

AIMS CDT - Signal Processing

Michaelmas Term 2023

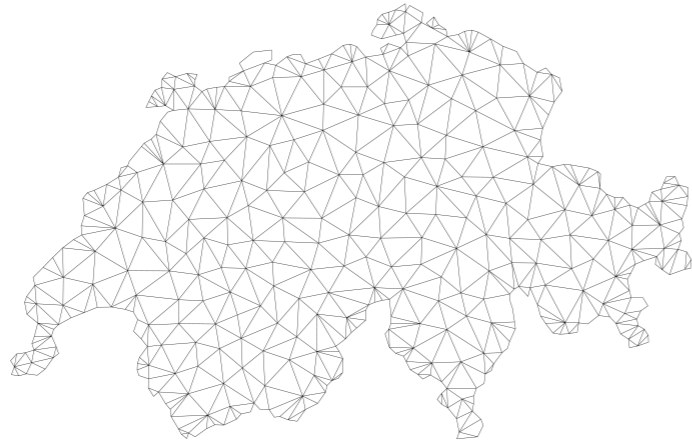
Xiaowen Dong

Department of Engineering Science



Introduction to Graphs Signal Processing

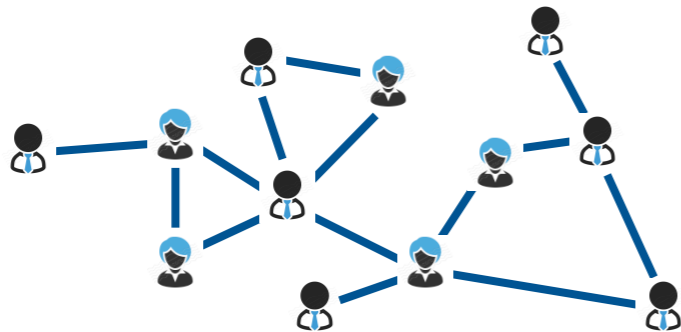
Networks are pervasive



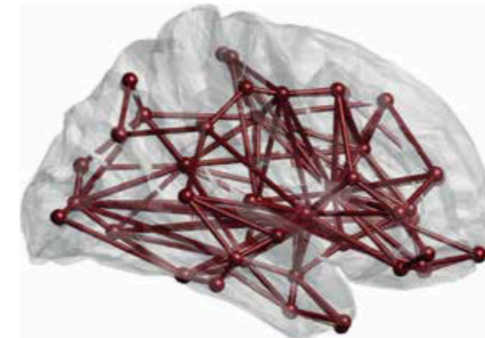
geographical network



traffic network



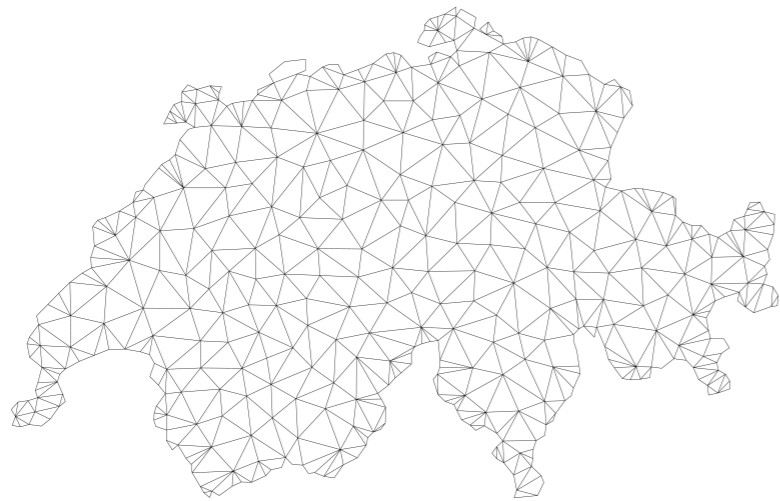
social network



brain network

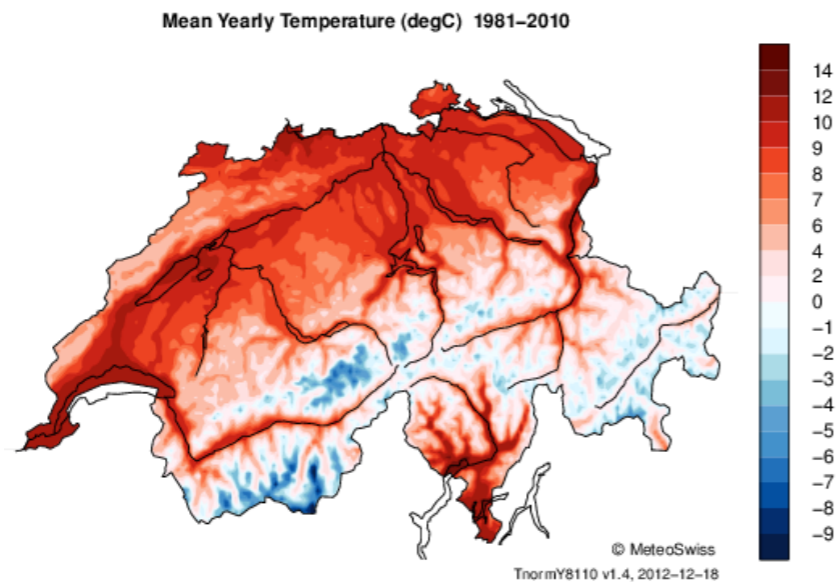
graphs provide mathematical representation of networks

Graph-structured data are pervasive



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

Graph-structured data are pervasive



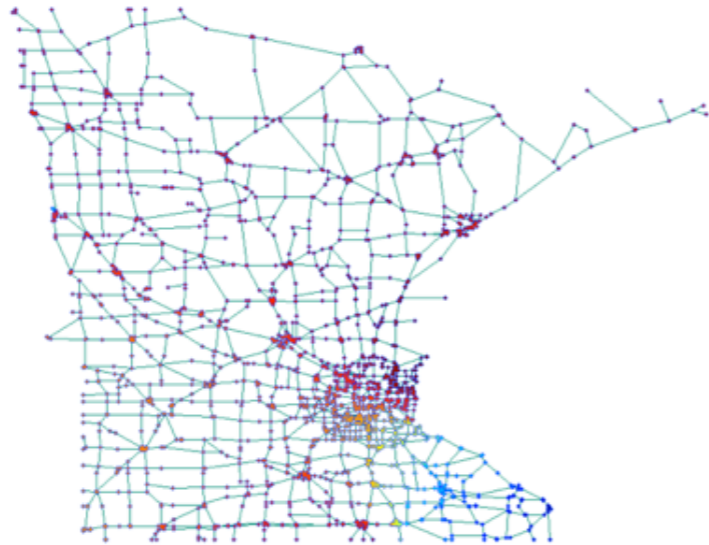
- vertices
 - geographical regions
- edges
 - geographical proximity between regions
- signal
 - temperature records in these regions

Graph-structured data are pervasive



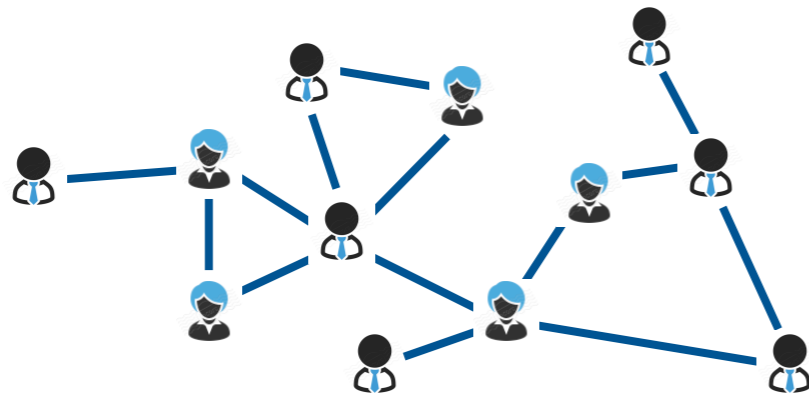
- vertices
 - road junctions
- edges
 - road connections

Graph-structured data are pervasive



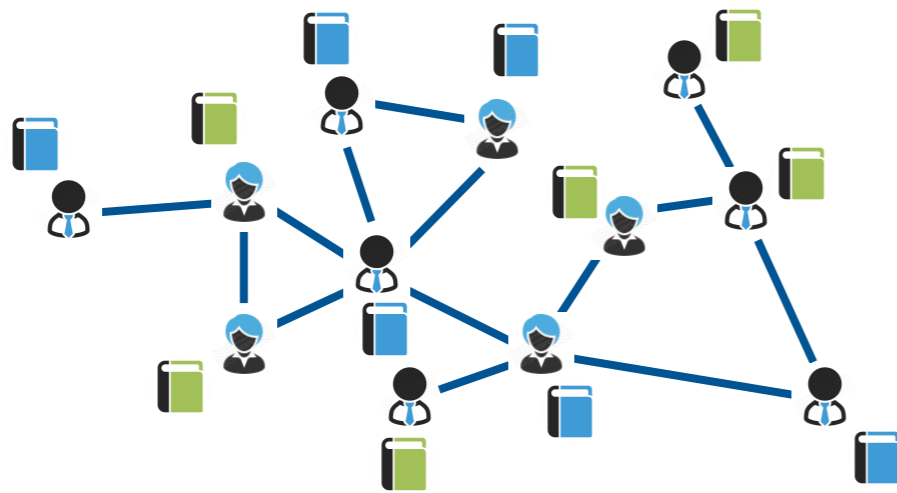
- vertices
 - road junctions
- edges
 - road connections
- signal
 - traffic congestion at junctions

Graph-structured data are pervasive



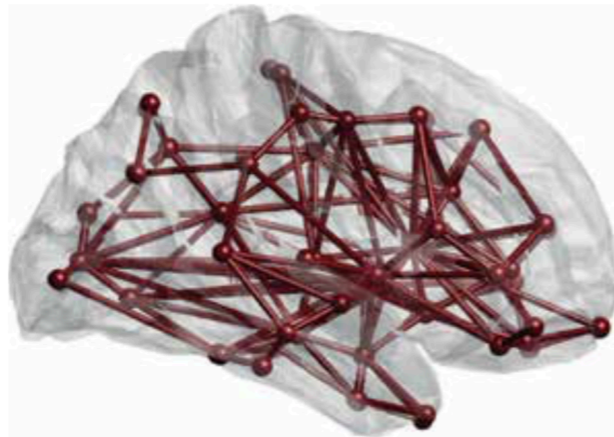
- vertices
 - individuals
- edges
 - friendship between individuals

Graph-structured data are pervasive



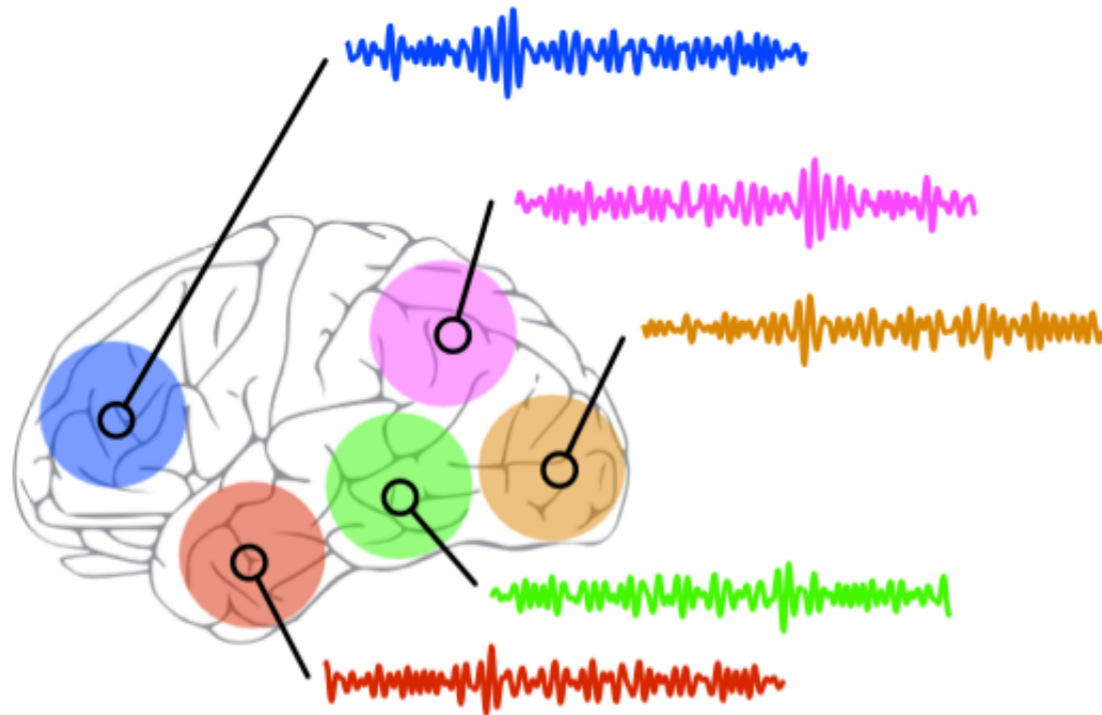
- vertices
 - individuals
- edges
 - friendship between individuals
- signal
 - personal interest

Graph-structured data are pervasive



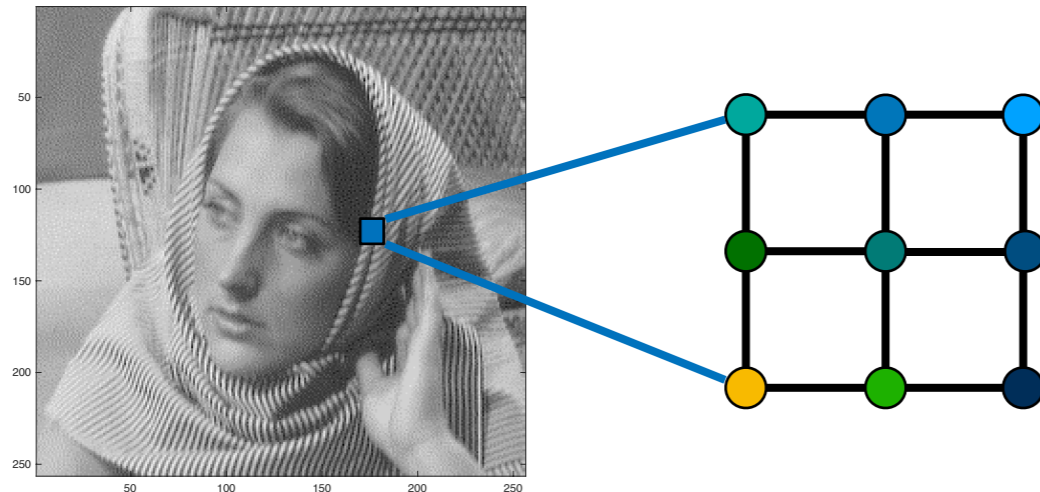
- vertices
 - brain regions
- edges
 - structural connectivity between brain regions

Graph-structured data are pervasive



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

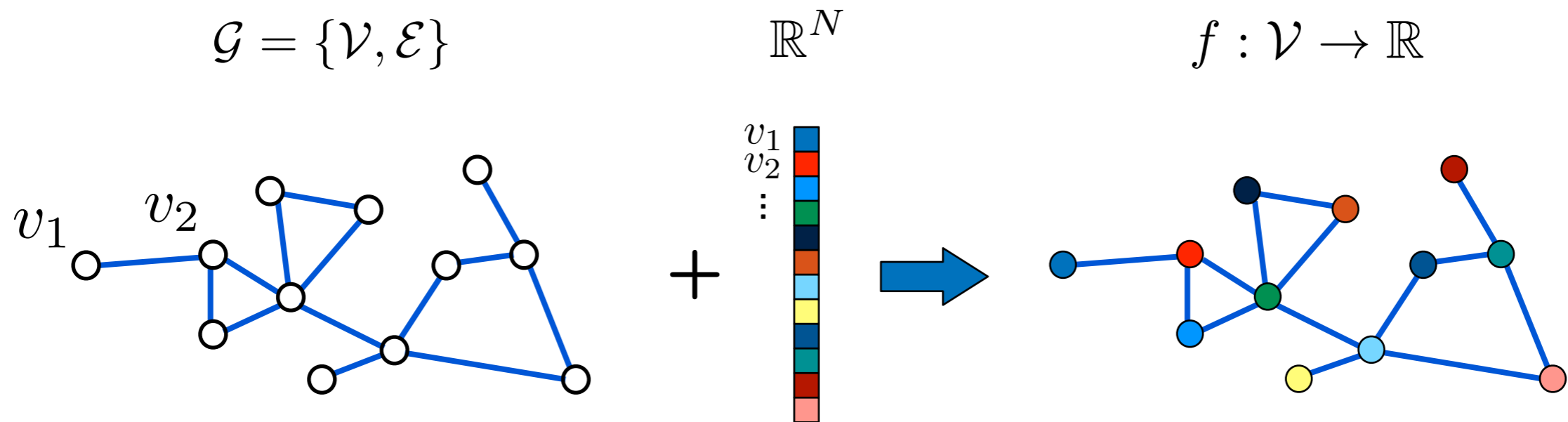
Graph-structured data are everywhere



- 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**

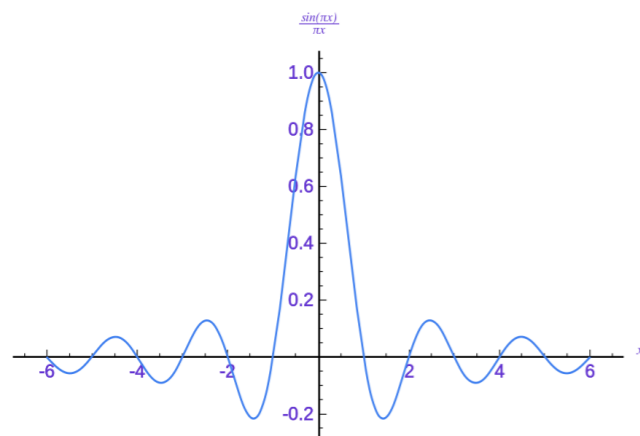


takes into account both **structure (edges)** and **data (values at nodes)**

Graph signal processing

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

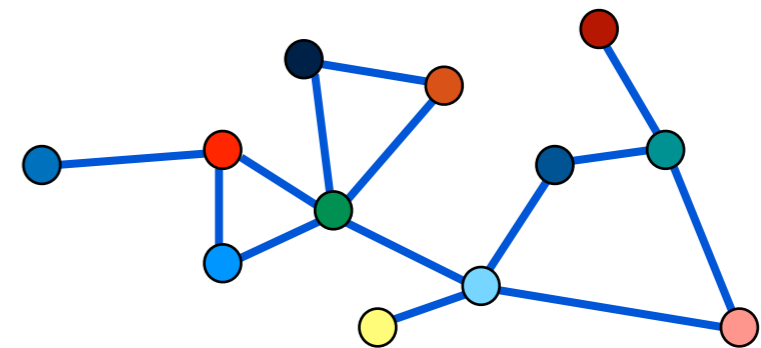
1D signal



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

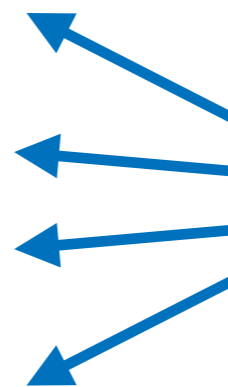
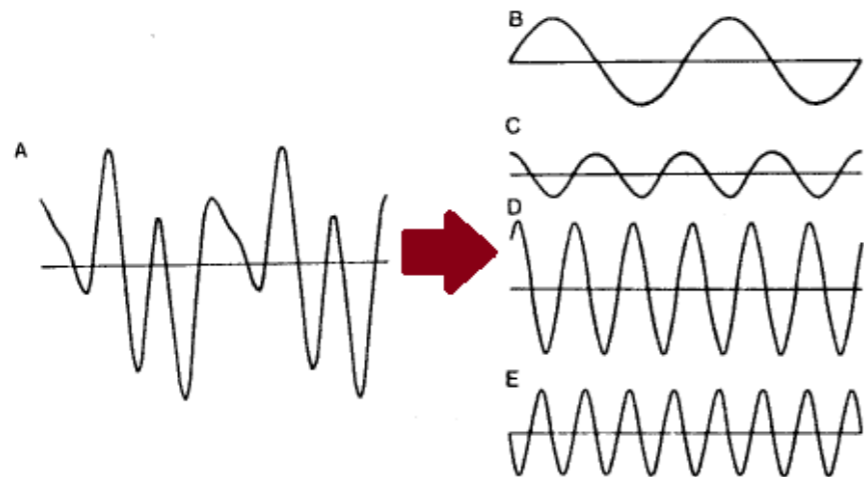
- Main GSP approaches can be categorised into two families:

- 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

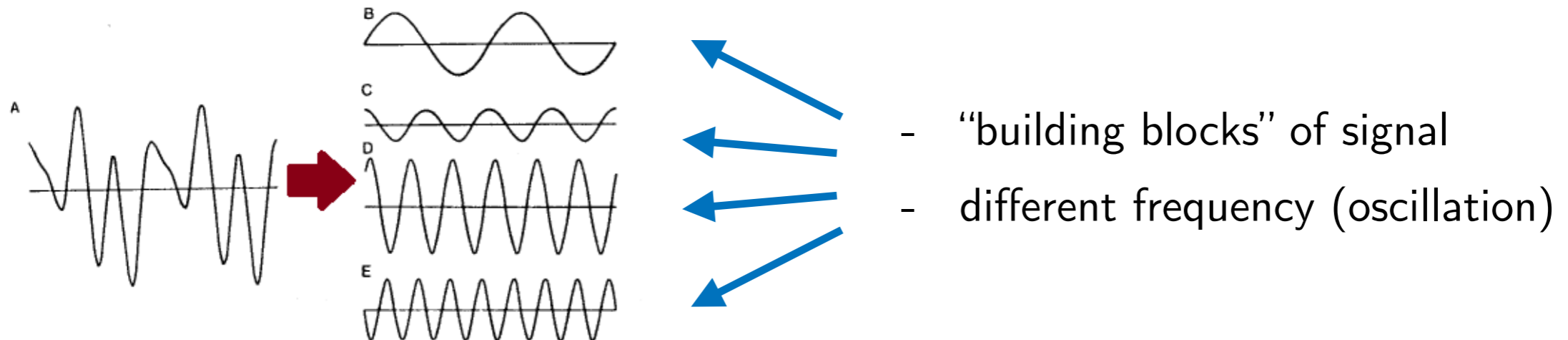
- Main GSP approaches can be categorised into two families:

- vertex (spatial) domain designs

- frequency (graph spectral) domain designs

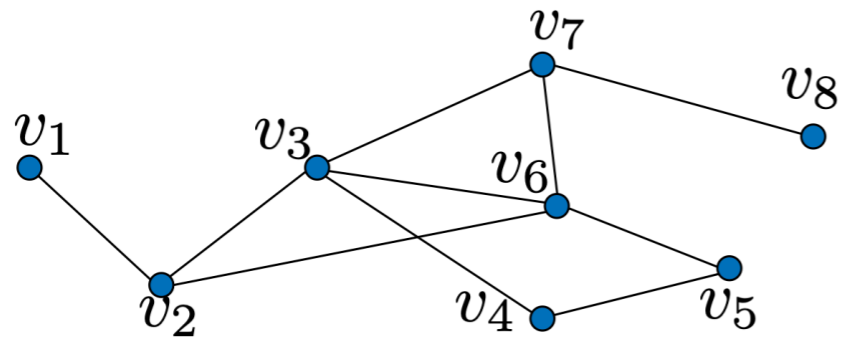
**important for
signal analysis**

- Classical Fourier transform provides frequency domain representation of signals



- What about a notion of frequency for graph signals?

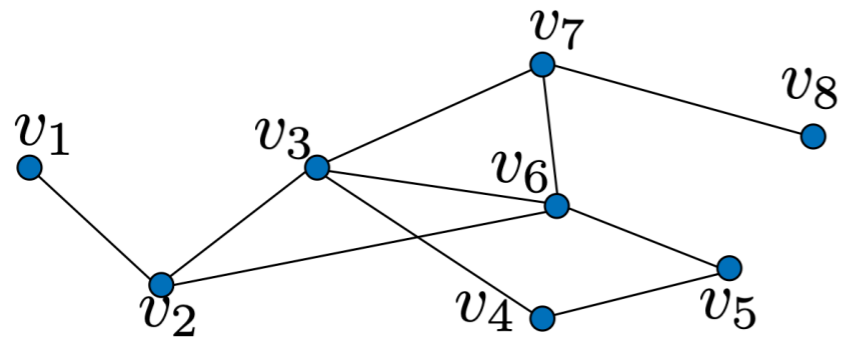
Graph Laplacian



weighted and undirected graph:

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

Graph Laplacian



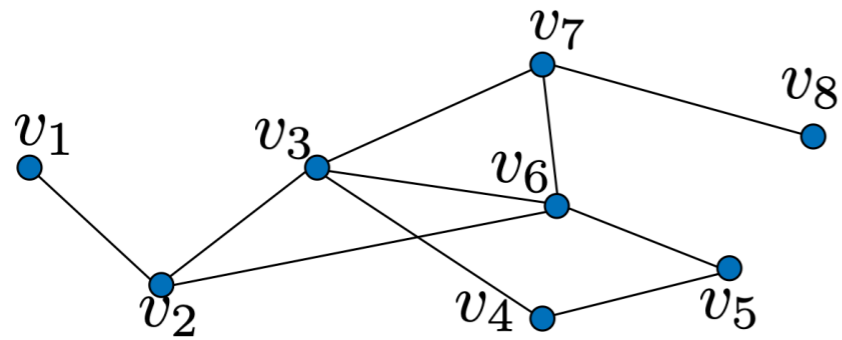
weighted and undirected graph:

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

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

W

Graph Laplacian



weighted and undirected graph:

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

$$D = \text{diag}(d(v_1), \dots, d(v_N))$$

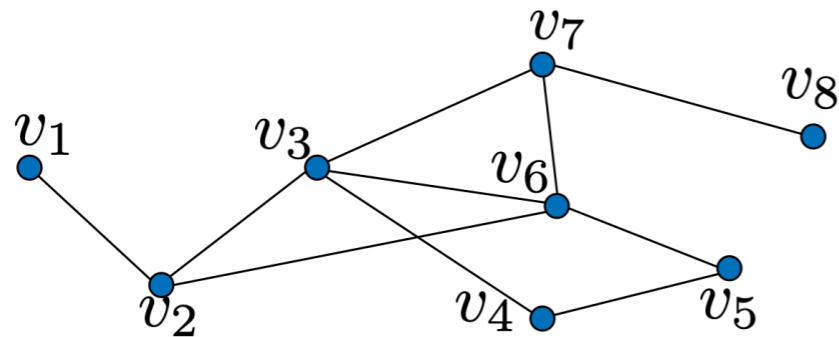
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

D

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

W

Graph Laplacian



weighted and undirected graph:

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

$$D = \text{diag}(d(v_1), \dots, d(v_N))$$

$$L = D - W \quad \text{equivalent to } W!$$

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

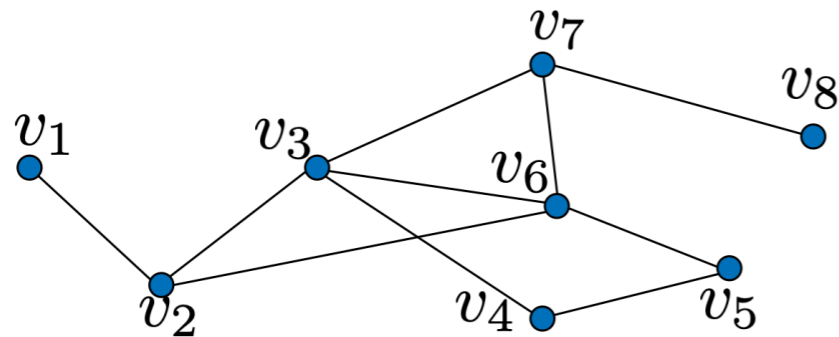
D

W

L

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

Graph Laplacian



weighted and undirected graph:

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

$$D = \text{diag}(d(v_1), \dots, d(v_N))$$

$$L = D - W \quad \text{equivalent to } W!$$

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

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

D

W

L

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

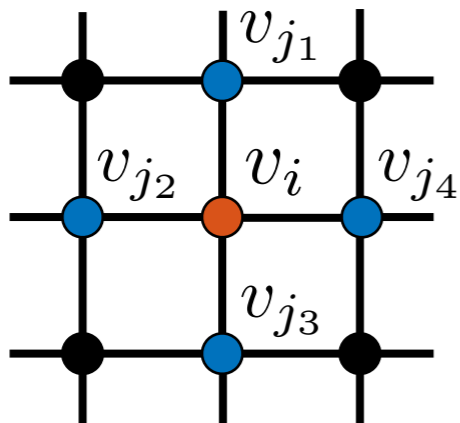
Graph Laplacian

Why graph Laplacian?

Graph Laplacian

Why graph Laplacian?

- provides an approximation of the Laplace operator



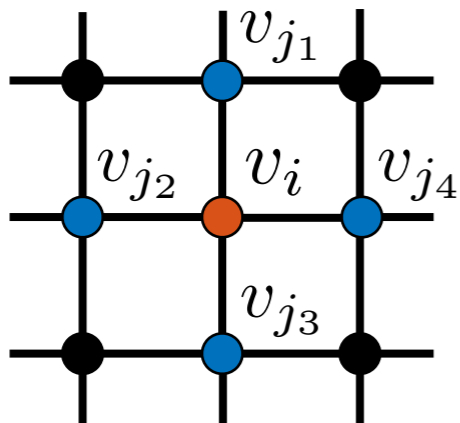
$$(Lf)(i) = (4f(i) - f(j_1) - f(j_2) - f(j_3) - f(j_4))/(\delta x)^2$$

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

Graph Laplacian

Why graph Laplacian?

- provides an approximation of the Laplace operator

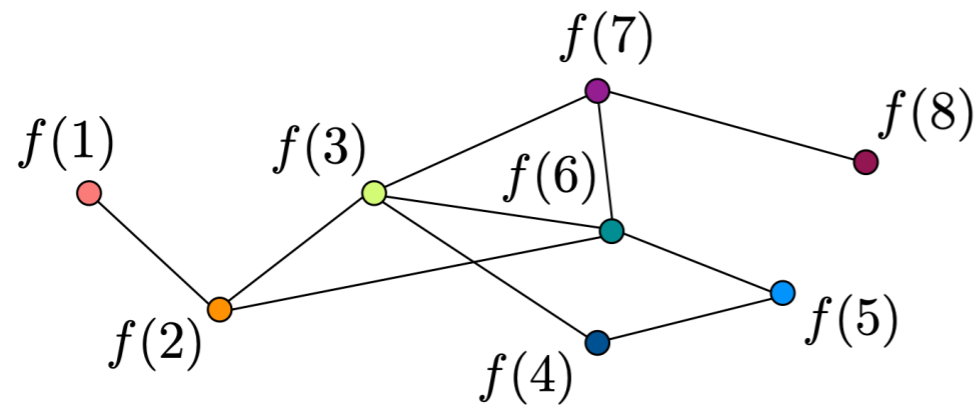


$$(Lf)(i) = (4f(i) - f(j_1) - f(j_2) - f(j_3) - f(j_4)) / (\delta x)^2$$

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

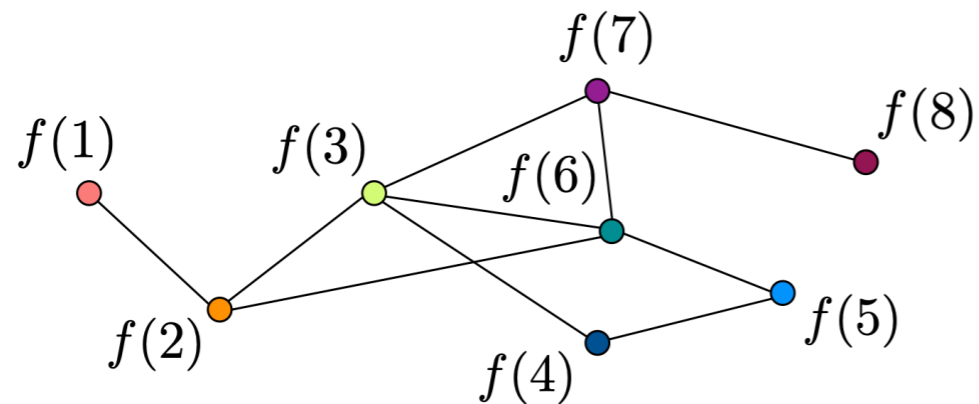
- converges to the Laplace-Beltrami operator (given certain conditions)
- provides a notion of “frequency” on graphs

Graph Laplacian



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

Graph Laplacian

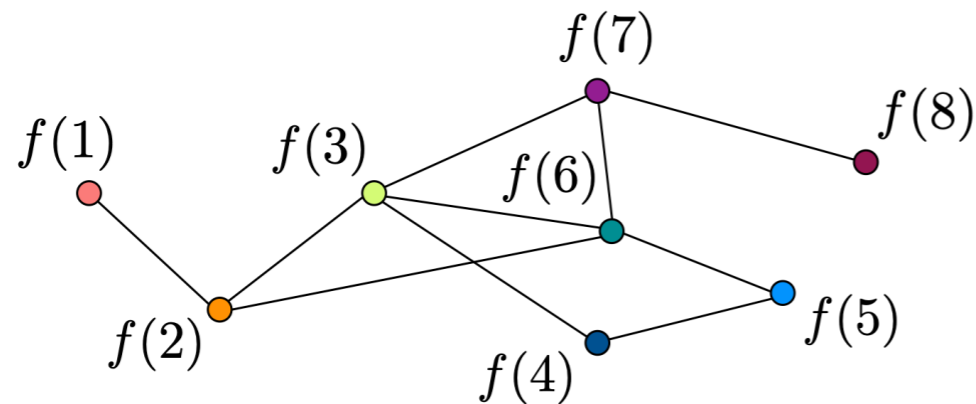


graph signal $f : \mathcal{V} \rightarrow \mathbb{R}$

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

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

Graph Laplacian



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

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

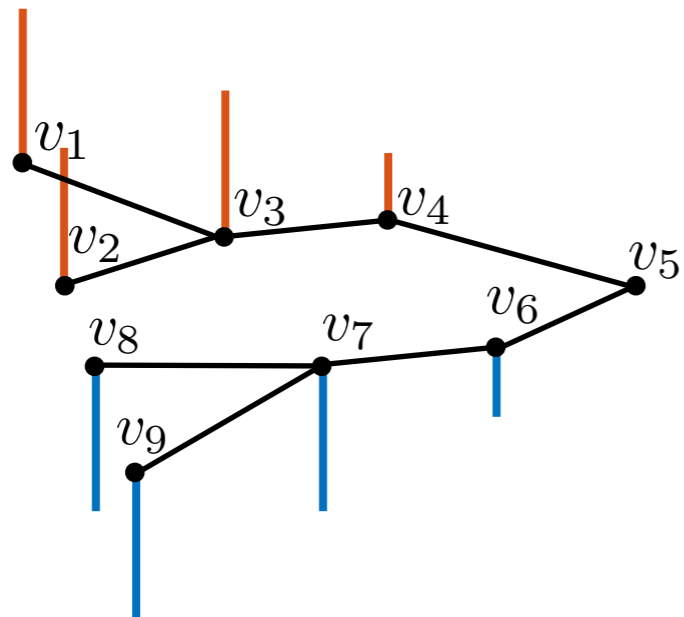
$$\begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}^T \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 &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

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

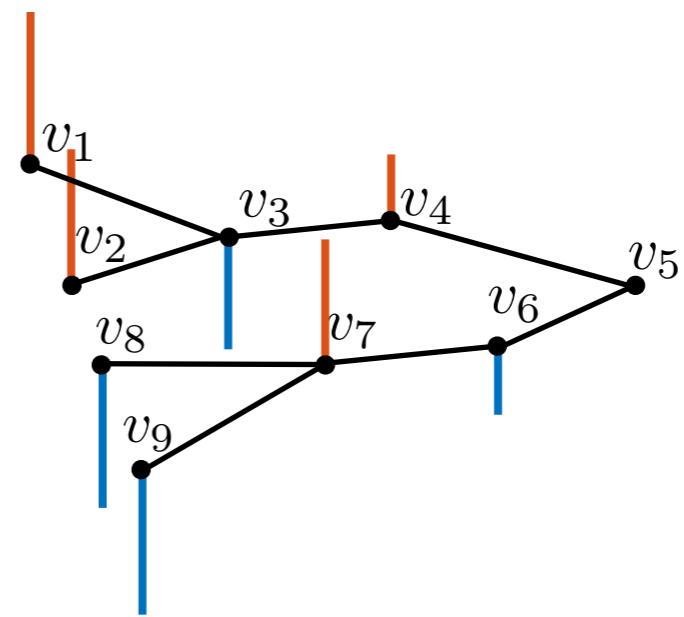
$$f^T Lf = \frac{1}{2} \sum_{i,j=1}^N W_{ij} (f(i) - f(j))^2$$

a measure of “smoothness”

Graph Laplacian



$$f^T L f = 1$$



$$f^T L f = 21$$

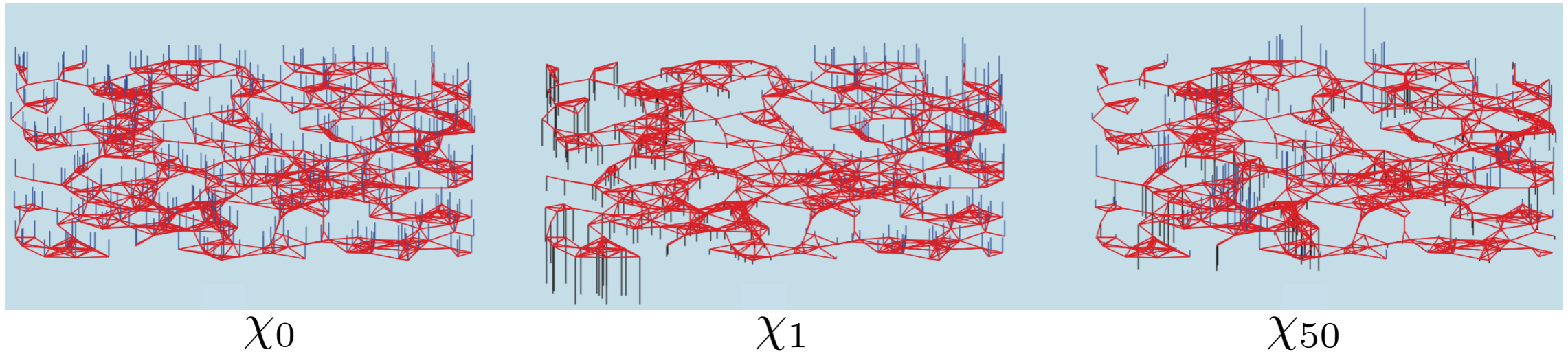
Graph Laplacian

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

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

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

Graph Fourier transform

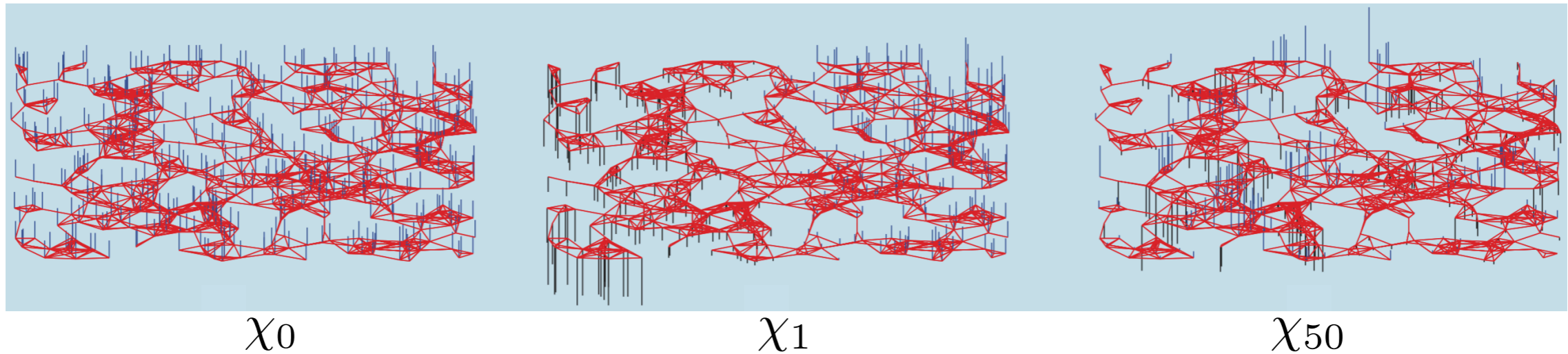


χ_0

χ_1

χ_{50}

Graph Fourier transform



low frequency

high frequency

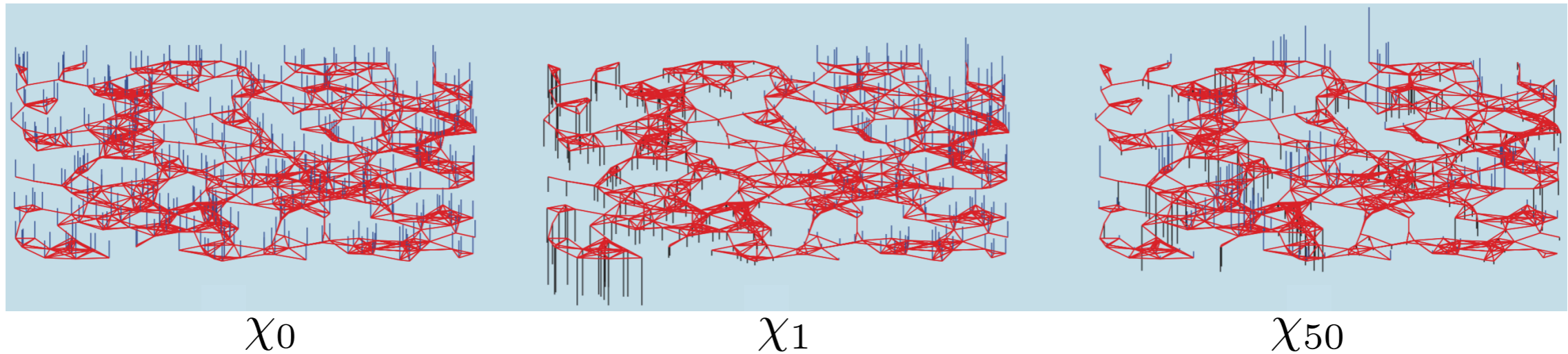
$$\chi_0^T L \chi_0 = \lambda_0 = 0$$

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

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

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

Graph Fourier transform



low frequency

high frequency

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

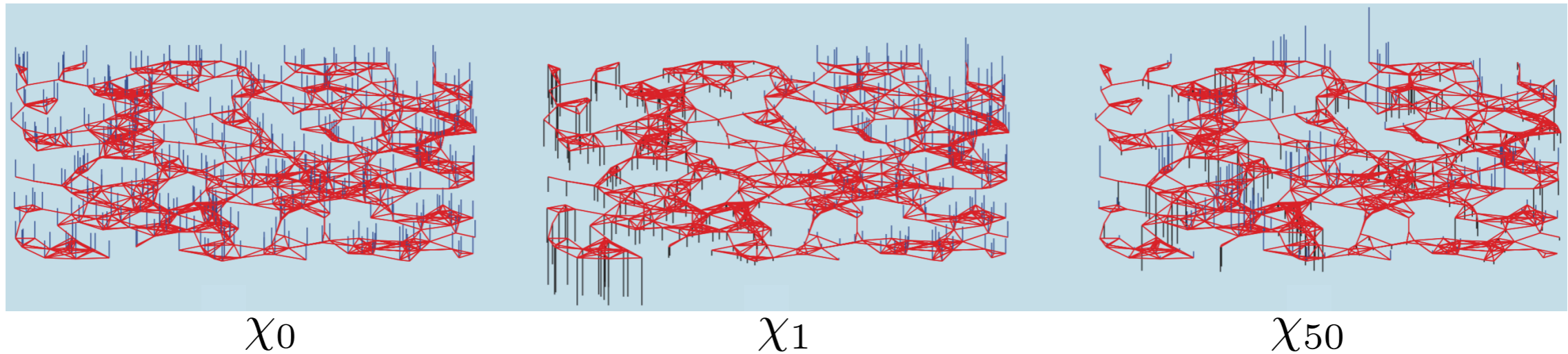
$$\chi_{50}^T L \chi_{50} = \lambda_{50}$$

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

graph Fourier transform:

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

Graph Fourier transform



low frequency

high frequency

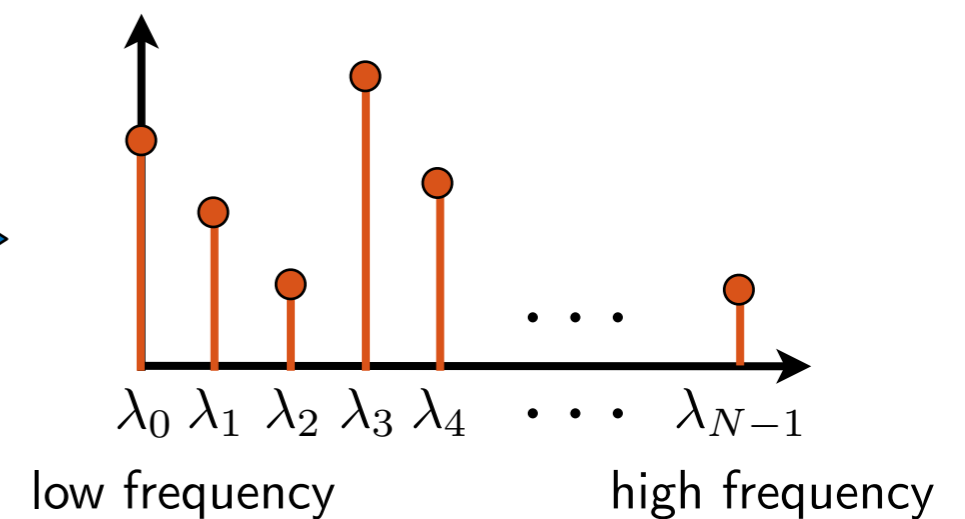
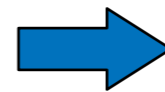
$$\chi_0^T L \chi_0 = \lambda_0 = 0$$

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

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

graph Fourier transform:

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



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



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

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: $-\nabla^2$



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

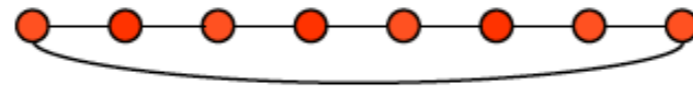


$f : V \rightarrow \mathbb{R}^N$

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

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

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:

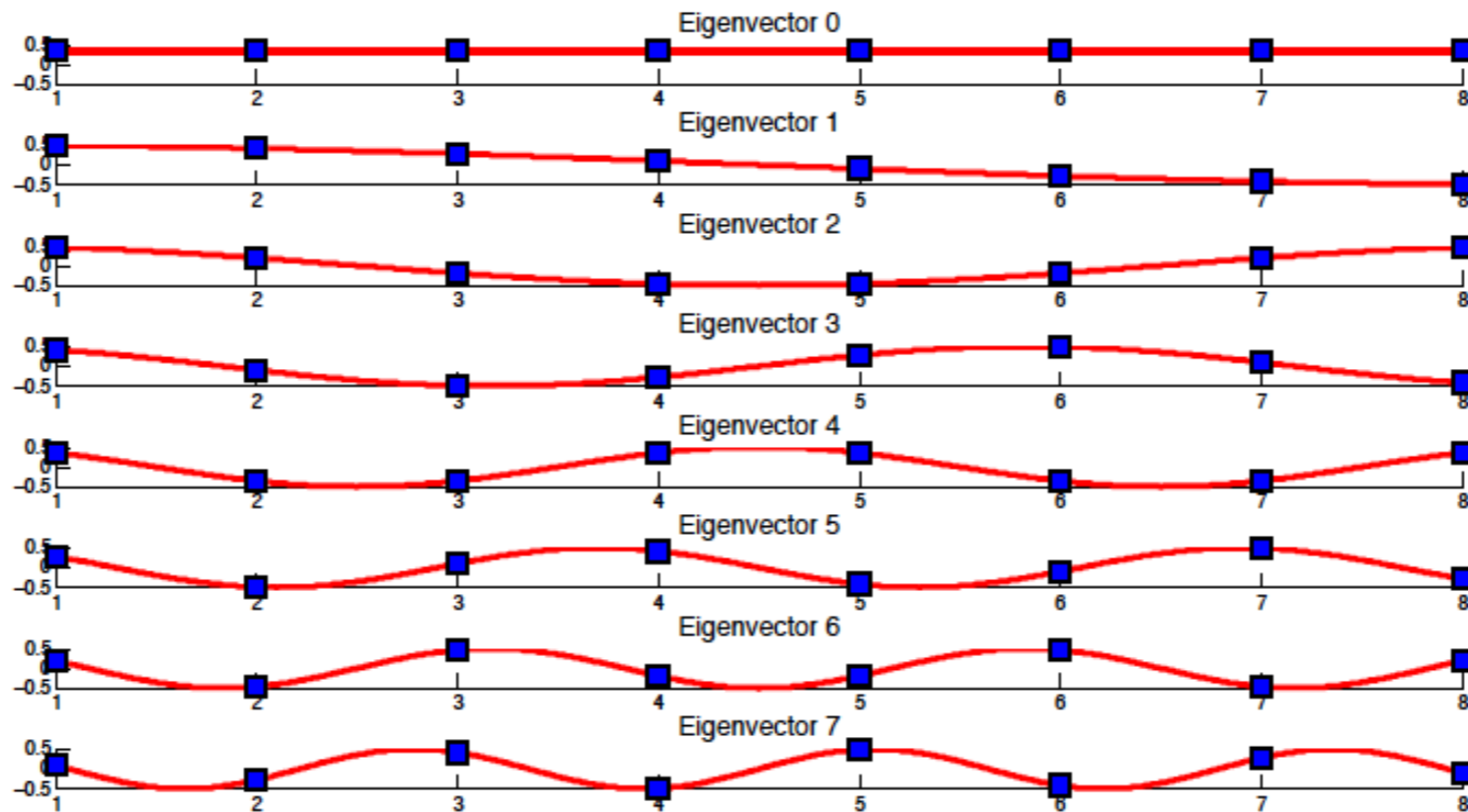
$$\chi_\ell = \left[1, \omega^\ell, \omega^{2\ell}, \dots, \omega^{(N-1)\ell} \right], \text{ where } \omega = e^{\frac{2\pi j}{N}}$$

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

Two special cases



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

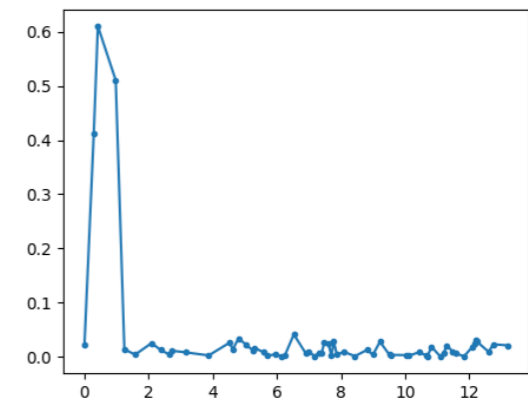
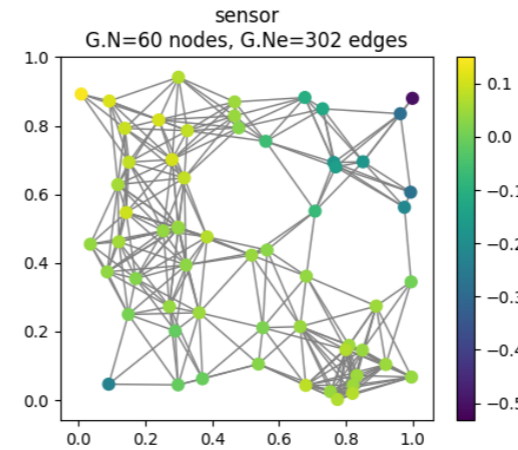
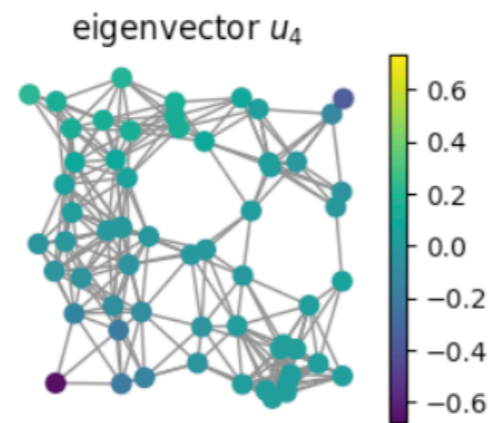
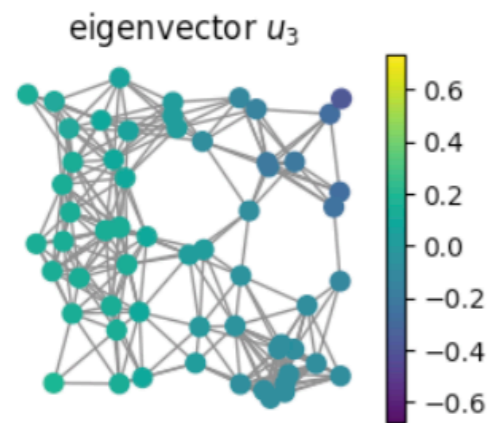
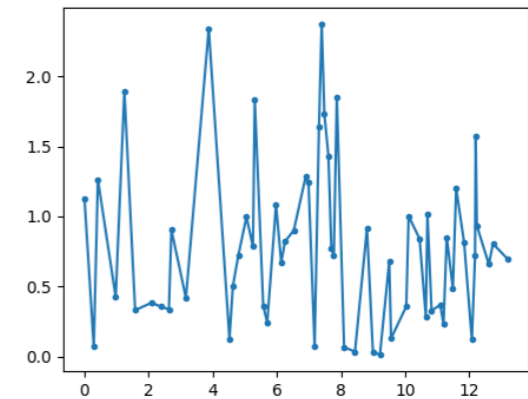
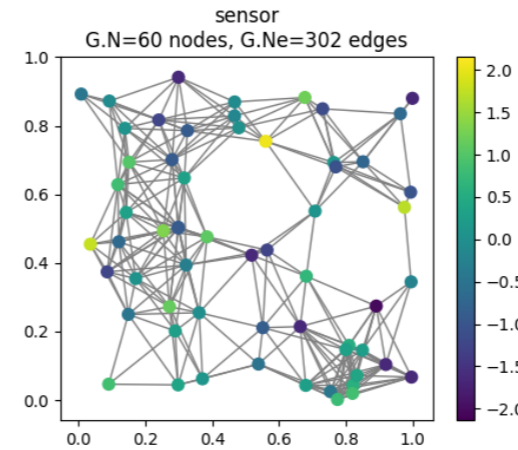
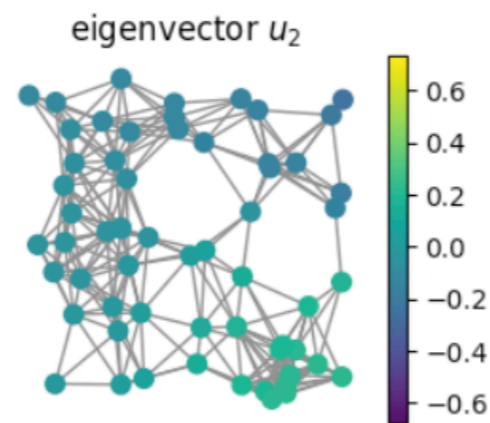
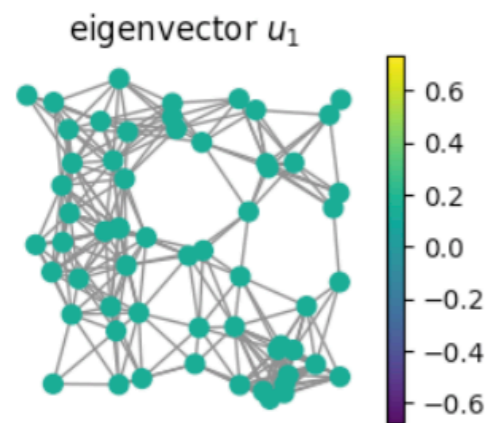


$\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$
 is the Discrete Cosine Transform matrix (DCT-II, Strang, 1999), which is used in JPEG image compression

Example on a general graph

GFT:

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



Outline

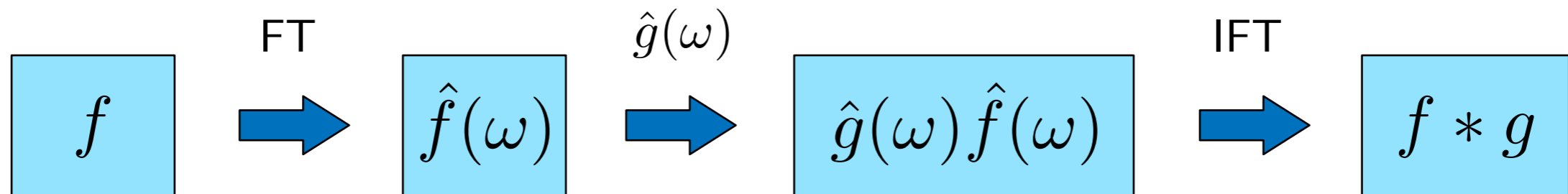
- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Representation of graph signals
- Convolutional neural networks on graphs
- Applications

Classical frequency filtering

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

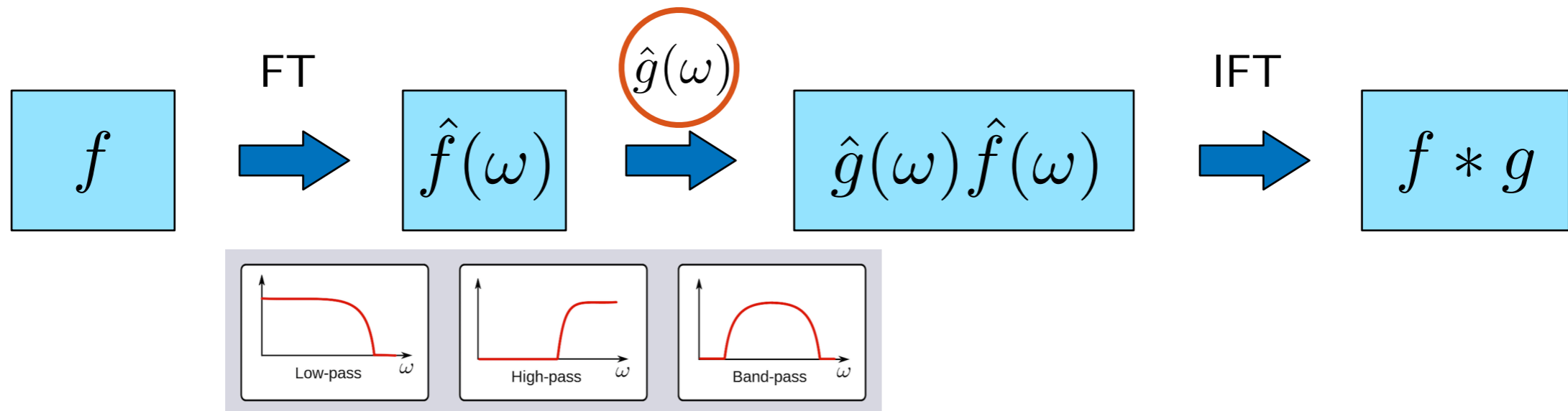
Classical frequency filtering

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



Classical frequency filtering

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

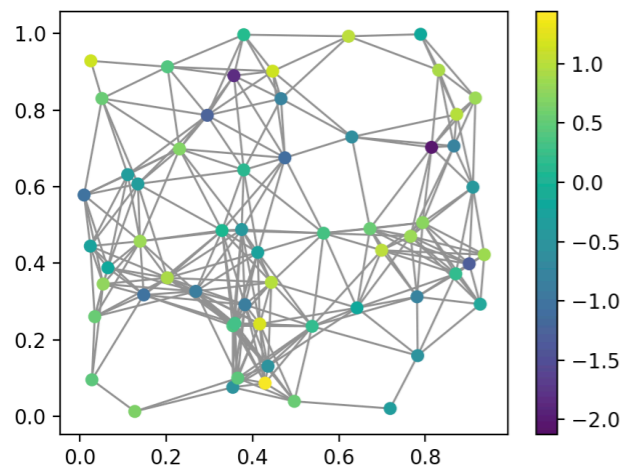
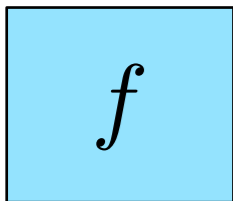


Graph spectral filtering

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

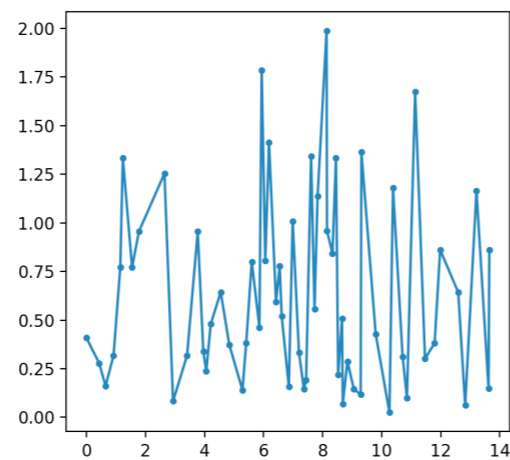
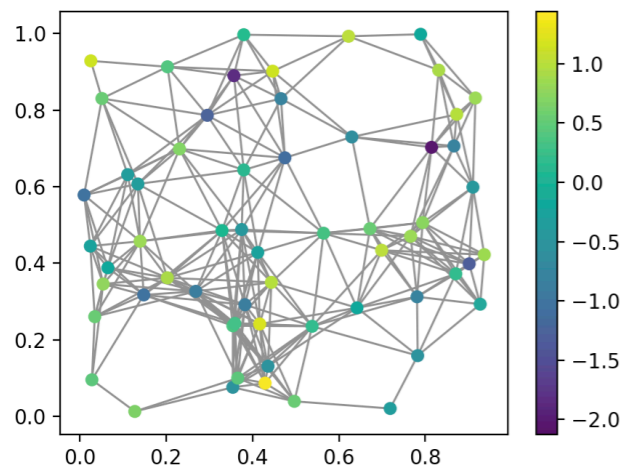
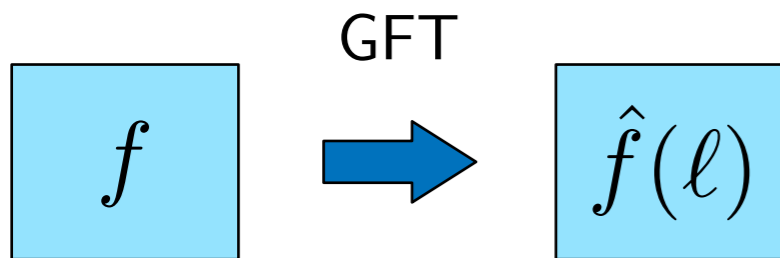
Graph spectral filtering

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



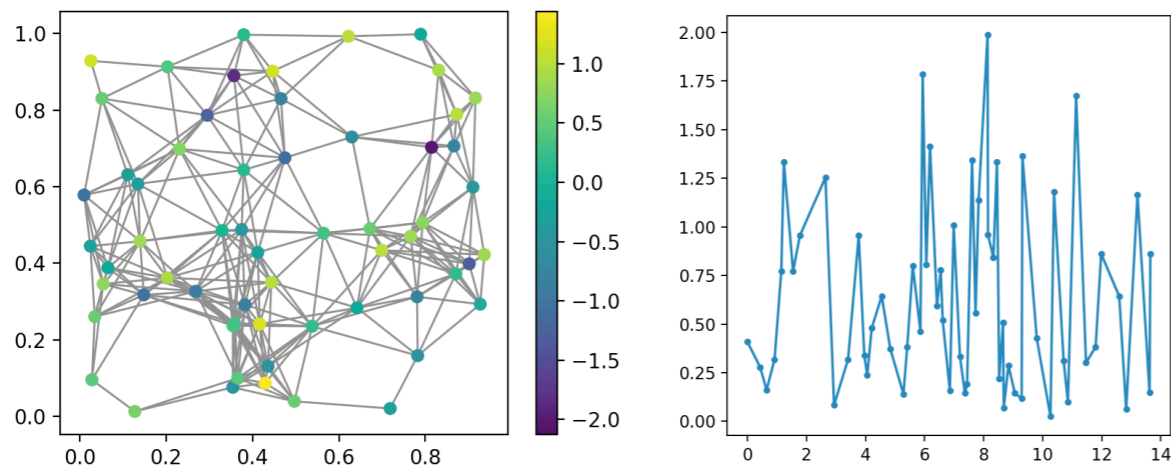
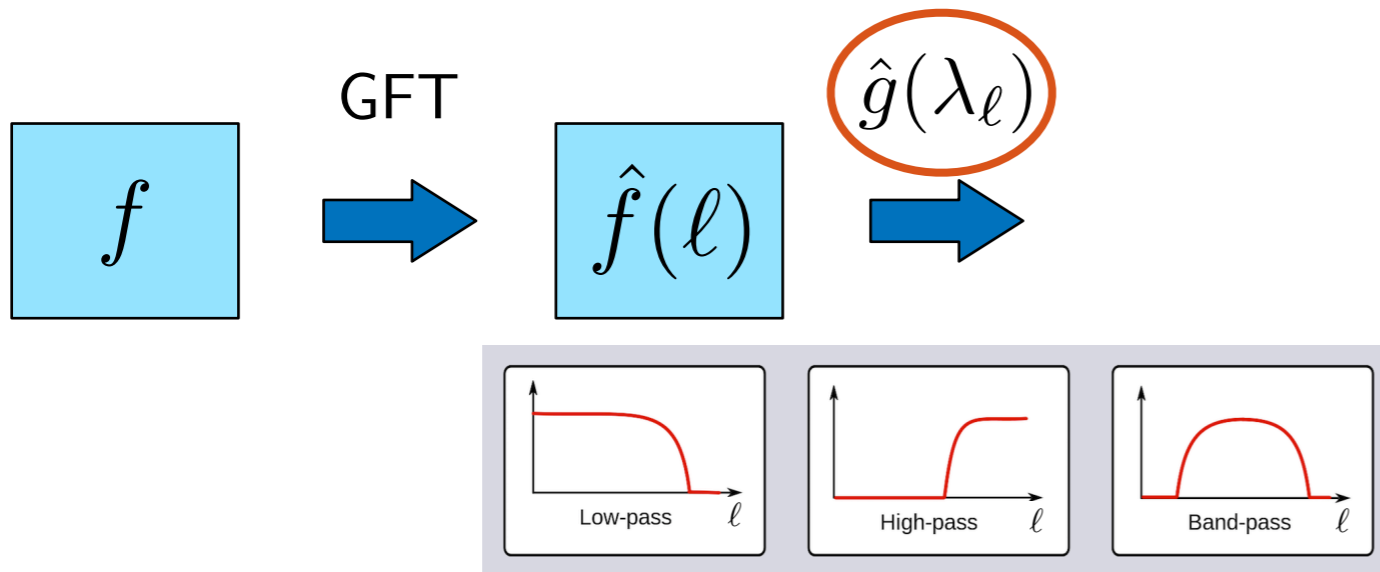
Graph spectral filtering

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



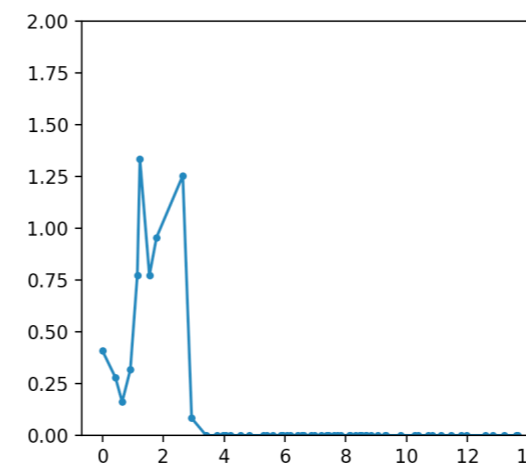
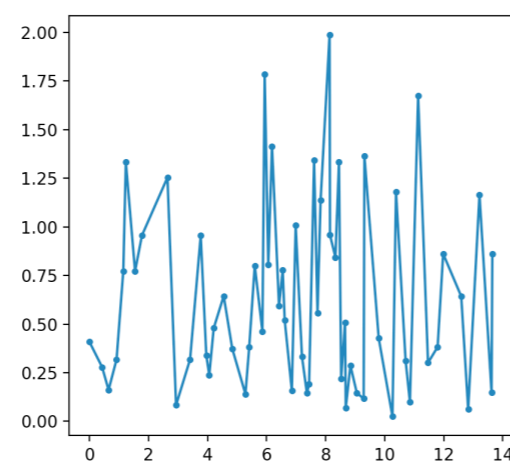
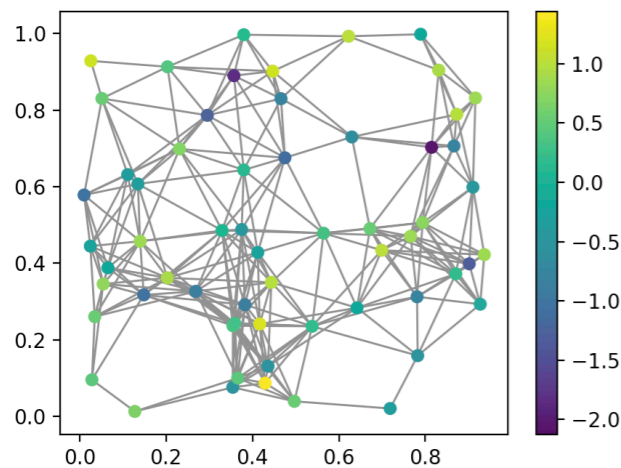
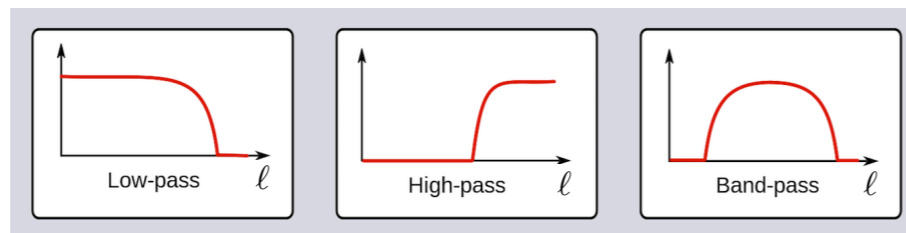
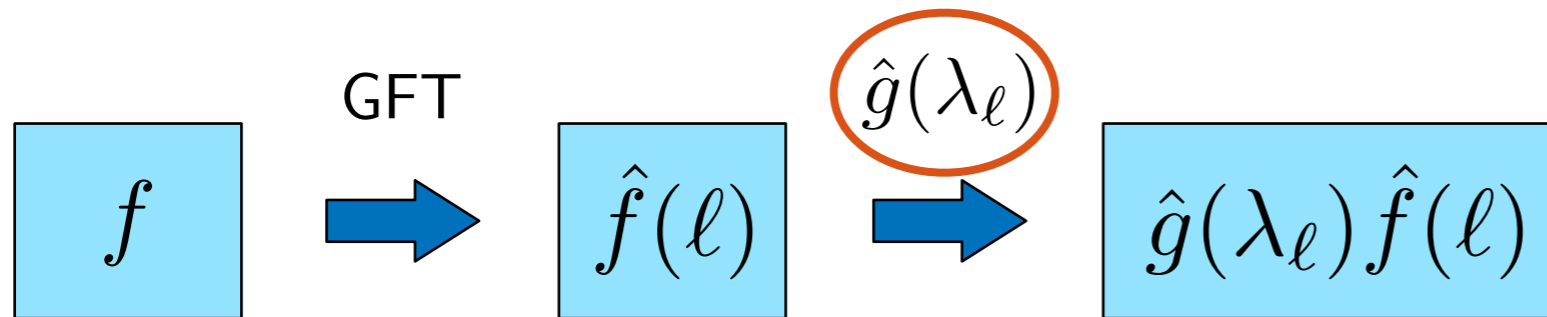
Graph spectral filtering

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



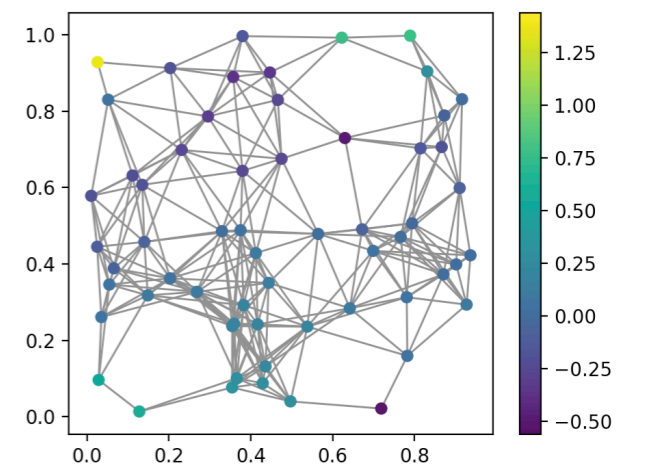
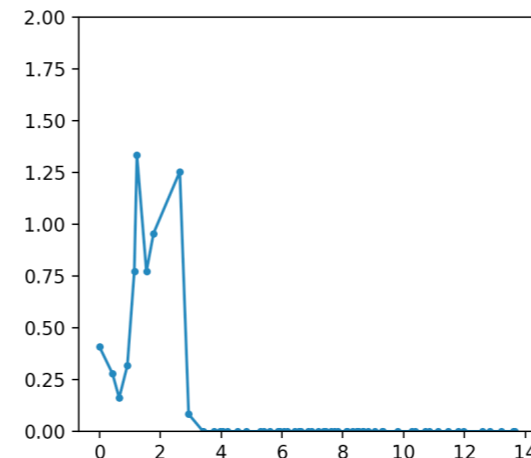
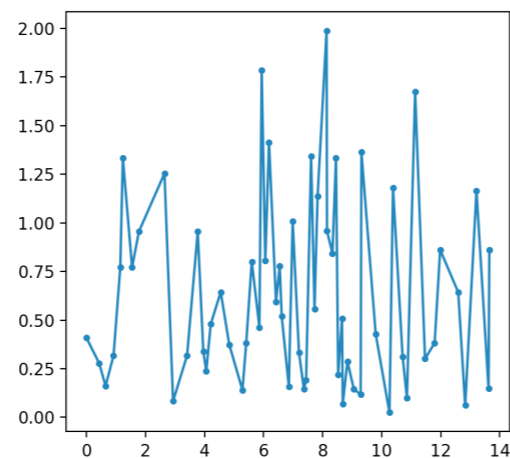
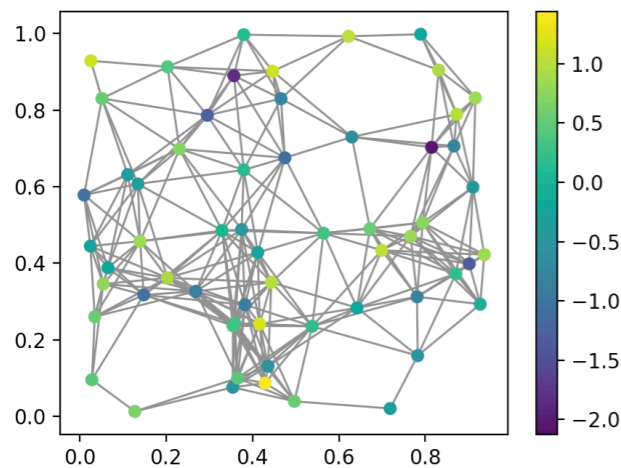
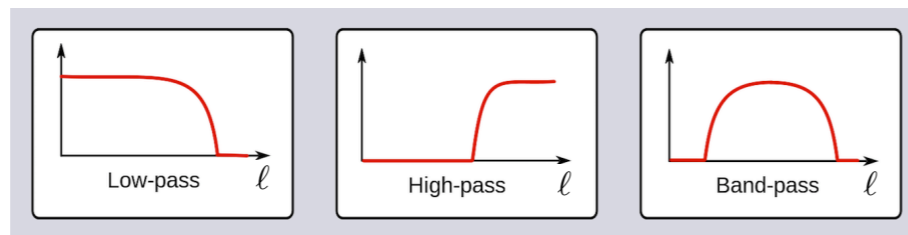
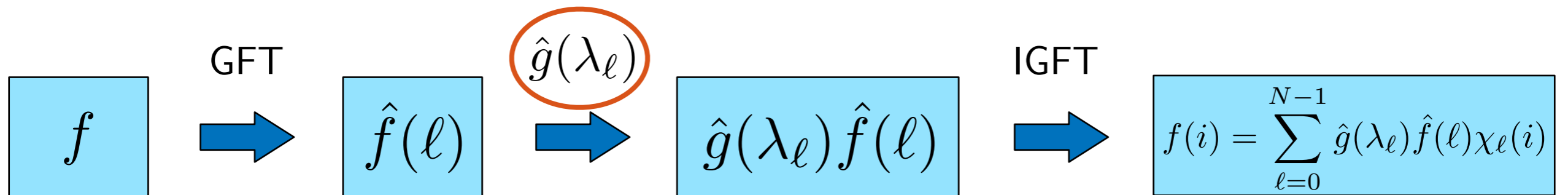
Graph spectral filtering

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



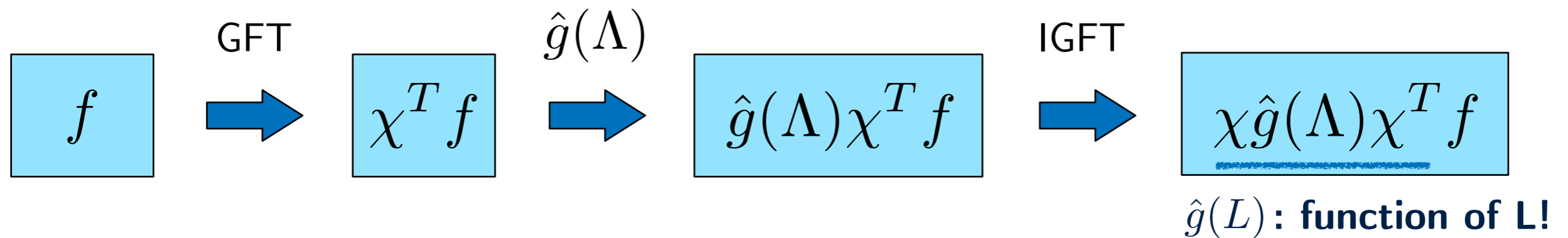
Graph spectral filtering

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



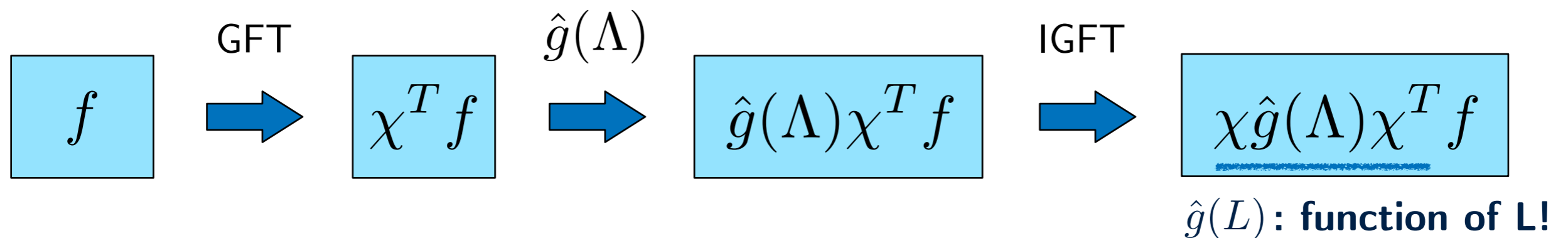
Graph transform/dictionary design

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



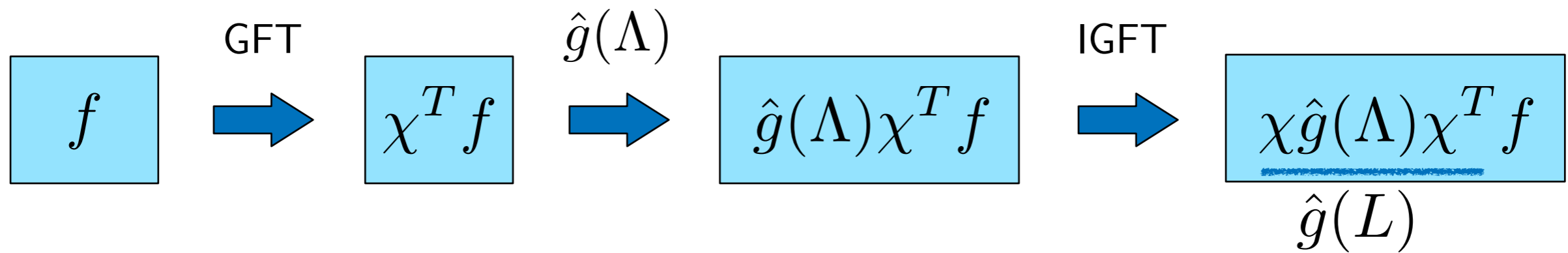
Graph transform/dictionary design

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

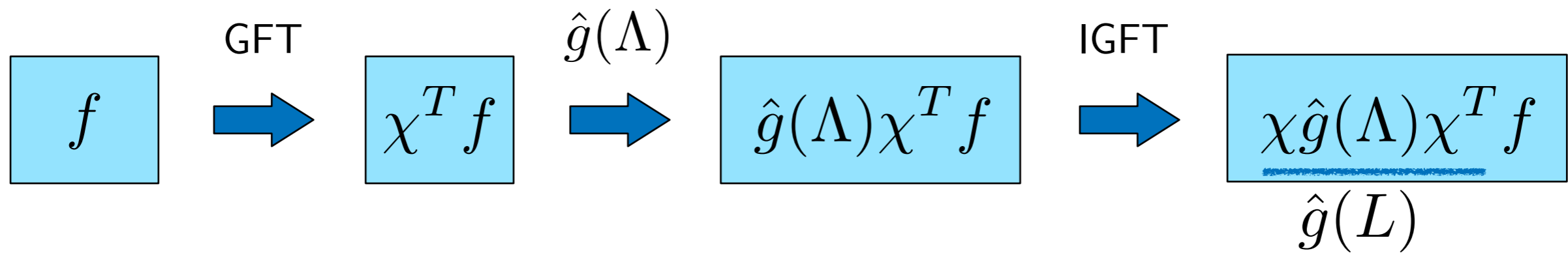


- Important properties can be achieved by properly defining $\hat{g}(L)$, such as localisation of atoms
- Closely related to kernels and regularisation on graphs

A practical example



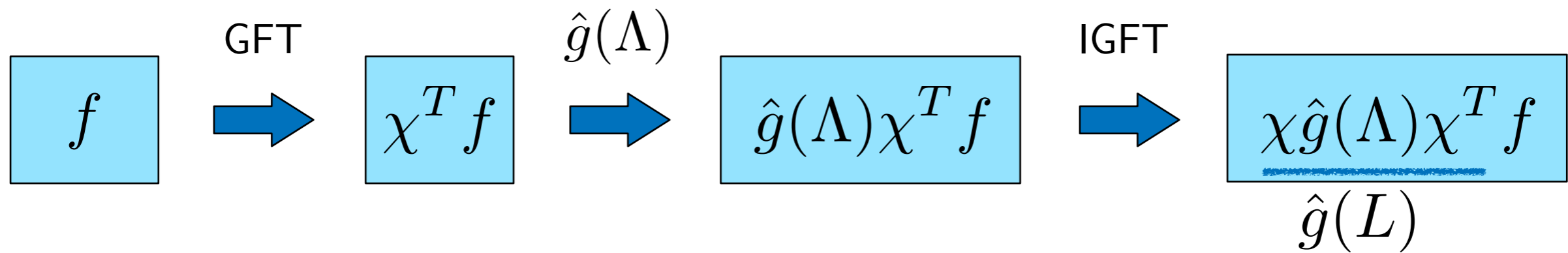
A practical example



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

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

A practical example



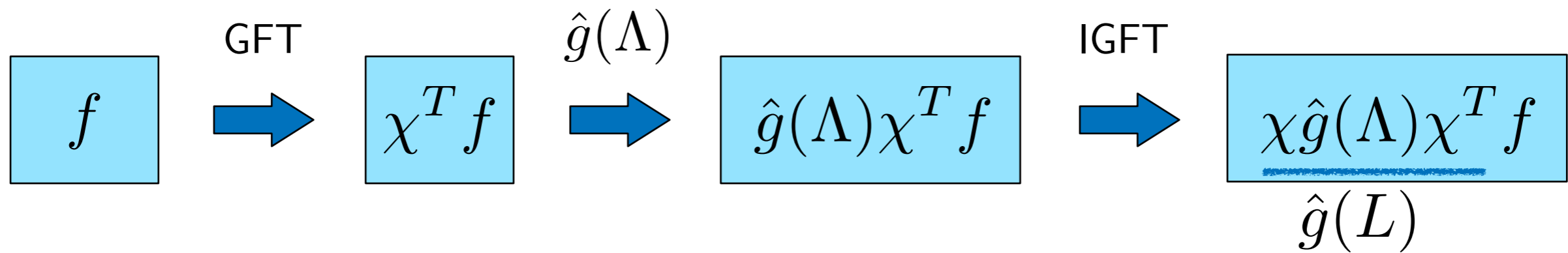
problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

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

data fitting term

"smoothness" assumption

A practical example

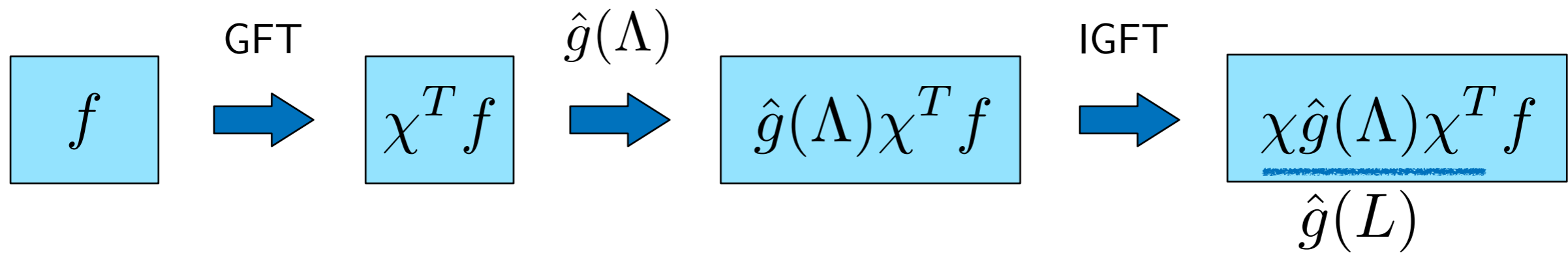


problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

$$y^* = \arg \min_y \{ \underbrace{\|y - f\|_2^2}_{\text{data fitting term}} + \underbrace{\gamma y^T L y}_{\text{"smoothness" assumption}} \}$$

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

A practical example



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

$$y^* = \arg \min_y \{ \underbrace{\|y - f\|_2^2}_{\text{data fitting term}} + \underbrace{\gamma y^T L y}_{\text{"smoothness" assumption}} \}$$

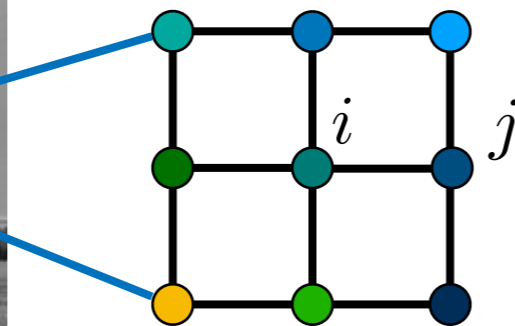
A blue arrow points from the text "data fitting term" to the $\|y - f\|_2^2$ term in the equation. An orange arrow points from the text "'smoothness' assumption" to the $\gamma y^T L y$ term.

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

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

A practical example

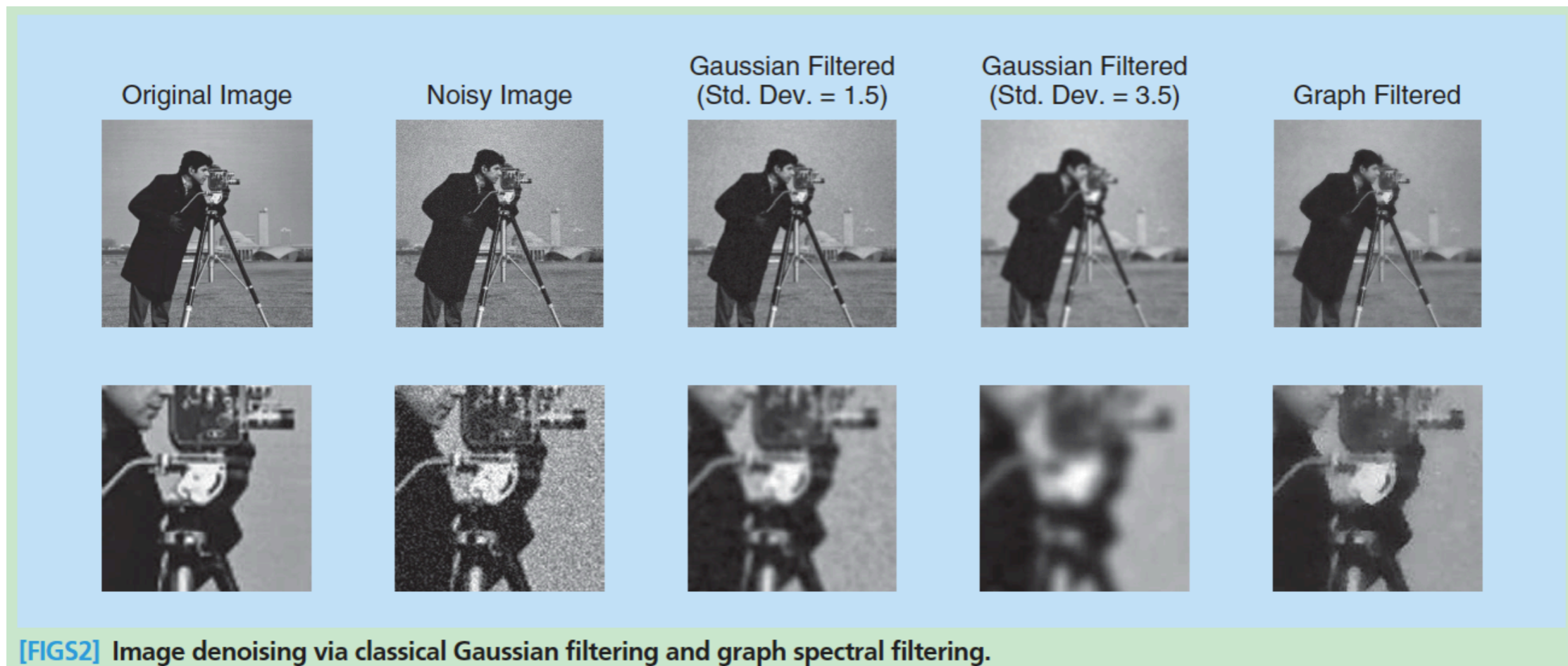
- noisy image as observed noisy graph signal
- regular grid graph (weights inversely proportional to pixel value difference)



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

A practical example

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

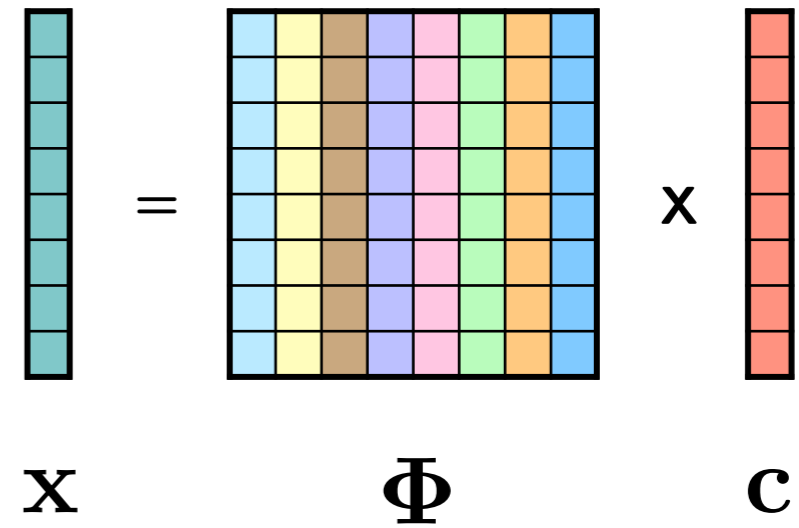


Outline

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

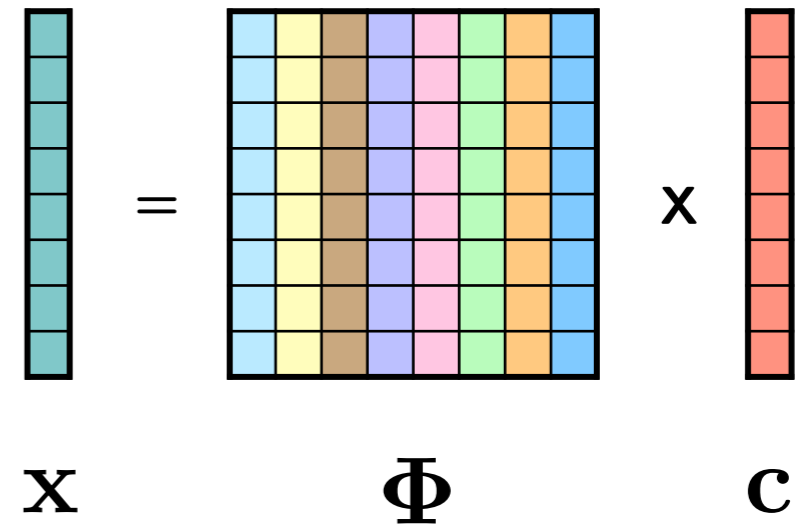
Classical vs. Graph dictionaries

classical signal

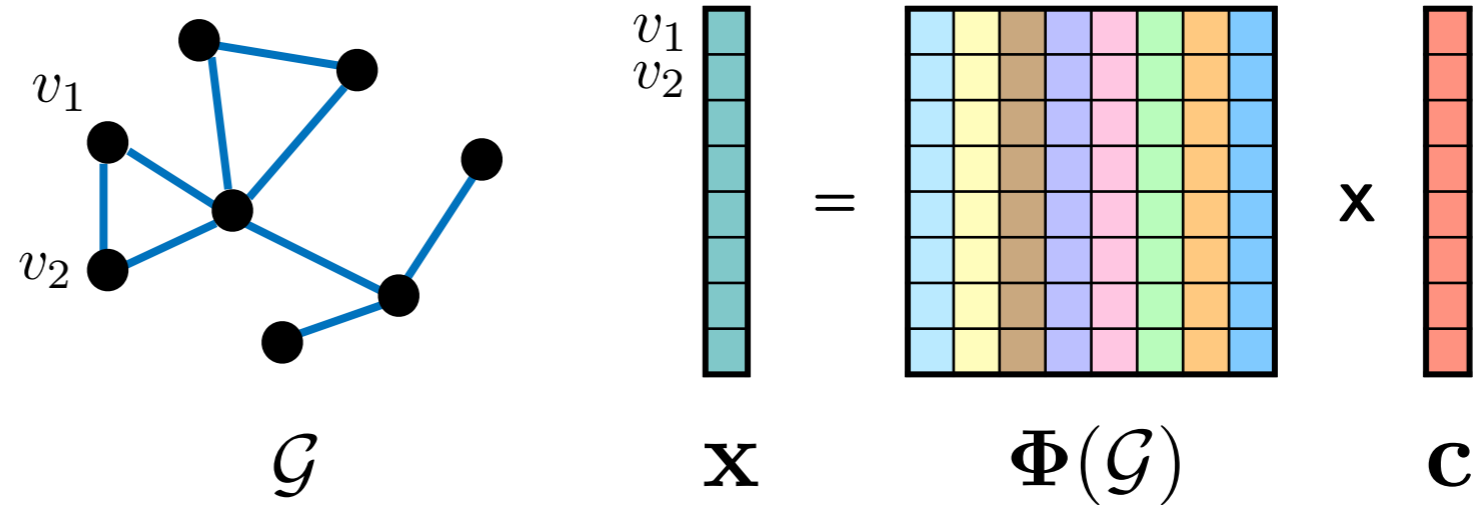


Classical vs. Graph dictionaries

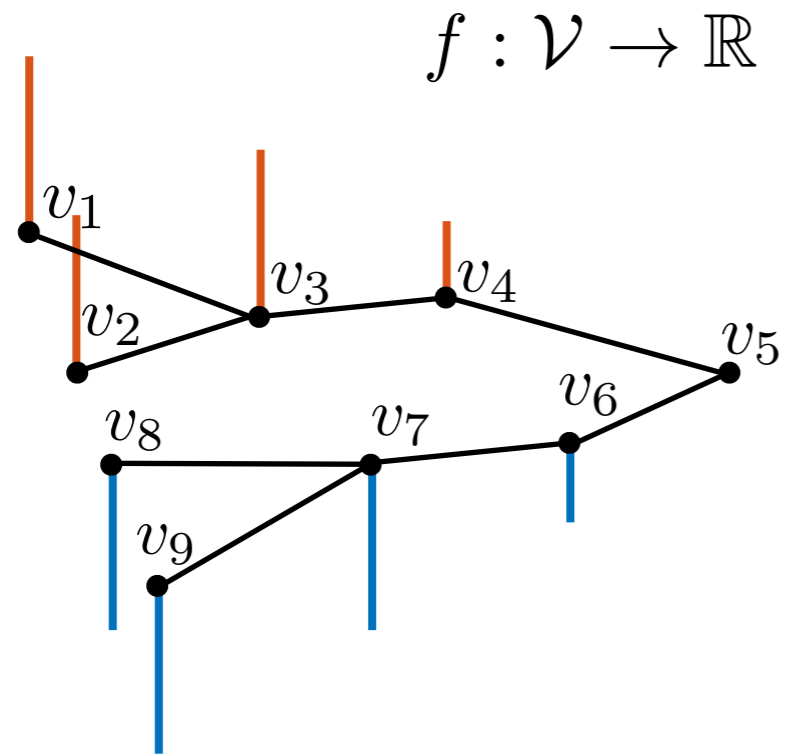
classical signal



graph signal

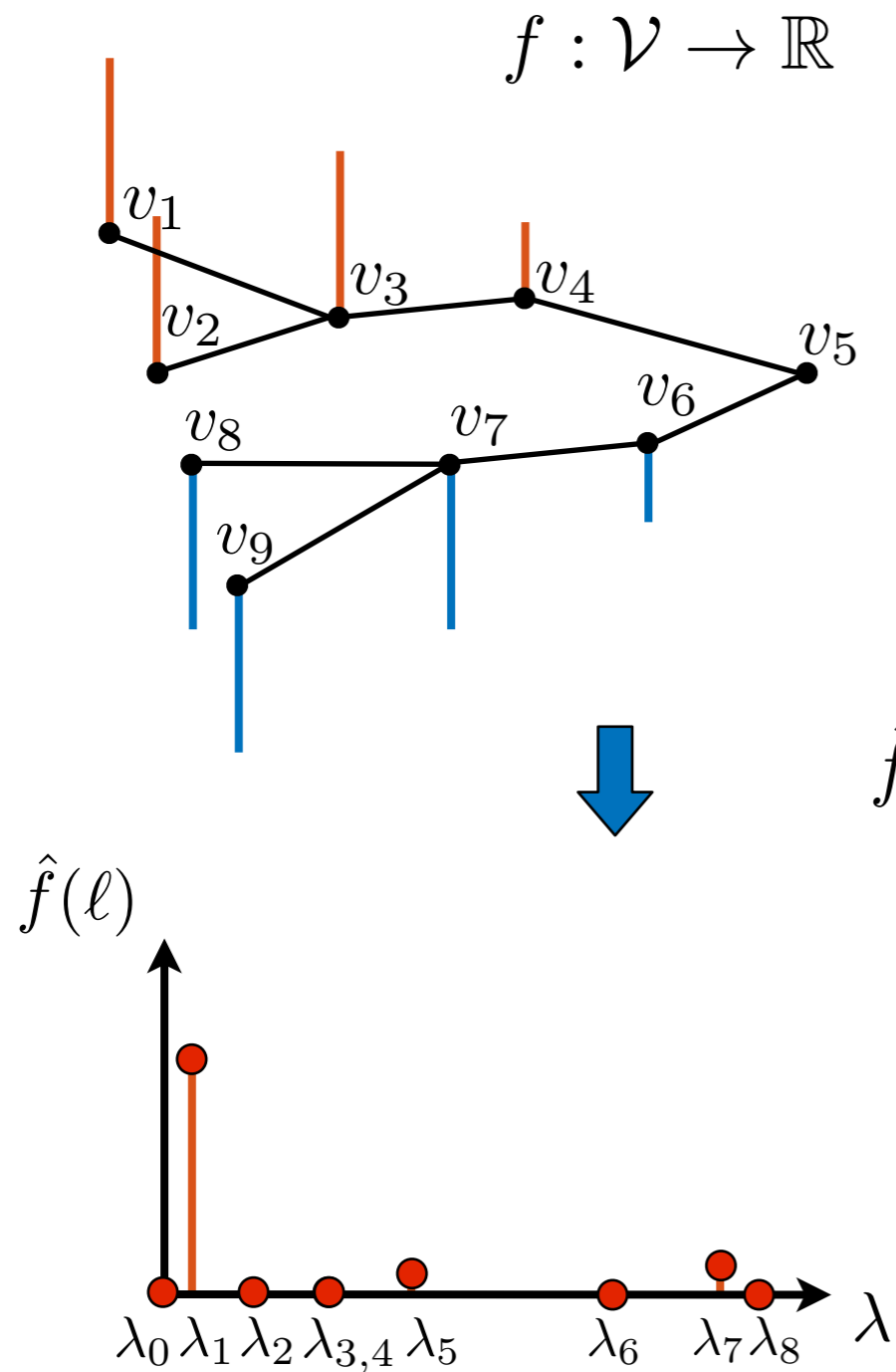


GFT provides a first graph dictionary



vertex domain

GFT provides a first graph dictionary

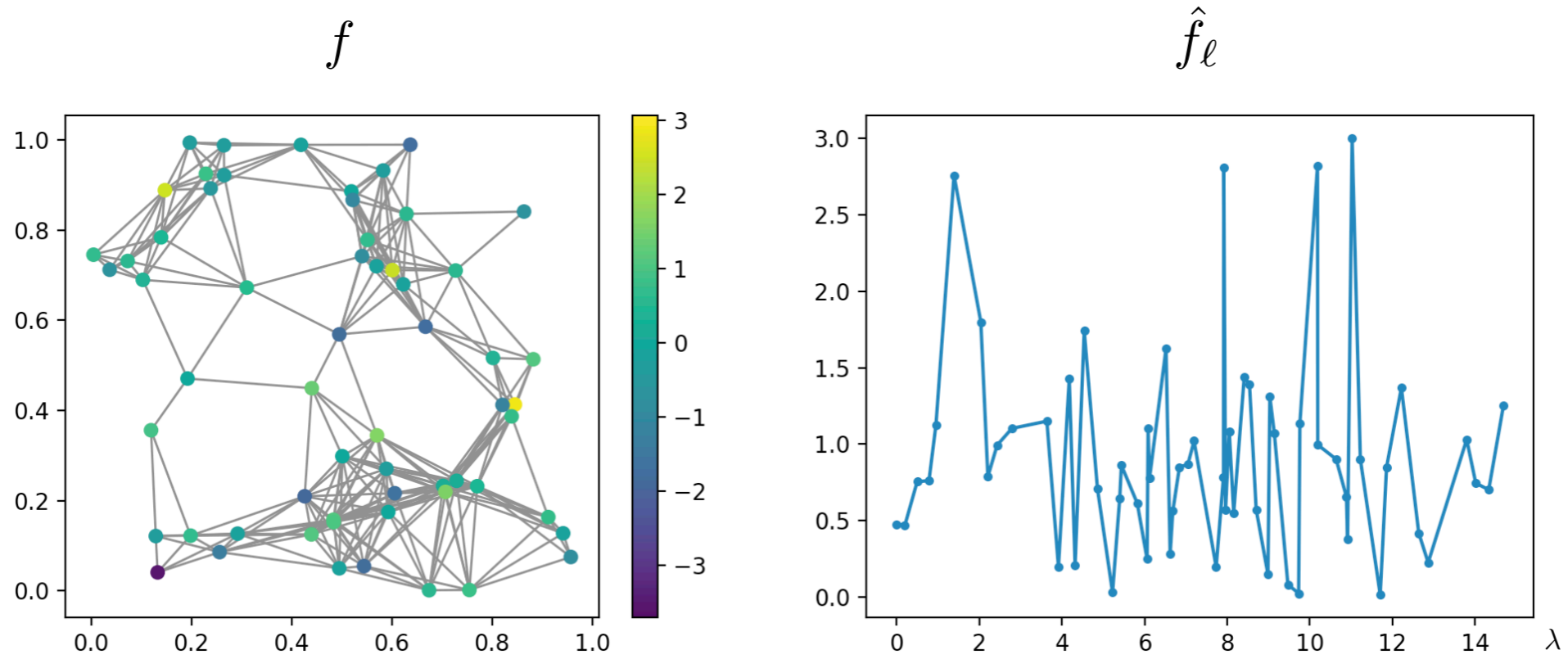


vertex domain

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

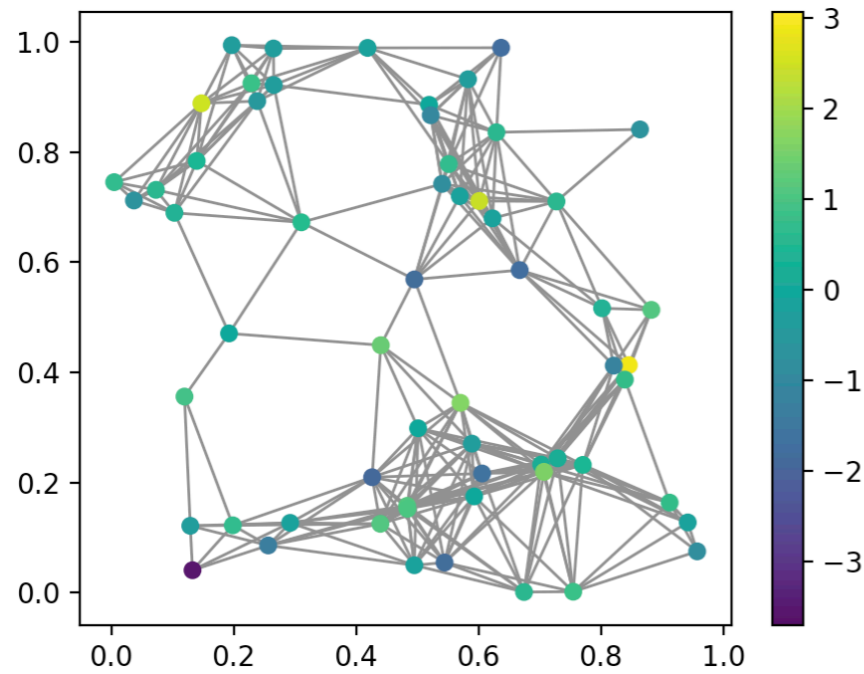
frequency (graph spectral) domain

GFT provides a first graph dictionary

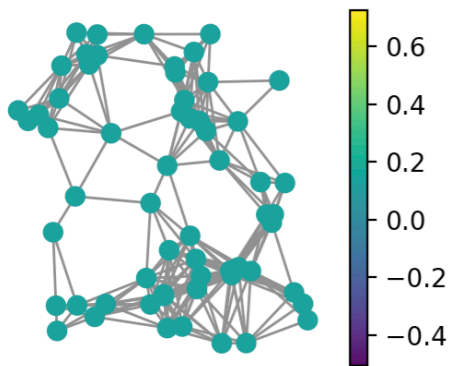
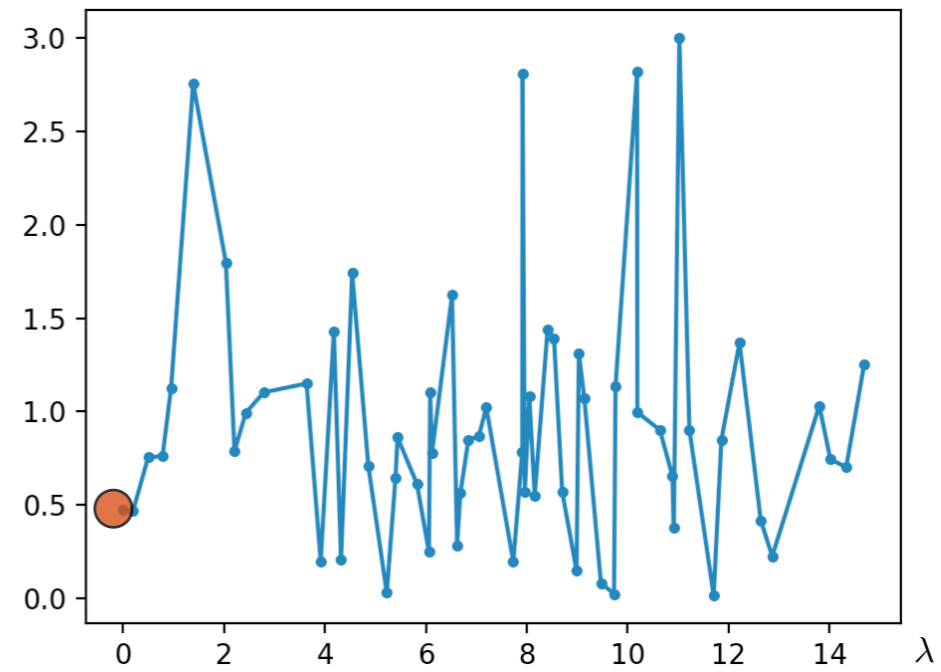


GFT provides a first graph dictionary

f



\hat{f}_ℓ

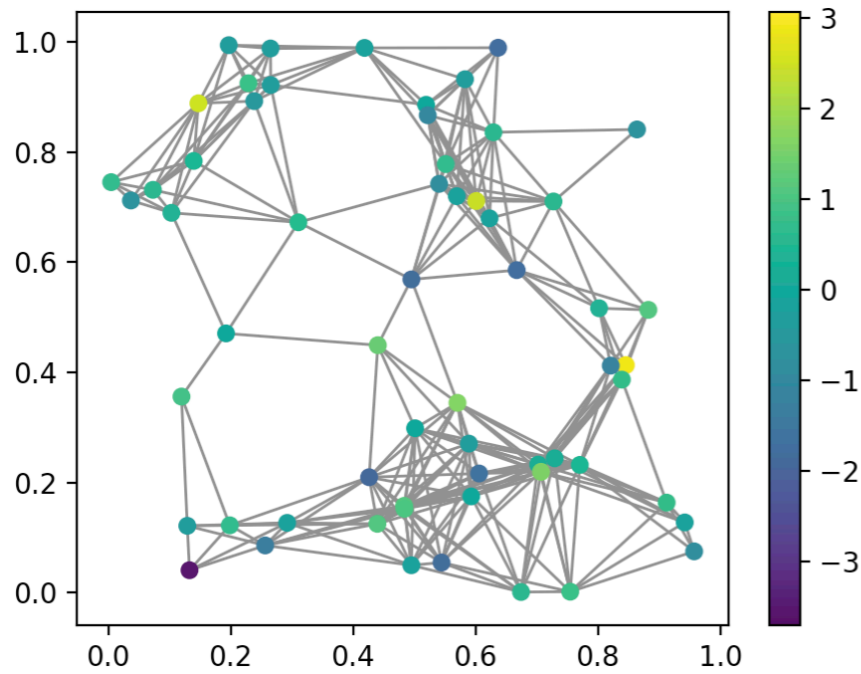


χ_0

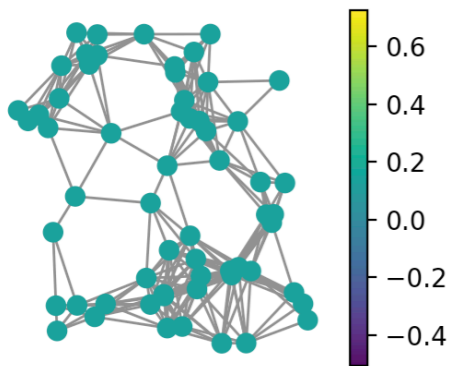
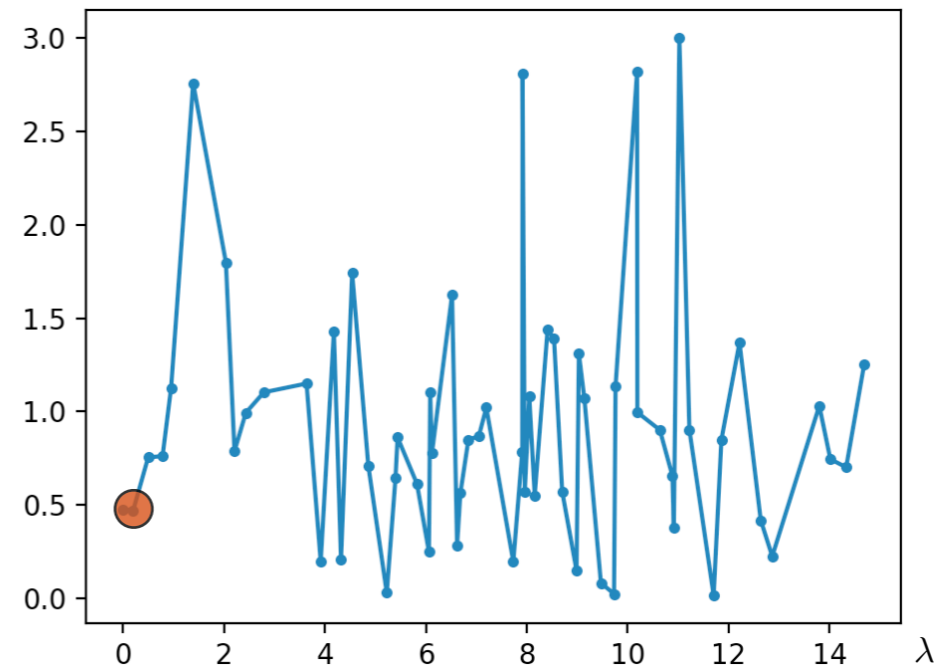
GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary

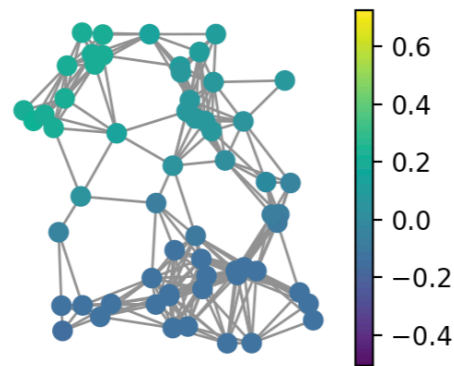
f



\hat{f}_ℓ



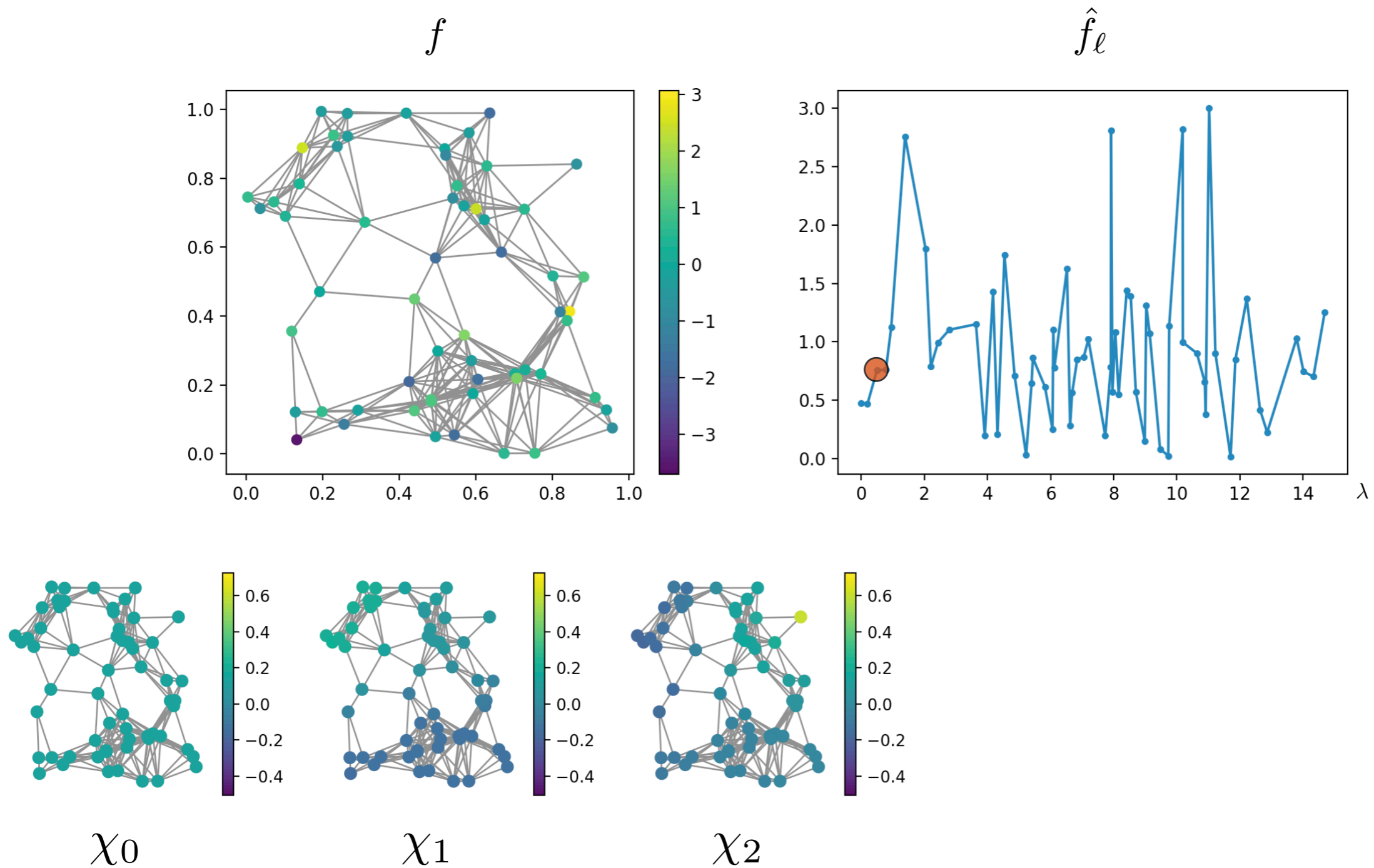
χ_0



χ_1

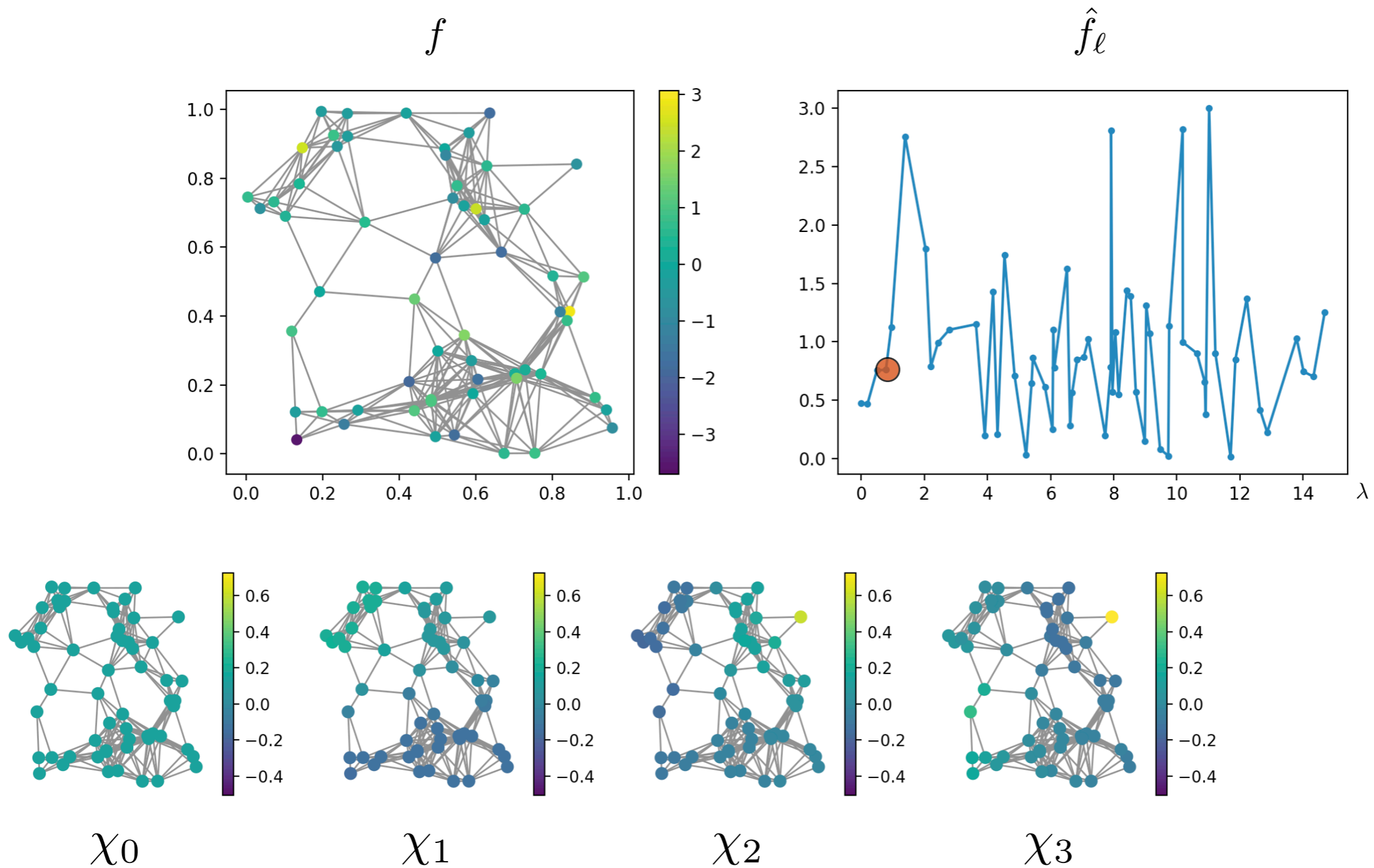
GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary



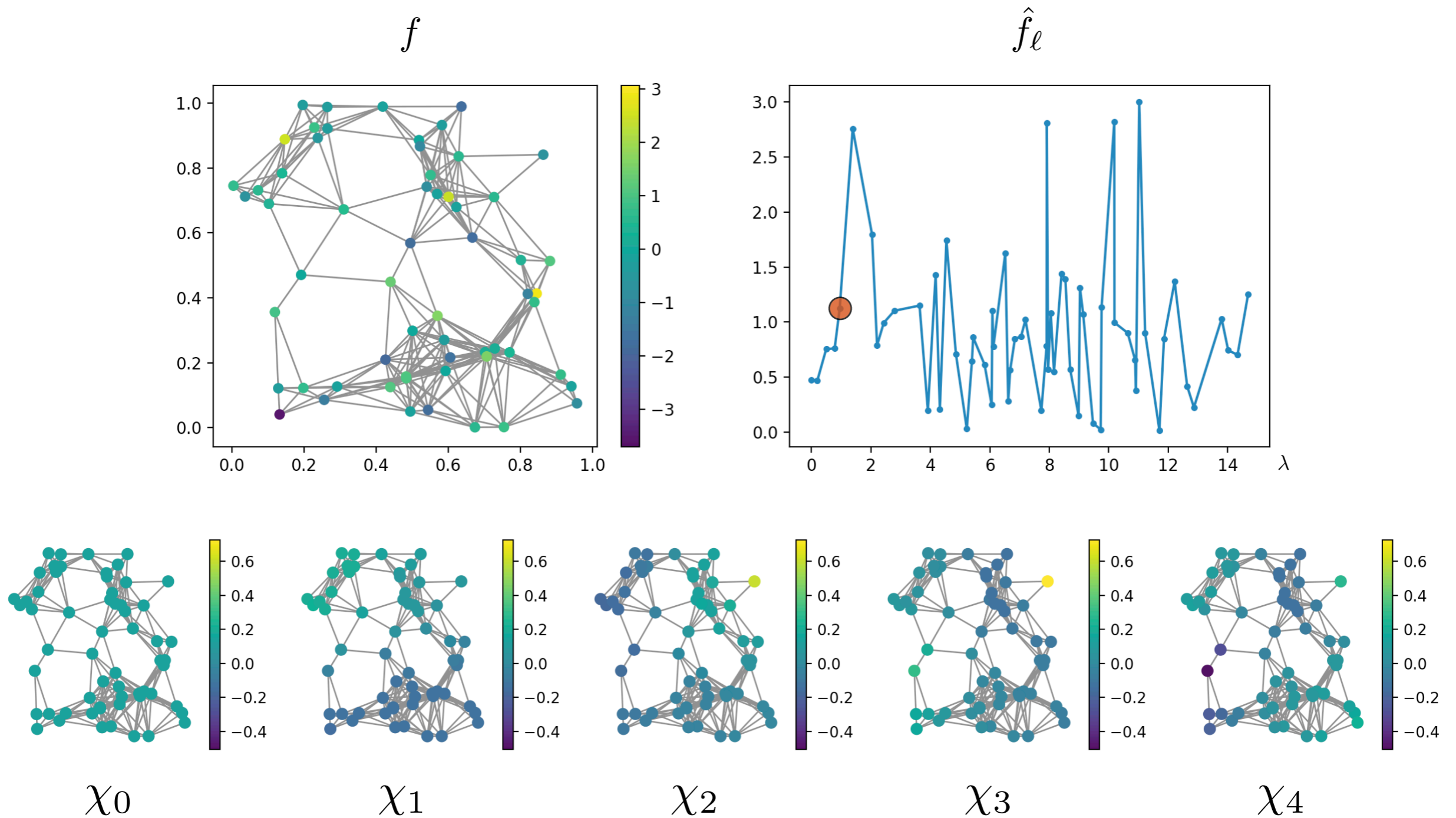
GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary



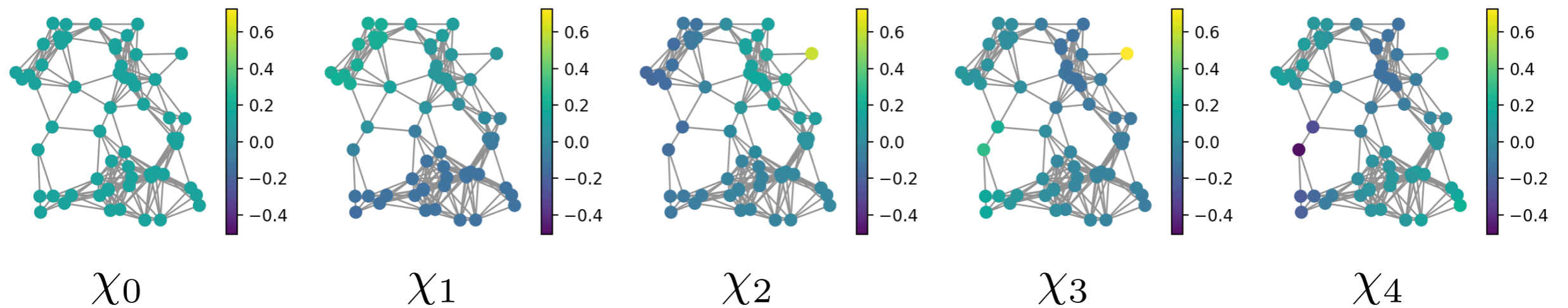
GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary

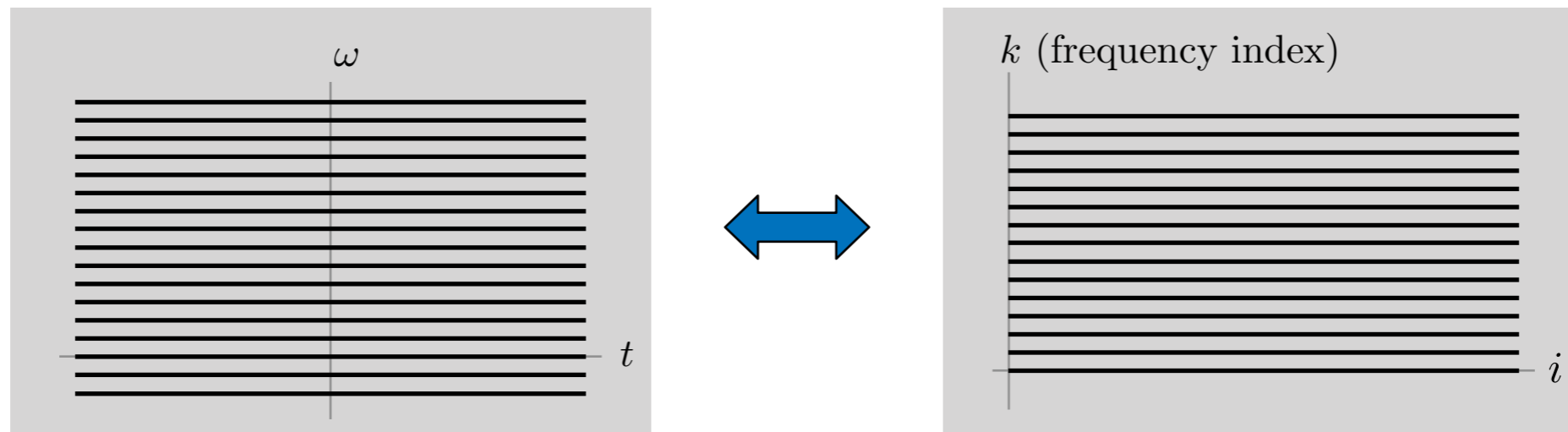
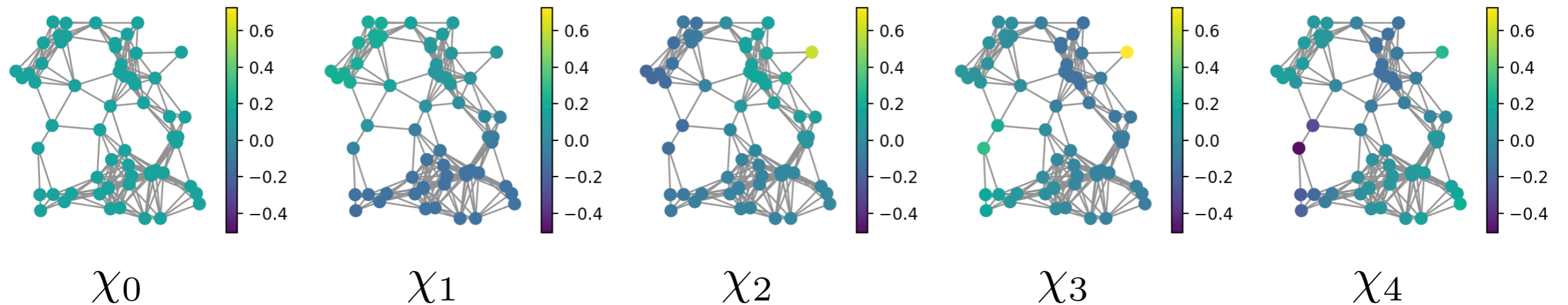


GFT atoms (corresponding to discrete frequencies)

GFT provides a first graph dictionary

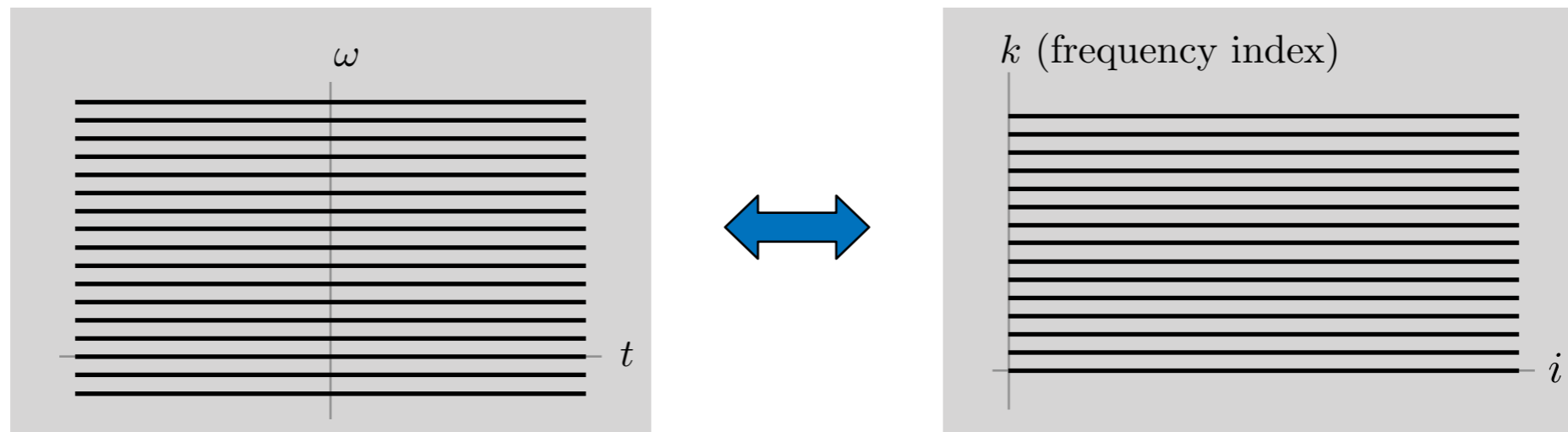
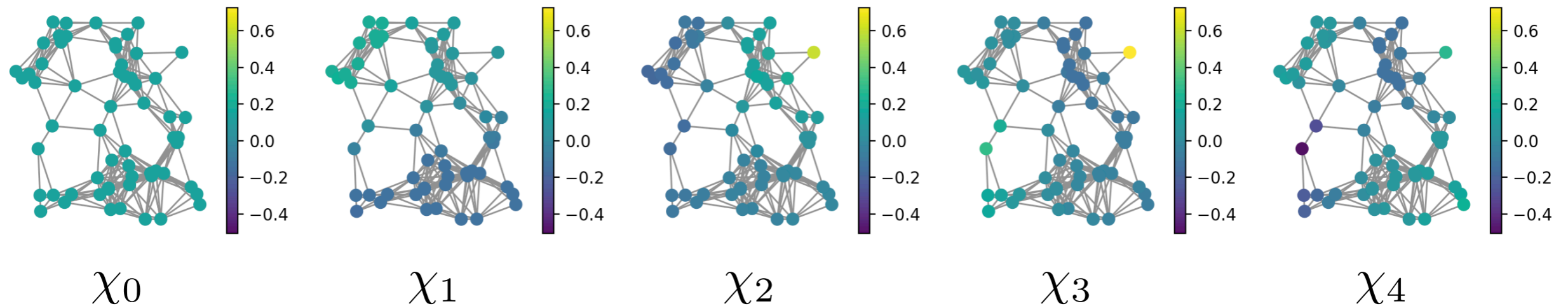


GFT provides a first graph dictionary



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

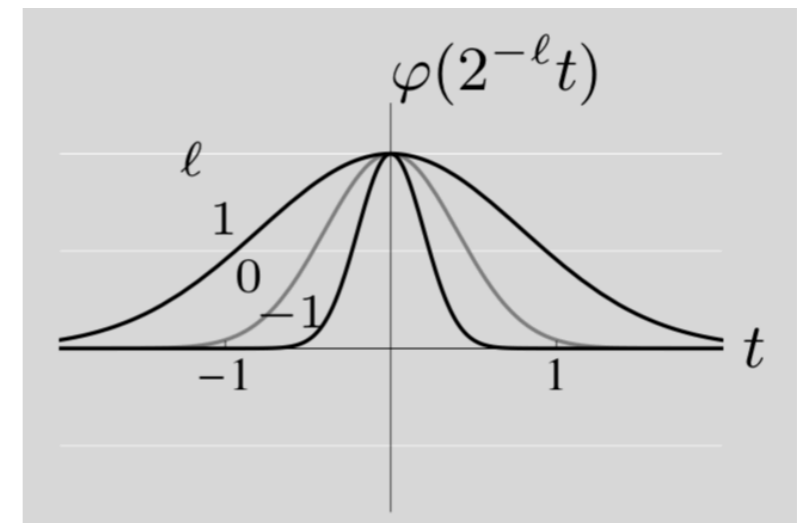
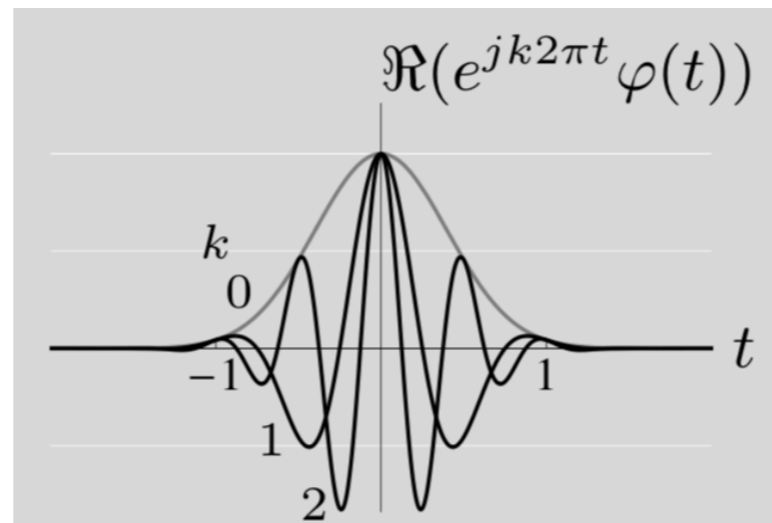
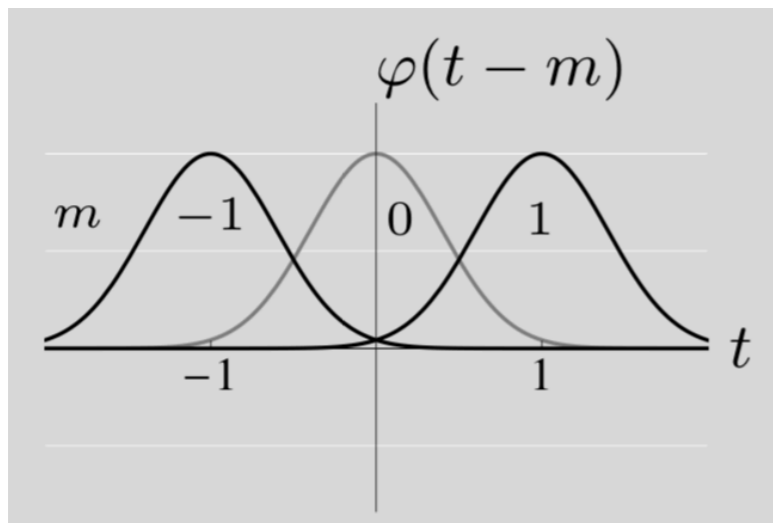
GFT provides a first graph dictionary



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

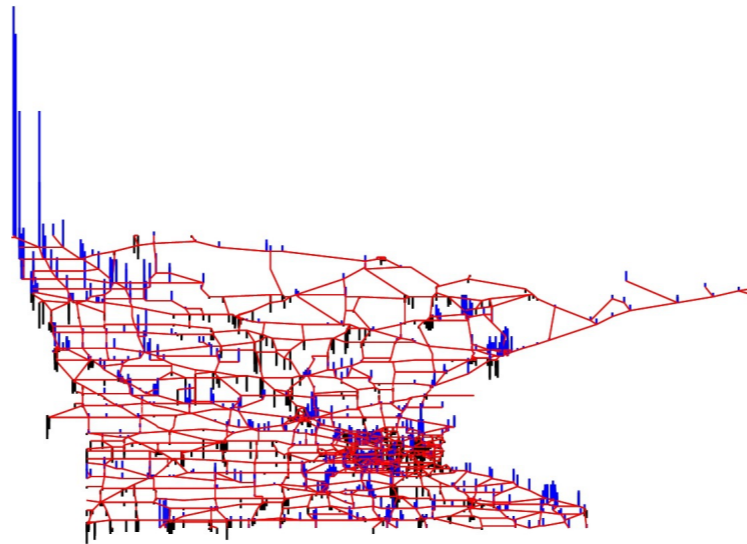
Basic operations for graph signals

basic operations in Euclidean domain



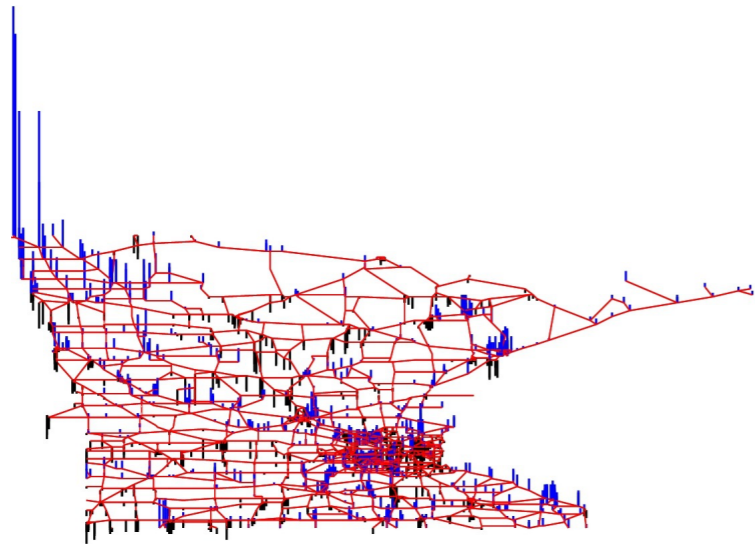
- recall that we used a set of structured functions (e.g., shifted and modulated) to produce localised items

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**

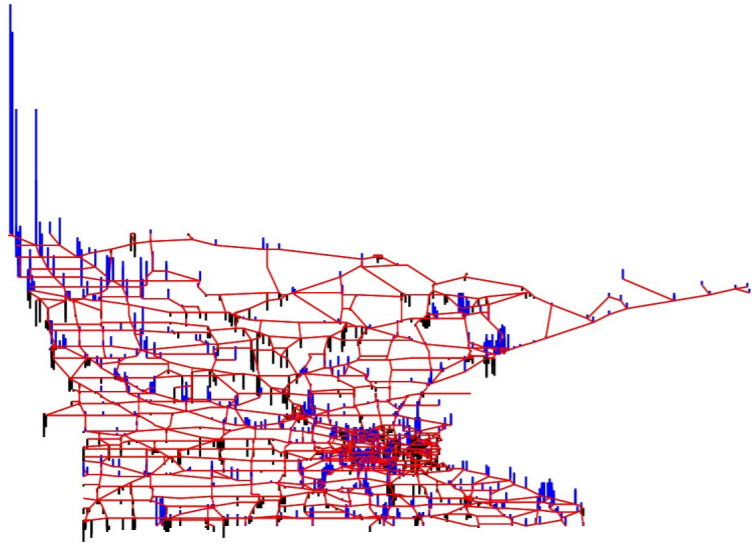
Convolution



classical convolution

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

Convolution

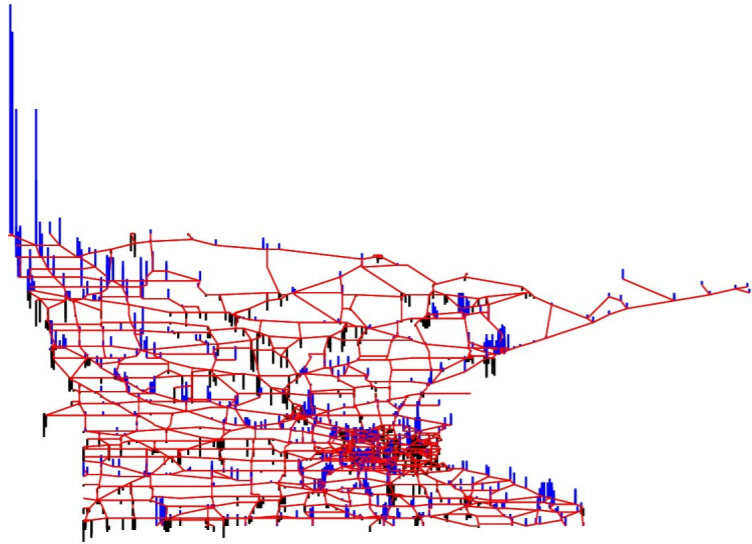


classical convolution

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

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

Convolution



classical convolution

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

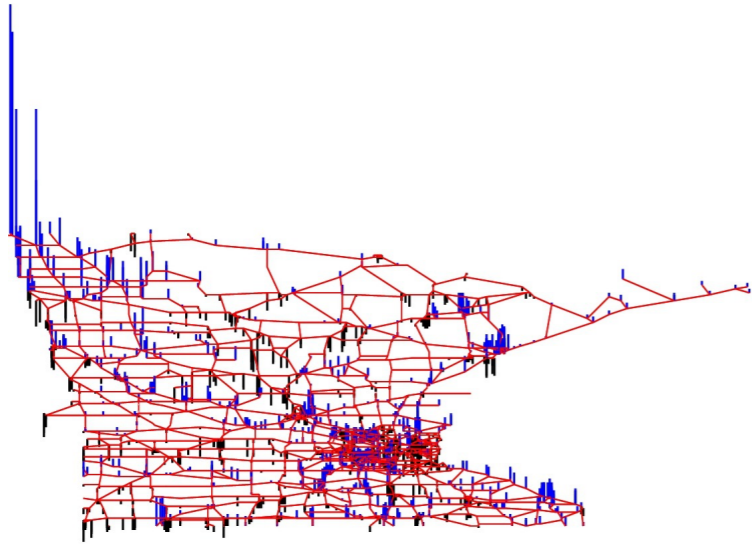
$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

graph convolution

multiplication in graph spectral domain

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

Convolution



classical convolution

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

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

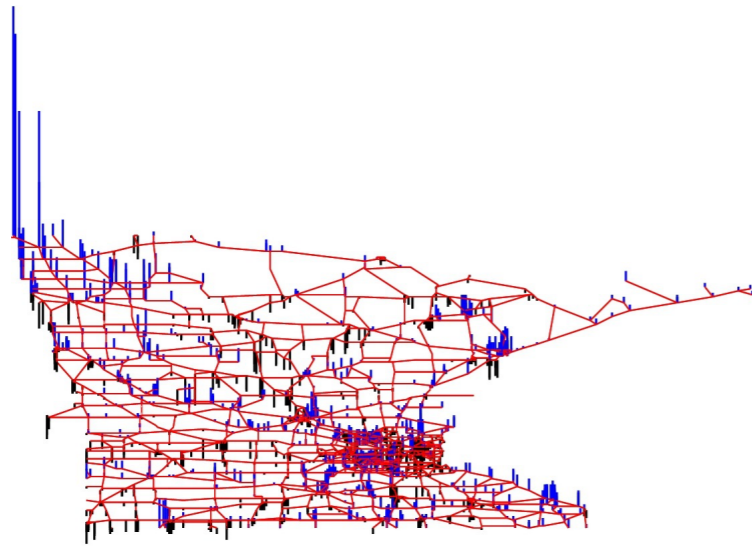
graph convolution

multiplication in graph spectral domain

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

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

Vertex-domain shift

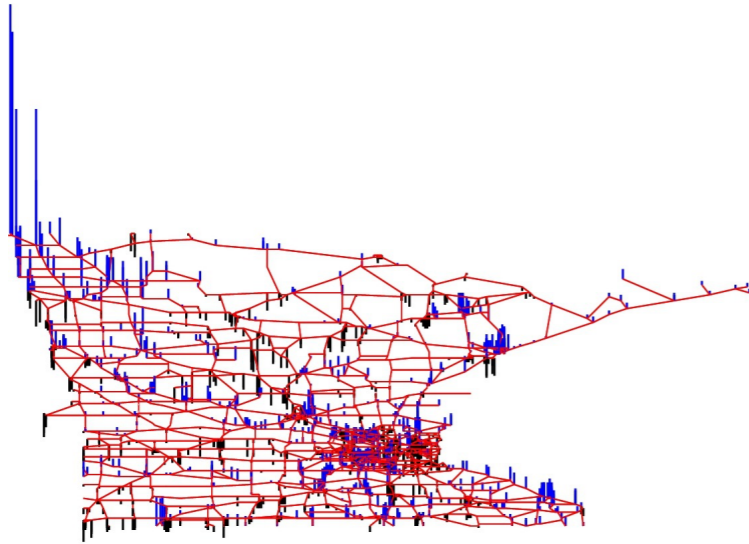


original signal

classical shift

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

Vertex-domain shift



original signal

classical shift

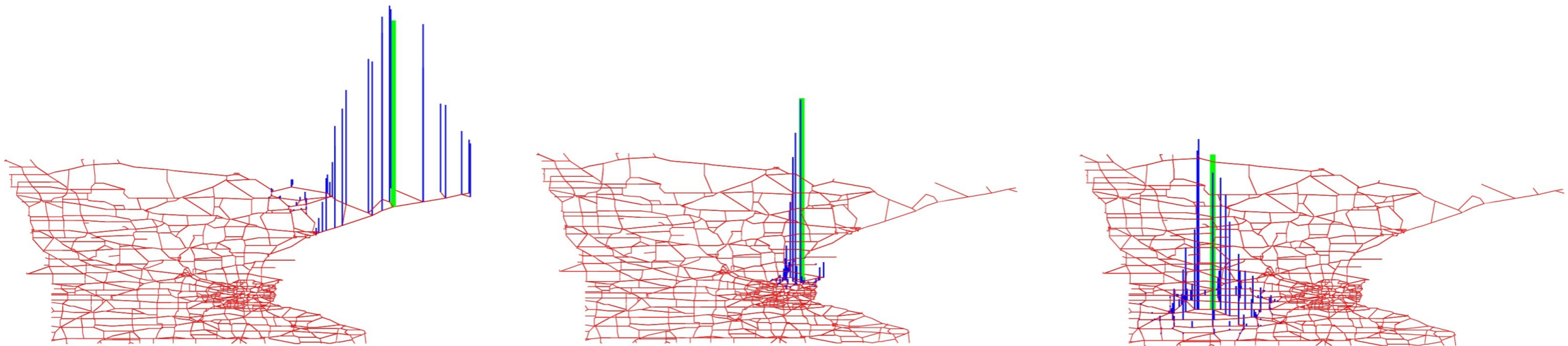
$$(T_u f)(t) := f(t - u) = (f * \delta_u)(t)$$

graph shift

convolution with a “delta” on graph

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

Vertex-domain shift



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

classical shift

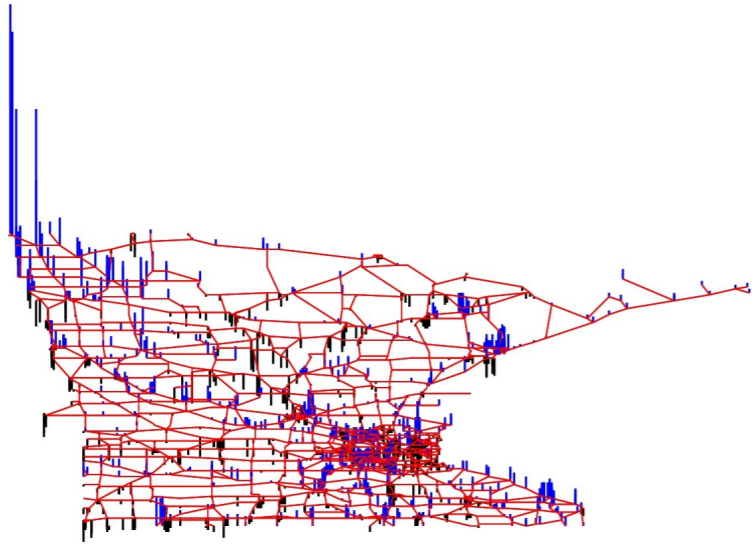
$$(T_u f)(t) := f(t - u) = (f * \delta_u)(t)$$

graph shift

convolution with a “delta” on graph

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

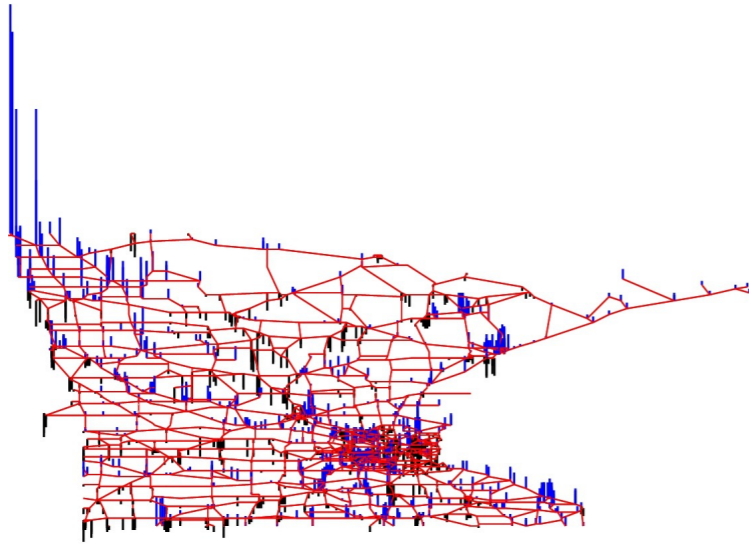
Modulation



classical modulation

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

Modulation



classical modulation

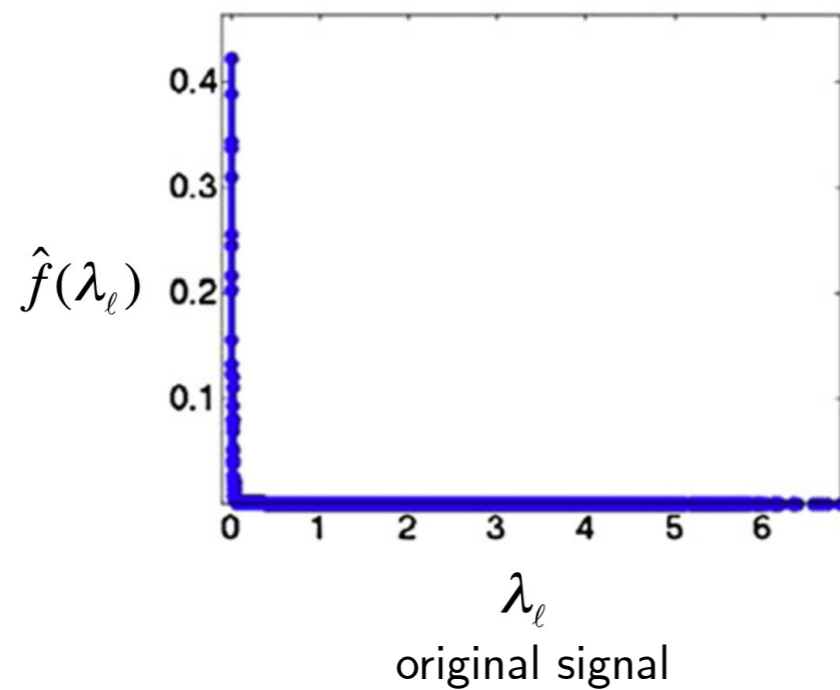
$$(M_{\xi} f)(t) := e^{j2\pi\xi t} f(t)$$

graph modulation

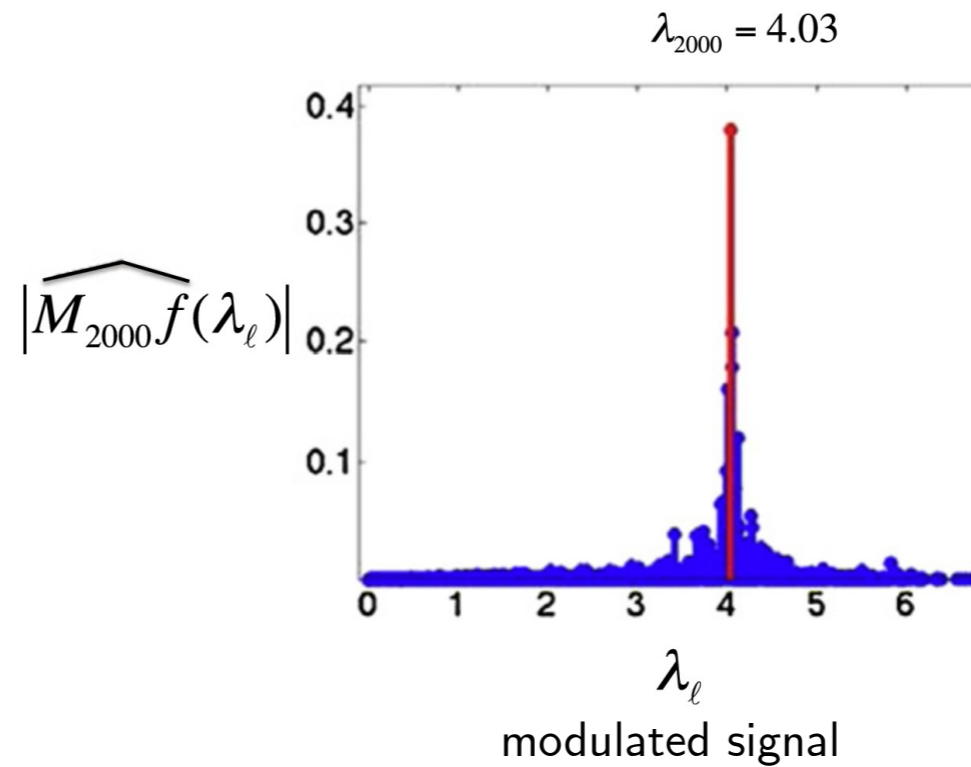
multiply by a graph Laplacian eigenvector

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

Modulation



classical modulation



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

graph modulation

multiply by a graph Laplacian eigenvector

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

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

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

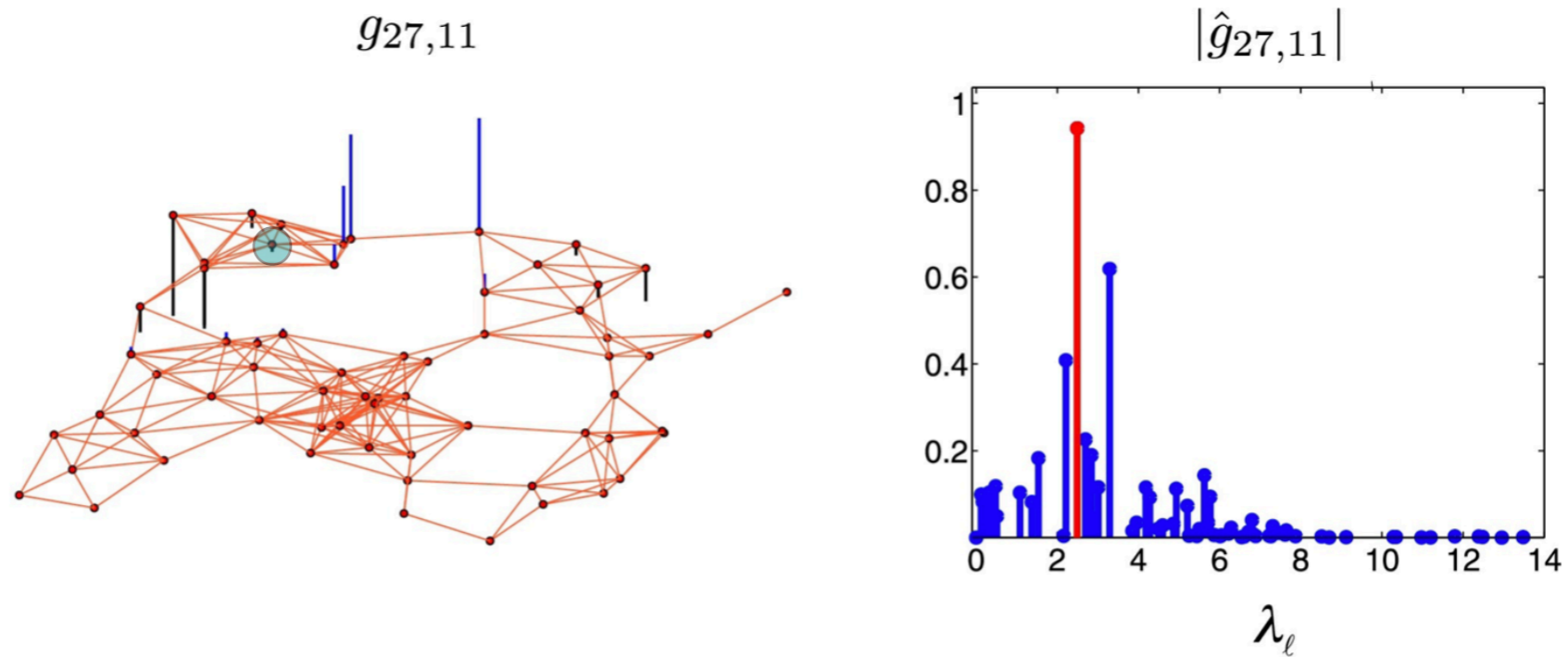
windowed graph
Fourier atom

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

windowed graph
Fourier transform

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

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)

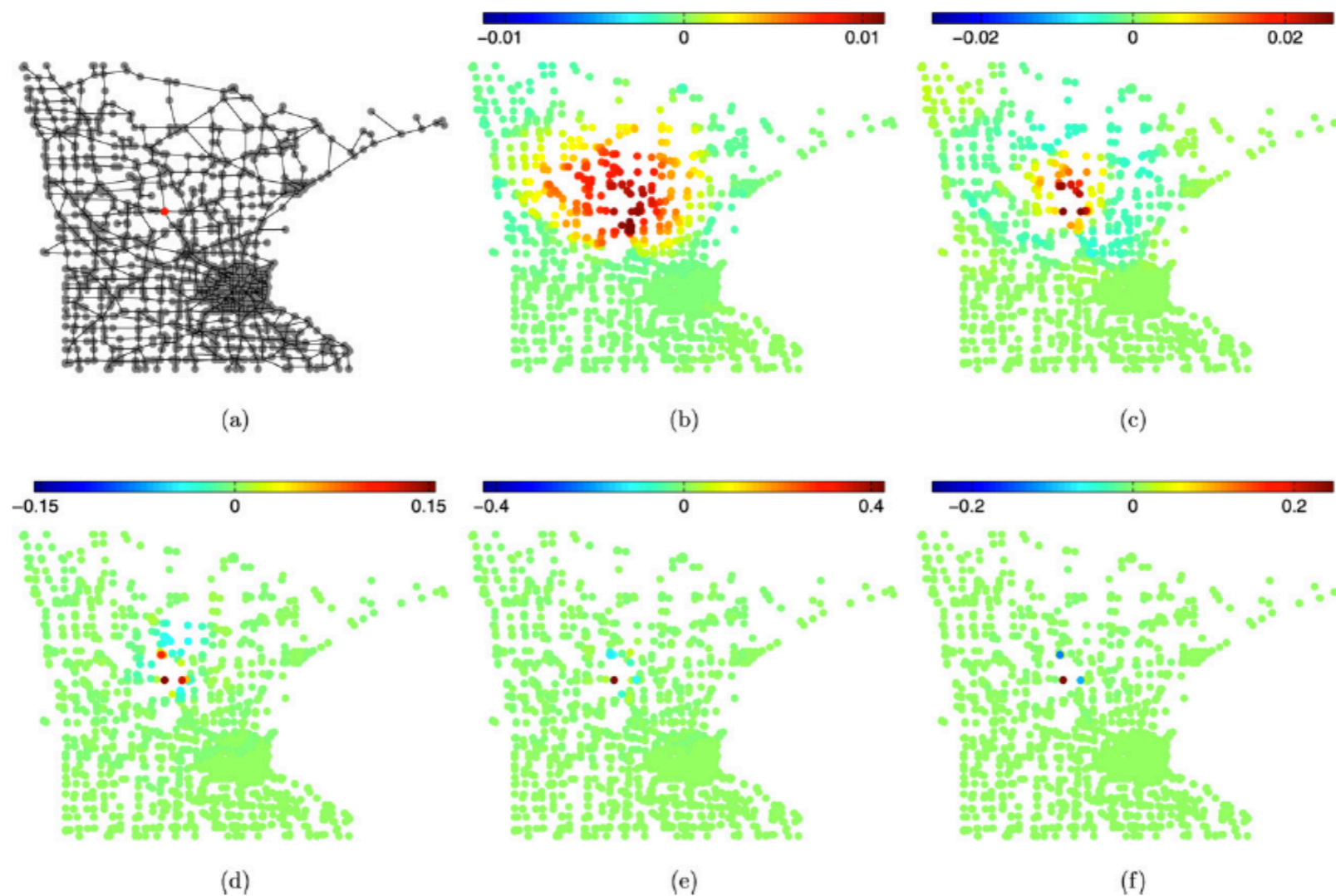


Fig. 4. Spectral graph wavelets on Minnesota road graph, with $K = 100$, $J = 4$ scales. (a) Vertex at which wavelets are centered, (b) scaling function, (c)–(f) wavelets, scales 1–4.

Wavelets on graphs

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

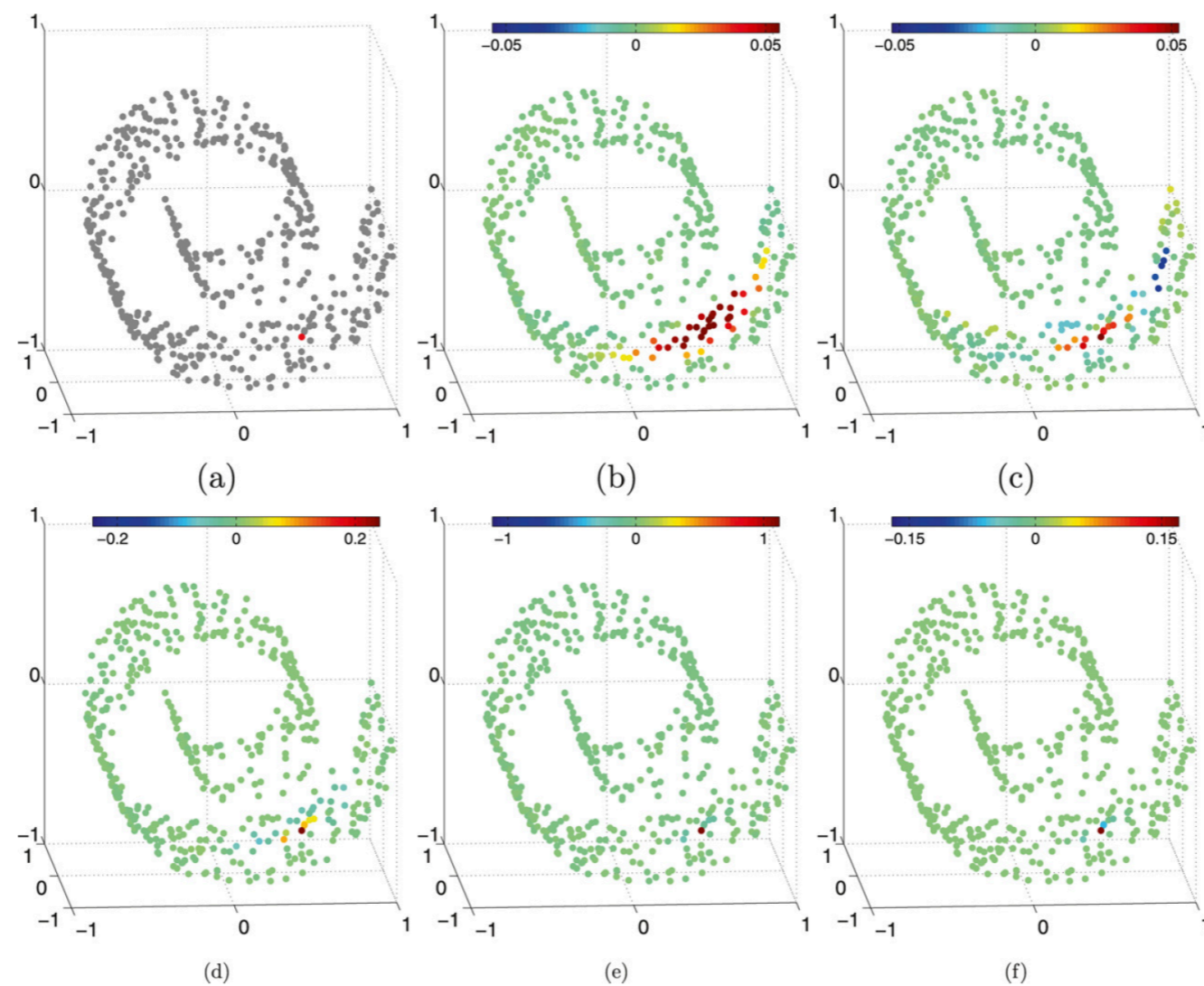


Fig. 3. Spectral graph wavelets on Swiss roll data cloud, with $J = 4$ wavelet scales. (a) Vertex at which wavelets are centered, (b) scaling function, (c)-(f) wavelets, scales 1-4.

WGFT vs SGWT atoms

WGFT atom

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

SGWT atom

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

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

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

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

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

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

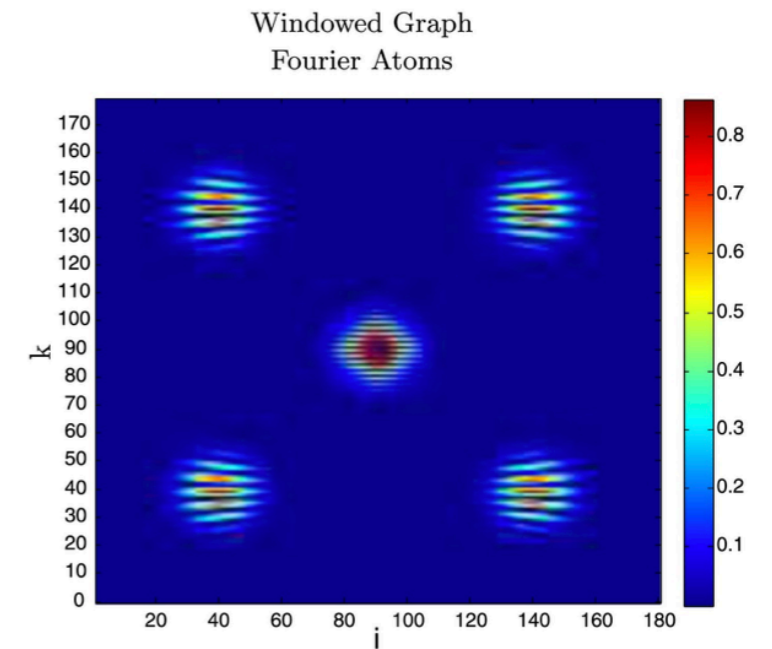
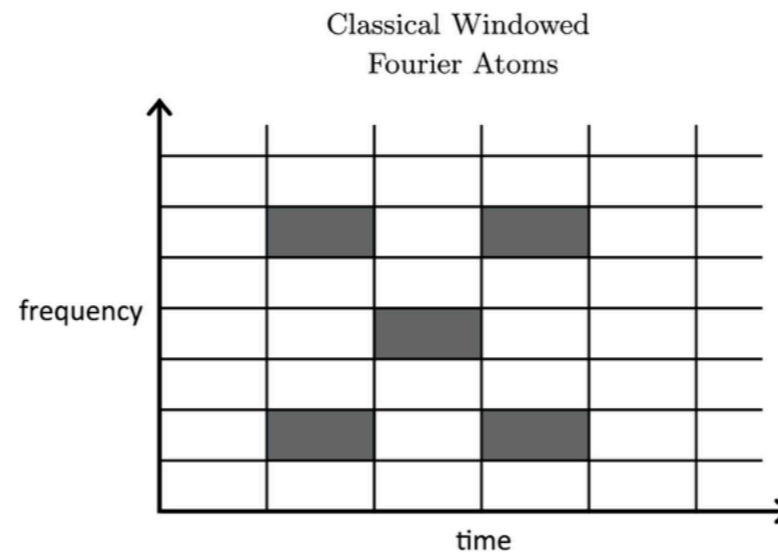
$$= \sum_{\ell=0}^{N-1} \hat{g}(s \lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$

WGFT vs SGWT atoms

WGFT atom

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

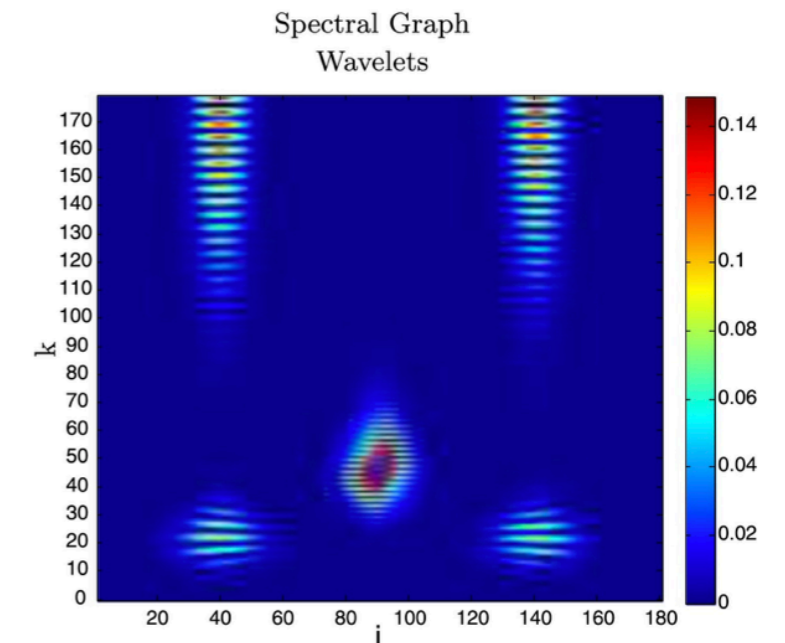
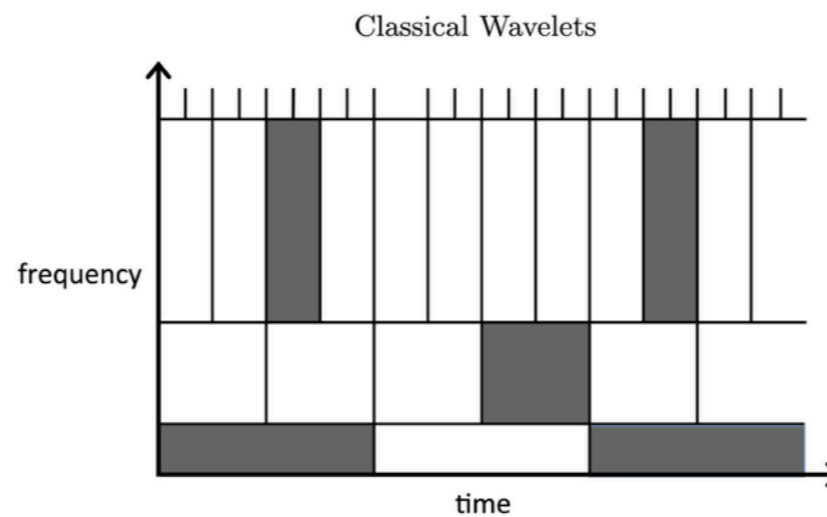
$$= N \chi_k(n) \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) \chi_\ell^*(i) \chi_\ell(n)$$



SGWT atom

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

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



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

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

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Convolution on graphs (revisited)

classical convolution

time domain

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

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Convolution on graphs (revisited)

classical convolution

time domain

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



frequency domain

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

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

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

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



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

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



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

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



graph spectral domain

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



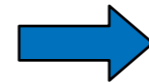
A parametric filter

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



parametric filter as polynomial of Laplacian

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



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

A parametric filter

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



parametric filter as polynomial of Laplacian

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

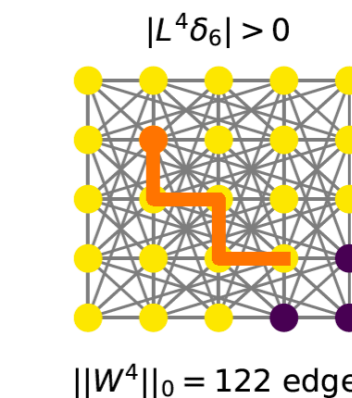
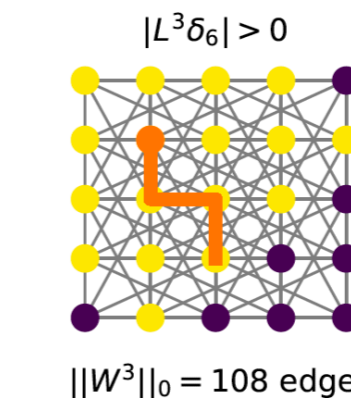
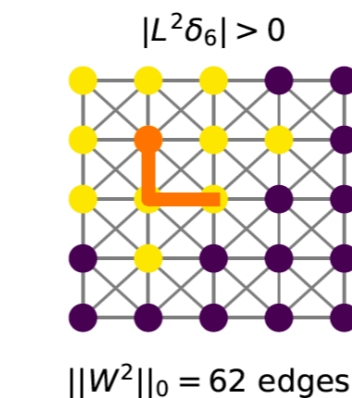
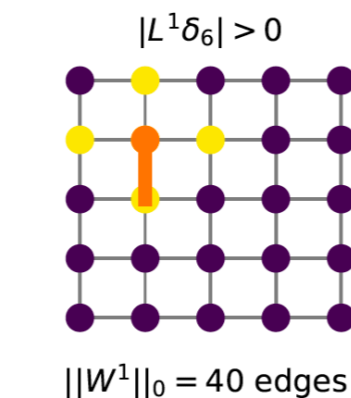
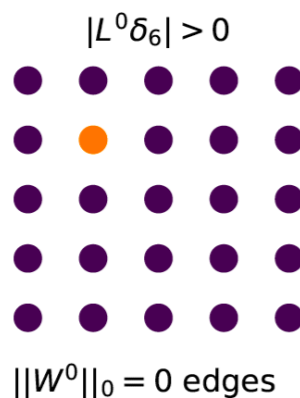
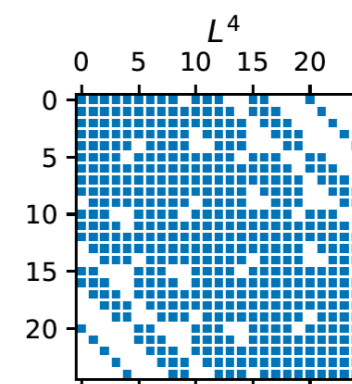
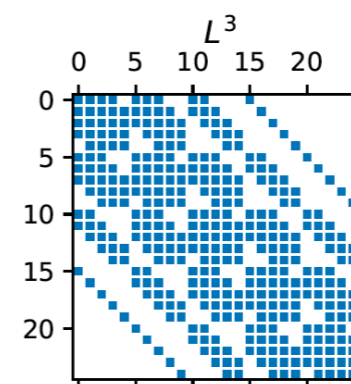
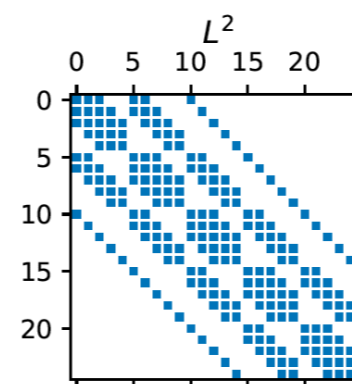
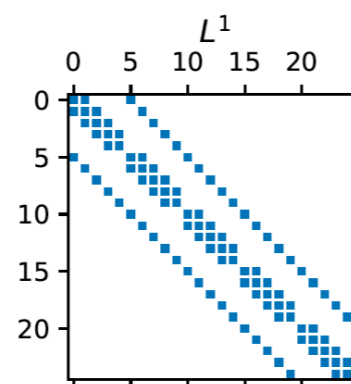
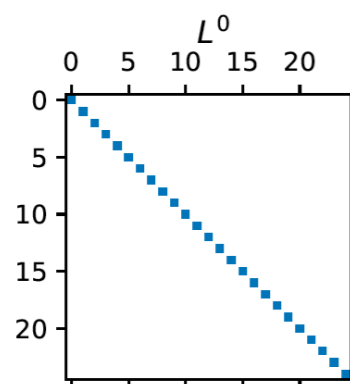


$$\hat{g}_\theta(L) = \sum_{j=0}^K \theta_j L^j$$

what do powers of graph Laplacian capture?

Powers of graph Laplacian

L^k defines the k -neighborhood



Localization: $d_G(v_i, v_j) > K$ implies $(L^K)_{ij} = 0$

(slide by Michaël Deferrard)

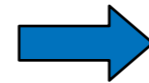
A parametric filter

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



parametric filter as polynomial of Laplacian

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



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

A parametric filter

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



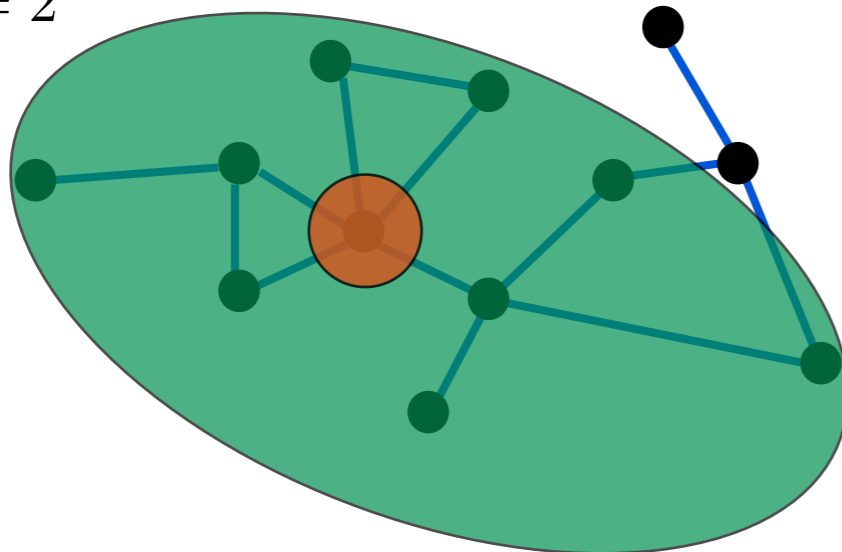
parametric filter as polynomial of Laplacian

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



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

$K = 2$



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

A simplified parametric filter

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

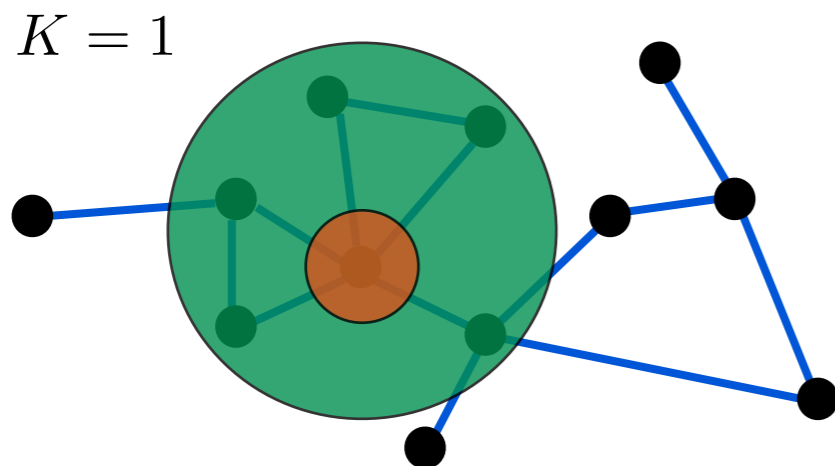


simplified parametric filter

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

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

(localisation within **1-hop** neighbourhood)



A simplified parametric filter

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

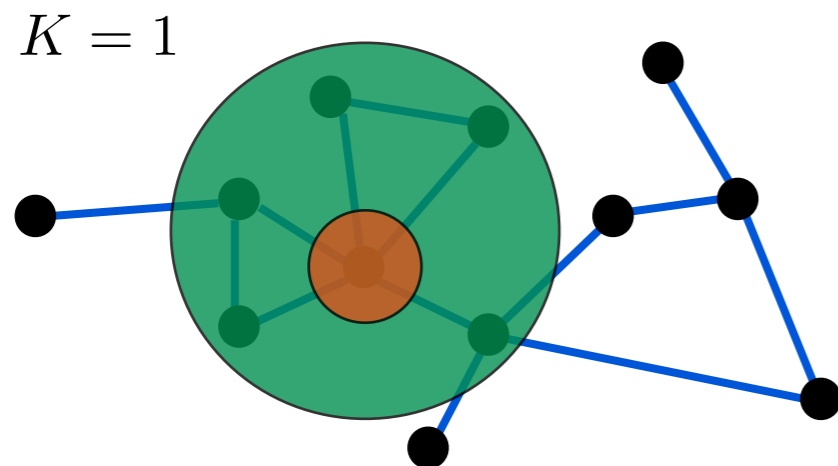


simplified parametric filter

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

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

(localisation within **1-hop** neighbourhood)



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

A simplified parametric filter

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



simplified parametric filter

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

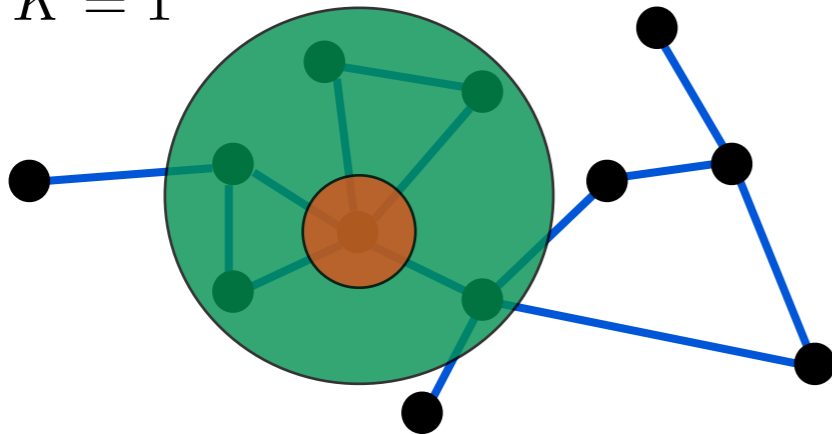
$K = 1$



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

(localisation within **1-hop** neighbourhood)

$K = 1$



$$\alpha = \theta_0 = -\theta_1$$



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

renormalisation



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

A simplified parametric filter

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



simplified parametric filter

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

A simplified parametric filter

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

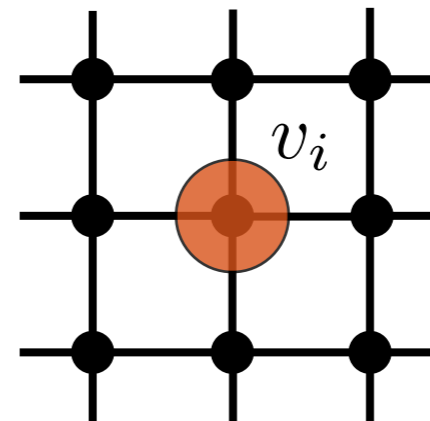


simplified parametric filter

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



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



A simplified parametric filter

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



simplified parametric filter

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

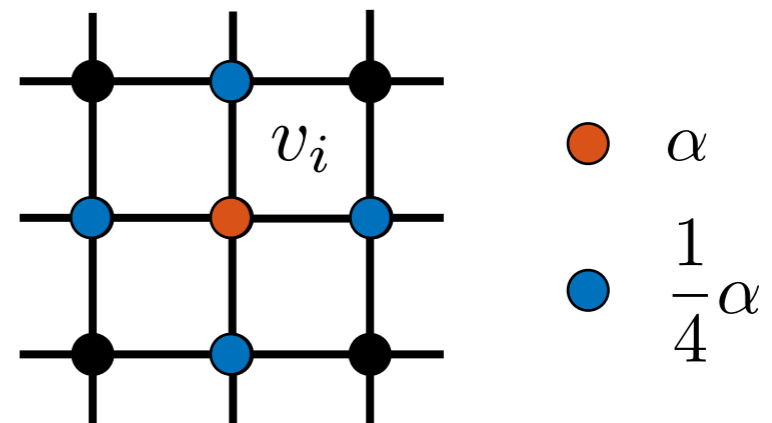


$$y_i = \alpha f_i + \alpha \frac{1}{\sqrt{d_i}} \sum_{j:(i,j) \in \mathcal{E}} w_{ij} \frac{1}{\sqrt{d_j}} f_j$$



unitary edge weights

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



A simplified parametric filter

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



simplified parametric filter

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



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

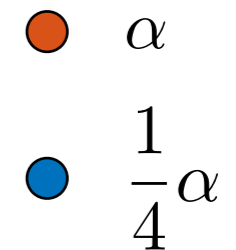
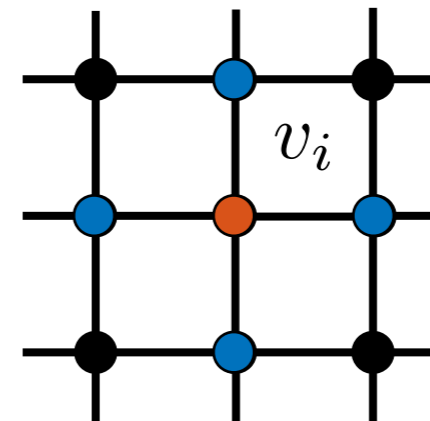


unitary edge weights

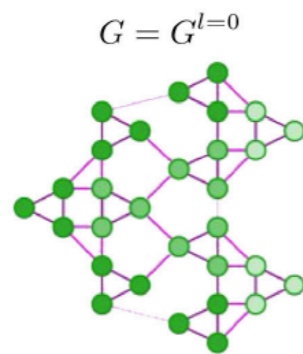
$$y_i = \alpha f_i + \frac{1}{4} \alpha \sum_{j:(i,j) \in \mathcal{E}} f_j$$

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0



CNN on graphs: Graph classification



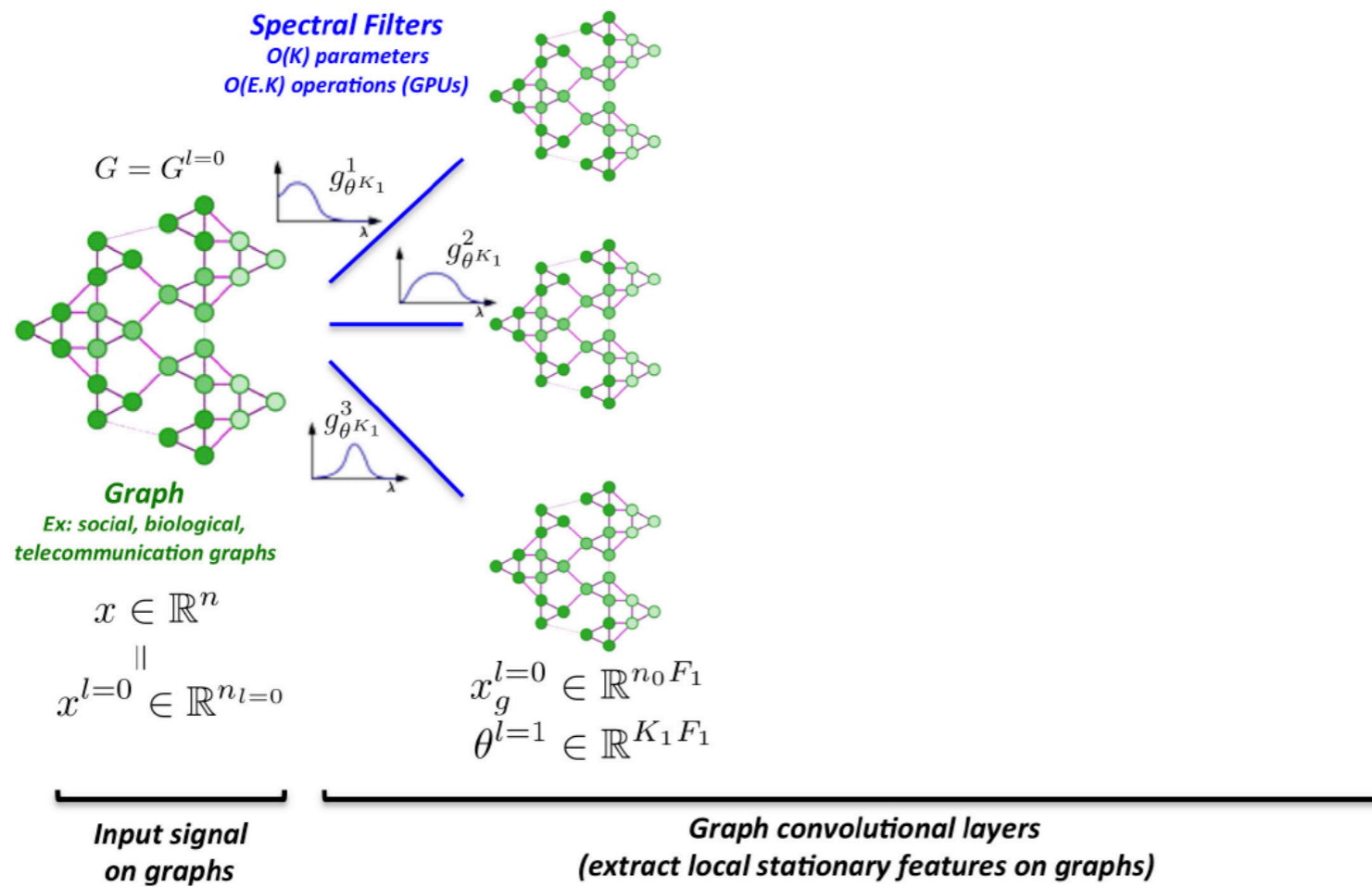
Graph

*Ex: social, biological,
telecommunication graphs*

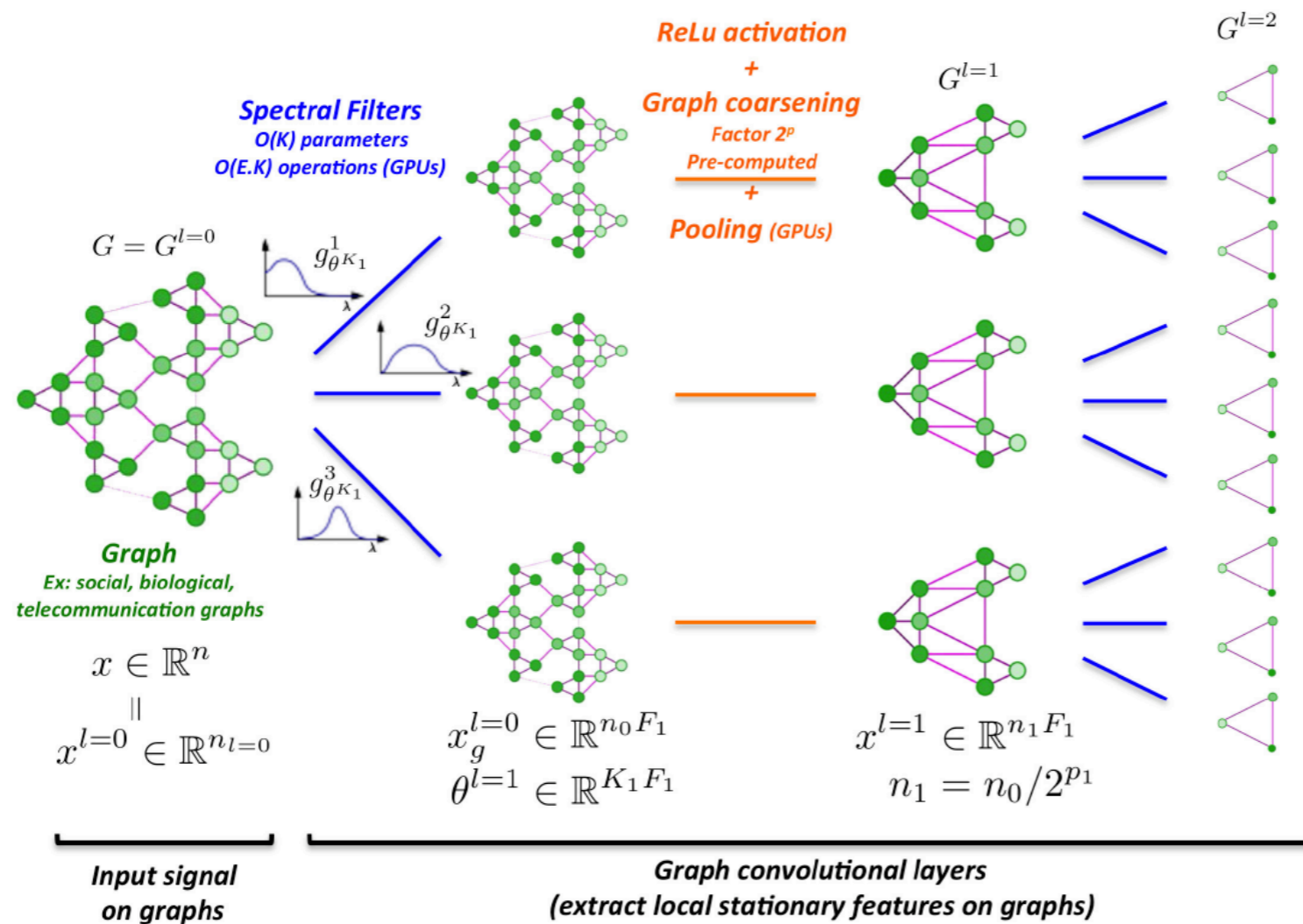
$$x \in \mathbb{R}^n$$
$$\parallel$$
$$x^{l=0} \in \mathbb{R}^{n_{l=0}}$$

**Input signal
on graphs**

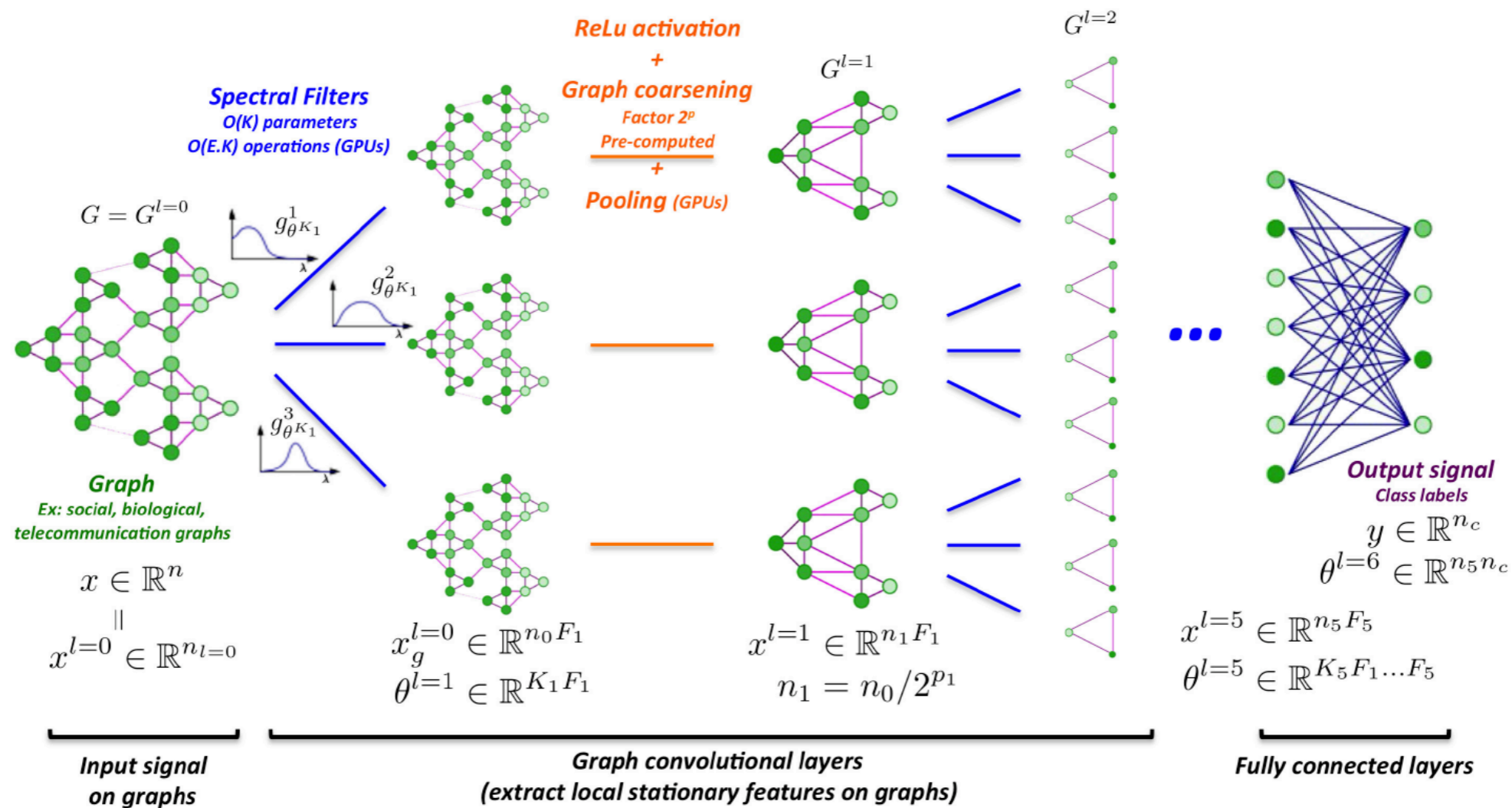
CNN on graphs: Graph classification



CNN on graphs: Graph classification

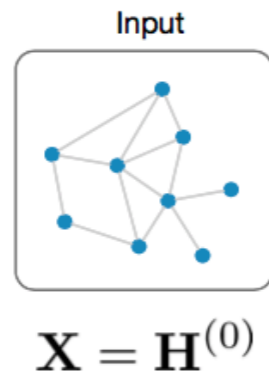


CNN on graphs: Graph classification



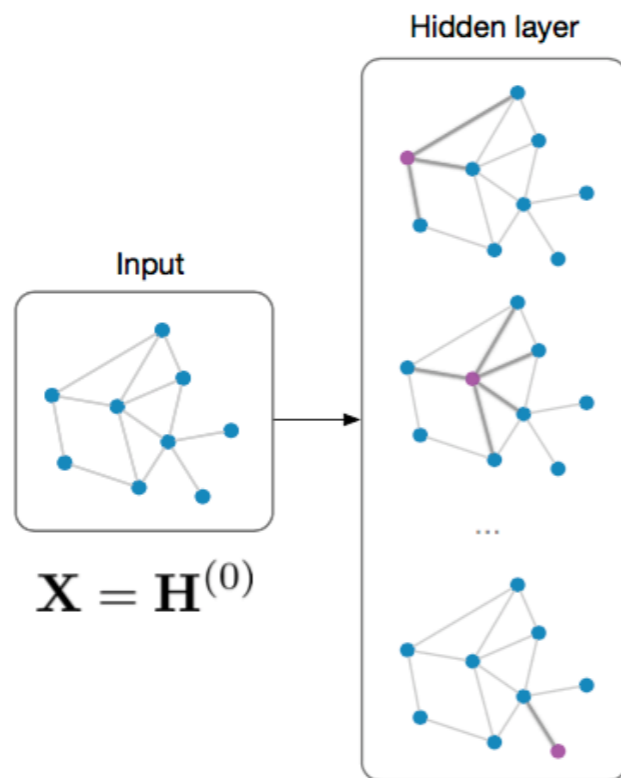
CNN on graphs: Node classification

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



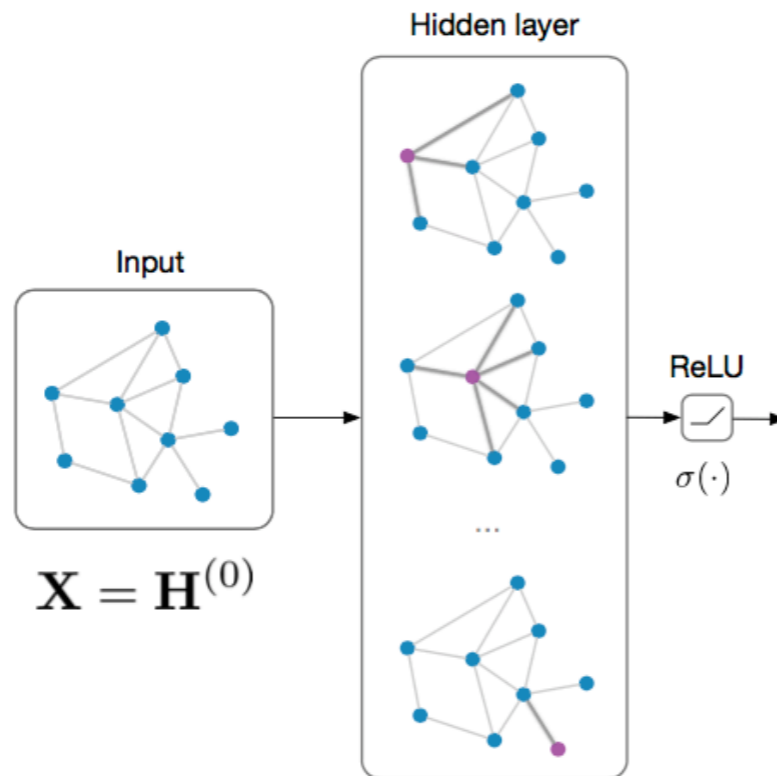
CNN on graphs: Node classification

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



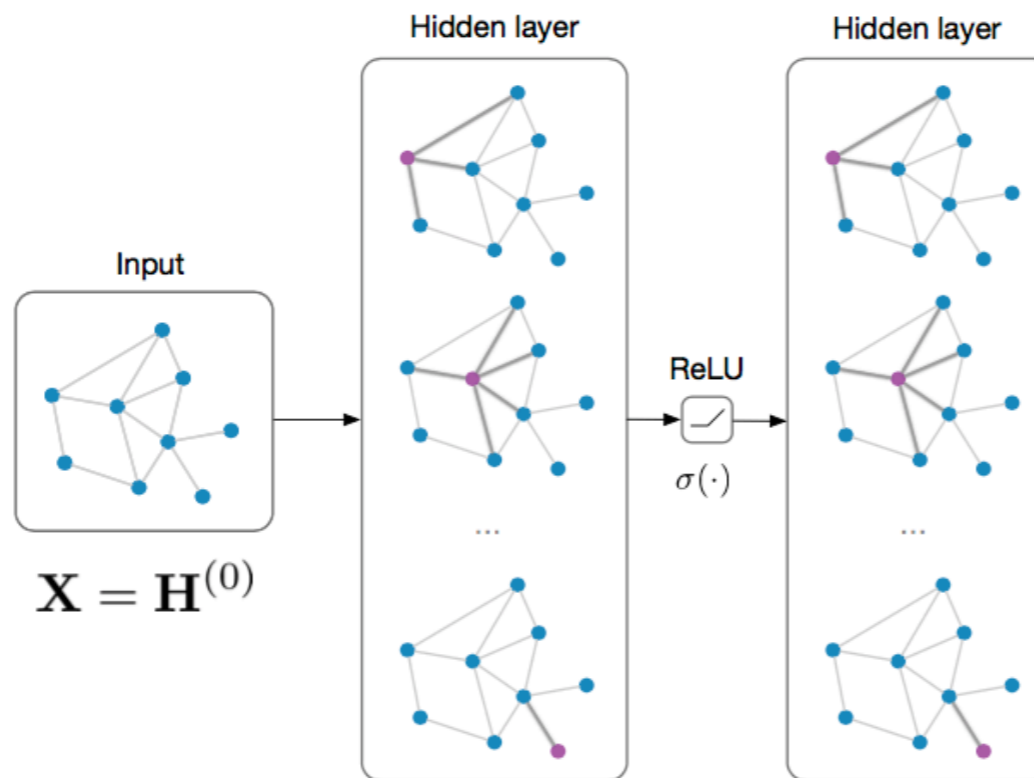
CNN on graphs: Node classification

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



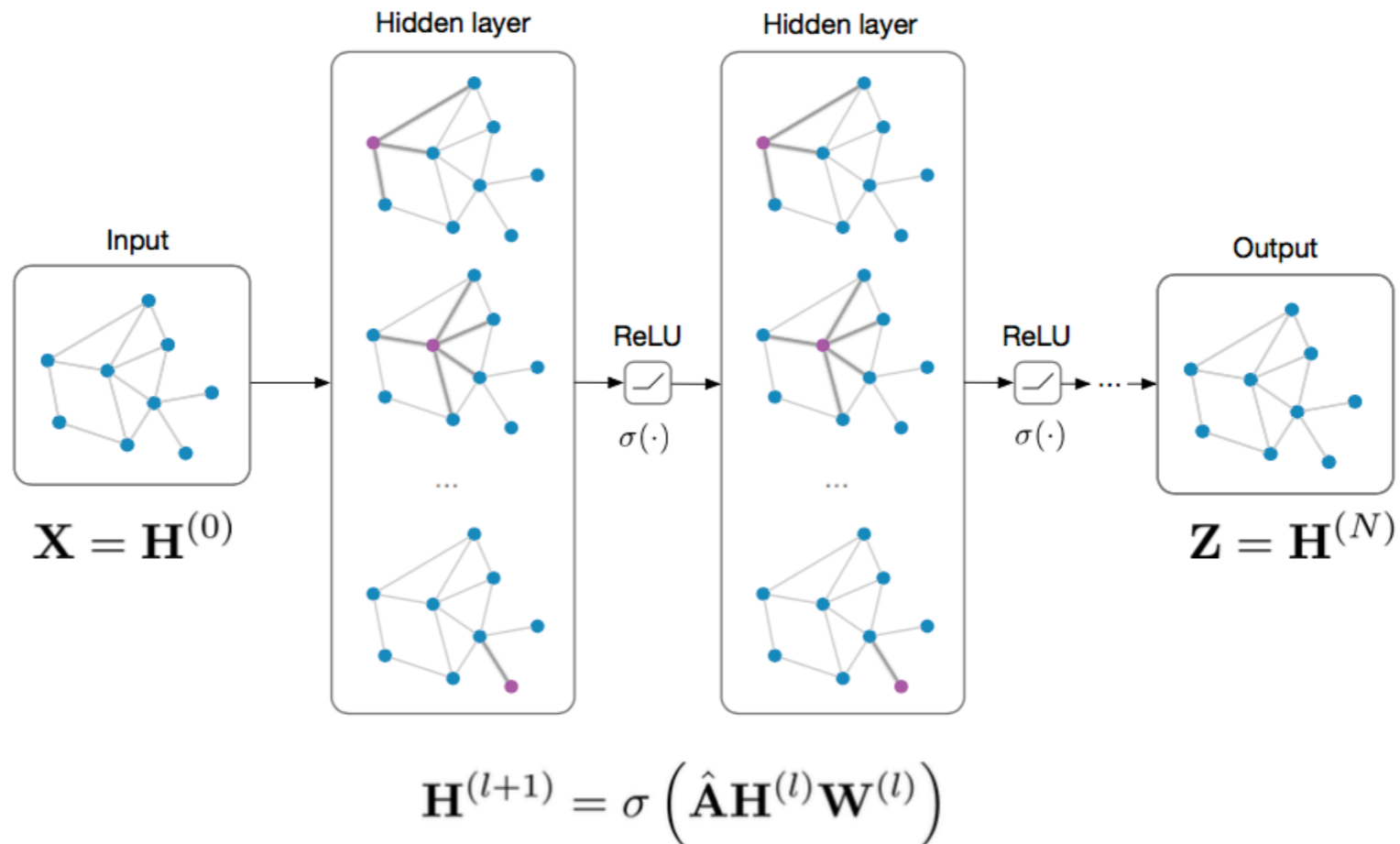
CNN on graphs: Node classification

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



CNN on graphs: Node classification

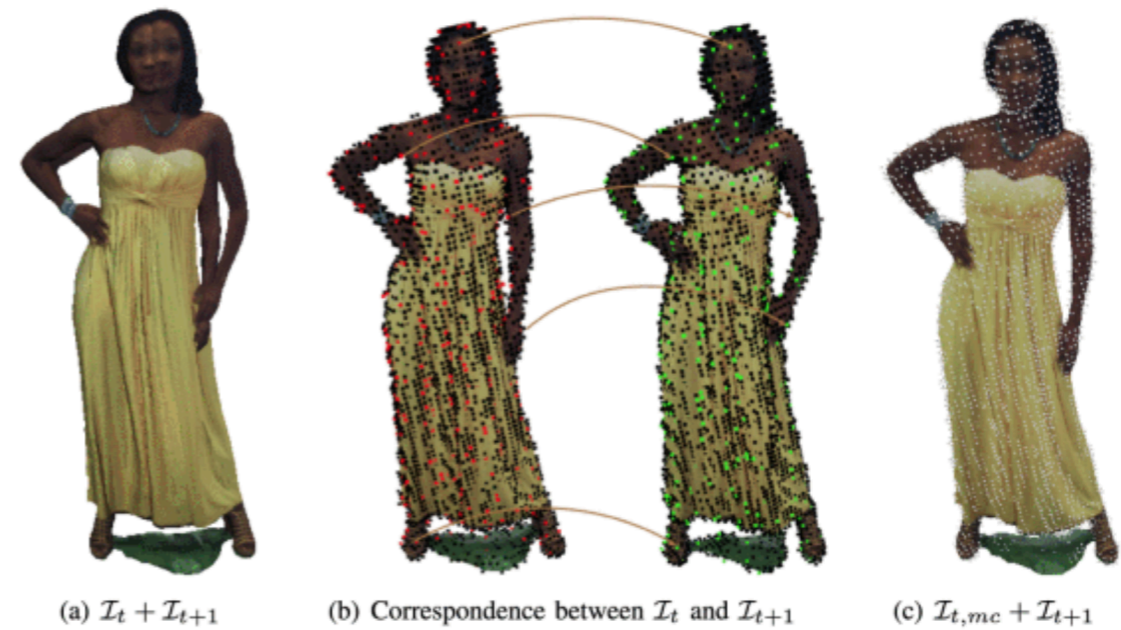
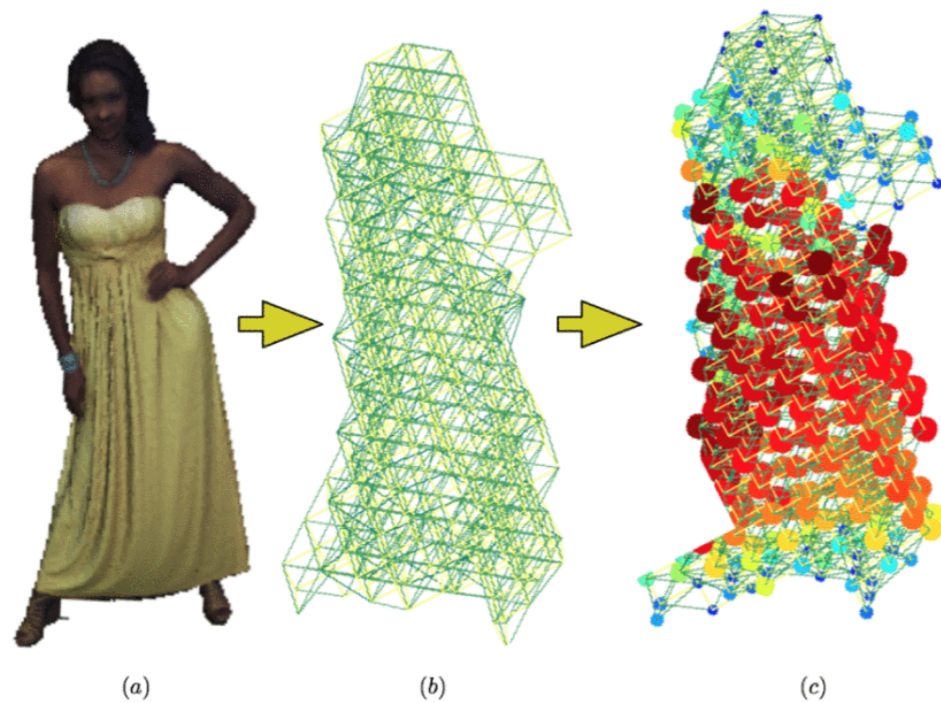
$$\hat{g}_{\theta^{(k+1)}}(L) \left(\text{ReLU}(\hat{g}_{\theta^{(k)}}(L)f) \right)$$



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

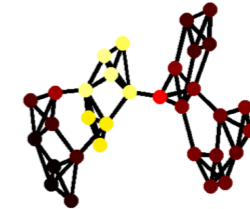
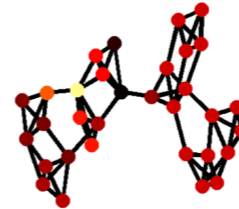


Application II: Community detection

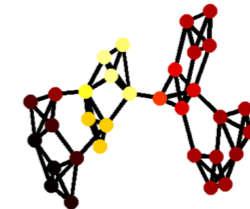
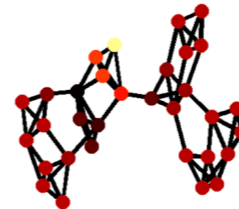
spectral graph wavelets
at different scales:

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

NODE
A:



NODE
B:



CORR.
COEF.:

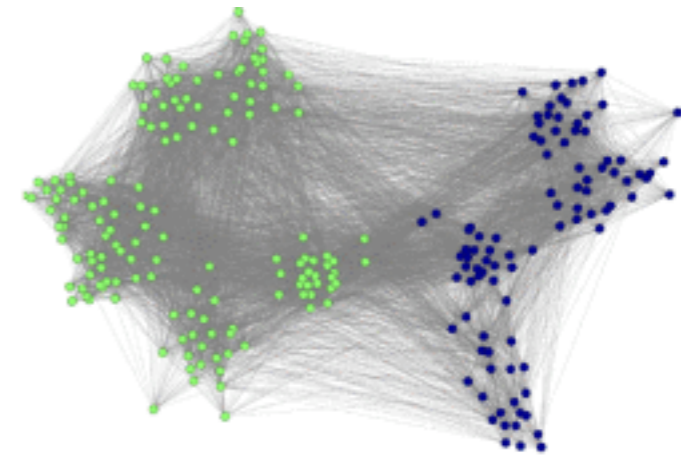
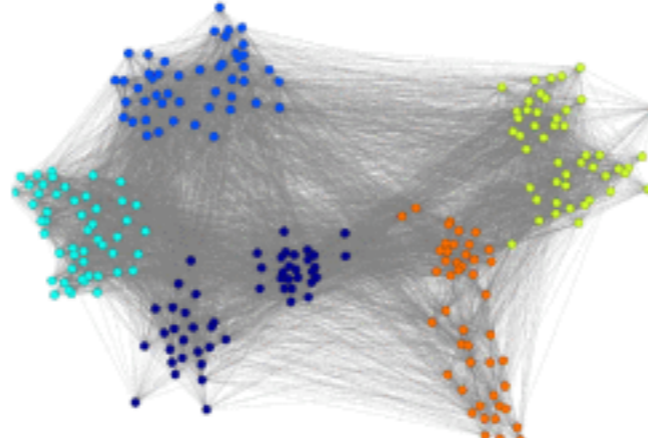
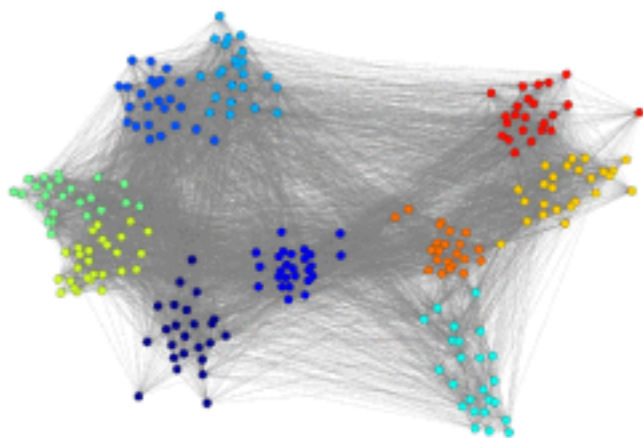
-0.50

0.97

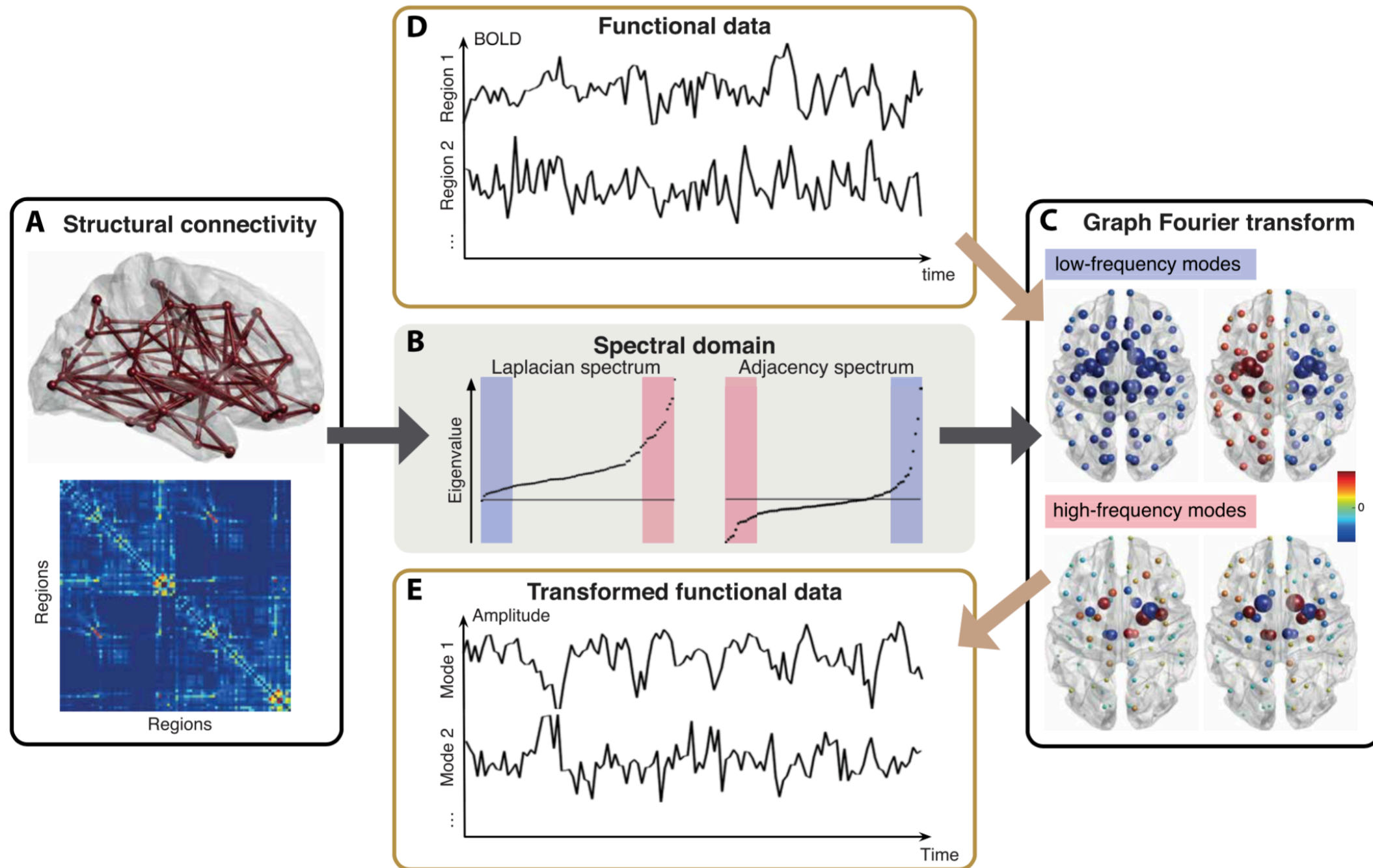
small scale

large scale

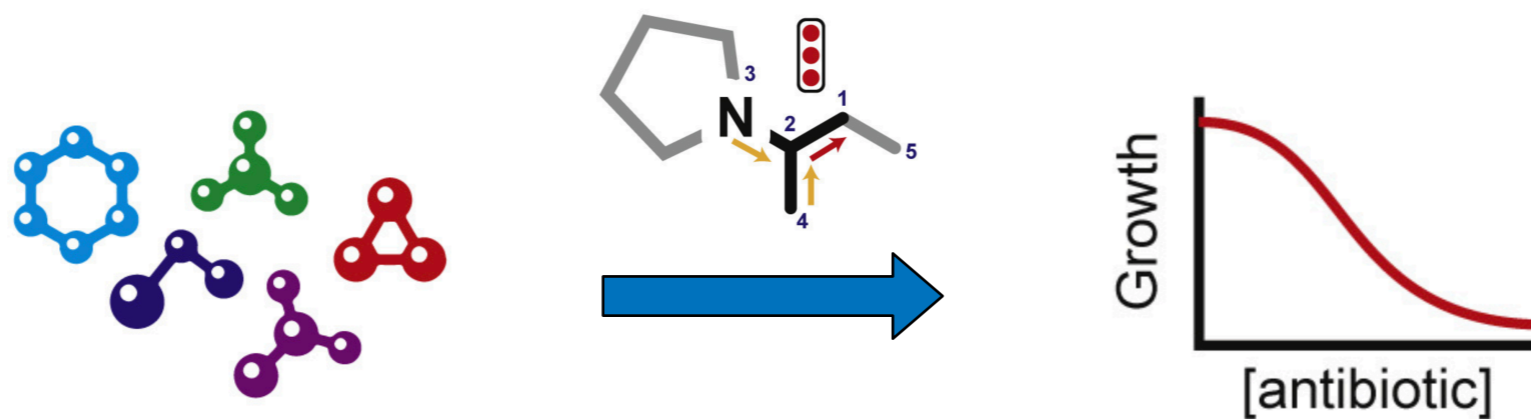
multi-scale community
detection:



Application III: Neuroscience



Application IV: Drug discovery

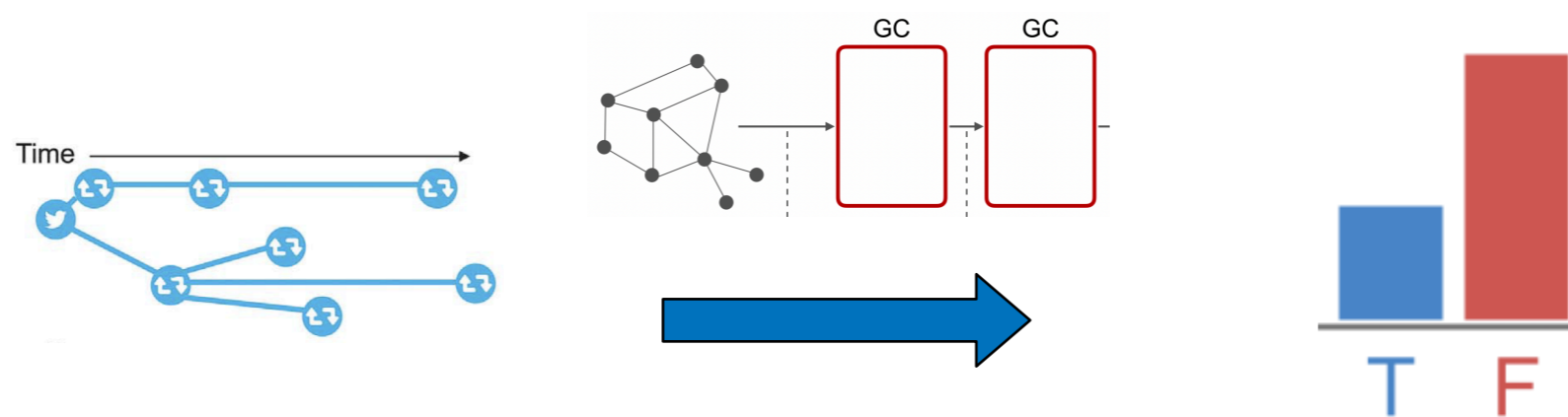


FINANCIAL TIMES
Artificial intelligence + Add to myFT
AI discovers antibiotics to treat drug-resistant diseases

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

Machine Learning for
Drug Discovery
Workshop
ICLR 2022

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

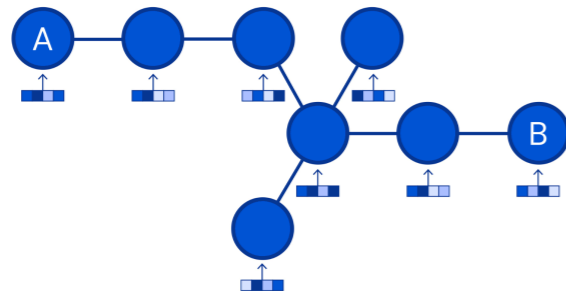
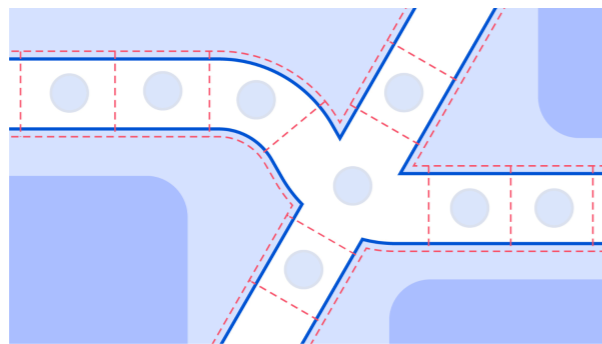
Home College and Campus Science Engineering Health

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

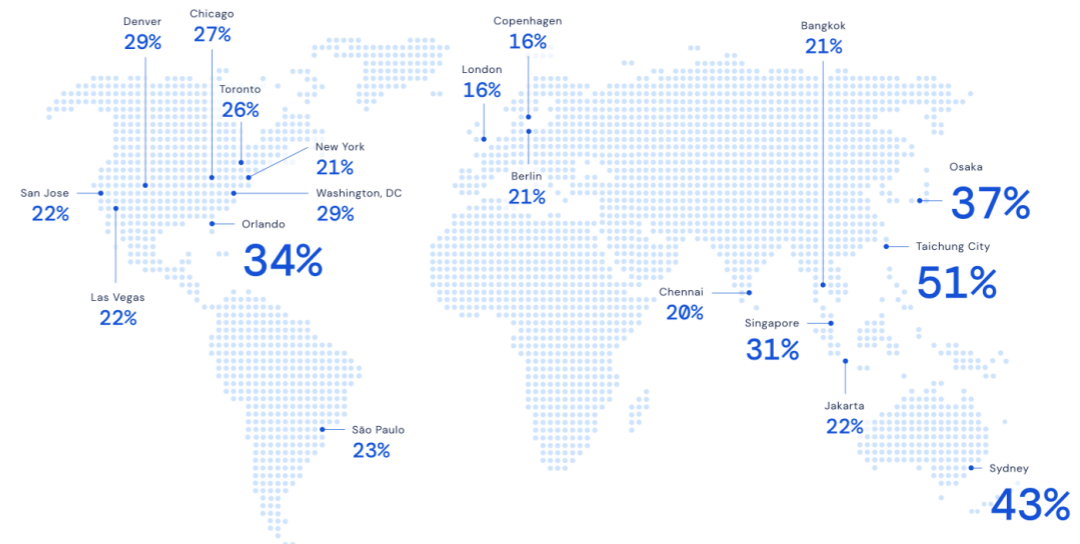
Vosoughi et al., "The spread of true and false news online," Science, 2018.

Monti et al., "Fake news detection on social media using geometric deep learning," ICLR Workshop, 2019.

Application VI: Traffic prediction



Google Maps ETA Improvements Around the World



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

In applications such as social, energy, transportation, sensor, and neuronal networks, high-dimensional data naturally reside on the vertices of weighted graphs. The emerging field of signal processing on graphs merges algebraic and spectral graph theoretic concepts with computational harmonic analysis to process such signals on graphs. In this tutorial overview, we outline the main challenges of the area, discuss different ways to define graph spectral domains, which are the analogs to the classical frequency domain, and highlight the importance of incorporating the irregular structures of graph data domains when processing signals on graphs. We then review methods to generalize fundamental operations such as filtering, translation, modulation, dilation, and downsampling to the graph setting and survey the localized, multiscale transforms that have

been proposed to efficiently extract information from high-dimensional data on graphs. We conclude with a brief discussion of open issues and possible extensions.

INTRODUCTION

Graphs are generic data representation forms that are useful for describing the geometric structures of data domains in numerous applications, including social, energy, transportation, sensor, and neuronal networks. The weight associated with each edge in the graph often represents the similarity between the two vertices it connects. The connectivities and edge weights are either dictated by the physics of the problem at hand or inferred from the data. For instance, the edge weight may be inversely proportional to the physical distance between nodes in the network. The data on these graphs can be visualized as a finite collection of samples, with one sample at each vertex in the graph. Collectively, we refer to these

1852-9876/16/0000-0000

IEEE SIGNAL PROCESSING MAGAZINE | MAY 2013

Geometric Deep Learning

Going beyond Euclidean data

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

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer networks. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DOF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

1852-9876/16/0000-0000

IEEE SIGNAL PROCESSING MAGAZINE | JULY 2017

1852-9876/17/0000-0000

Contents lists available at ScienceDirect

Applied and Computational Harmonic Analysis

ELSEVIER

www.elsevier.com/locate/acha

Vertex-frequency analysis on graphs

David I Shuman^{a,*}, Benjamin Ricaud^b, Pierre Vandergheynst^{b,1}

^a Department of Mathematics, Statistics, and Computer Science, Macalester College, Saint Paul, MN, USA

^b Signal Processing Laboratory (LTSI), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 28 July 2013
Received in revised form 28 December 2014
Accepted 22 February 2015
Available online 23 February 2015
Communicated by Patrick Flandrin

Keywords:
Signal processing on graphs
Time-frequency analysis
Generalized translation and modulation
Spectral graph theory
Localization
Clustering

ABSTRACT

One of the key challenges in the area of signal processing on graphs is to design dictionaries and transform methods to identify and exploit structure in signals on weighted graphs. To do so, we need to account for the intrinsic geometric structure of the underlying graph data domain. In this paper, we generalize one of the most important signal processing tools – windowed Fourier analysis – to the graph setting. Our approach is to first define generalized convolution, translation, and modulation operators for signals on graphs, and explore related properties such as the localization of translated and modulated graph kernels. We then use these operators to define a windowed graph Fourier transform, enabling vertex-frequency analysis. When we apply this transform to a signal with frequency components that vary along a path graph, the resulting spectrum matches our intuition from classical discrete-time signal processing. Yet, our construction is fully generalized and can be applied to analyze signals on any undirected, connected, weighted graph.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

In applications such as social networks, electricity networks, transportation networks, and sensor networks, data naturally reside on the vertices of weighted graphs. Moreover, weighted graphs are a flexible tool that can be used to describe similarities between data points in statistical learning problems, functional connectivities between different regions of the brain, and the geometric structures of countless other topologically-complex data domains.

* This work was supported by FET-Open grant number 255981 UNLOX.

¹ Part of the work reported here was presented at the IEEE Statistical Signal Processing Workshop, August 2012, Ann Arbor, MI.

Corresponding author.
E-mail addresses: dshuman@macalester.edu (D.I. Shuman), benjamin.ricaud@epfl.ch (B. Ricaud), pierre.vandergheynst@epfl.ch (P. Vandergheynst).

The authors would like to thank the anonymous reviewers for their constructive comments on this article.

http://dx.doi.org/10.1016/j.acha.2015.02.005
1862-5207/© 2015 Elsevier Inc. All rights reserved.

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 32, NO. 1, JANUARY 2021

A Comprehensive Survey on Graph Neural Networks

Zonghan Wu^a, Shirui Pan^a, Member, IEEE, Fengwen Chen, Guodong Long^a, Chengqi Zhang^a, Senior Member, IEEE, and Philip S. Yu, Life Fellow, IEEE

Abstract—Deep learning has revolutionized many machine learning tasks in recent years, ranging from image classification and video processing to speech recognition and natural language understanding. The data in these tasks are typically represented in the Euclidean space. However, there is an increasing number of applications, where data are generated from non-Euclidean domains and are represented as graphs with complex relationships and interdependency between objects. The complexity of graph data has imposed significant challenges on the existing machine learning algorithms. Recently, many studies on extending deep learning approaches for graph data have emerged. In this article, we provide a comprehensive overview of graph neural networks (GNNs) in data mining and machine learning fields. We propose a new taxonomy to divide the state-of-the-art GNNs into four categories, namely, recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial-temporal GNNs. We further discuss the applications of GNNs across various domains and summarize the open-source codes, benchmark data sets, and model evaluation of GNNs. Finally, we propose potential research directions in this rapidly growing field.

Index Terms—Deep learning, graph autoencoder (GAE), graph convolutional networks (GCNs), graph neural networks (GNNs), graph representation learning, network embedding.

I. INTRODUCTION

THE recent success of neural networks has boosted research on pattern recognition and data mining. Many machine learning tasks, such as object detection [1], [2], machine translation [3], [4], and speech recognition [5], which once heavily relied on handcrafted feature engineering to extract informative feature sets, have recently been revolutionized by various end-to-end deep learning paradigms, e.g., convolutional neural networks (CNNs) [6], recurrent neural networks (RNNs) [7], and autoencoders [8]. The success of deep learning in many domains is partially attributed to the rapidly developing computational resources (e.g., GPU), the availability of big training data, and the effectiveness of deep learning to extract latent representations from the Euclidean data (e.g., images, text, and videos). Taking image data as an example, we can represent an image as a regular grid in the Euclidean space. CNN is able to exploit the shift-invariance, local connectivity, and compositionality of image data [9]. As a result, CNN can extract local meaningful features that are shared with the entire data sets for various image analyses.

While deep learning effectively captures hidden patterns of Euclidean data, there are an increasing number of applications, where data are represented in the form of graphs. For example, in e-commerce, a graph-based learning system can identify the interactions between users and products to make highly accurate recommendations. In chemistry, molecules are modeled as graphs, and their bioactivity needs to be identified for drug discovery. In a citation network, articles are linked to each other via citations, and they need to be categorized into different groups. The complexity of graph data has imposed significant challenges on the existing machine learning algorithms. As graphs can be irregular, a graph may have a variable size of unordered nodes, and nodes from a graph may have a different number of neighbors, resulting in some important operations (e.g., convolutions) being easy to compute in the image domain but difficult to apply to the graph domain. Furthermore, a core assumption of existing machine learning algorithms is that instances are independent of each other. This assumption no longer holds for graph data because each instance (node) is related to others by links of various types, such as citations, friendships, and interactions.

Recently, there is increasing interest in extending deep learning approaches for graph data. Motivated by CNNs, RNNs, and autoencoders from deep learning, new generalizations and definitions of important operations have been rapidly developed over the past few years to handle the complexity of graph data. For example, a graph convolution can be generalized from a 2-D convolution. As illustrated in Fig. 1, an image can be considered as a special case of graphs, where pixels are connected by adjacent pixels. Similar to 2-D convolution, one may perform graph convolutions by taking the weighted average of a node's neighborhood information.

There are a limited number of existing reviews on the topic of graph neural networks (GNNs). Using the term geometric deep learning, Bronstein et al. [9] give an overview

2162-237X © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications_standards_info.html for more information.

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^a, Fellow IEEE, PASCAL FROSSARD^a, Fellow IEEE, JELENA KOVAČEVIĆ^b, Fellow IEEE, JOSÉ M. F. MOURA^c, Fellow IEEE, and PIERRE VANDERGHEYNST

ABSTRACT | Research in graph signal processing (GSP) aims to develop tools for processing data defined on irregular graph domains. In this paper, we first provide an overview of core ideas in GSP and their connection to conventional digital signal processing, along with a brief historical perspective to highlight how concepts recently developed in GSP build on top of prior research in other areas. We then summarize recent advances in developing basic GSP tools, including methods for sampling, filtering, or graph learning. Next, we review progress in several application areas using GSP, including processing and analysis of sensor network data, biological data, and applications to image processing and machine learning.

KEYWORDS | Graph signal processing (GSP), network science and graphs, sampling, signal processing

I. INTRODUCTION AND MOTIVATION

Data is all around us, and massive amounts of it. Almost every aspect of human life is now being recorded at all levels: from the marking and recording of processing inside the cells starting with the advent of Bioassess markets, to our personal data through health monitoring devices and apps, financial and banking data, our social networks, mobility and traffic patterns, marketing preferences, fads, and many more. The complexity of such networks [1] and interactions means that the data now reside on irregular and complex structures that do not lend themselves to standard tools.

Research received November 20, 2020; revised March 20, 2020; accepted March 23, 2020. Date of current version April 26, 2020. Corresponding author: Antonio Ortega (a.ortega@univ-lorraine.fr).
A. Ortega and P. Vandergheynst are with EPFL, Lausanne, Switzerland 1015.
A. Ortega, and J. M. F. Moura are with Carnegie Mellon University, Pittsburgh, PA 15213.
Digital Object Identifier: 10.1109/SPAC.2020.2920204

© 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications_standards_info.html for more information.

808 PROCEEDINGS OF THE IEEE | Vol. 108, No. 5, May 2020

Graph Signal Processing for Machine Learning

A review and new perspectives

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

Graph Signal Processing for Machine Learning

A review and new perspectives

The effective representation, processing, analysis, and visualization of large-scale structured data, especially those related to complex domains, such as networks and graphs, are one of the key questions in modern machine learning. Graph signal processing (GSP), a vibrant branch of signal processing models and algorithms that aims at handling data supported on graphs, opens new paths of research to address this challenge. In this article, we review a few important contributions made by GSP concepts and tools, such as graph filters and transforms, to the development of novel machine learning algorithms. In particular, our discussion focuses on the following three aspects: exploiting data structure and relational priors, improving data and computational efficiency, and enhancing model interpretability. Furthermore, we provide new perspectives on the future development of GSP techniques that may serve as a bridge between applied mathematics and signal processing on one side and machine learning and network science on the other. Cross-fertilization across these different disciplines may help unlock the numerous challenges of complex data analysis in the modern age.

Introduction

We live in a connected society. Data collected from large-scale interactive systems, such as biological, social, and financial networks, become largely available. In parallel, the past few decades have seen a significant amount of interest in the machine learning community for network data processing and analysis. Networks have an intrinsic structure that conveys very specific properties to data, e.g., interdependencies between data entities in the form of pairwise relationships. These properties are traditionally captured by mathematical representations such as graphs.

In this context, new trends and challenges have been developing fast. Let us consider, for example, a network of protein-protein interactions and the expression level of individual genes at every point in time. Some typical tasks in network biology relating to this type of data are 1) discovery of key genes (via protein grouping) affected by the infection and 2) prediction of how the host organism reacts (in terms of gene expression)

1852-9876/16/0000-0000

IEEE SIGNAL PROCESSING MAGAZINE | November 2020

117