(n)$$

$$= \sum_{\ell=0}^{N-1} \hat{g}(s\lambda_\ell)\chi_\ell^*(i)\chi_\ell(n)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

WGFT vs SGWT atoms

WGFT atom

$$g_{i,k}(n) := (M_k T_i g)(n)$$

$$= N\chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell)\chi_\ell^*(i)\chi_\ell(n)$$

SGWT atom

$$\psi_{i,s}(n) := (T_i D_s g)(n)$$

$$= \sum_{\ell=0}^{N-1} \hat{g}(s\lambda_\ell)\chi_\ell^*(i)\chi_\ell(n)$$

Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

WGFT vs SGWT atoms

WGFT atom

$g_{i,k}(n) := (M_k T_i g)(n)$

$$= N \chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$

SGWT atom

$\psi_{i,s}(n) := (T_i D_s g)(n)$

$$= \sum_{\ell=0}^{N-1} \hat{g}(s\lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$

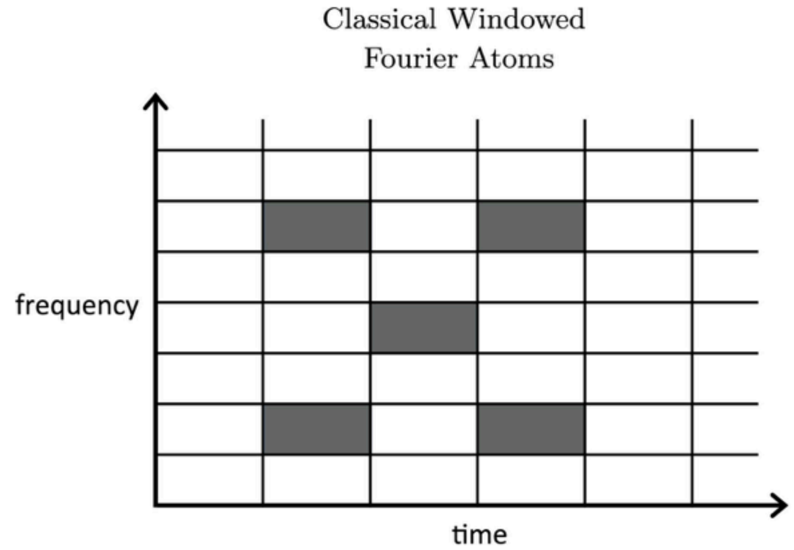
Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

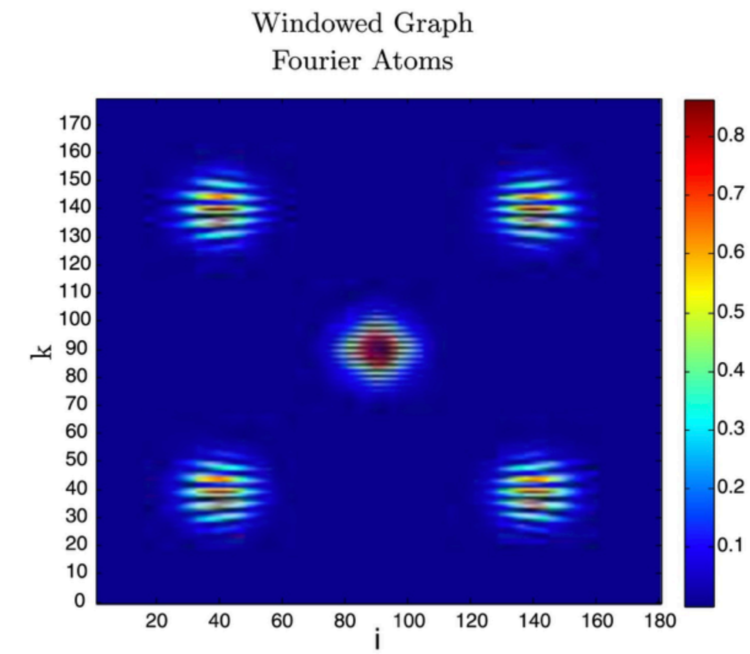
WGFT vs SGWT atoms

WGFT atom

$$g_{i,k}(n) := (M_k T_i g)(n)$$

$$= N \chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$

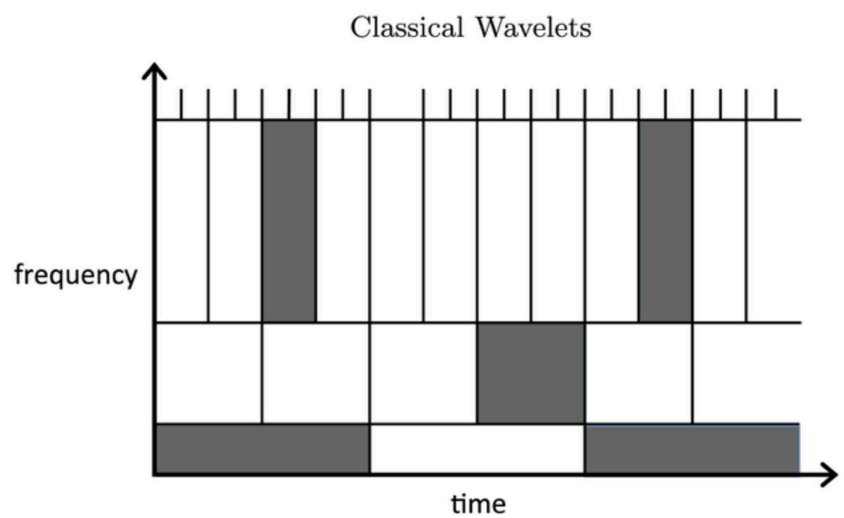


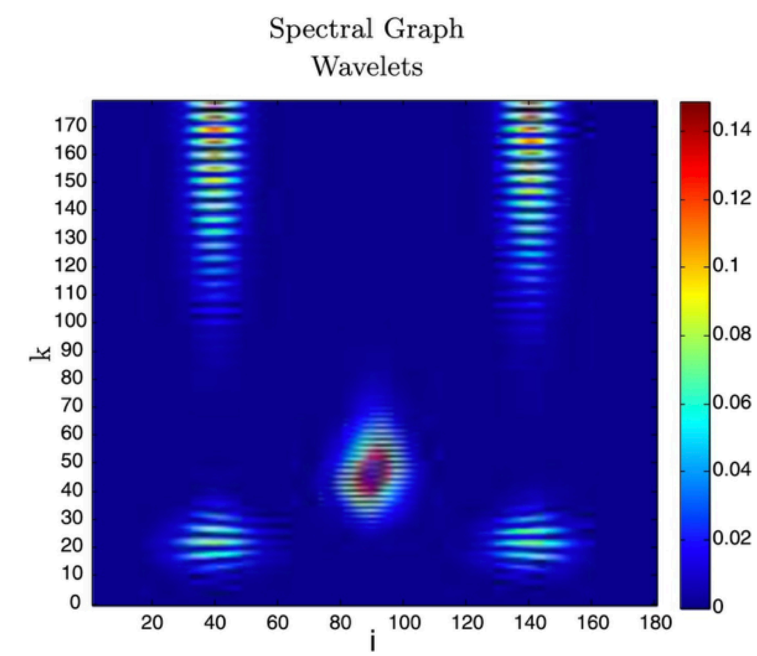


SGWT atom

$$\psi_{i,s}(n) := (T_i D_s g)(n)$$

$$= \sum_{\ell=0}^{N-1} \hat{g}(s\lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$





Shuman et al., "Vertex-frequency analysis on graphs," ACHA, 2016.

Outline

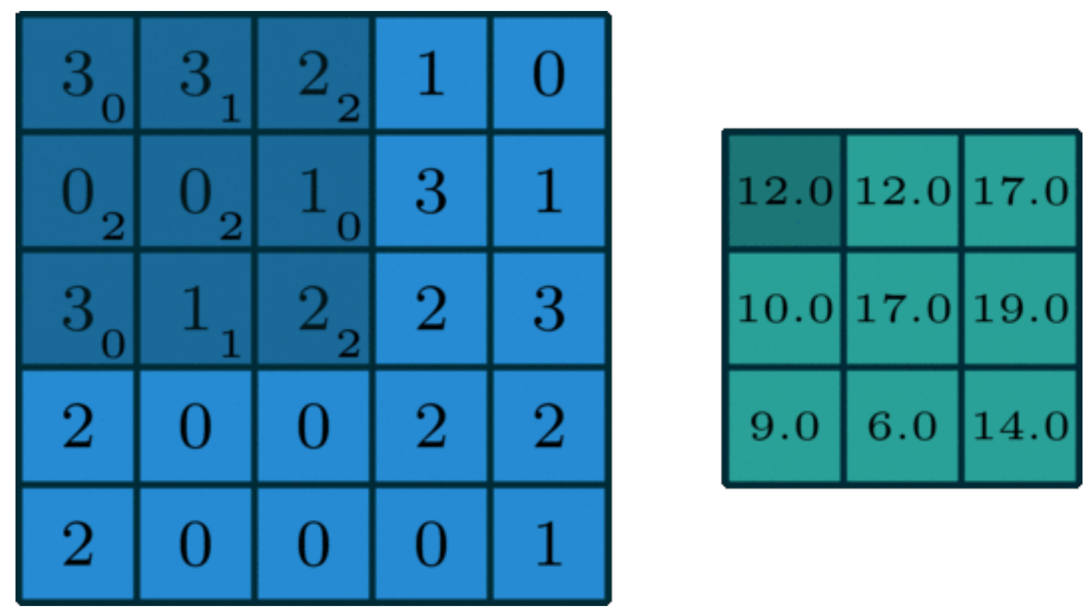
- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

Convolution on graphs (revisited)

classical convolution

time domain

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

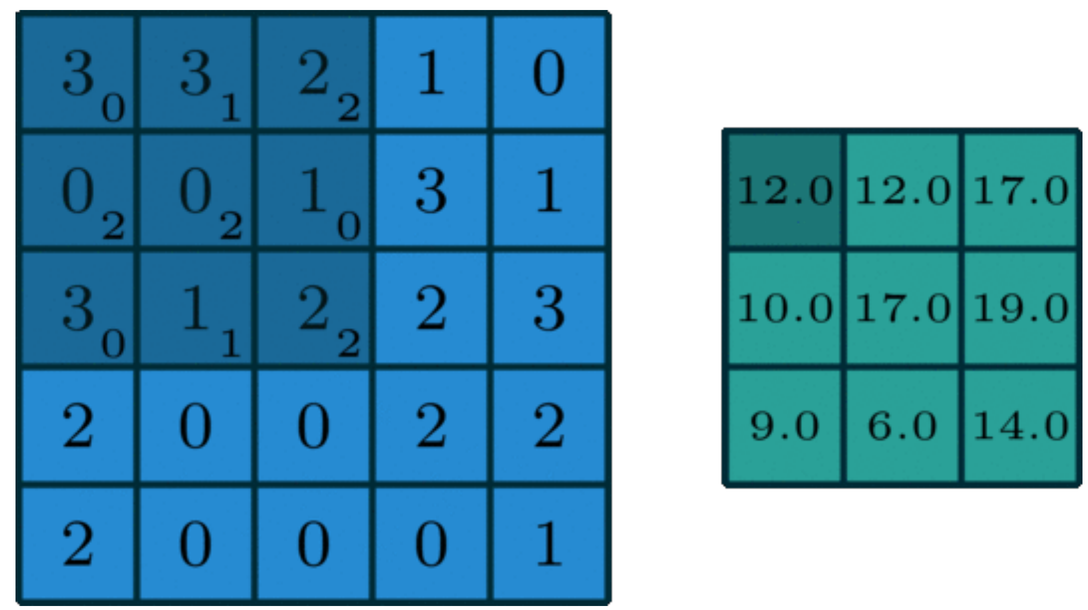


Convolution on graphs (revisited)

classical convolution

time domain

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$



Convolution on graphs (revisited)

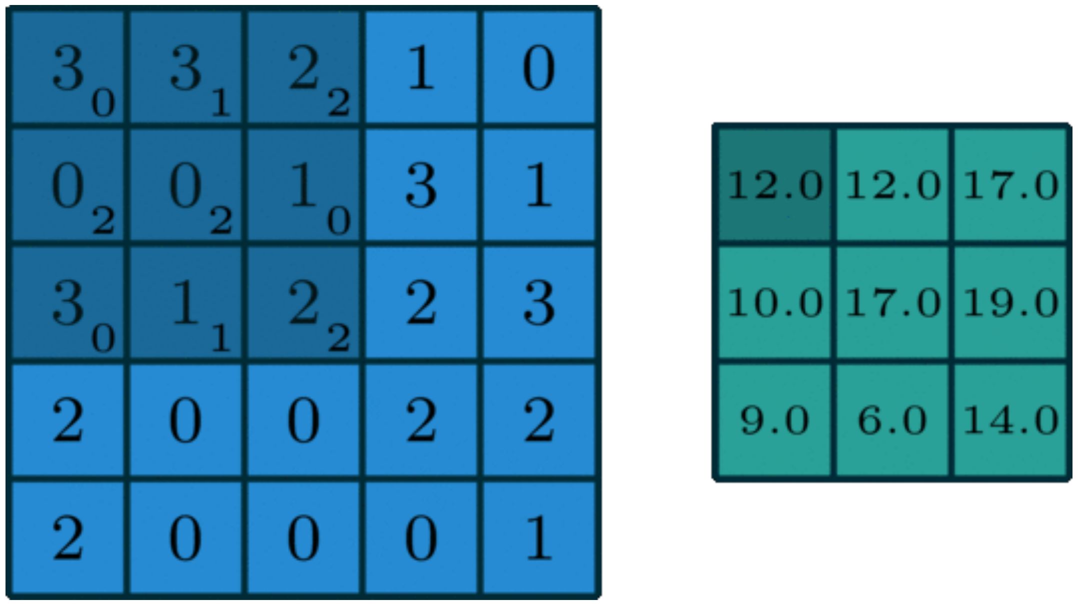
classical convolution

time domain

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

frequency domain

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$



Convolution on graphs (revisited)

classical convolution

time domain

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

frequency domain

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

convolution on graphs

graph spectral domain

$$\widehat{(f * g)}(\lambda) = \big((\chi^T f) \circ \hat{g}\big)(\lambda)$$

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution on graphs (revisited)

classical convolution

time domain

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

frequency domain

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

convolution on graphs

spatial (node) domain

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

graph spectral domain

$$\widehat{(f * g)}(\lambda) = \big((\chi^T f) \circ \hat{g}\big)(\lambda)$$

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution on graphs (revisited)

classical convolution

time domain

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

frequency domain

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

convolution on graphs

spatial (node) domain

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L)f$$

convolution = filtering

graph spectral domain

$$\widehat{(f * g)}(\lambda) = \big((\chi^T f) \circ \hat{g}\big)(\lambda)$$

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

A parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

parametric filter as polynomial of Laplacian

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \; \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

A parametric filter

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

⇓

parametric filter as polynomial of Laplacian

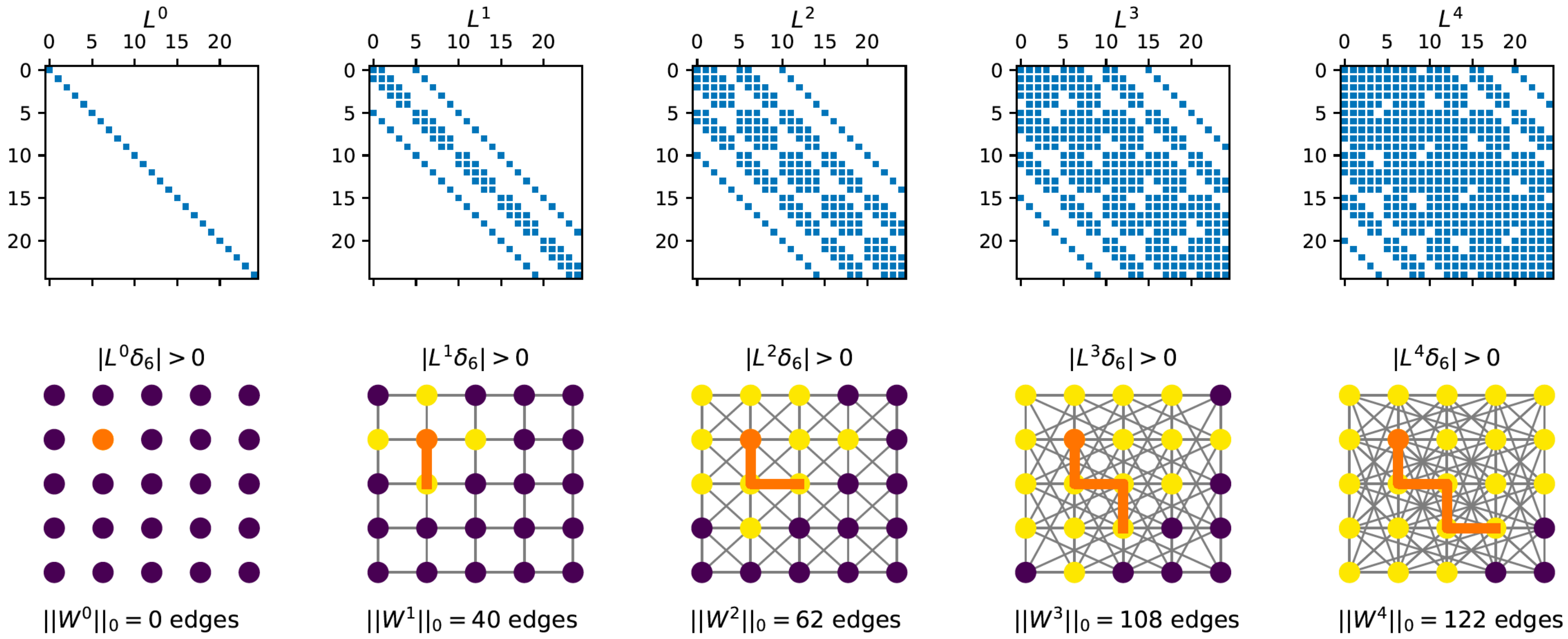
$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \; \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

what do powers of graph Laplacian capture?

Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

Powers of graph Laplacian

L^k defines the k-neighborhood



Localization: $d_{\mathcal{G}}(v_i, v_j) > K$ implies $(L^K)_{ij} = 0$

(slide by Michaël Deferrard)

Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

A parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

⬇

parametric filter as polynomial of Laplacian

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

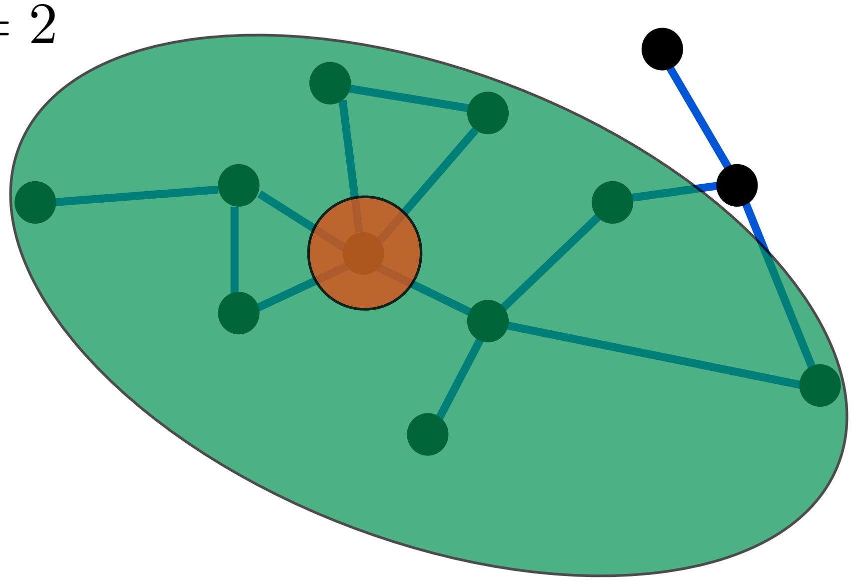
A parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

parametric filter as polynomial of Laplacian

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

$K = 2$



- convolution is expressed in the graph spectral domain
- localisation within *K*-hop neighbourhood

Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

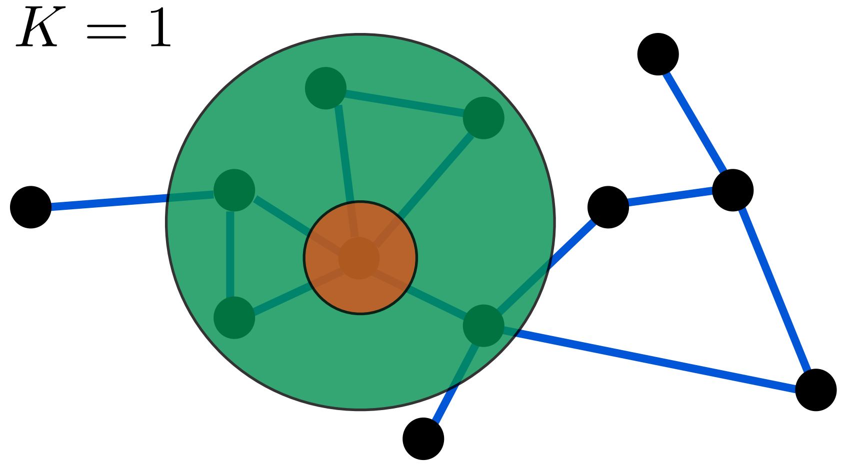
A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

simplified parametric filter

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j \quad \xrightarrow{K=1} \quad = \theta_0 I - \theta_1 (D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

(localisation within 1-hop neighbourhood)



A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

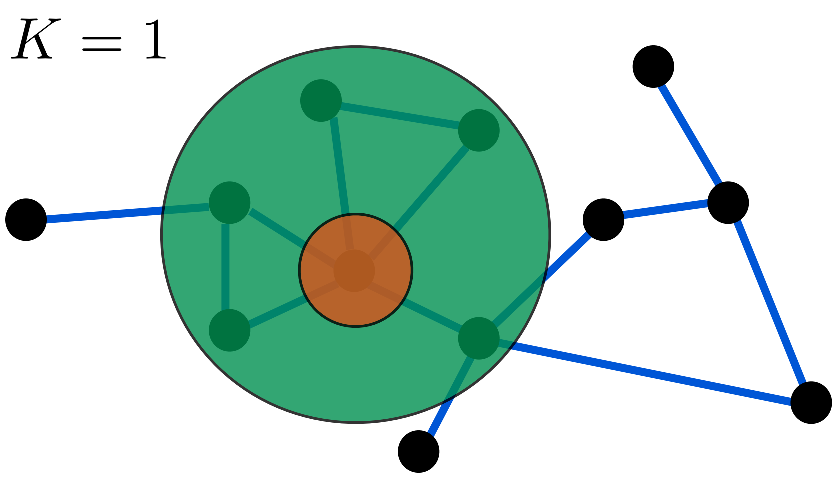
simplified parametric filter

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

$K = 1$

$$= \theta_0 I - \theta_1 (D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

(localisation within 1-hop neighbourhood)



$\alpha = \theta_0 = -\theta_1$

$$= \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

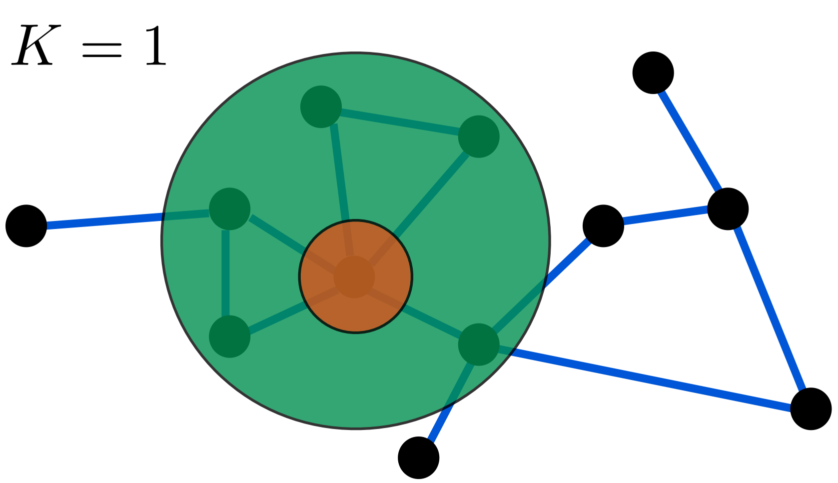
simplified parametric filter

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

$K = 1$

$$= \theta_0 I - \theta_1 (D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

(localisation within 1-hop neighbourhood)



$\alpha = \theta_0 = -\theta_1$

$$= \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

renormalisation

$$\Rightarrow \alpha(\tilde{D}^{-\frac{1}{2}} \tilde{W} \tilde{D}^{-\frac{1}{2}})$$

Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

⬇

simplified parametric filter

$$\hat{g}_\alpha(L) = \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

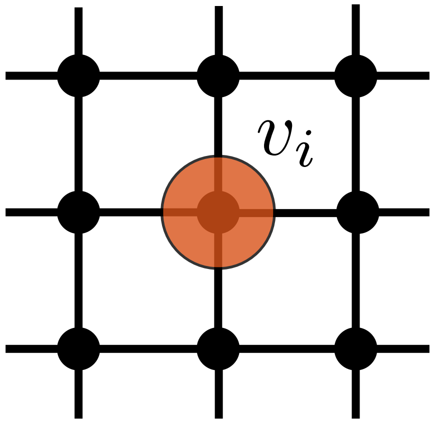
A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

simplified parametric filter

$$\hat{g}_\alpha(L) = \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

$$y_i = \alpha f_i + \alpha \frac{1}{\sqrt{d_i}} \sum_{j:(i,j)\in\mathcal{E}} w_{ij} \frac{1}{\sqrt{d_j}} f_j$$



Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

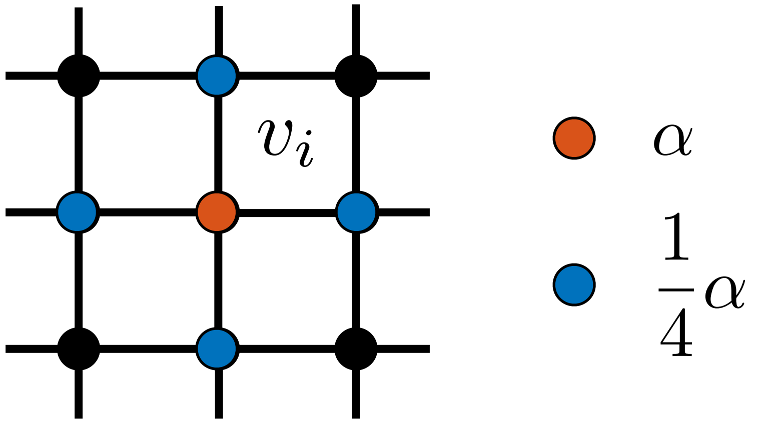
simplified parametric filter

$$\hat{g}_\alpha(L) = \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

$$y_i = \alpha f_i + \alpha \frac{1}{\sqrt{d_i}} \sum_{j:(i,j)\in\mathcal{E}} w_{ij} \frac{1}{\sqrt{d_j}} f_j$$

unitary edge weights

$$y_i = \alpha f_i + \frac{1}{4}\alpha \sum_{j:(i,j)\in\mathcal{E}} f_j$$



Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

A simplified parametric filter

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

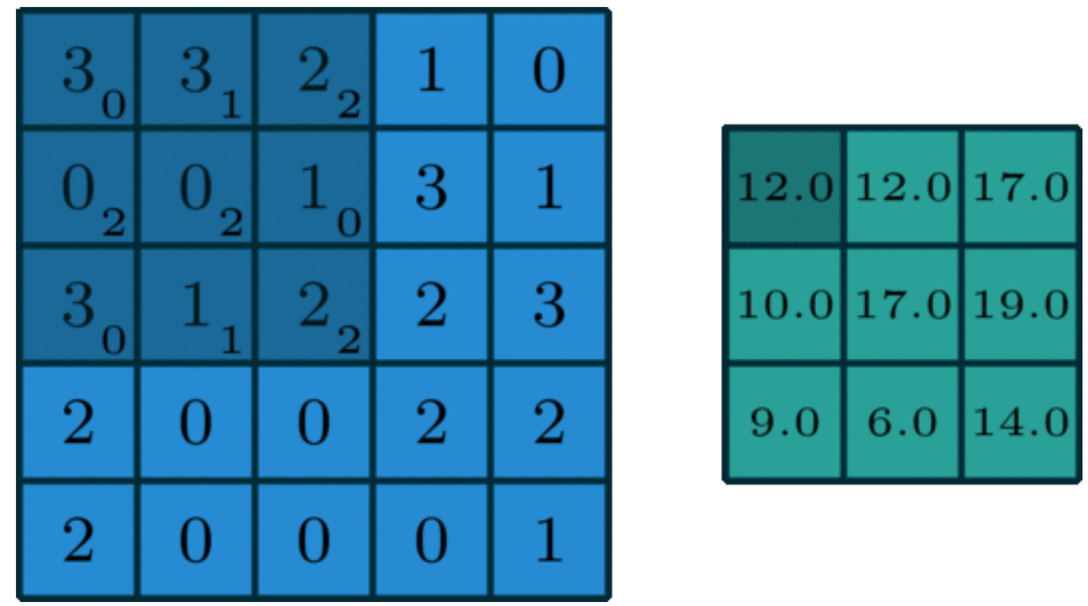
simplified parametric filter

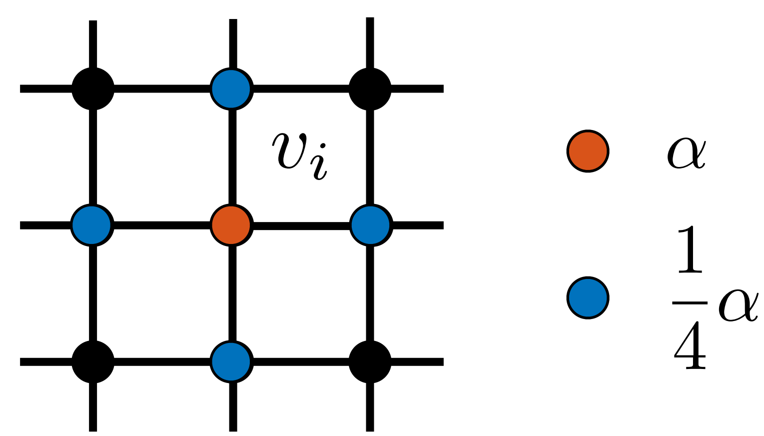
$$\hat{g}_\alpha(L) = \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

$$y_i = \alpha f_i + \alpha \frac{1}{\sqrt{d_i}} \sum_{j:(i,j)\in\mathcal{E}} w_{ij} \frac{1}{\sqrt{d_j}} f_j$$

unitary edge weights

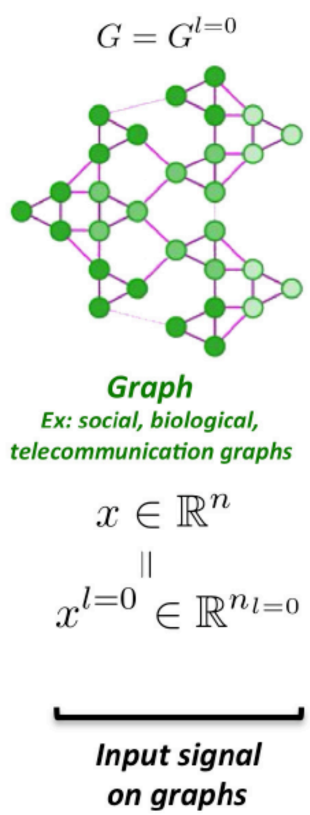
$$y_i = \alpha f_i + \frac{1}{4}\alpha \sum_{j:(i,j)\in\mathcal{E}} f_j$$





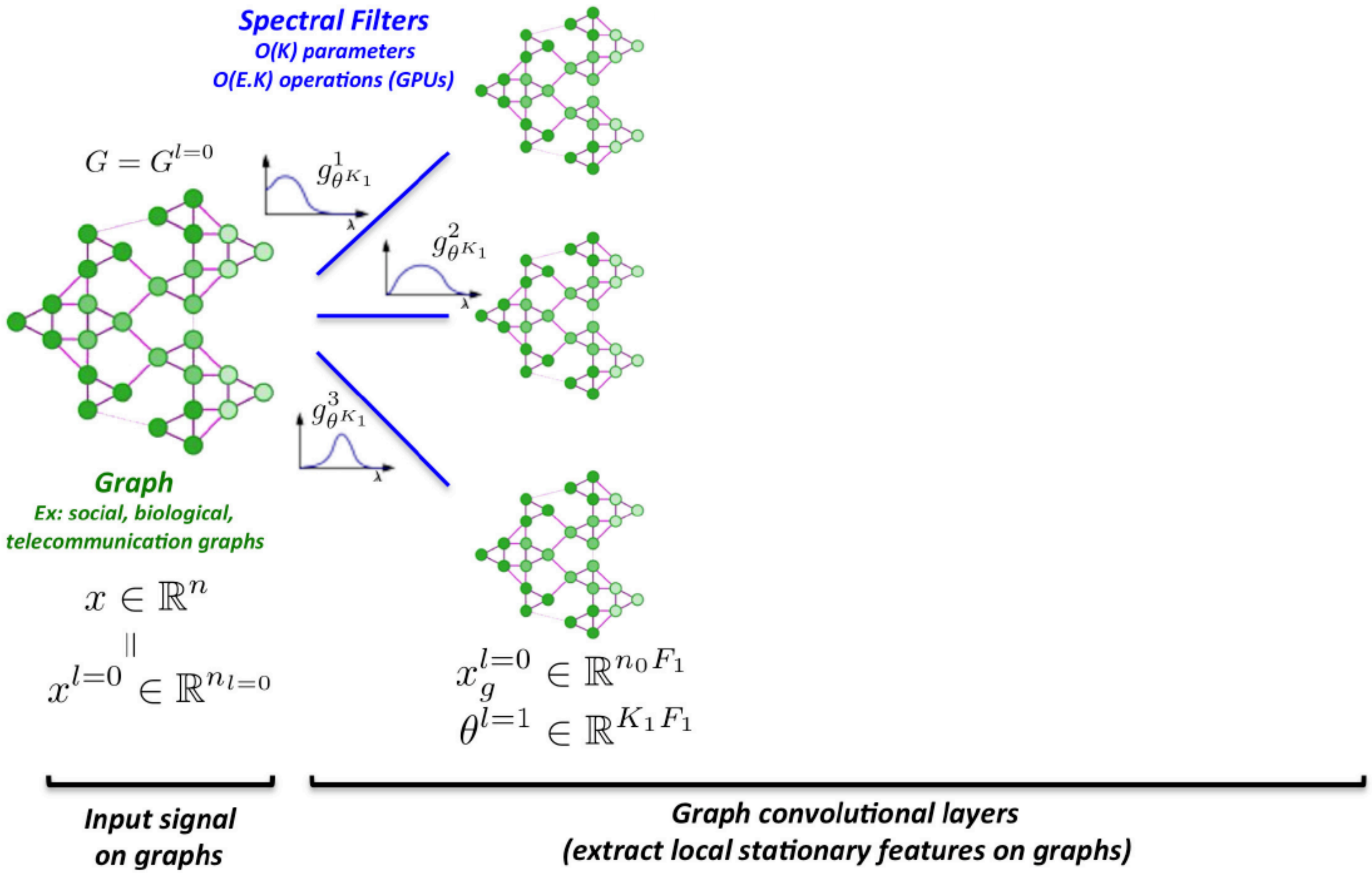
Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

CNN on graphs: Graph classification



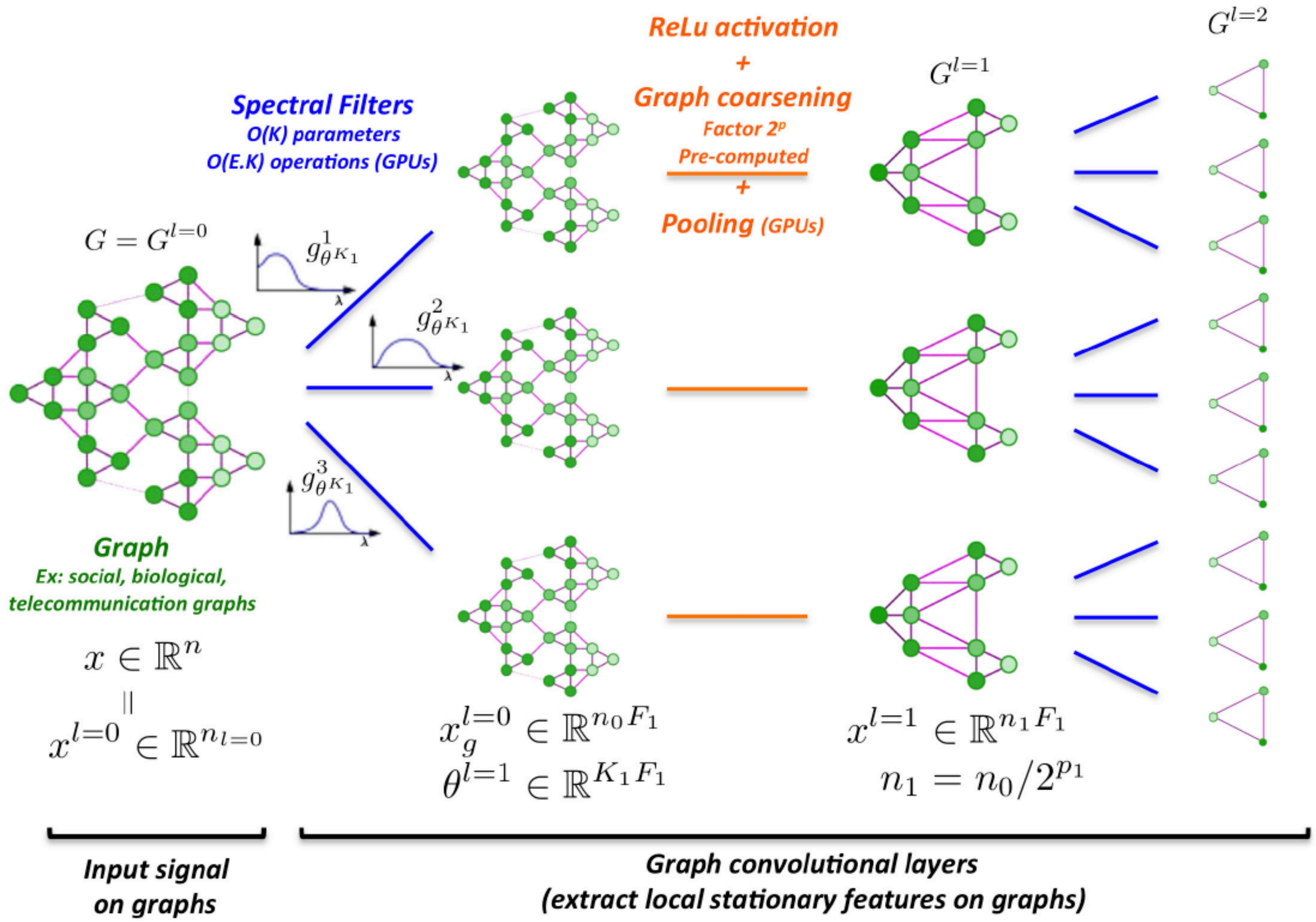
Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNN on graphs: Graph classification



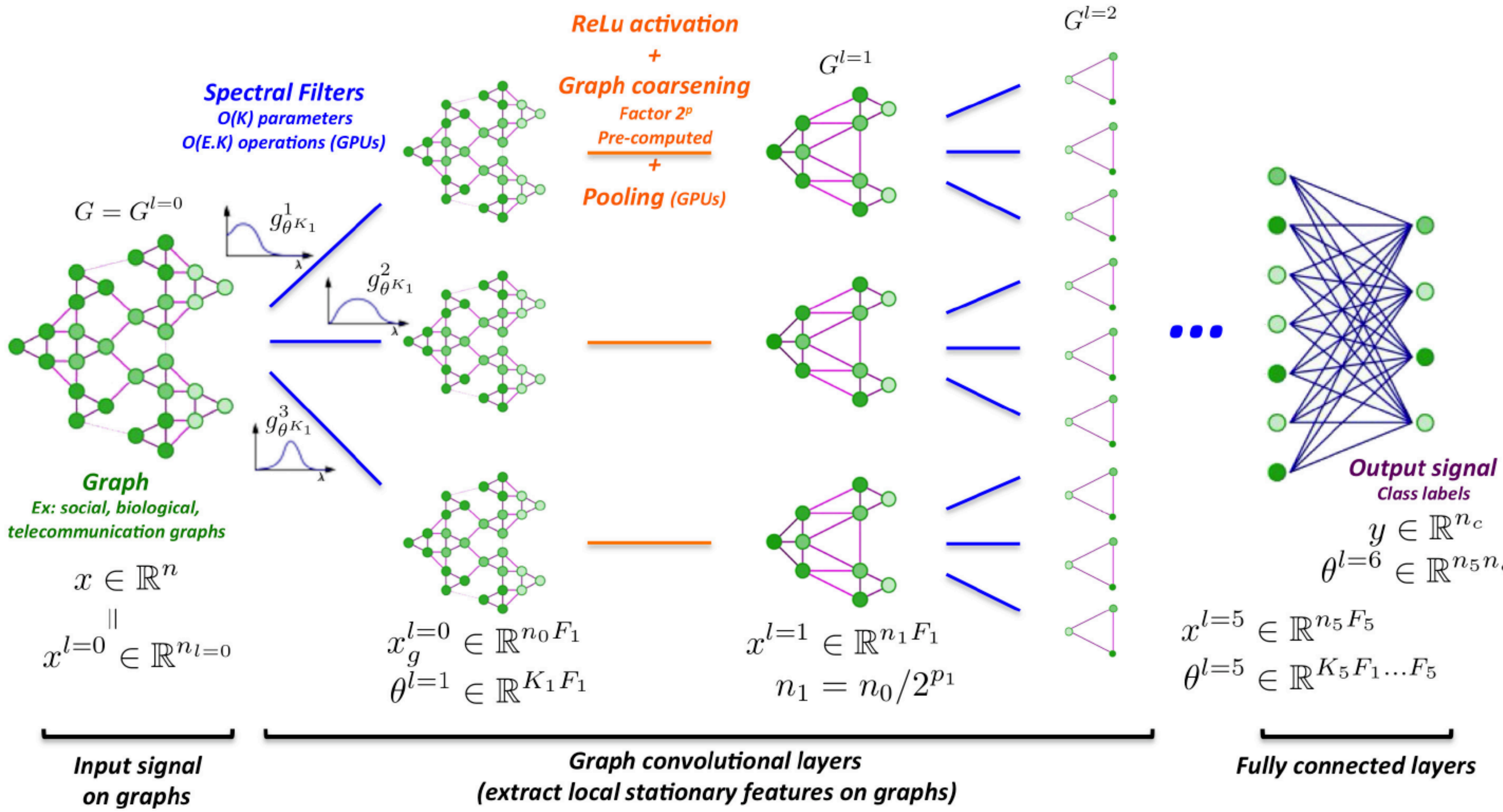
Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNN on graphs: Graph classification



Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNN on graphs: Graph classification



Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNN on graphs: Node classification

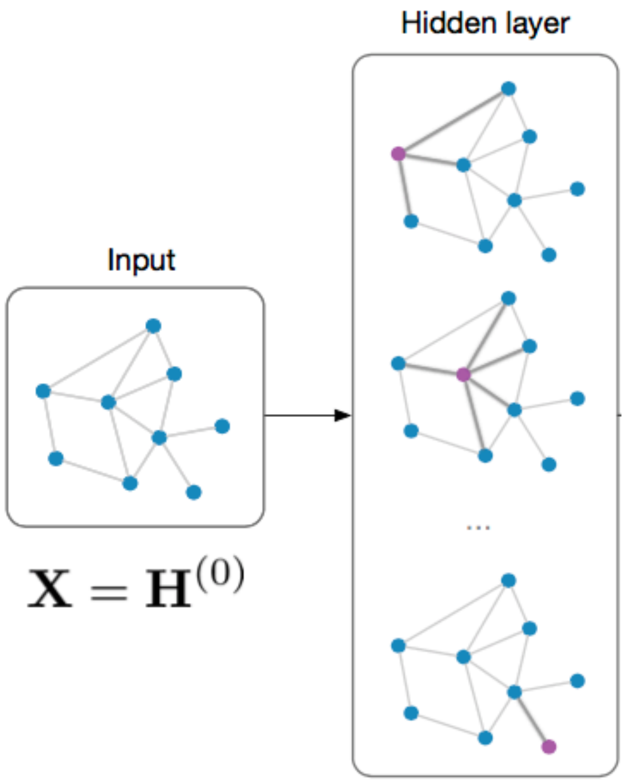
$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$

$\mathbf{X} = \mathbf{H}^{(0)}$

Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

CNN on graphs: Node classification

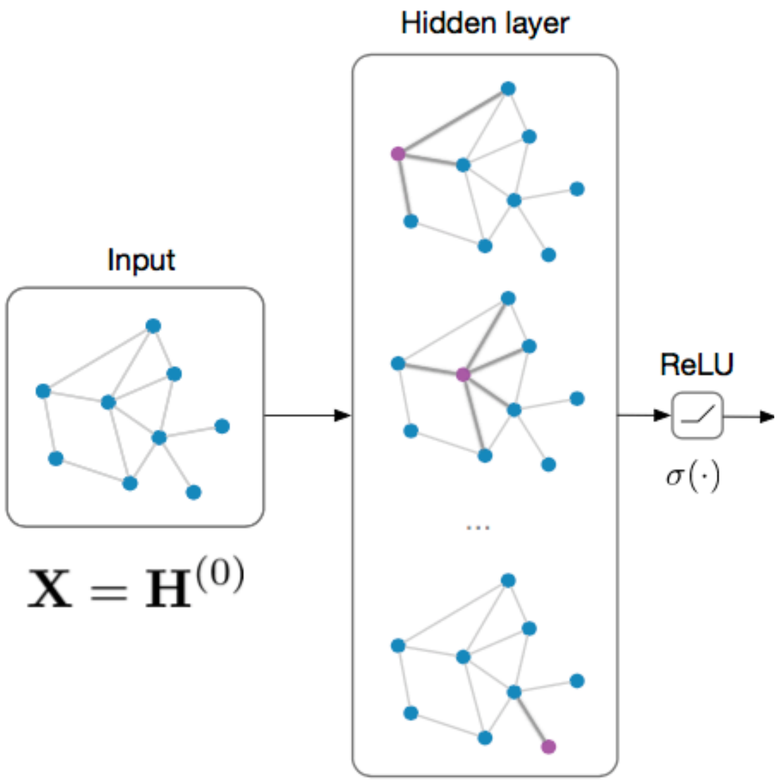
$$\hat{g}_{\theta^{(k+1)}}(L)\left(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\right)$$



Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

CNN on graphs: Node classification

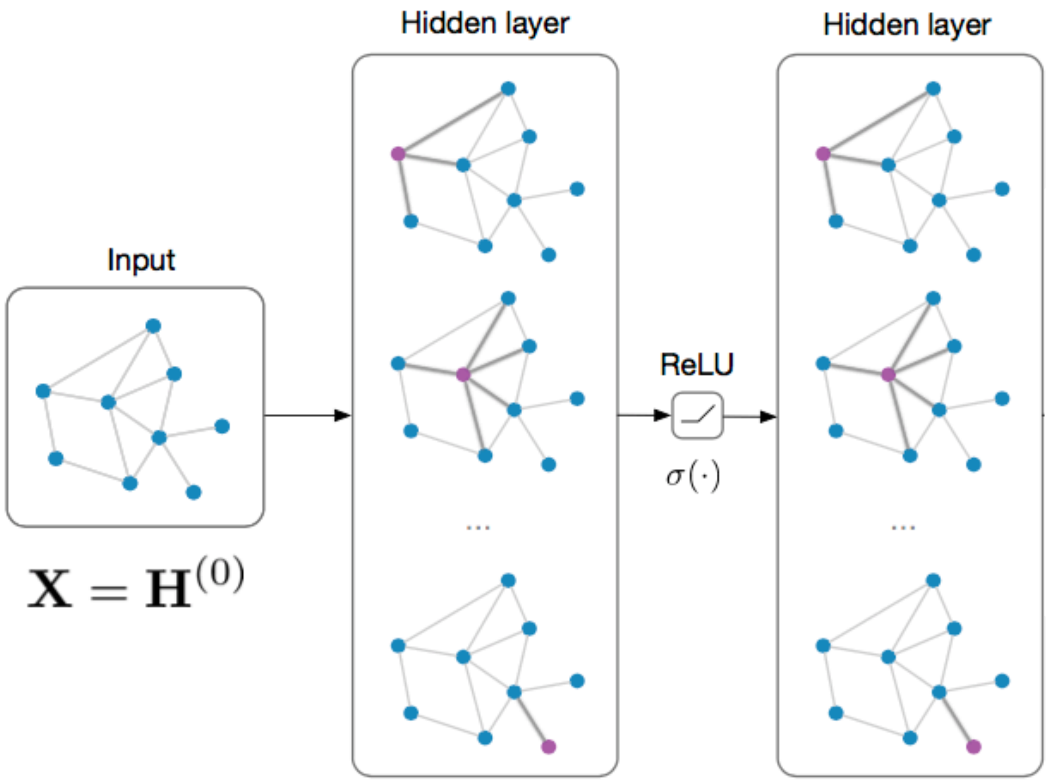
$$\hat{g}_{\theta^{(k+1)}}(L)\left(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\right)$$



Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

CNN on graphs: Node classification

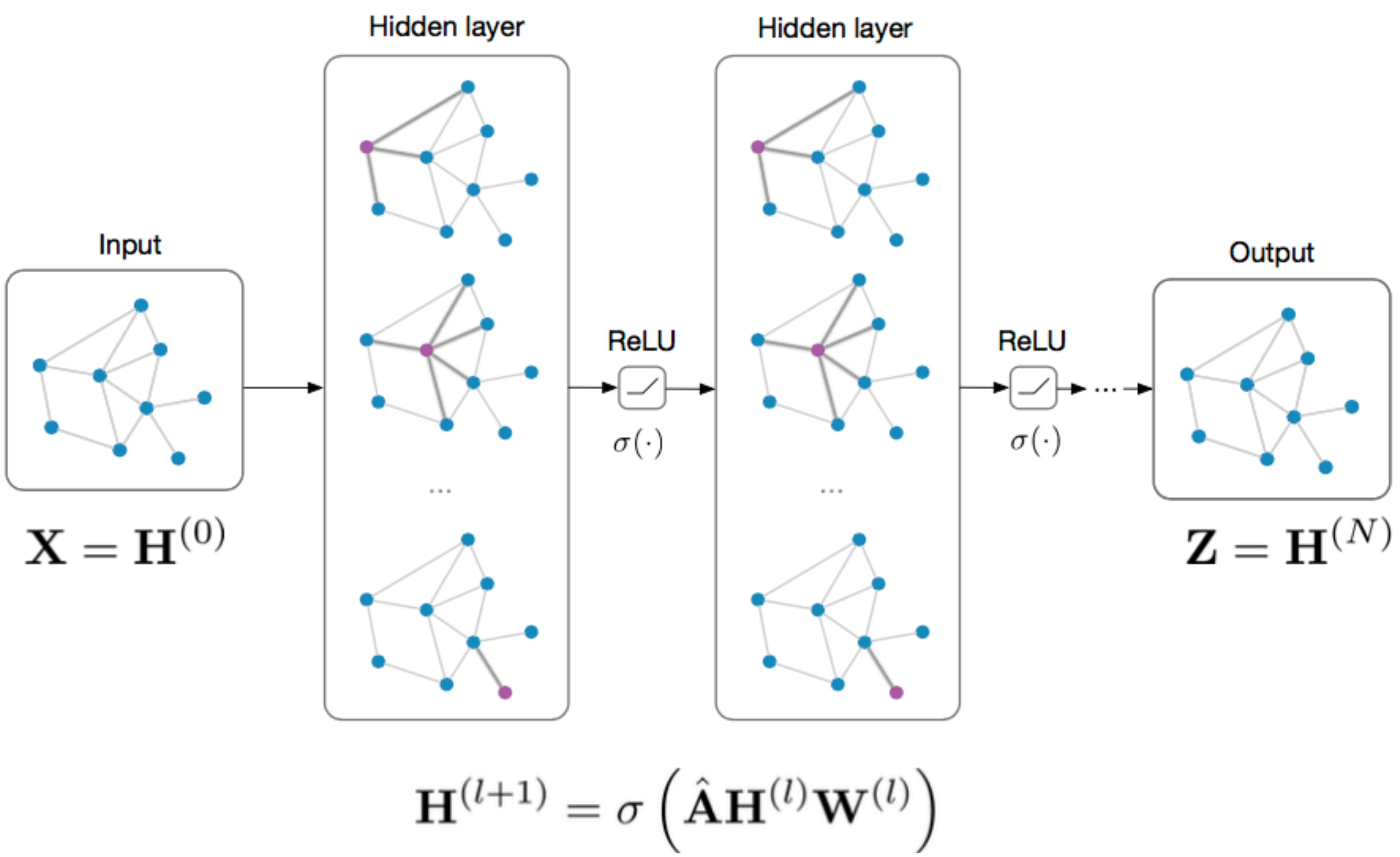
$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$



Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

CNN on graphs: Node classification

$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\text{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$

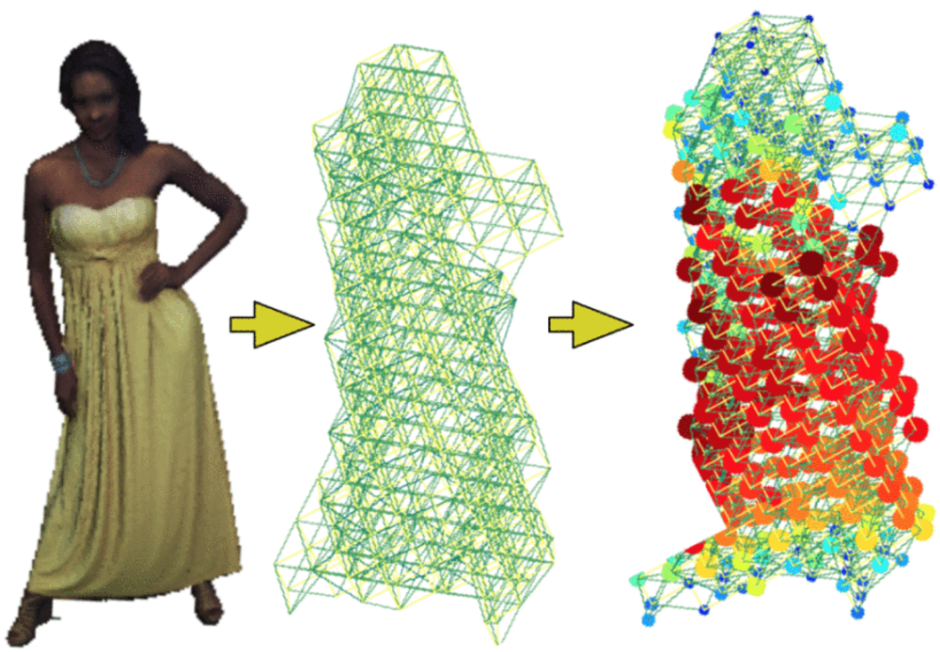


Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

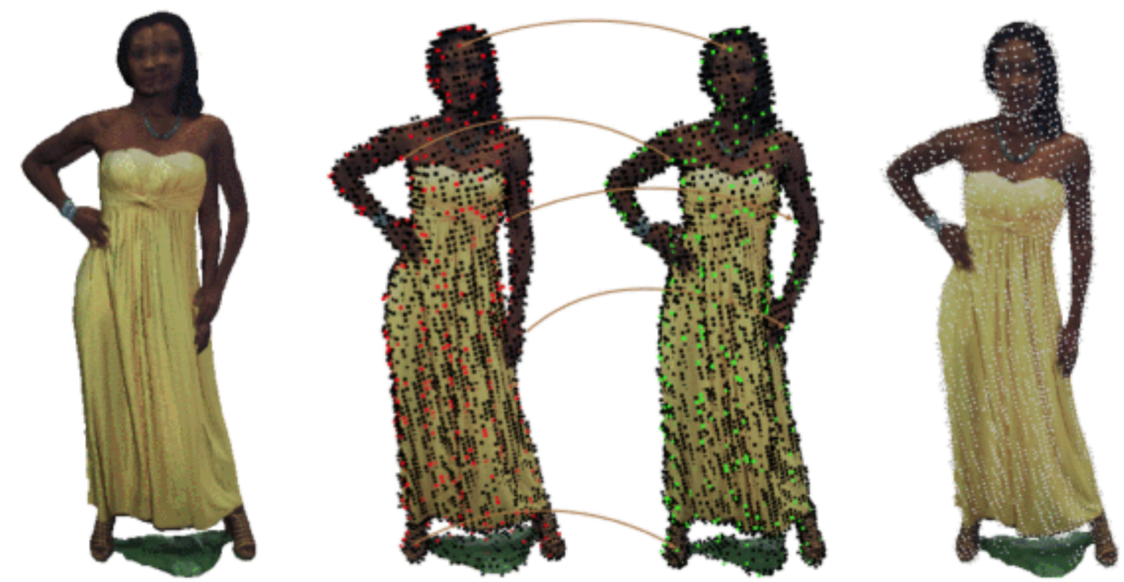
Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

Application I: 3D point cloud analysis



(a) (b) (c)



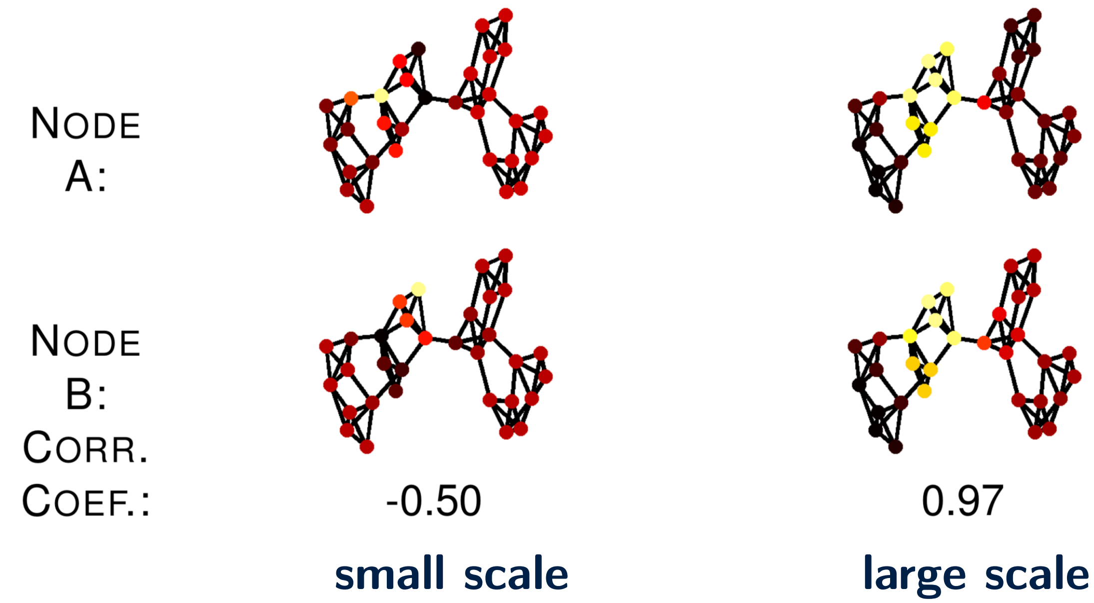
(a) $\mathcal{I}_t + \mathcal{I}_{t+1}$ (b) Correspondence between $\mathcal{I}_t$ and $\mathcal{I}_{t+1}$ (c) $\mathcal{I}_{t,mc} + \mathcal{I}_{t+1}$

Thanou et al., "Graph-based compression of dynamic 3D point cloud sequences," IEEE TIP, 2016.

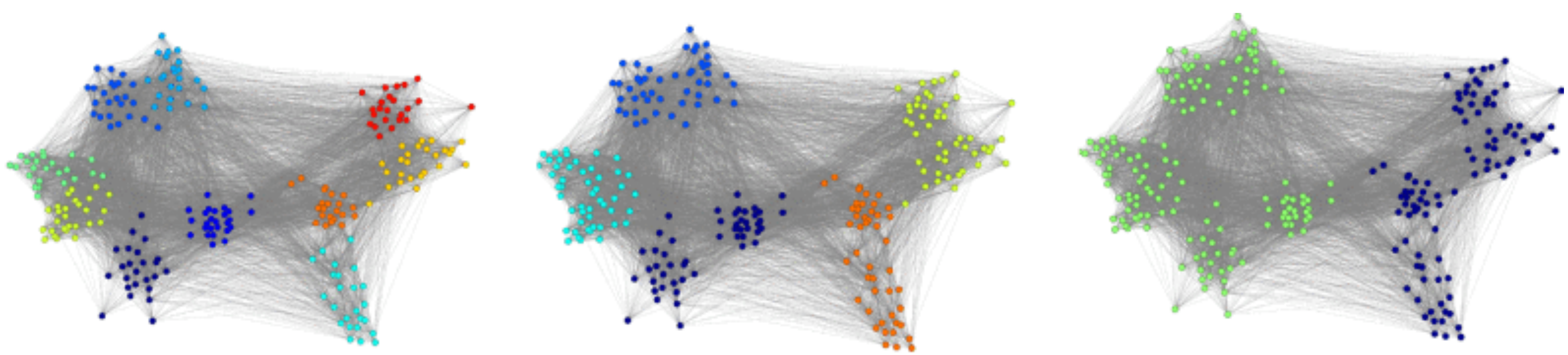
Application II: Community detection

spectral graph wavelets
at different scales:

$$D_s(a,b) = 1 - \frac{\psi_{s,a}^\top \psi_{s,b}}{\|\psi_{s,a}\|_2 \|\psi_{s,b}\|_2}.$$

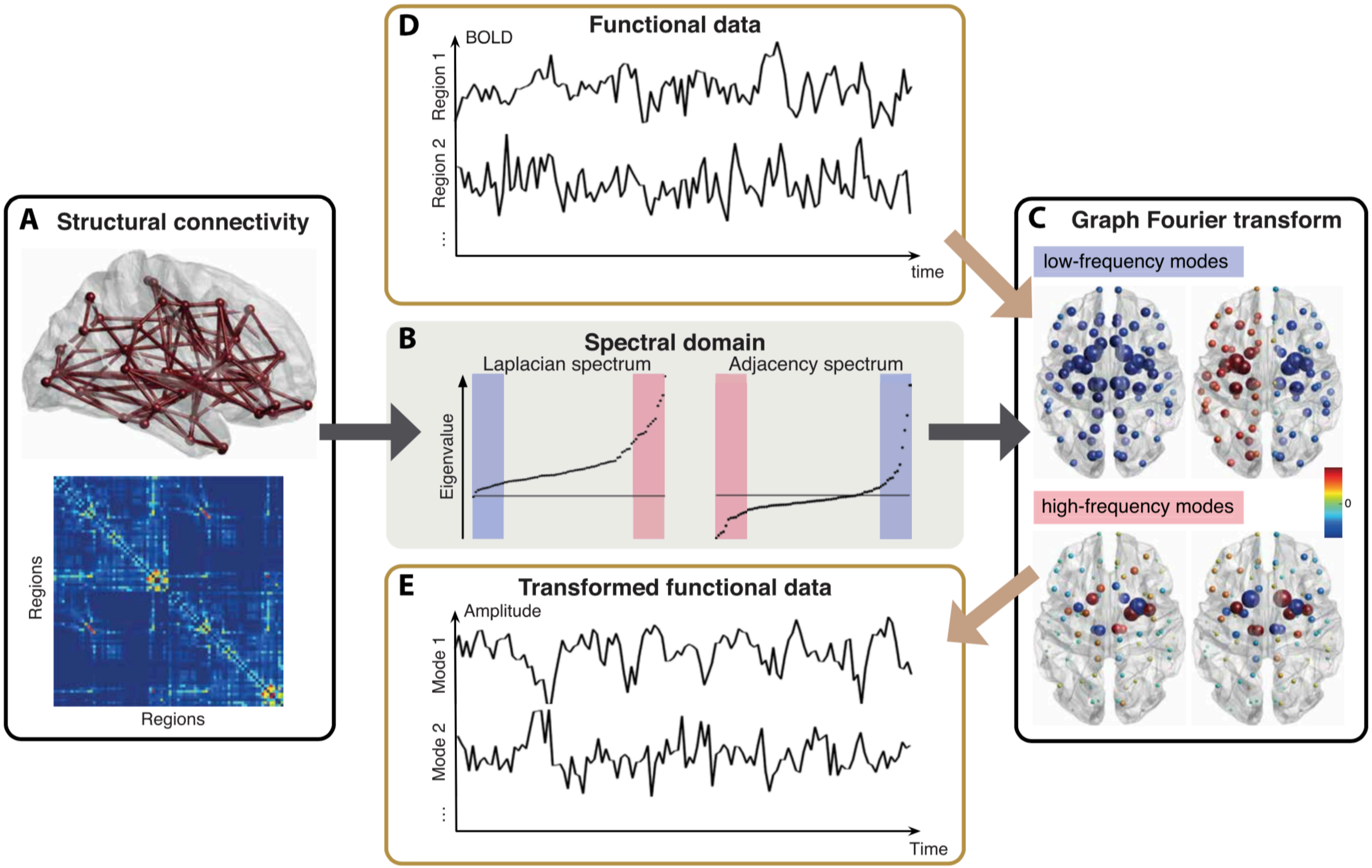


multi-scale community
detection:



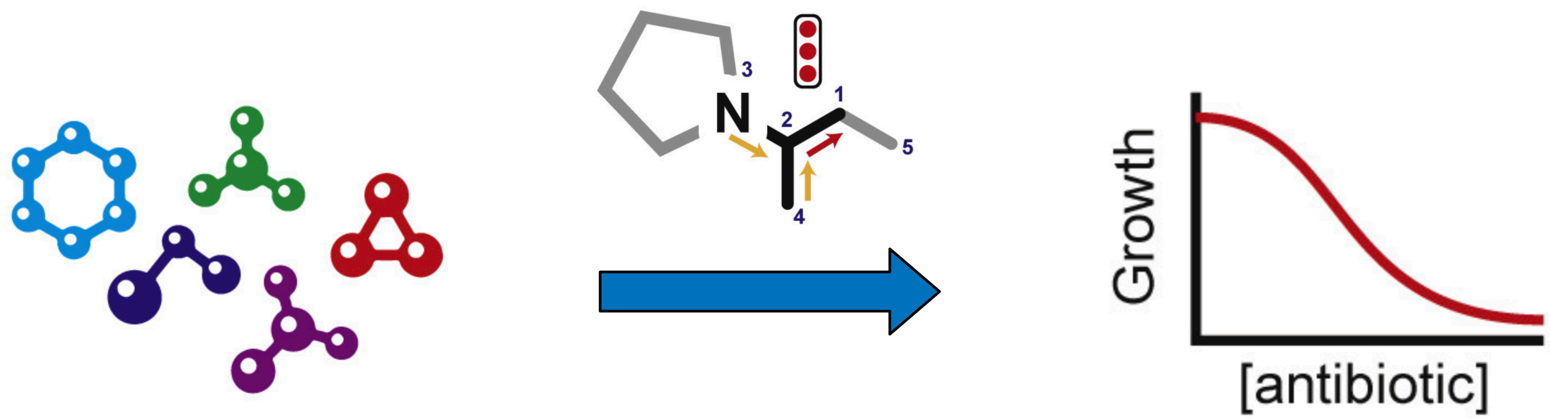
Tremblay et al., "Graph wavelets for multiscale community mining," IEEE TSP, 2014.

Application III: Neuroscience



Huang et al., "A graph signal processing perspective on functional brain imaging," Proceedings of the IEEE, 2018.

Application IV: Drug discovery

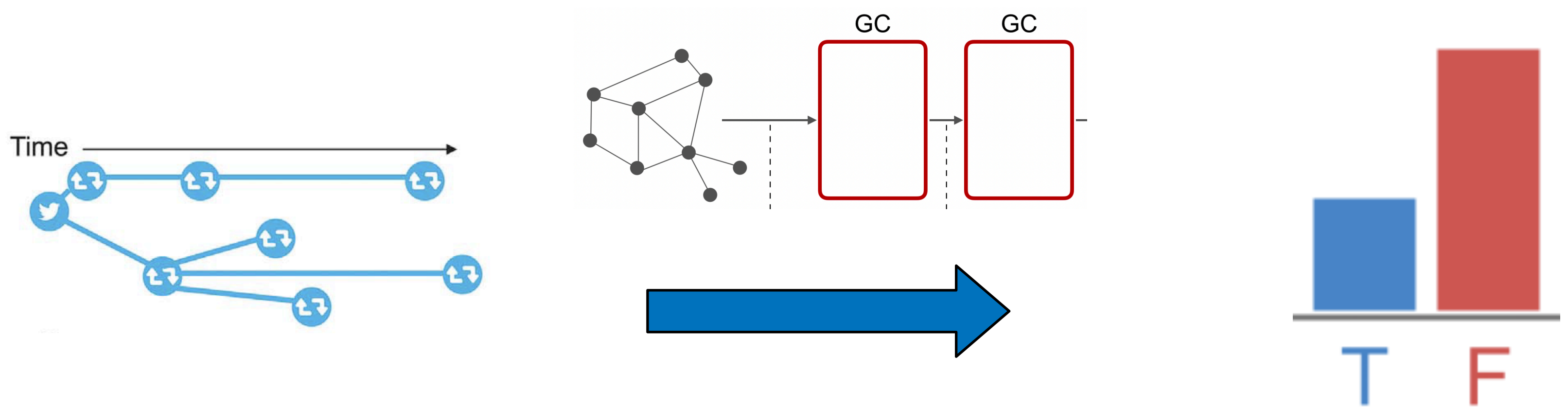


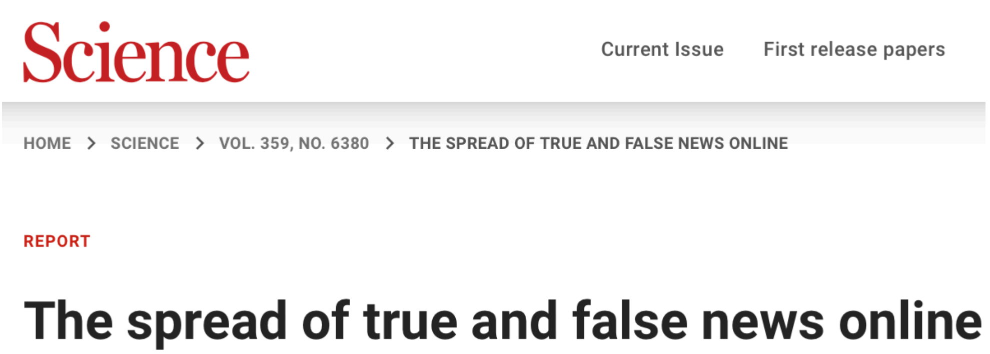

FINANCIAL TIMES
Artificial intelligence + Add to myFT
AI discovers antibiotics to treat drug-resistant diseases

BBC | Your account | Home | News | Sport
NEWS
Home | Coronavirus | Climate | UK | World | Business | Politics | Tech | Science
Health
Scientists discover powerful antibiotic using AI

Stokes et al., "A deep learning approach to antibiotic discovery," Cell, 2020.

Application V: Fake news detection



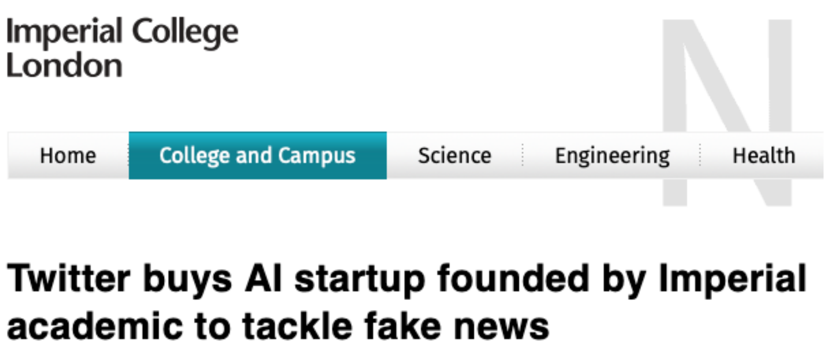

Science

Current Issue First release papers

HOME > SCIENCE > VOL. 359, NO. 6380 > THE SPREAD OF TRUE AND FALSE NEWS ONLINE

REPORT

The spread of true and false news online

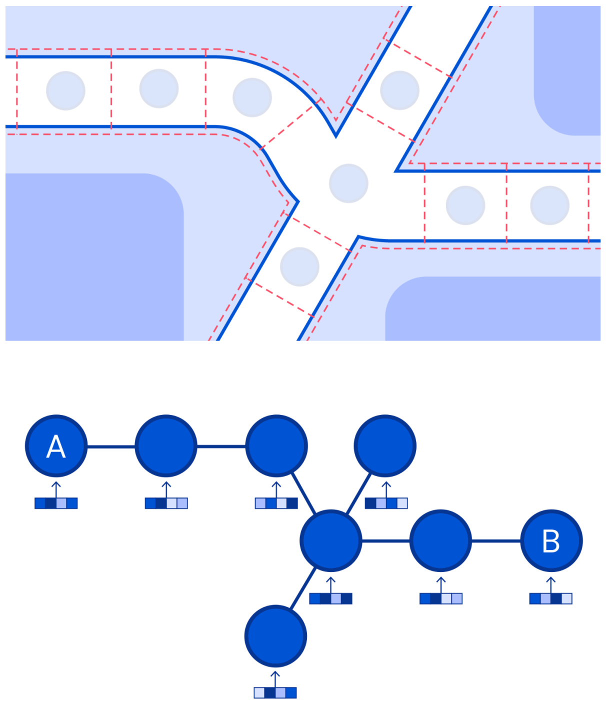

Imperial College London

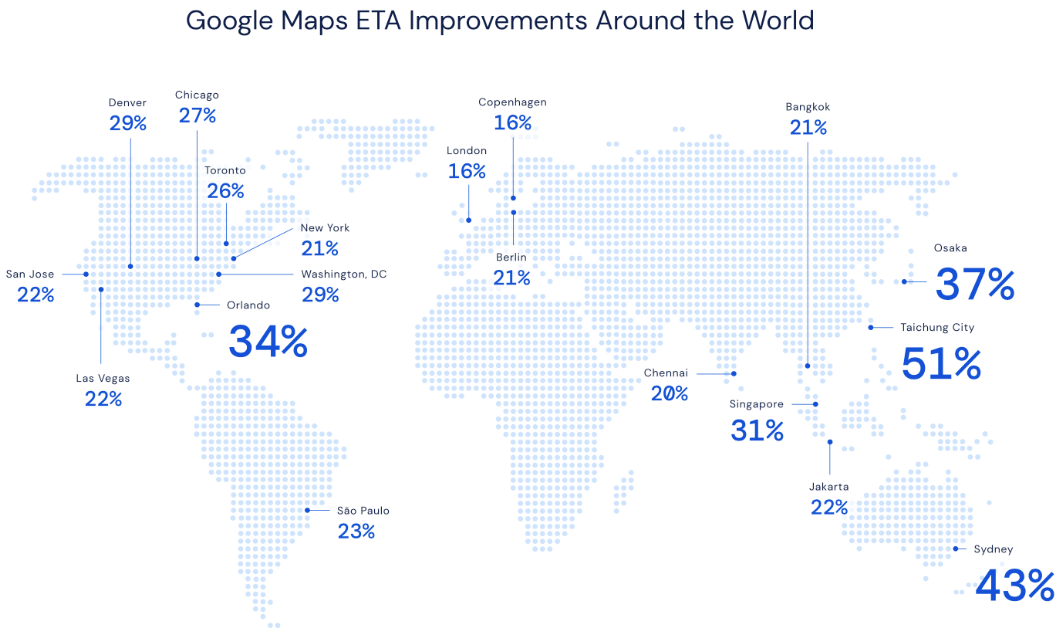
Home | College and Campus | Science | Engineering | Health

Twitter buys AI startup founded by Imperial academic to tackle fake news

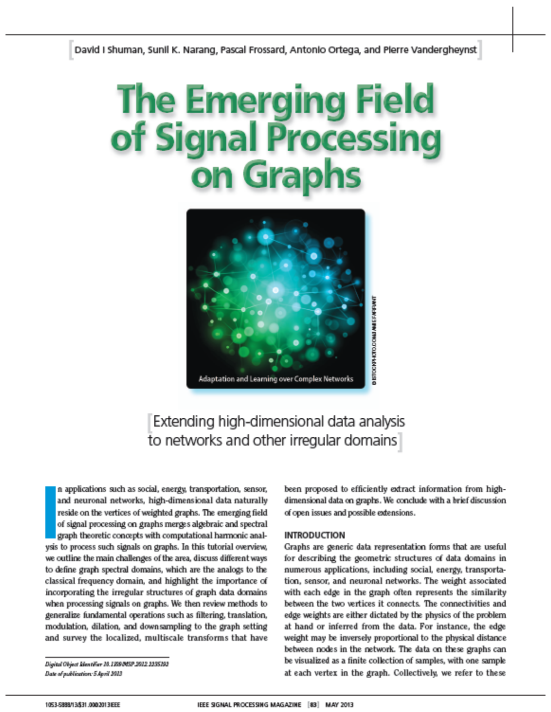
Vosoughi et al., "The spread of true and false news online," Science, 2018.
Monti et al., "Fake news detection on social media using geometric deep learning," ICLR Workshop, 2019.

Application VI: Traffic prediction





References


David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst

The Emerging Field of Signal Processing on Graphs

Extending high-dimensional data analysis to networks and other irregular domains


Appl. Comput. Harmon. Anal. 40 (2016) 260–291

Applied and Computational Harmonic Analysis

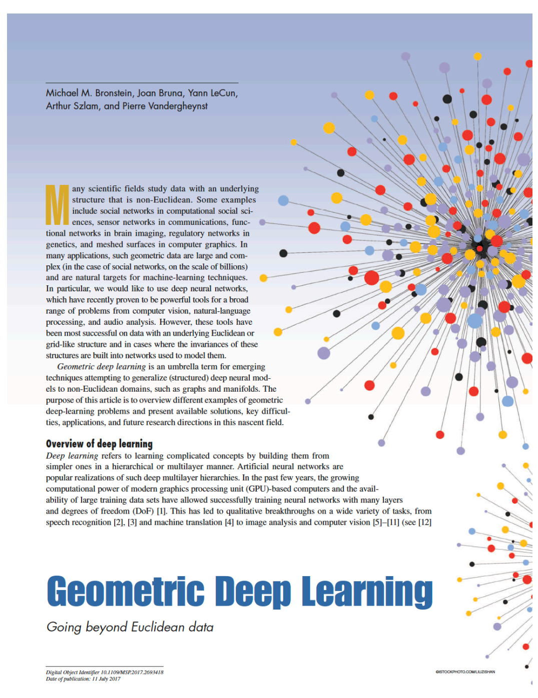
Vertex-frequency analysis on graphs

David I Shuman, Benjamin Ricaud, Pierre Vandergheynst


Graph Signal Processing: Overview, Challenges, and Applications

This article presents methods to process data associated to graphs (graph signals) extending techniques (transforms, sampling, and others) that are used for conventional signals.

By Antonio Ortega, Pascal Frossard, Jelena Kovačević, José M. F. Moura, and Pierre Vandergheynst

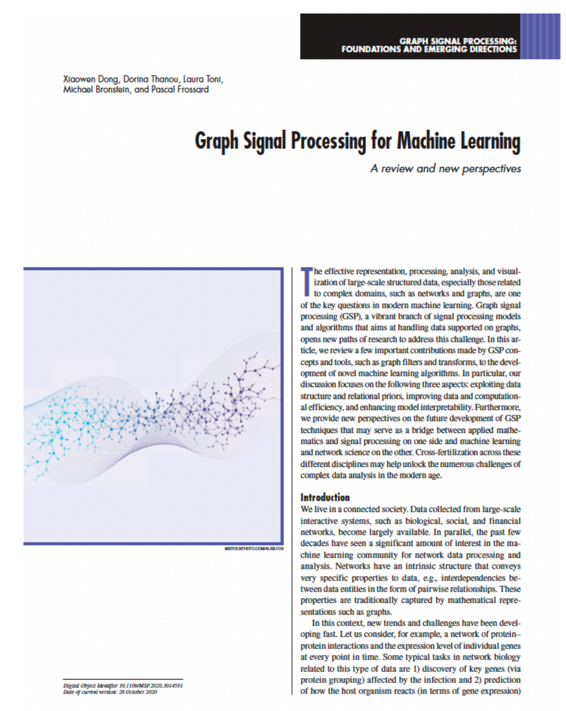

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Geometric Deep Learning

Going beyond Euclidean data


A Comprehensive Survey on Graph Neural Networks

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu


Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard

Graph Signal Processing for Machine Learning

A review and new perspectives