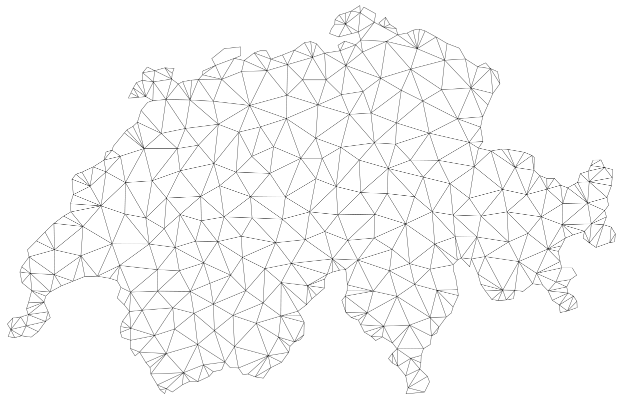
Recent Advances in Learning with Graphs

Xiaowen Dong
Department of Engineering Science
University of Oxford

Clarkson University, March 2021

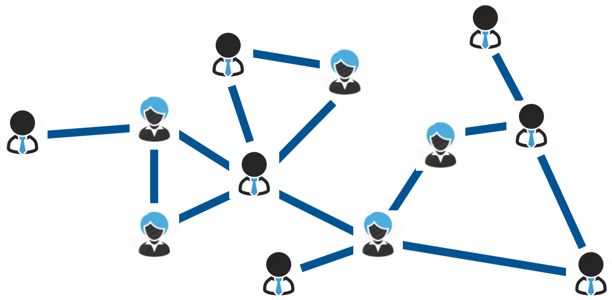
UNIVERSITY OF OXFORD

Networks are pervasive



geographical network

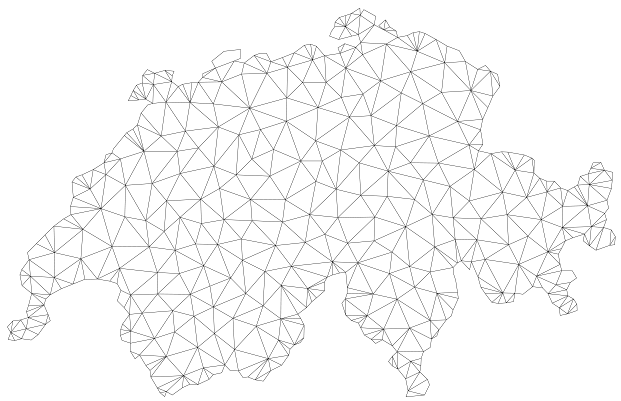
traffic network



social network

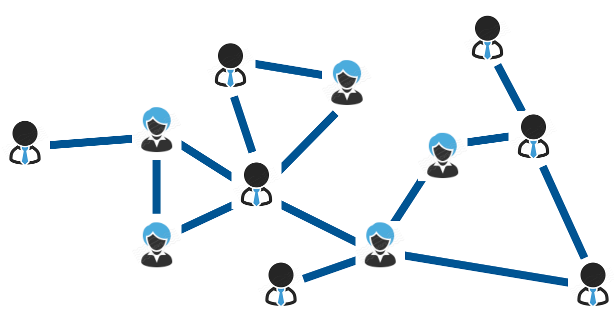
brain network

Networks are pervasive



geographical network

traffic network



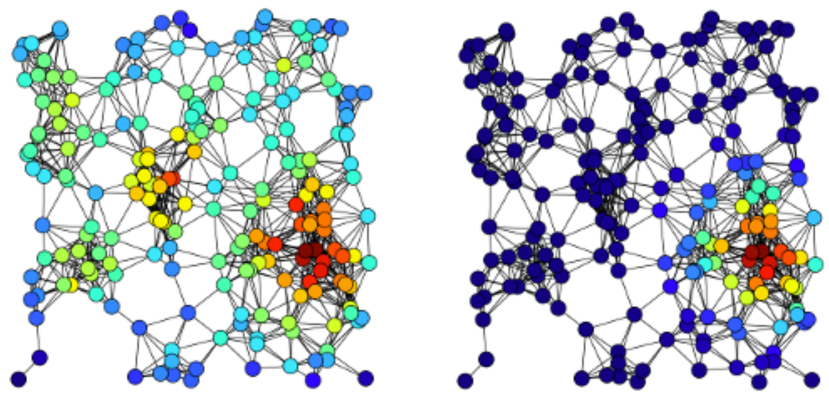
social network

brain network

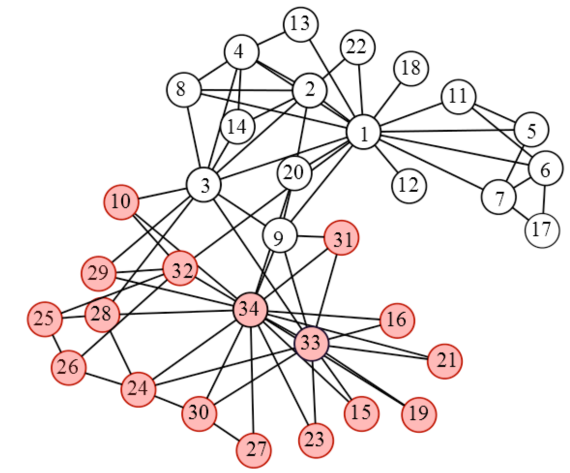
graphs provide mathematical representation of networks

The field of network science

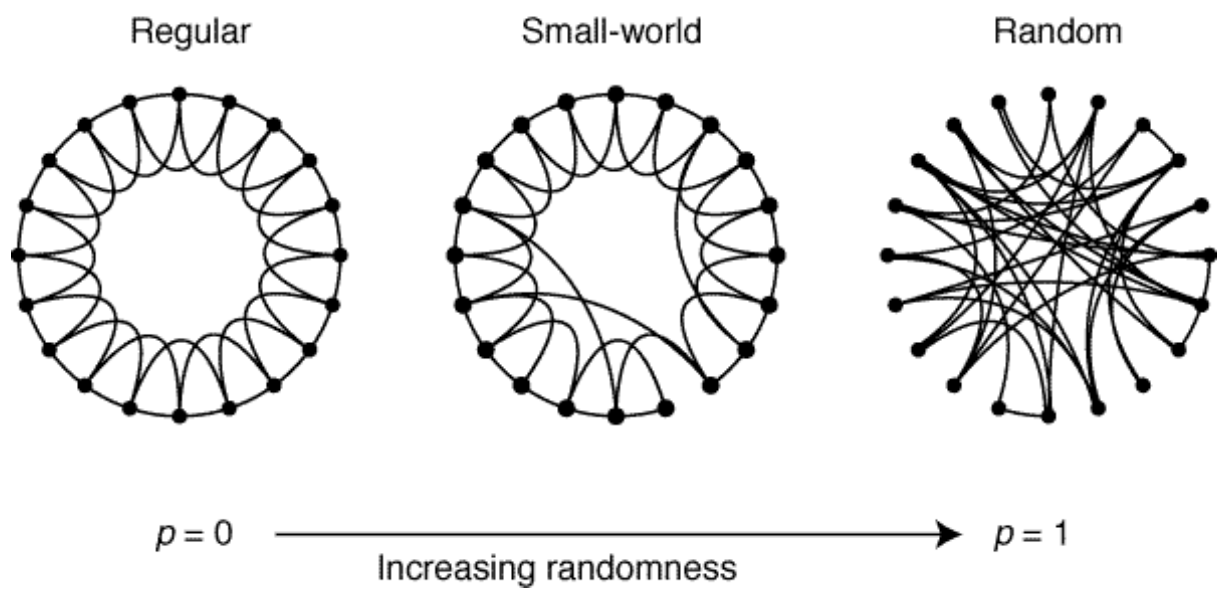
network centrality



community detection

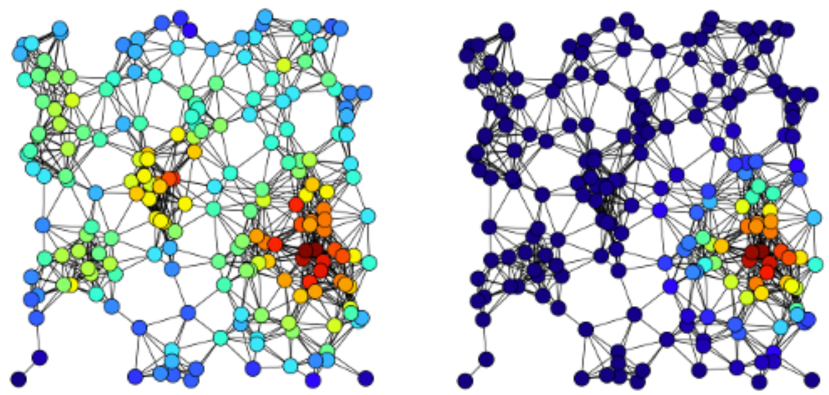


random graph models

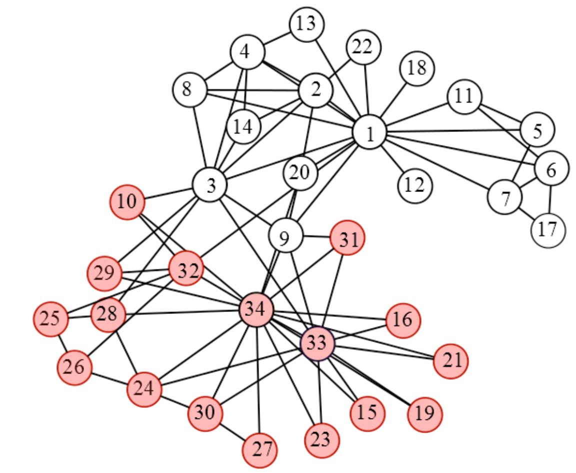


The field of network science

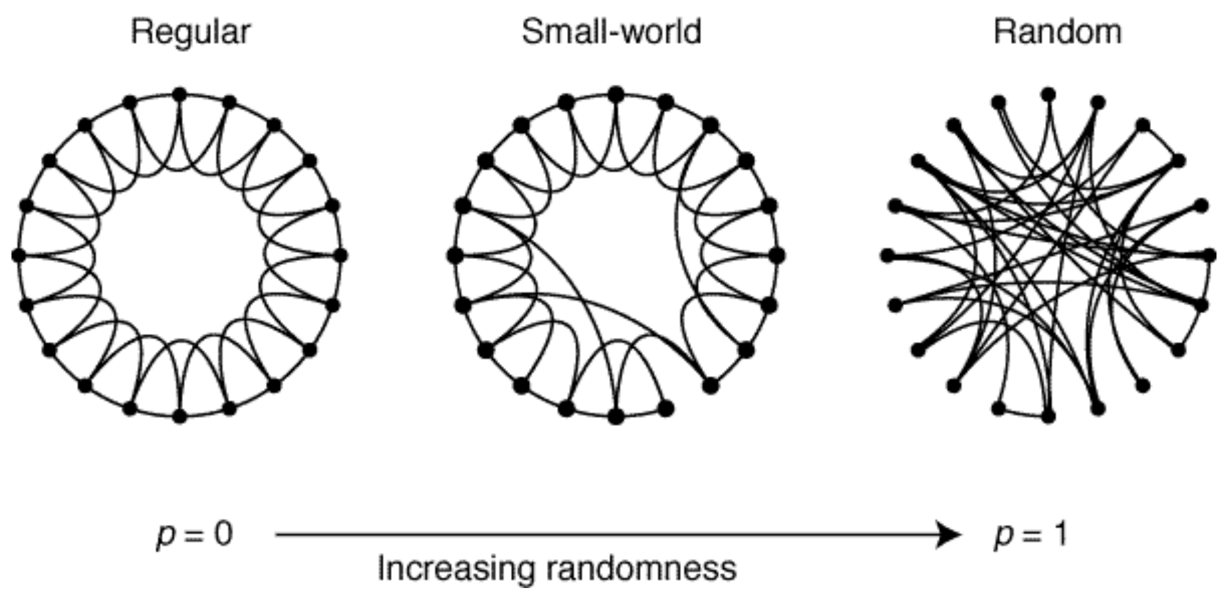
network centrality



community detection



random graph models

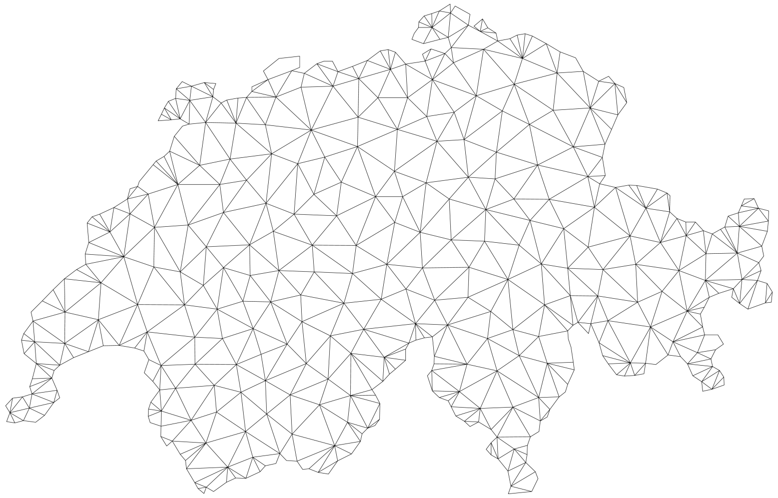


from edge attributes to node attributes

from graphs to graph-structured data

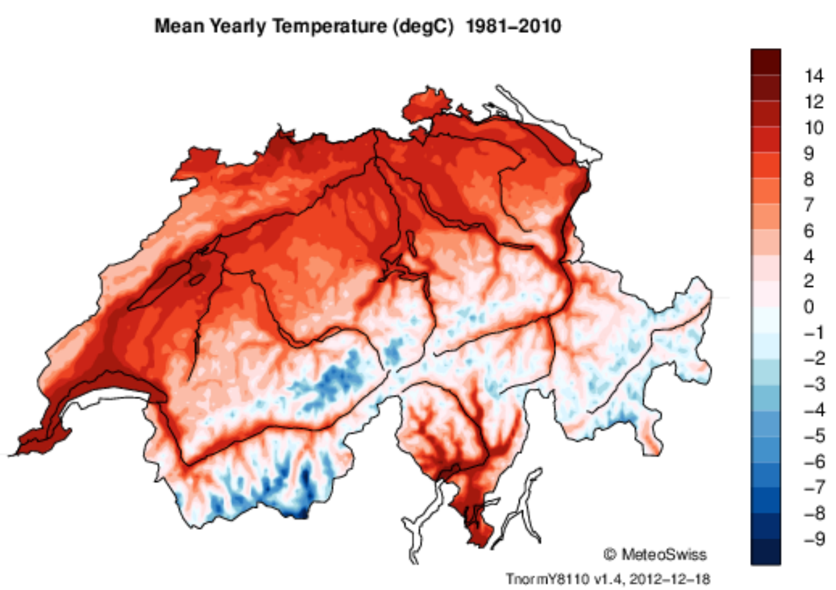
Newman, "An introduction," Oxford University Press, 2010.

Graph-structured data are pervasive



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

Graph-structured data are pervasive

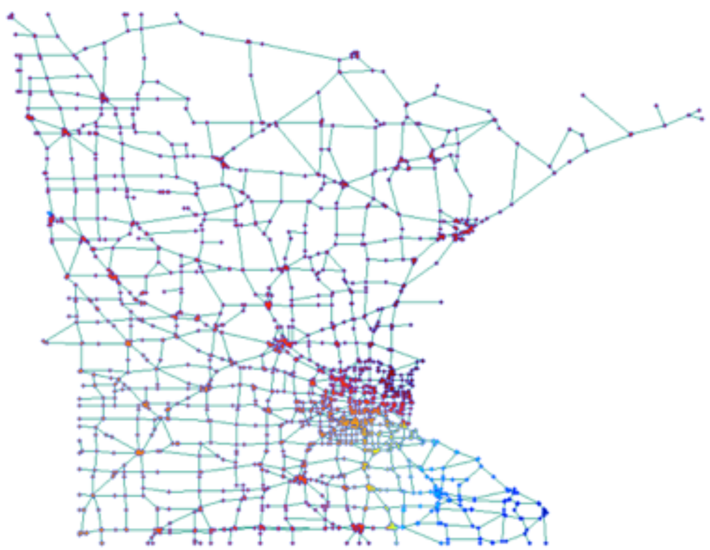


- vertices
  - geographical regions
- edges
  - geographical proximity between regions
- signal
  - temperature records in these regions

Graph-structured data are pervasive

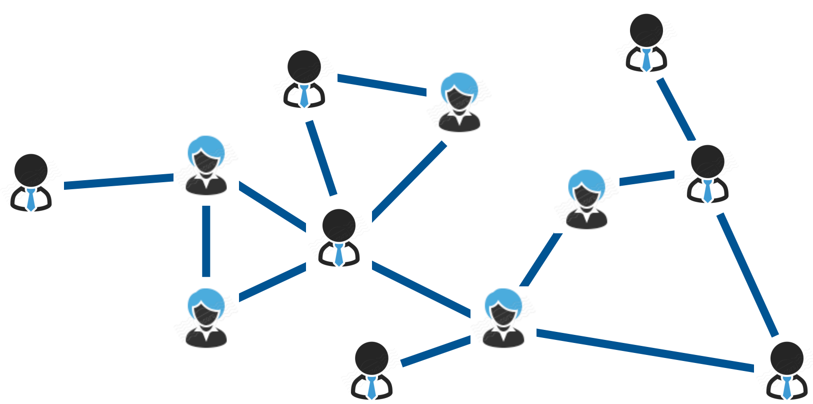
- nodes
  - road junctions
- edges
  - road connections

Graph-structured data are pervasive



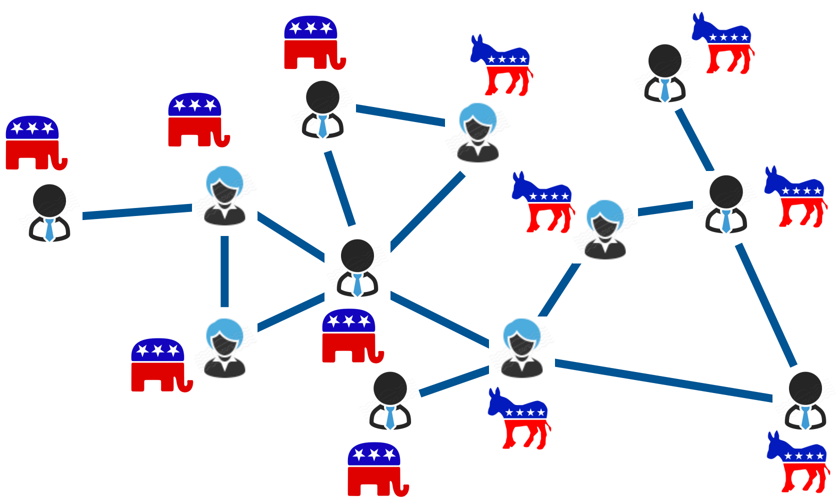
- nodes
  - road junctions
- edges
  - road connections
- signal
  - traffic congestion at junctions

Graph-structured data are pervasive



- nodes
  - individuals
- edges
  - friendship between individuals

Graph-structured data are pervasive

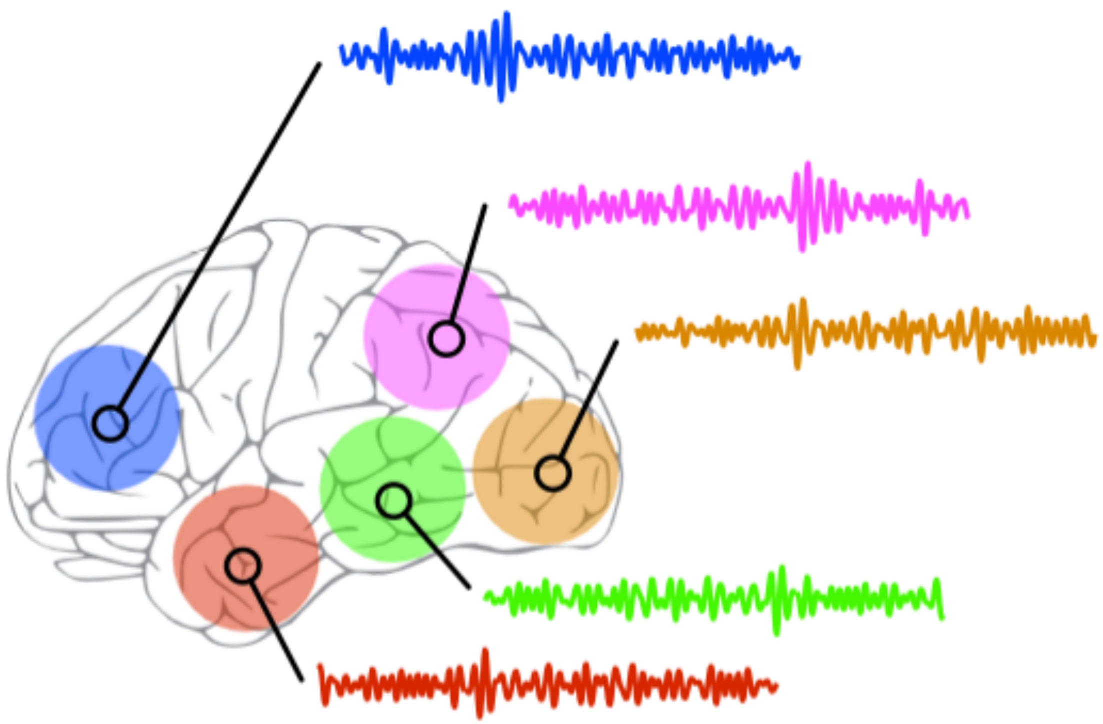


- nodes
  - individuals
- edges
  - friendship between individuals
- signal
  - political view

Graph-structured data are pervasive

- nodes
  - brain regions
- edges
  - structural connectivity between brain regions

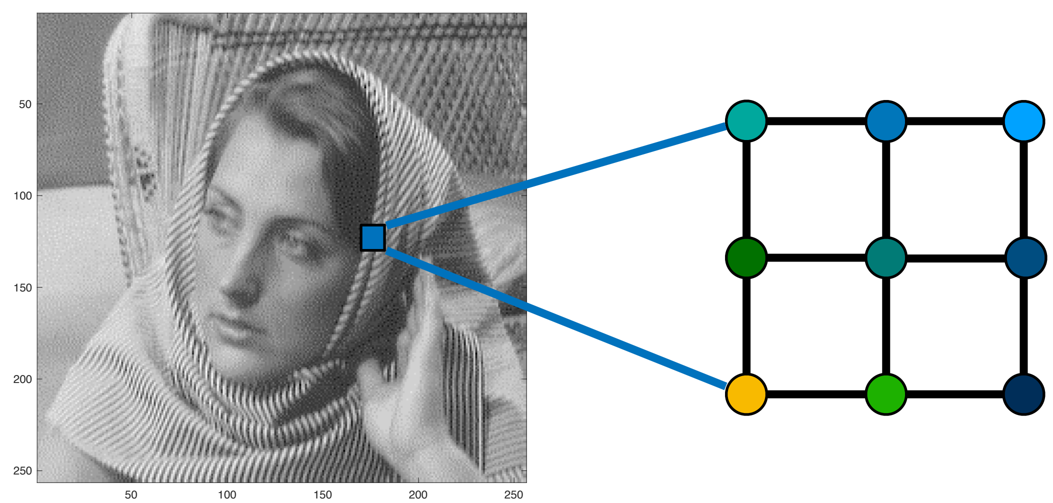
Graph-structured data are pervasive



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

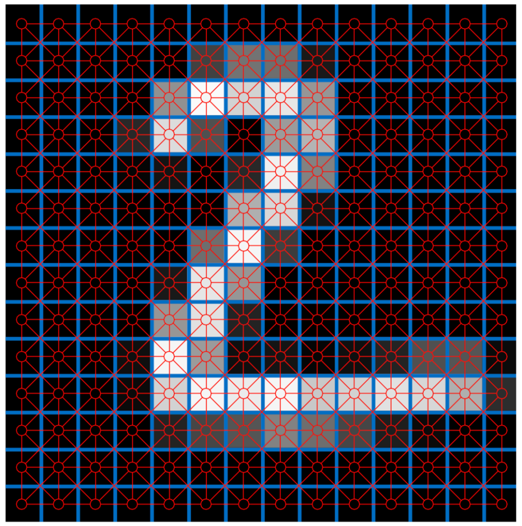
Richiardi et al., "Machine learning with brain graphs," IEEE Signal Processing Magazine, 2013.

Graph-structured data are pervasive



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

Learning with graph-structured data

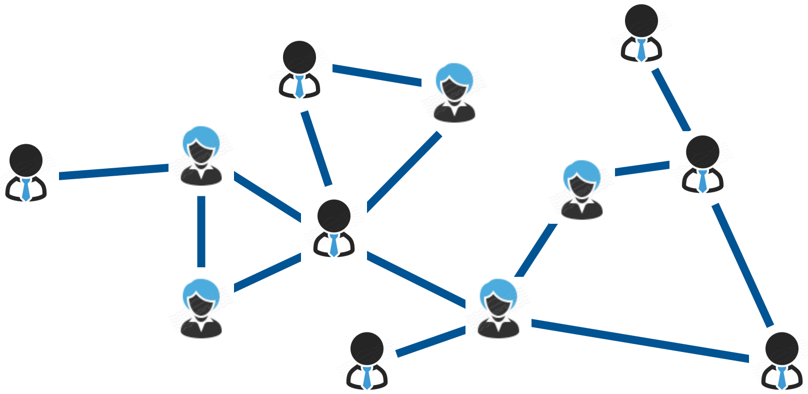


is it a 2?
is it a 4?

condition?
no condition?

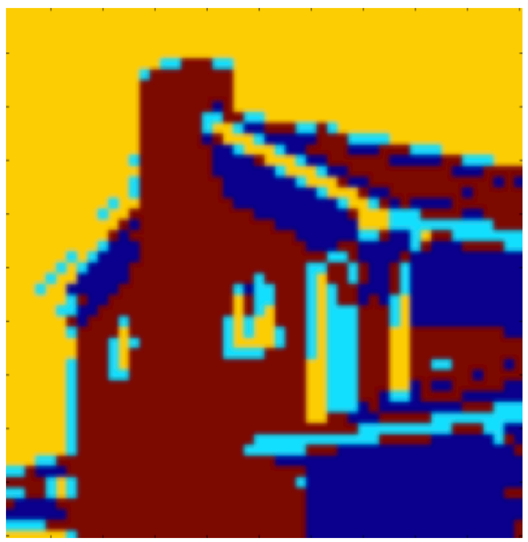
(supervised) graph-wise classification

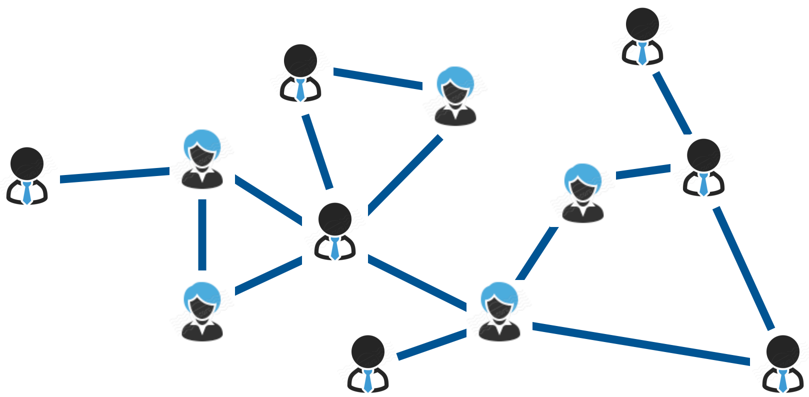
Learning with graph-structured data



(semi-supervised) node-wise classification

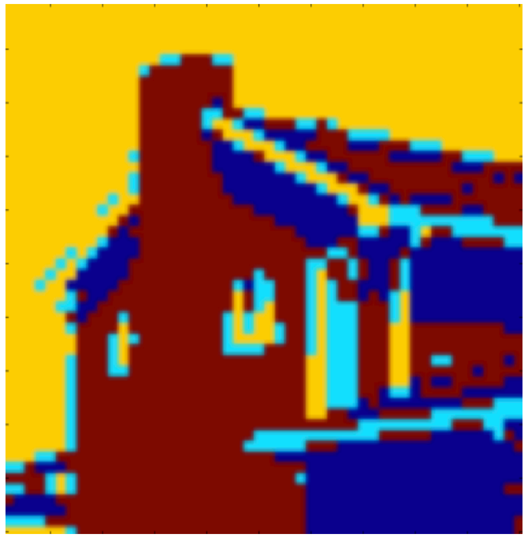
Learning with graph-structured data

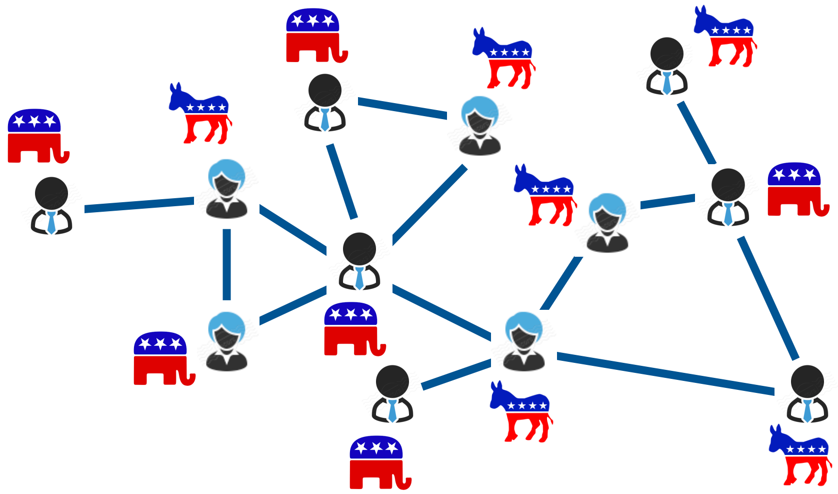




(semi-supervised) node-wise classification

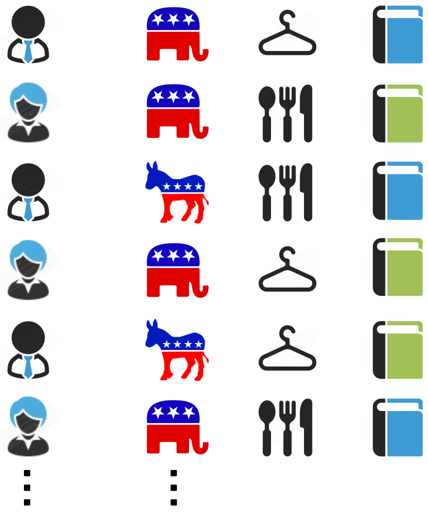
Learning with graph-structured data





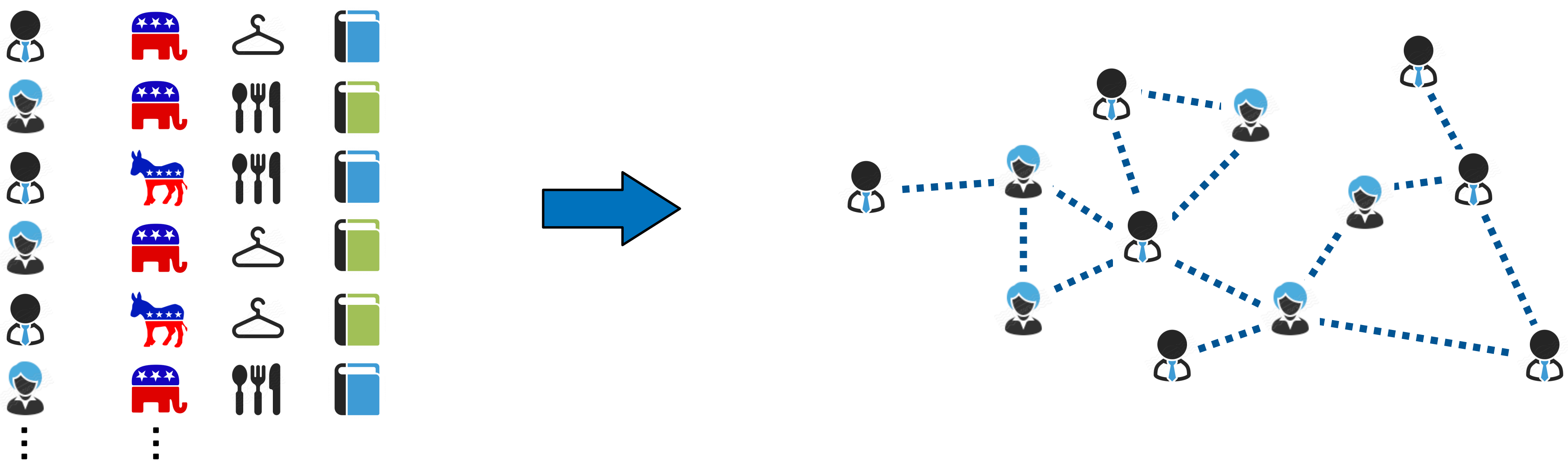
(semi-supervised) node-wise classification

Learning with graph-structured data



learning graph structure from data

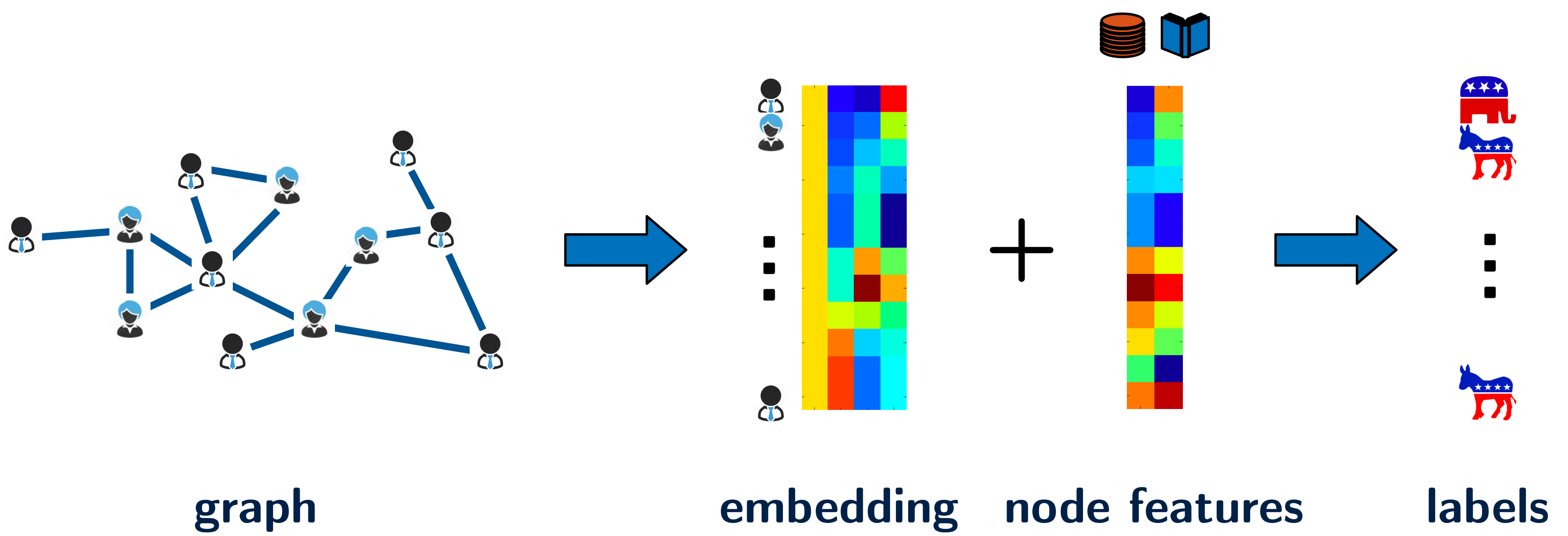
Learning with graph-structured data



learning graph structure from data

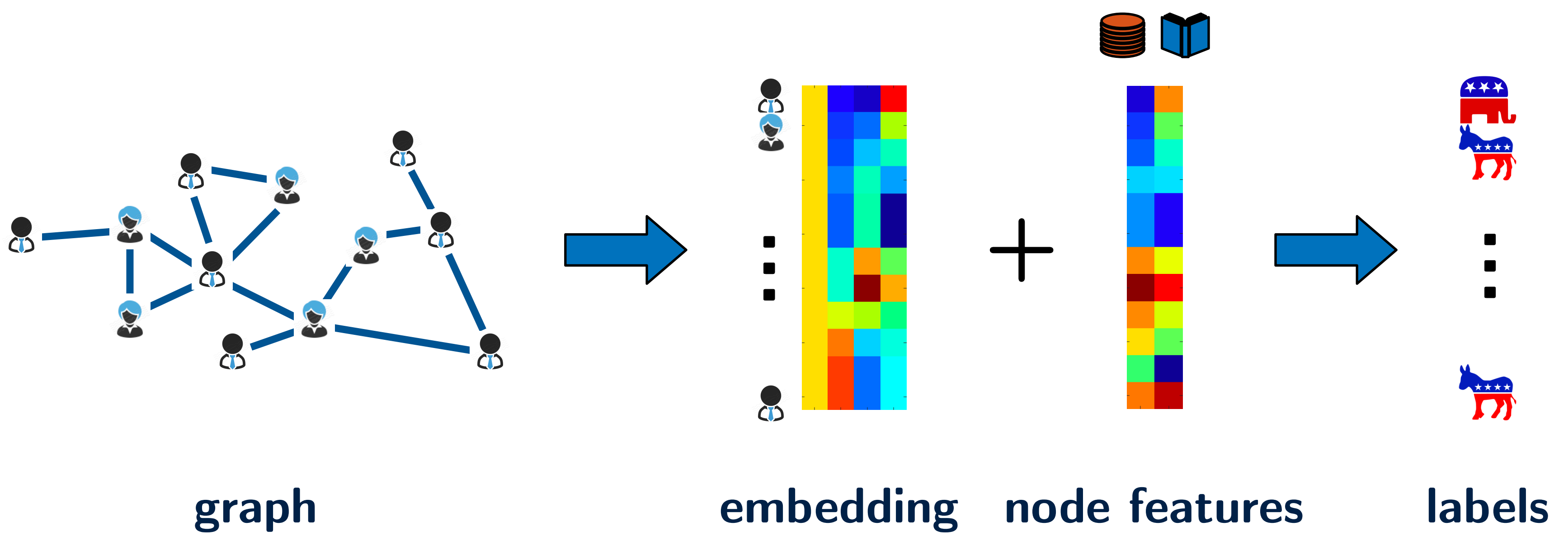
How to incorporate graph into learning?

- Embedding of graph structure leads to information loss



How to incorporate graph into learning?

- Embedding of graph structure leads to information loss



- Need for new models that directly incorporate structure in data analysis
  - graph signal processing (GSP)
  - graph neural networks (GNN)

Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Graph neural networks (GNNs)
- Applications

Outline

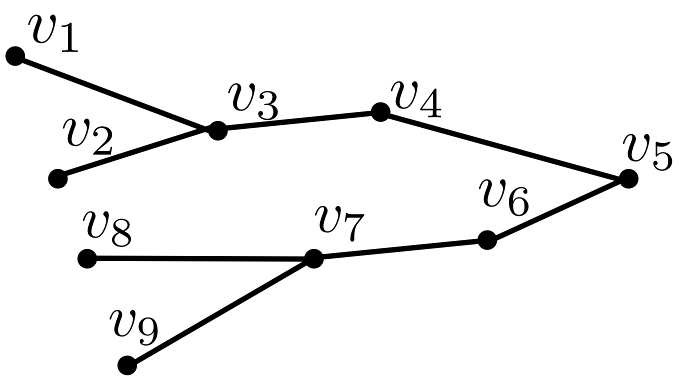
- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tools of GSP
- Graph neural networks (GNNs)
- Applications

Graph signal processing (GSP)

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

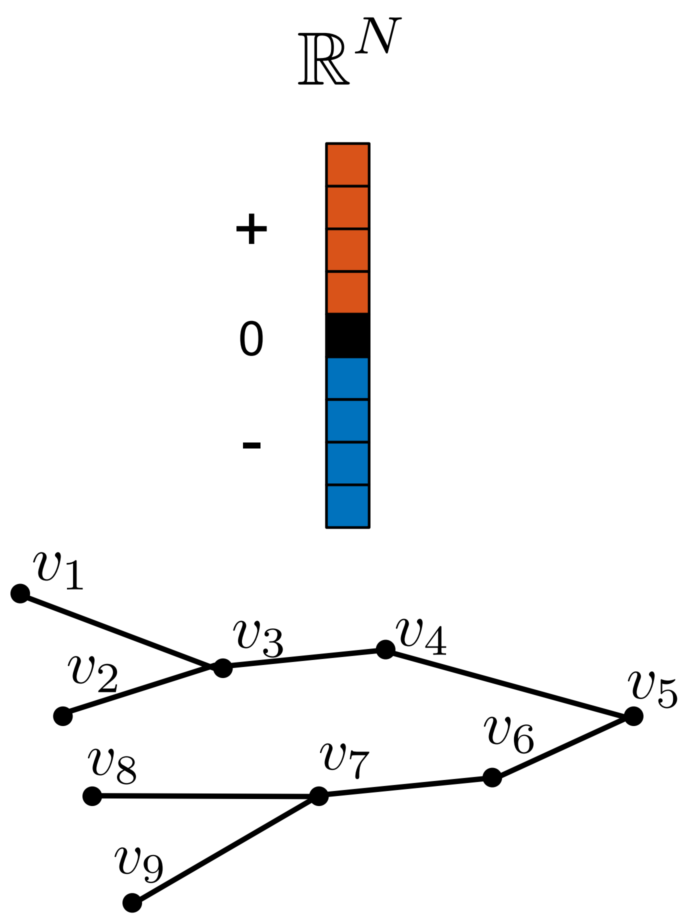
Graph signal processing (GSP)

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



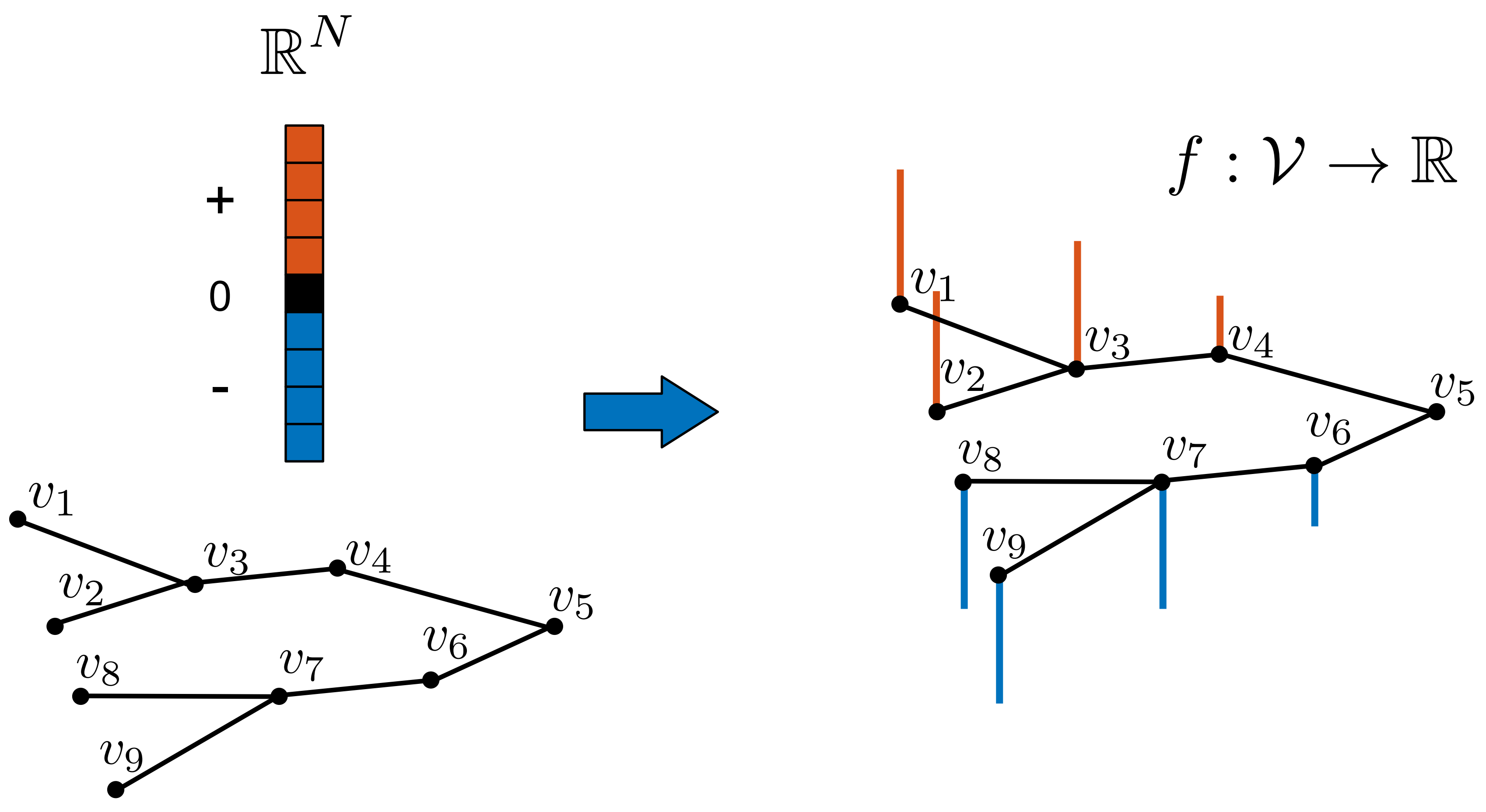
Graph signal processing (GSP)

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



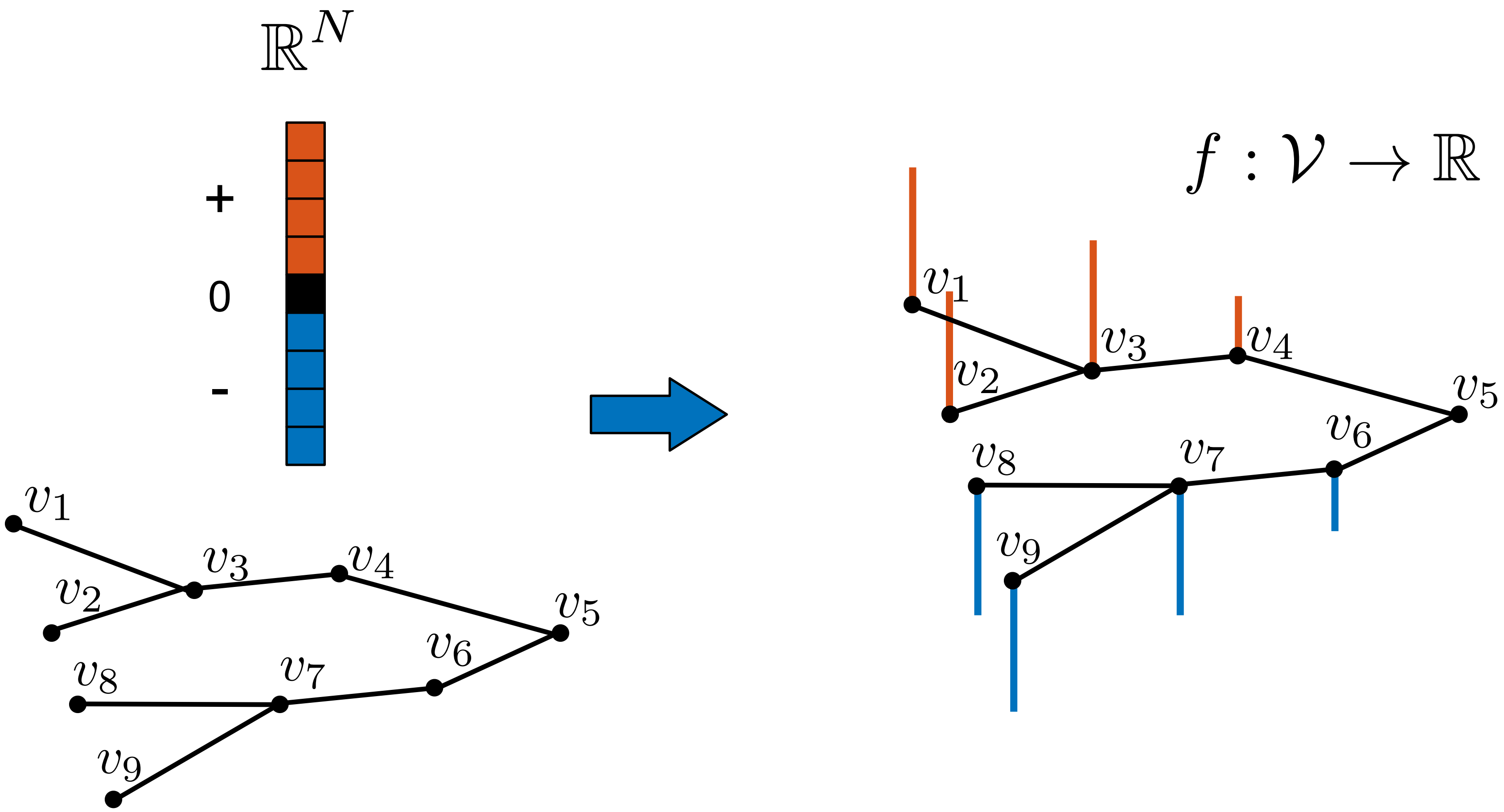
Graph signal processing (GSP)

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



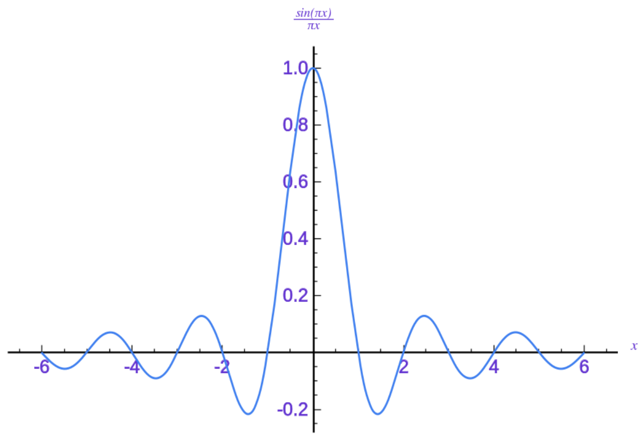
Graph signal processing (GSP)

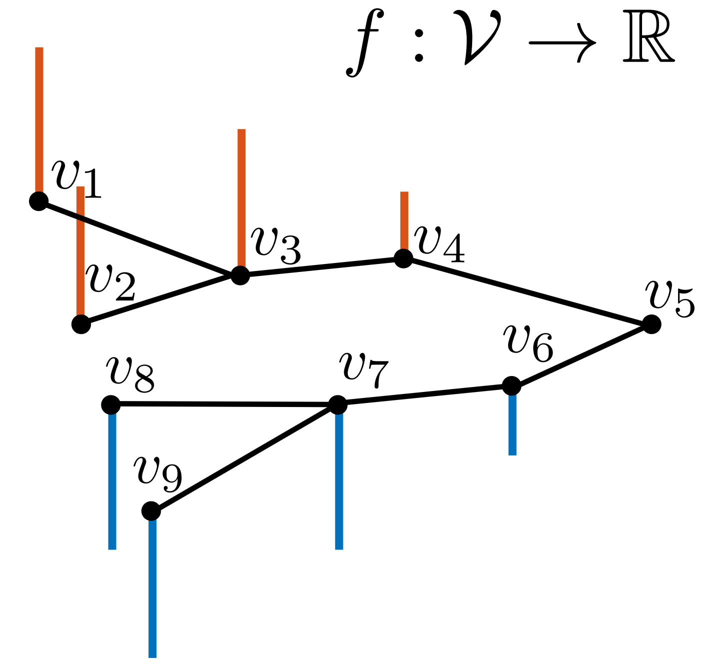
- Graph-structured data can be represented by signals defined on graphs or **graph signals**



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

Graph signal processing (GSP)





how to generalise classical signal processing tools on irregular domains such as graphs?

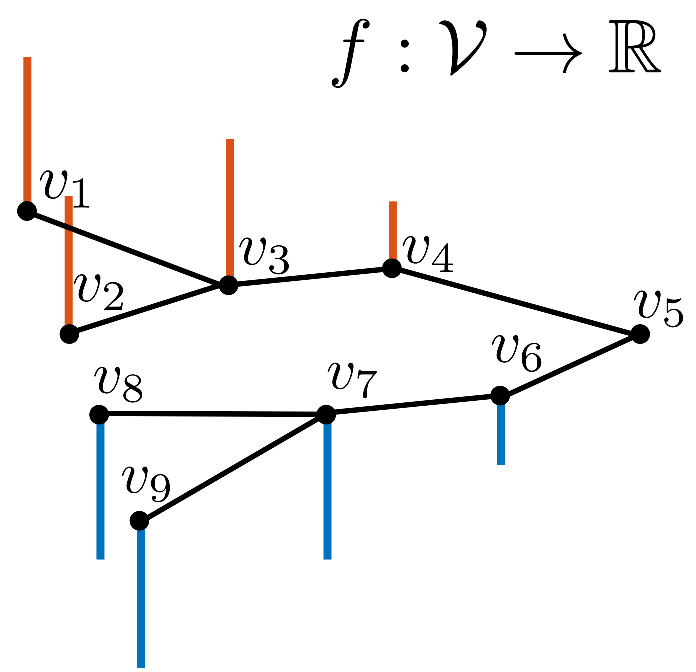
Graph signal processing (GSP)

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

Graph signal processing (GSP)

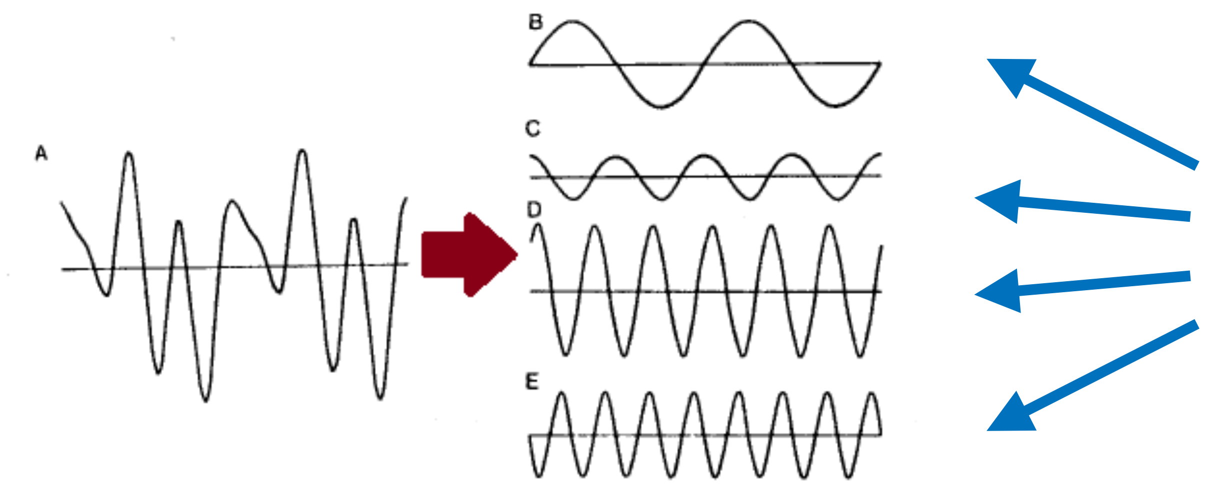
- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

important for analysis of signal properties



Graph signal processing (GSP)

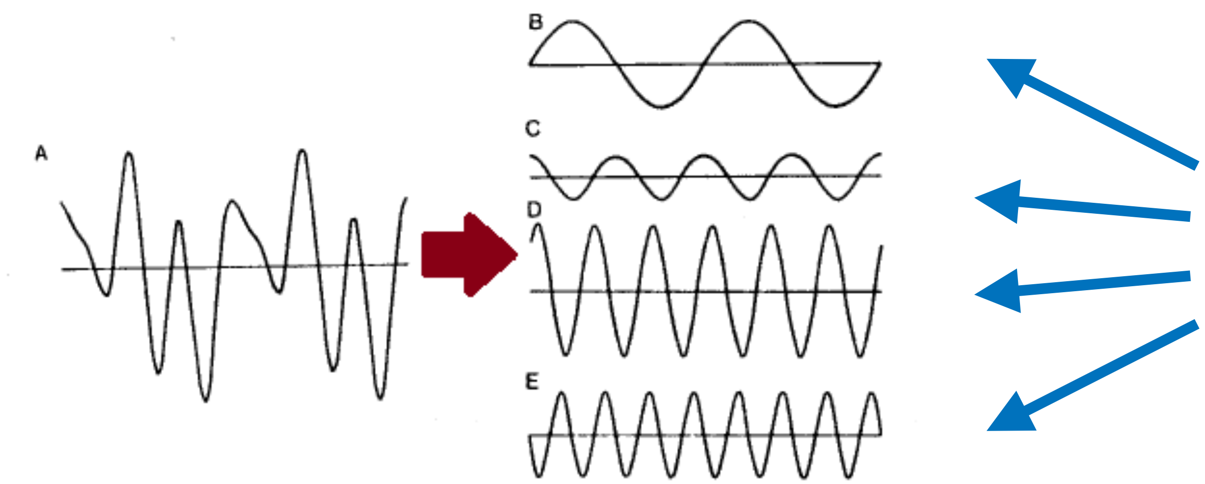
- Classical Fourier transform provides frequency domain representation of signals



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

Graph signal processing (GSP)

- Classical Fourier transform provides frequency domain representation of signals

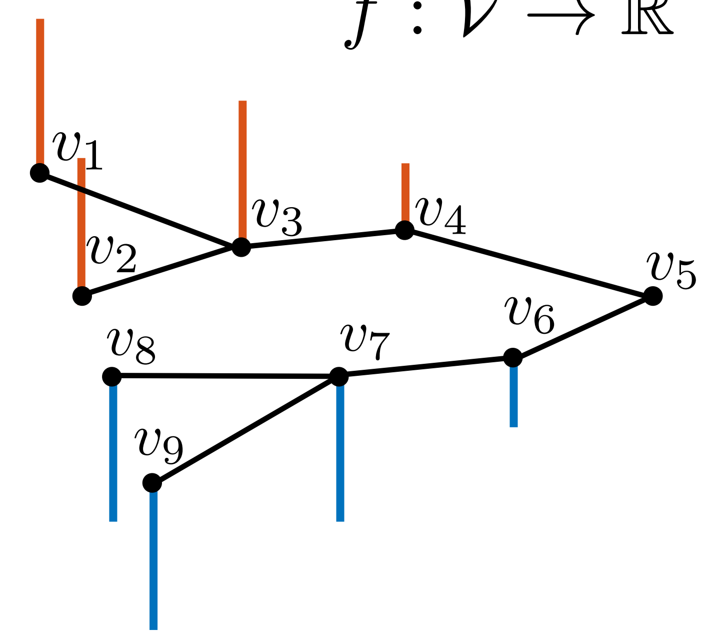


- "building blocks" of signal
- different frequency (oscillation)

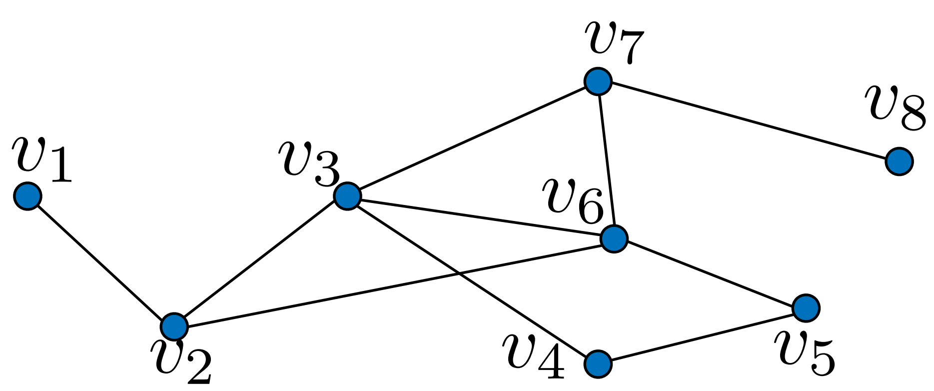
- What about a notion of frequency for graph signals?

  we need the graph Laplacian matrix

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



Graph Laplacian



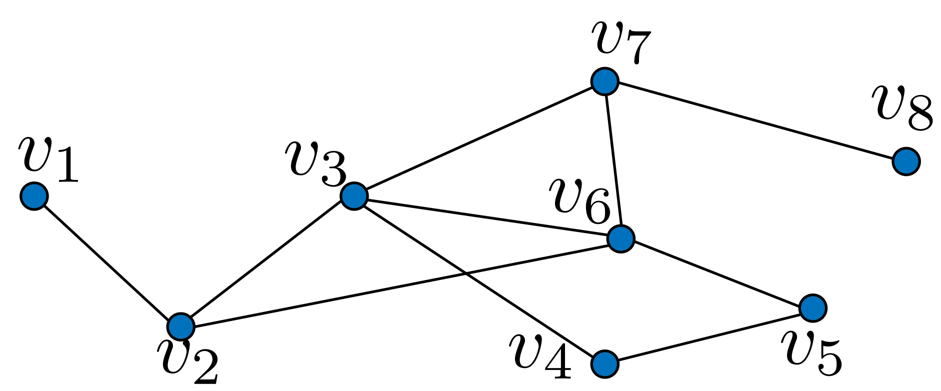
weighted and undirected graph:

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

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

$$W$$

Graph Laplacian



weighted and undirected graph:

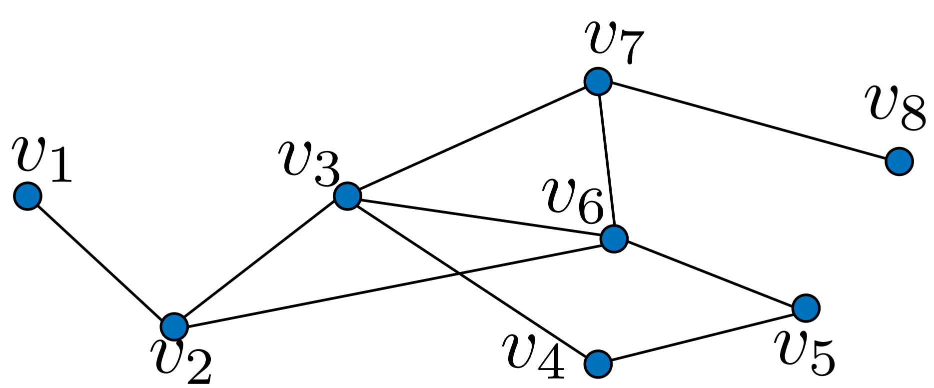
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

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

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

D $\qquad$ W

Graph Laplacian



weighted and undirected graph:

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

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

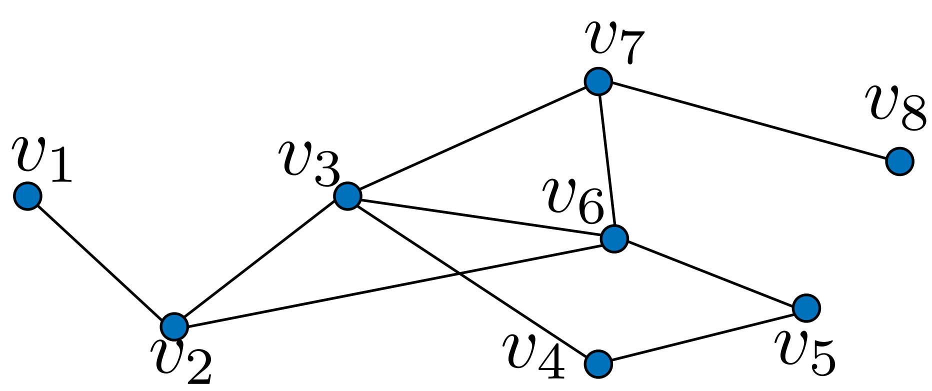
$$L = D - W$$ **equivalent to G!**

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

$$D \qquad\qquad W \qquad\qquad L$$

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

Graph Laplacian



weighted and undirected graph:

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

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

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

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

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

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

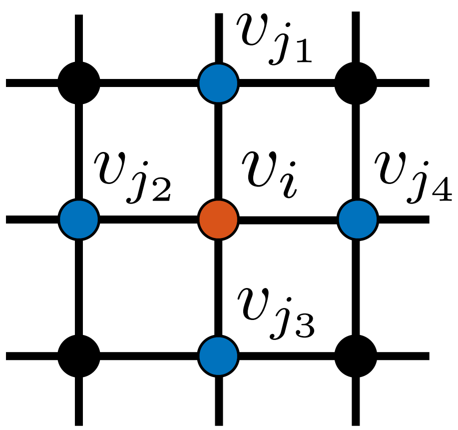
Graph Laplacian

Why graph Laplacian?

Graph Laplacian

Why graph Laplacian?

- approximation of the Laplace operator



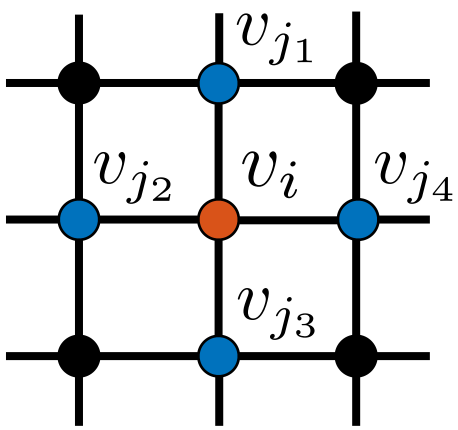
$$(Lf)(i) = 4f(i) - [f(j_1) + f(j_2) + f(j_3) + f(j_4)]$$

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

Graph Laplacian

Why graph Laplacian?

- approximation of the Laplace operator

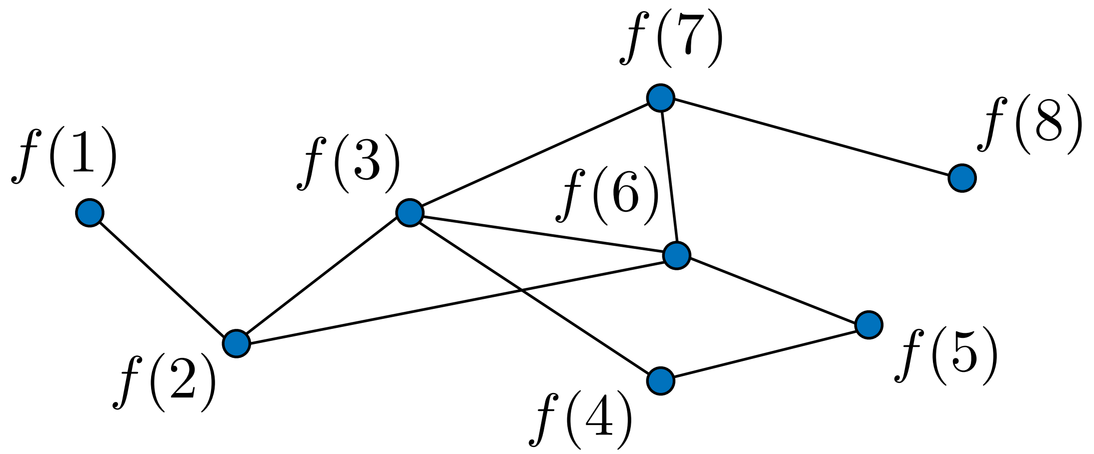


$$(Lf)(i) = 4f(i) - [f(j_1) + f(j_2) + f(j_3) + f(j_4)]$$

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

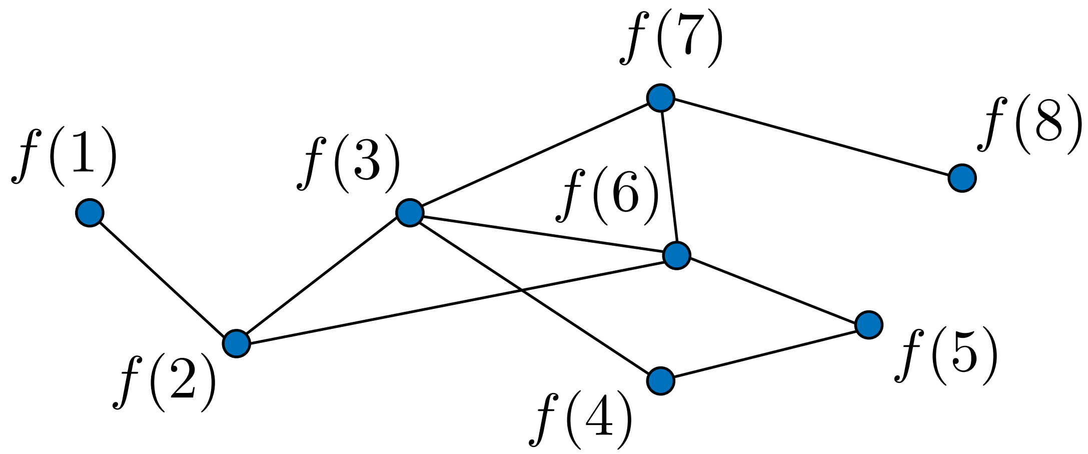
- converges to the Laplace-Beltrami operator (given certain conditions)
- provides a notion of "frequency" on graphs

Graph Laplacian



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

Graph Laplacian

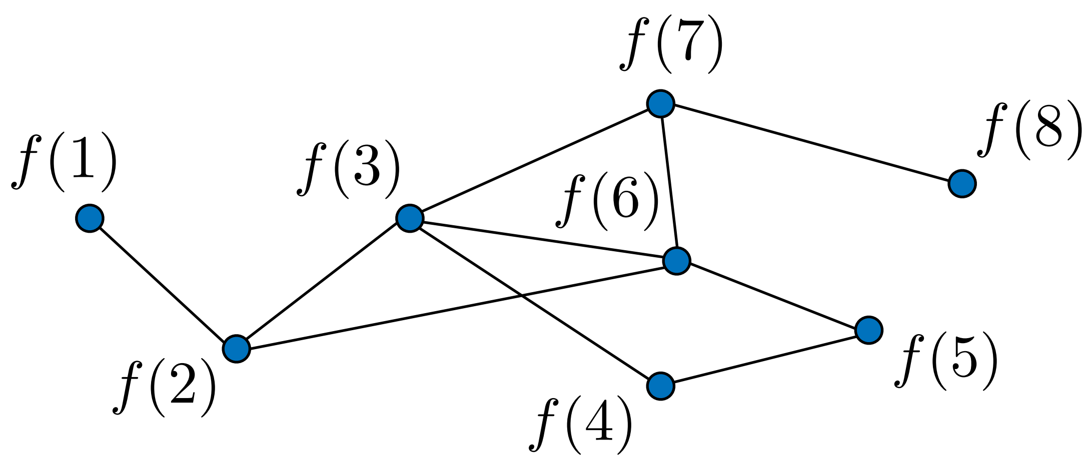


graph signal $f : \mathcal{V} \rightarrow \mathbb{R}^N$

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

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

Graph Laplacian



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

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

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

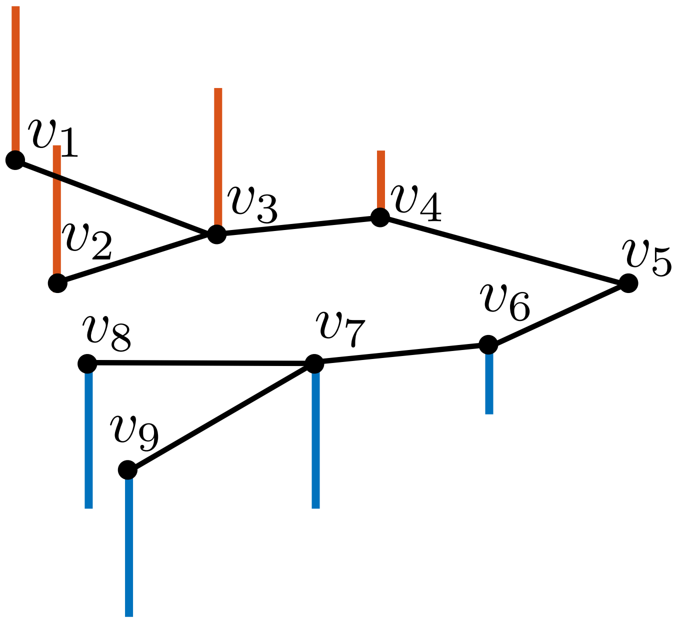
$$Lf(i) = \sum_{j=1}^{N} W_{ij}(f(i) - f(j))$$

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

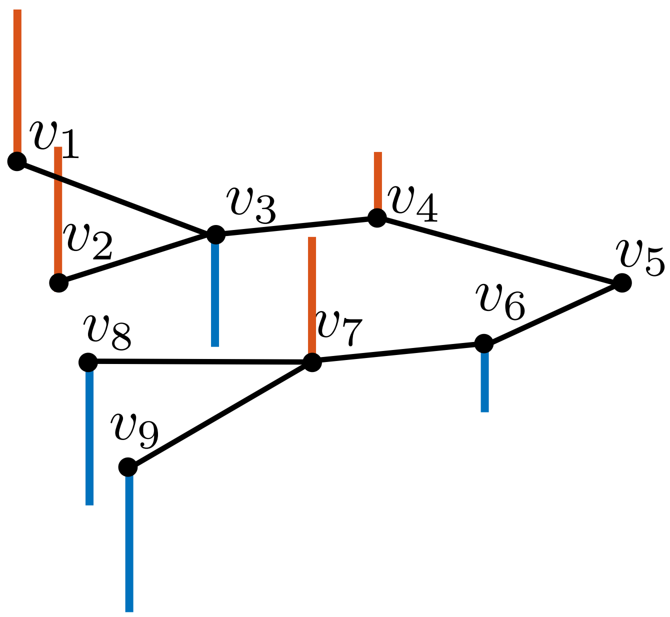
a measure of "smoothness"

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

Graph Laplacian



$f^T L f = 1$



$f^T L f = 21$

Graph Laplacian

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

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

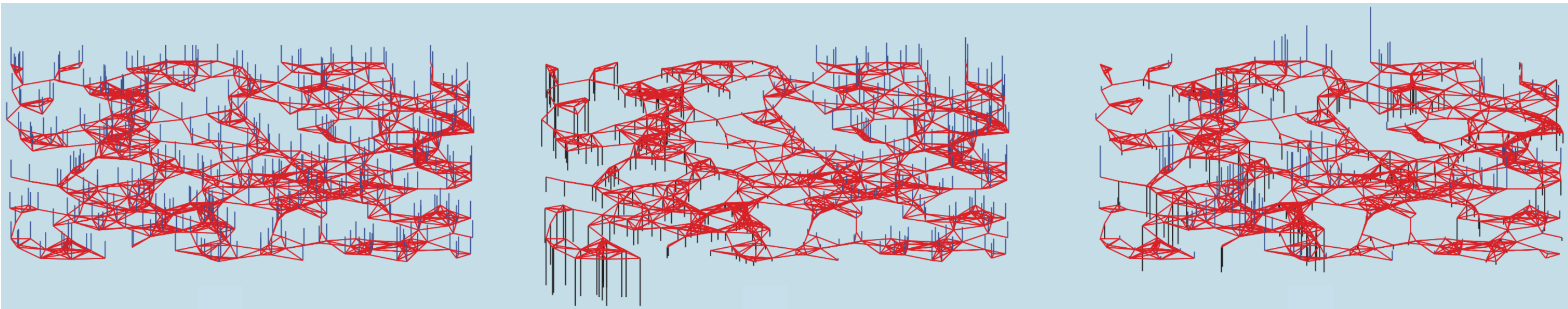
Graph Laplacian

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

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

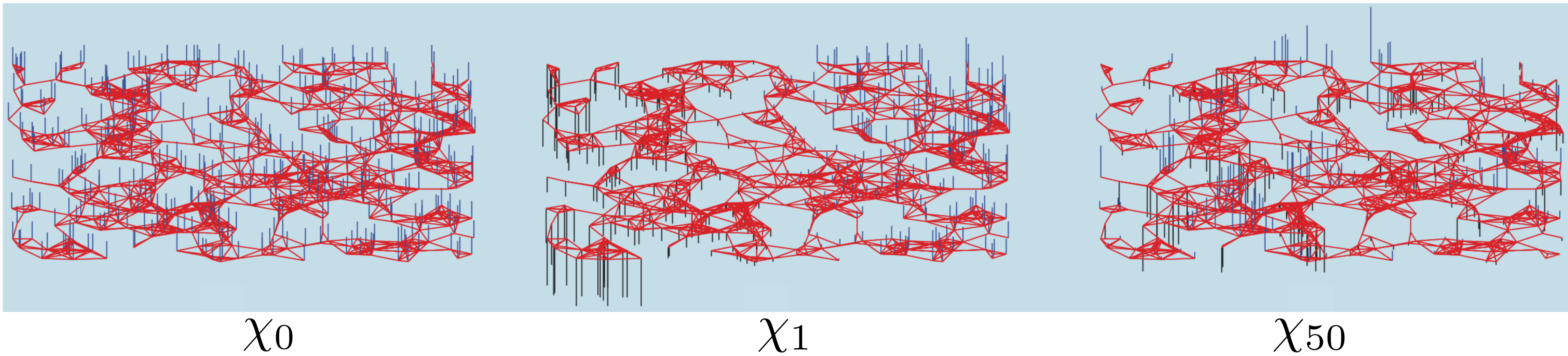
- Eigenvalues are usually sorted increasingly: $0 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{N-1}$

Graph Fourier transform



χ_0 χ_1 χ_{50}

Graph Fourier transform



low frequency → high frequency

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

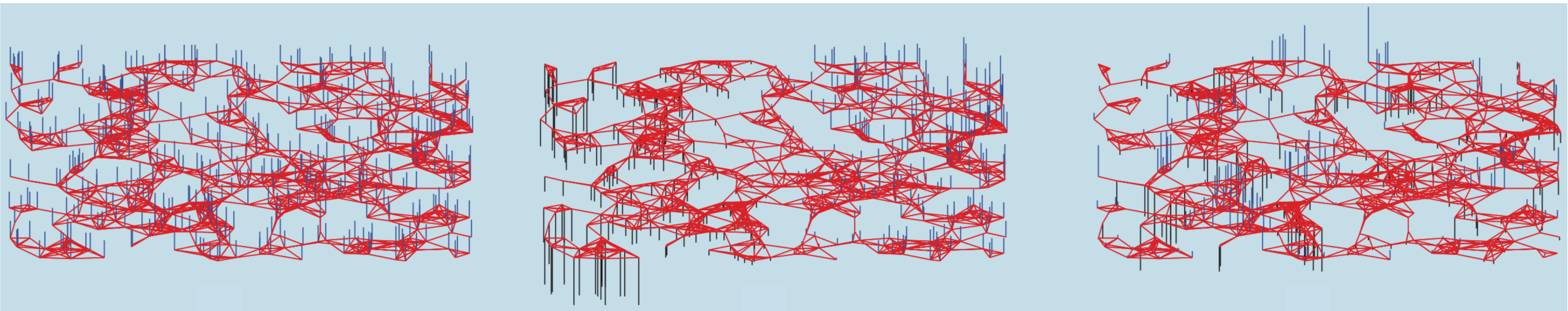
$\chi_0^T L \chi_0 = \lambda_0 = 0$

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

- 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



χ_0 χ_1 χ_{50}

low frequency → high frequency

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

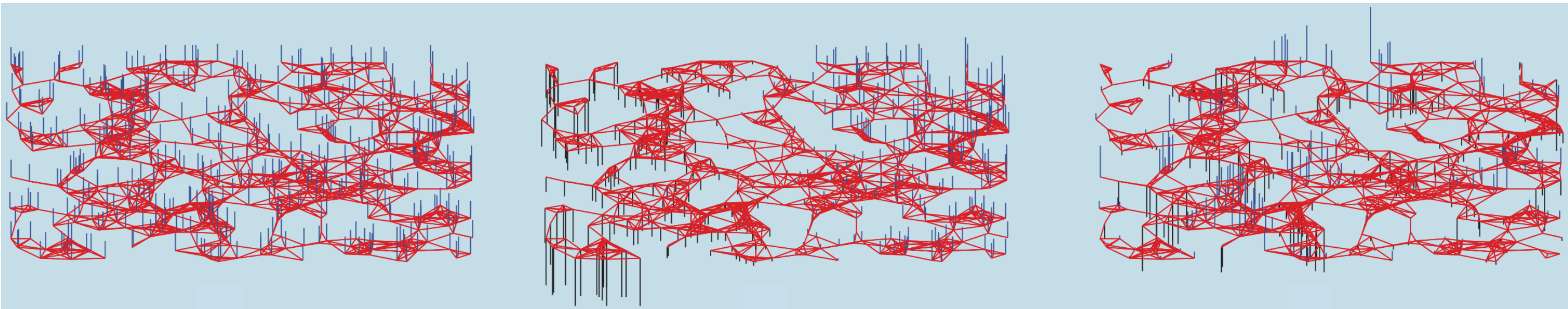
$\chi_0^T L \chi_0 = \lambda_0 = 0$

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

graph Fourier transform:
[Hammond11]

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

Graph Fourier transform



χ_0 χ_1 χ_{50}

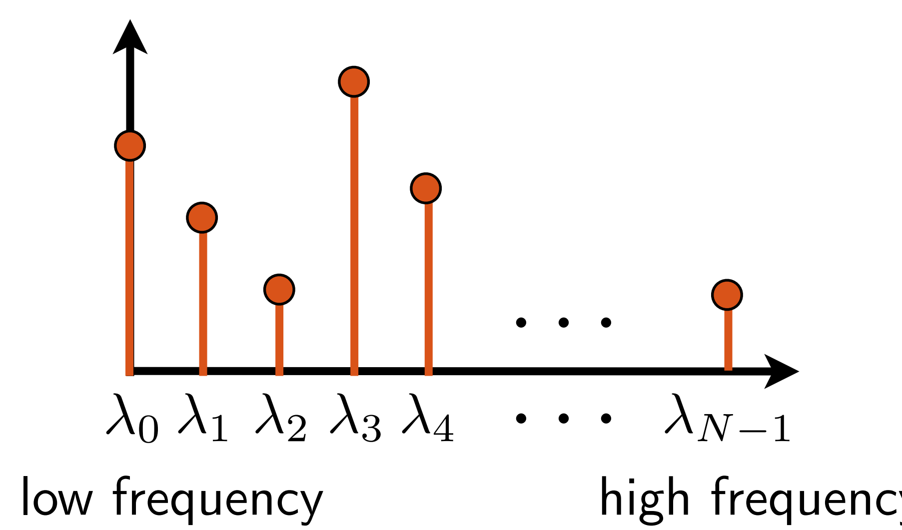
low frequency → high frequency

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

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

graph Fourier transform:
[Hammond11]

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



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: $-\nabla^2$

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

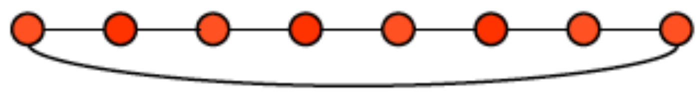
eigenvectors: χ_ℓ

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

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

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

Two special cases



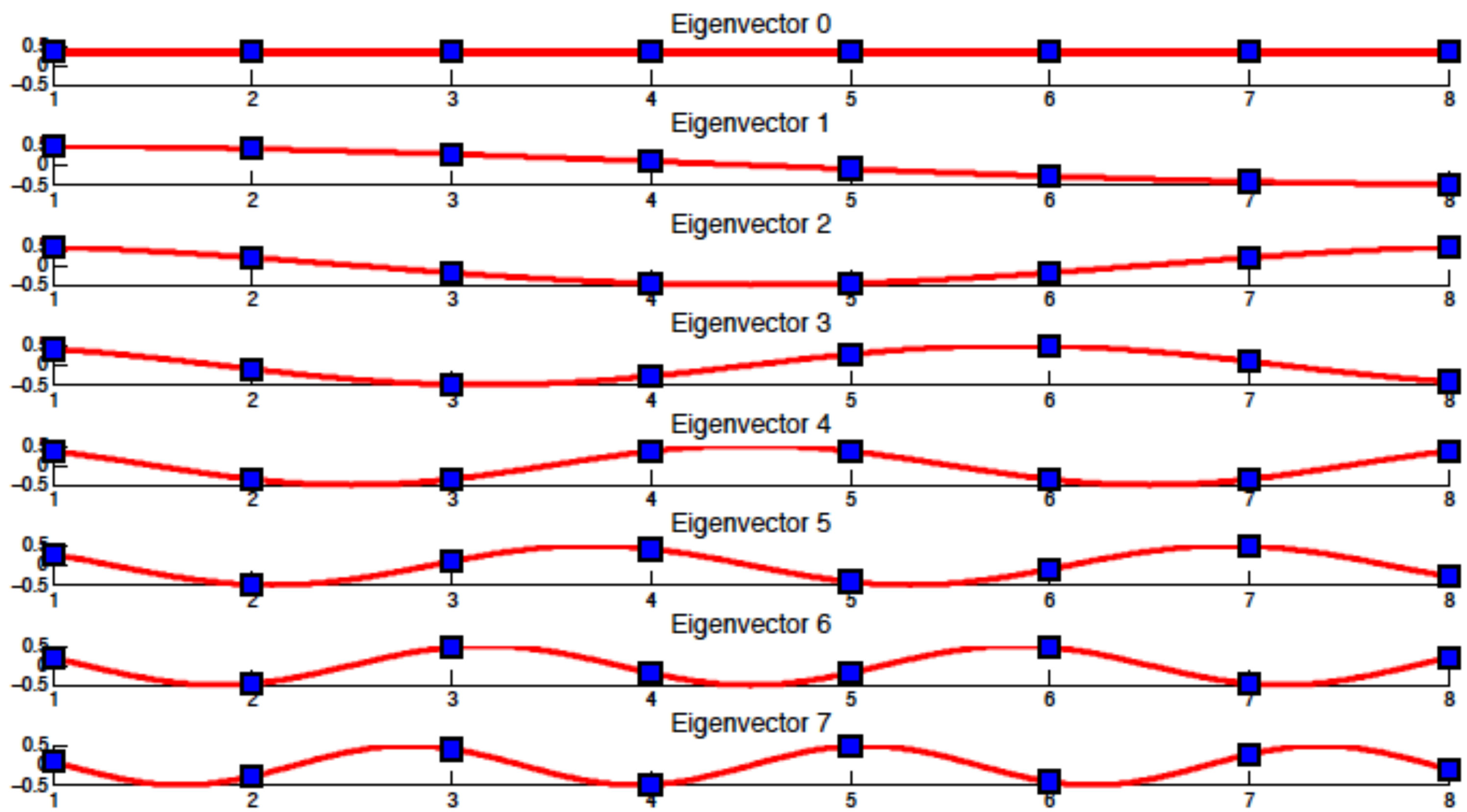
- (Unordered) Laplacian eigenvalues: $\lambda_\ell = 2 - 2\cos\left(\frac{2\ell\pi}{N}\right)$
- One possible choice of orthogonal Laplacian eigenvectors:

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

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

Two special cases

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

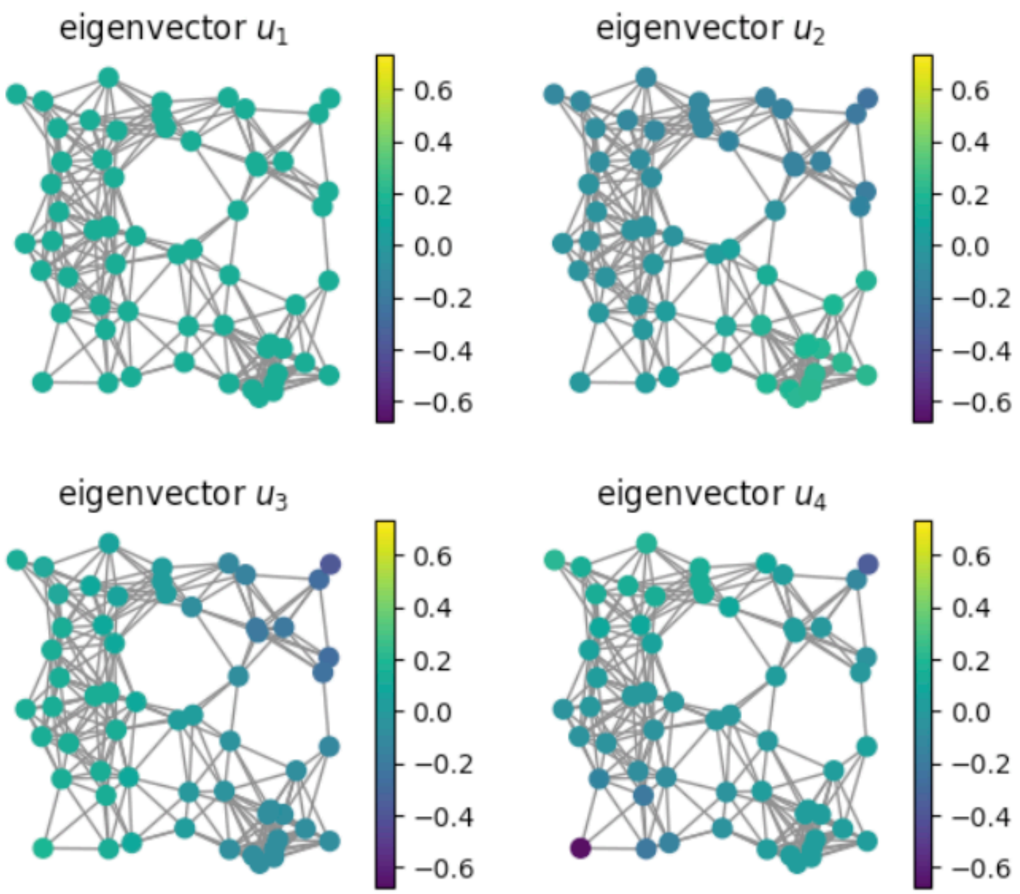


$$\begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N-1} \\ | & & | \end{bmatrix}$$ is the Discrete Cosine Transform matrix (DCT-II, Strang, 1999), which is used in JPEG image compression

Graph Fourier transform

- Example on a simple graph

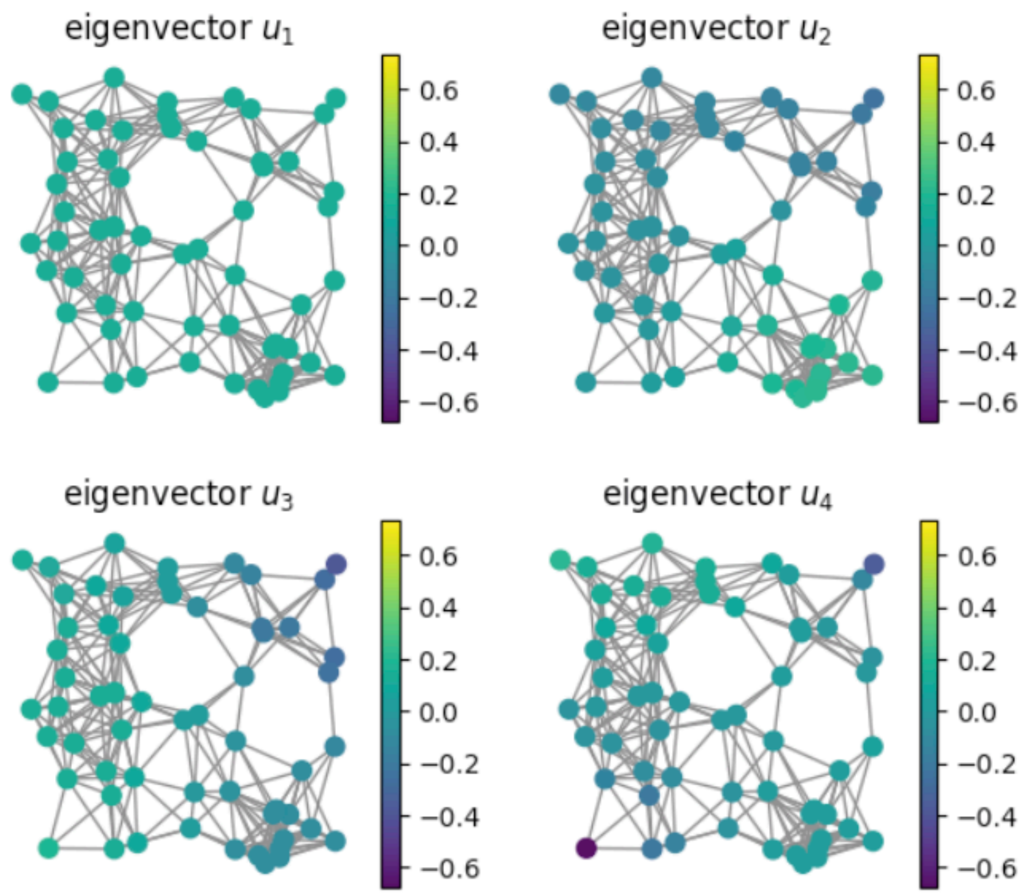
$$\hat{f}(\ell) = \langle \chi_\ell, f \rangle : \quad \begin{bmatrix} | & & | \\ \chi_0 & \cdots & \chi_{N\text{-}1} \\ | & & | \end{bmatrix}^T \begin{matrix} | \\ f \\ | \end{matrix}$$

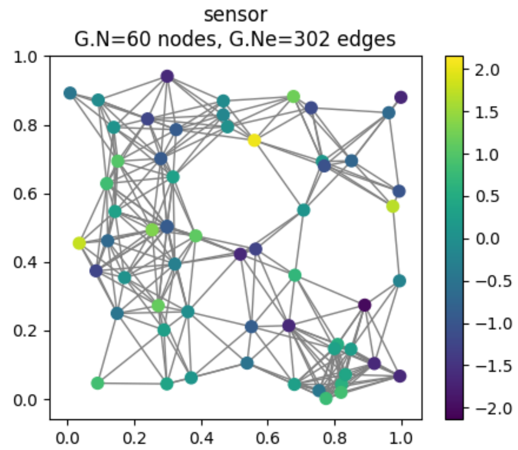


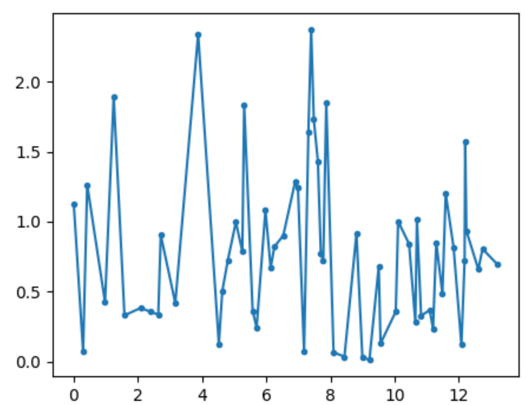
Graph Fourier transform

- Example on a simple graph

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



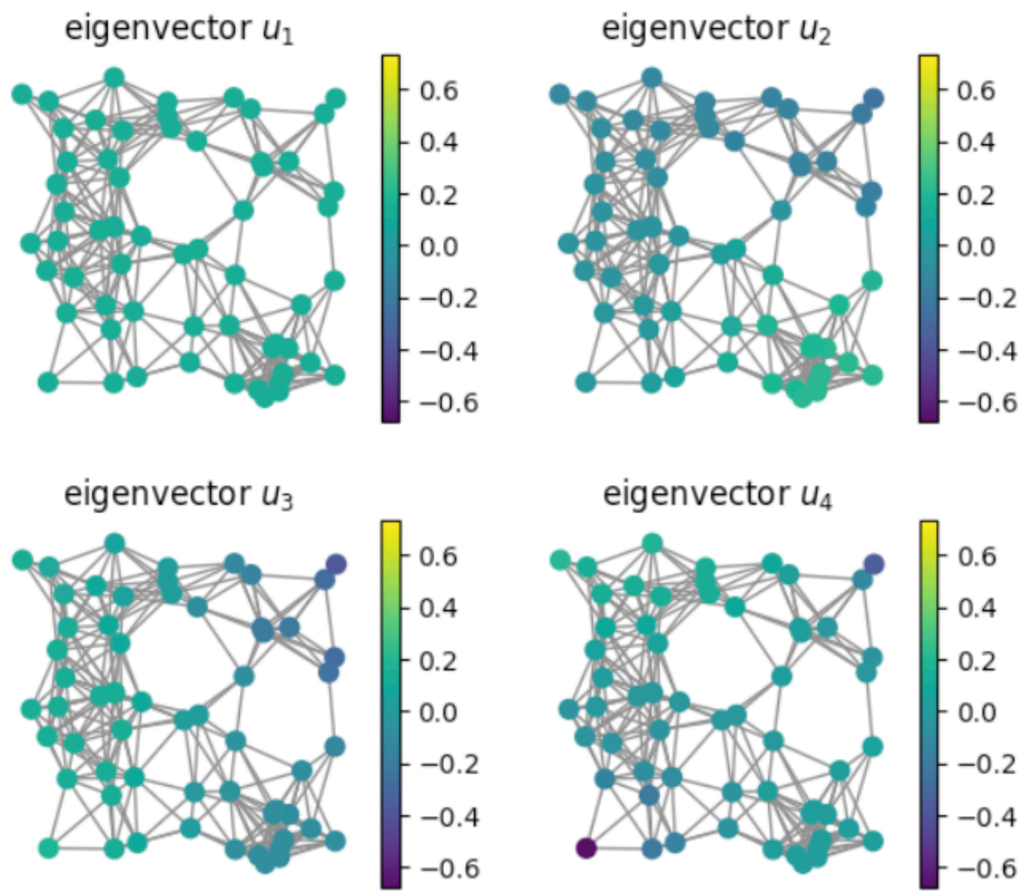


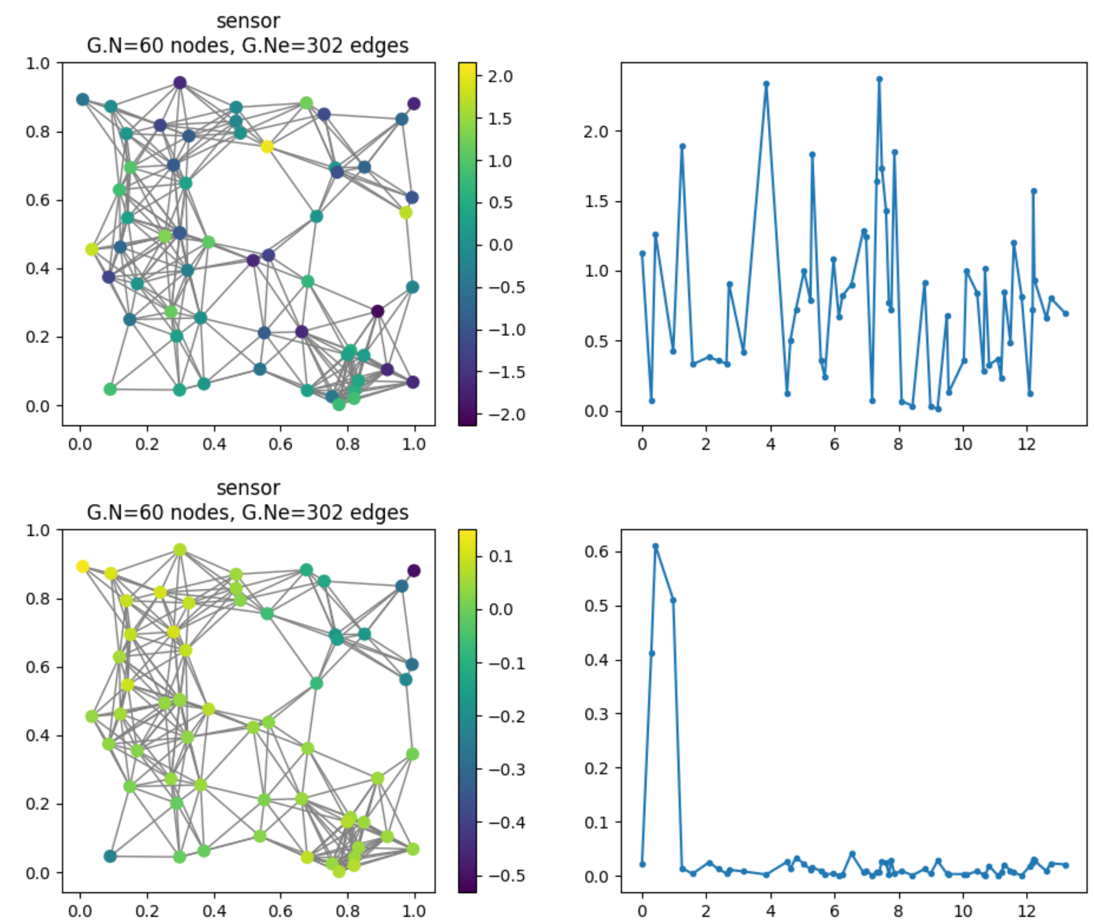


Graph Fourier transform

- Example on a simple graph

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





Outline

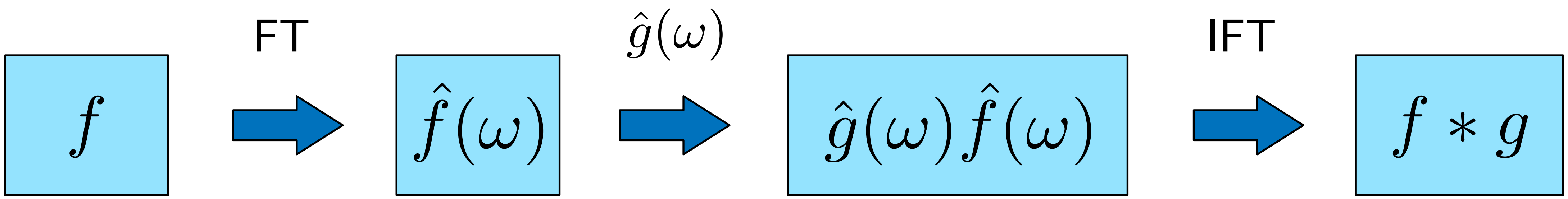
- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tool of GSP
- Graph neural networks (GNNs)
- Applications

Classical frequency filtering

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

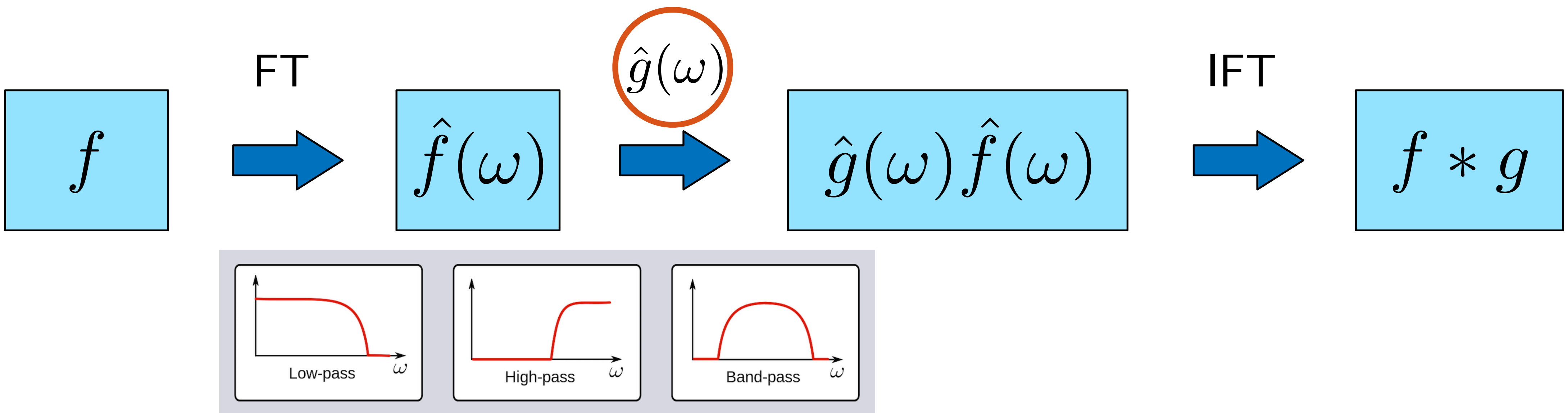
Classical frequency filtering

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



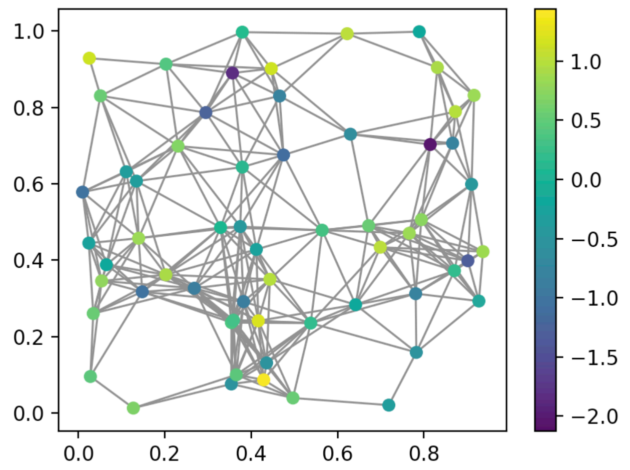
Graph spectral filtering

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

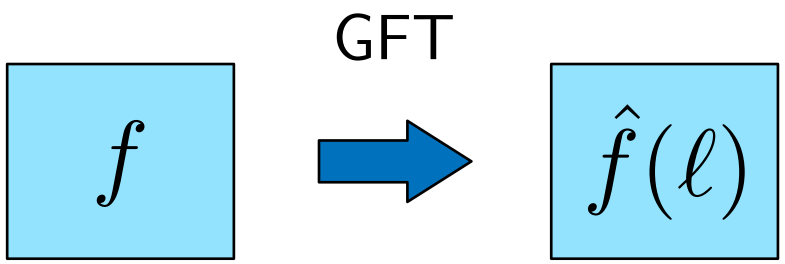
GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$

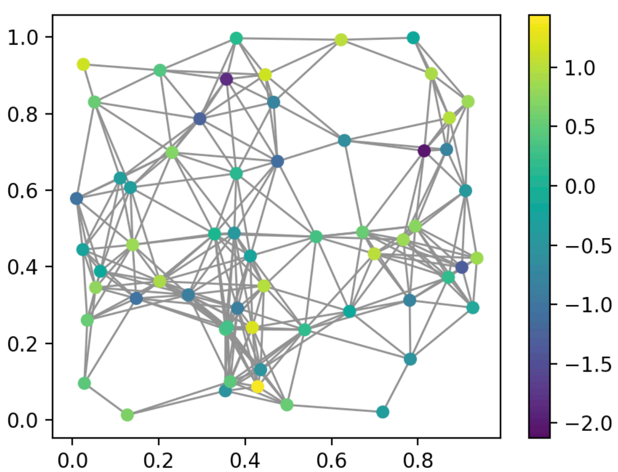
f

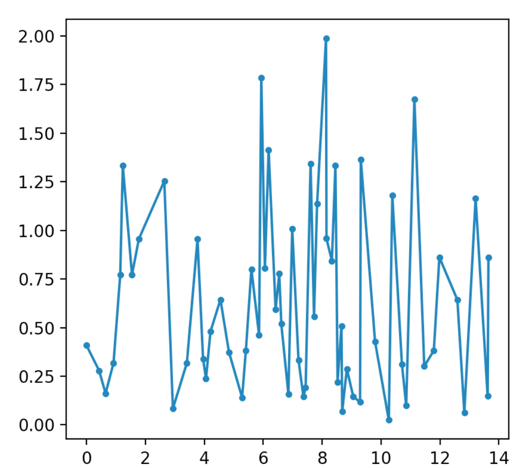


Graph spectral filtering

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

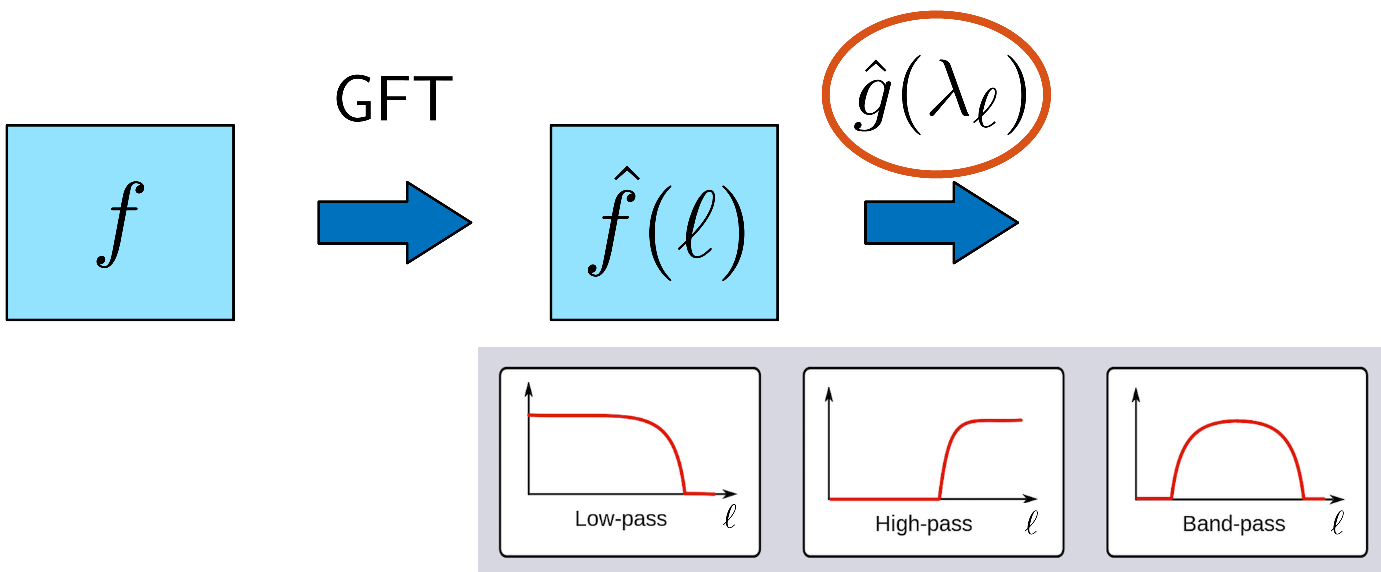


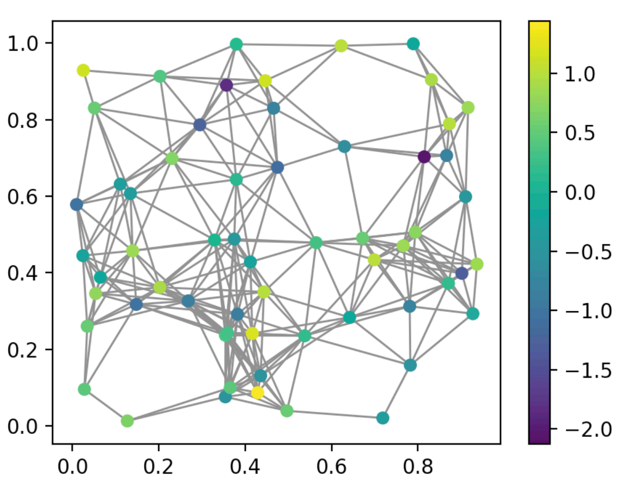


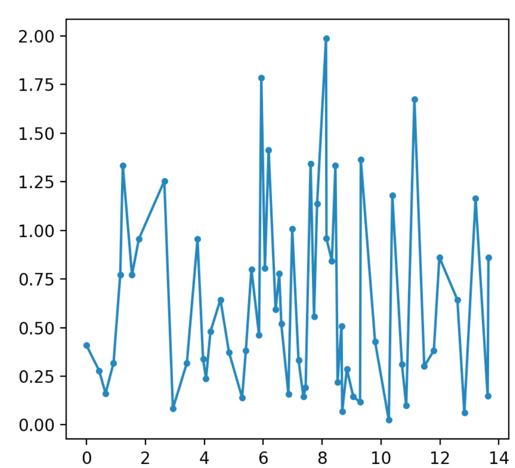


Graph spectral filtering

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

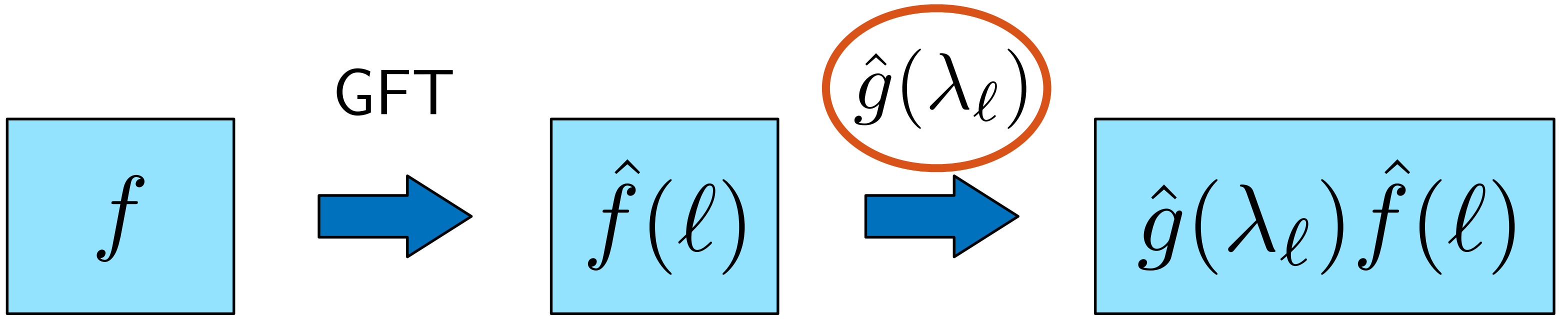


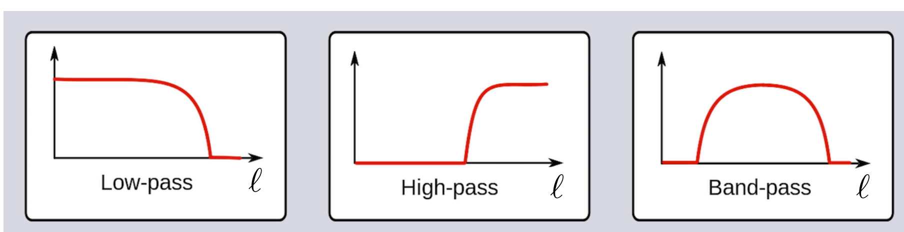


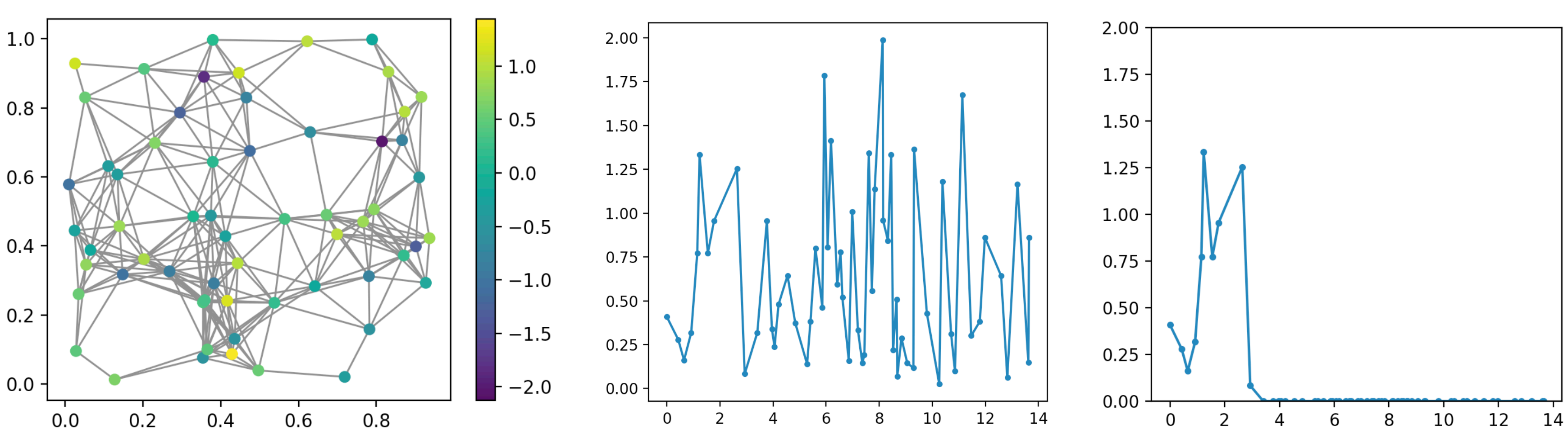


Graph spectral filtering

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



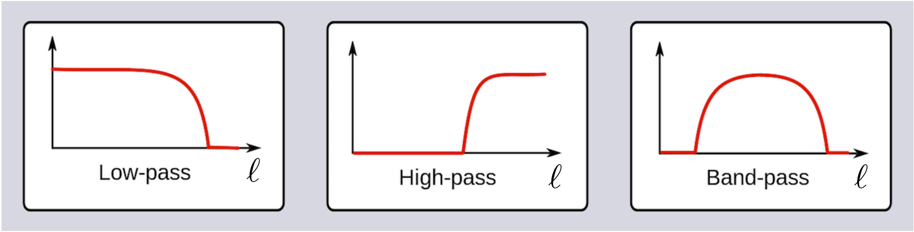


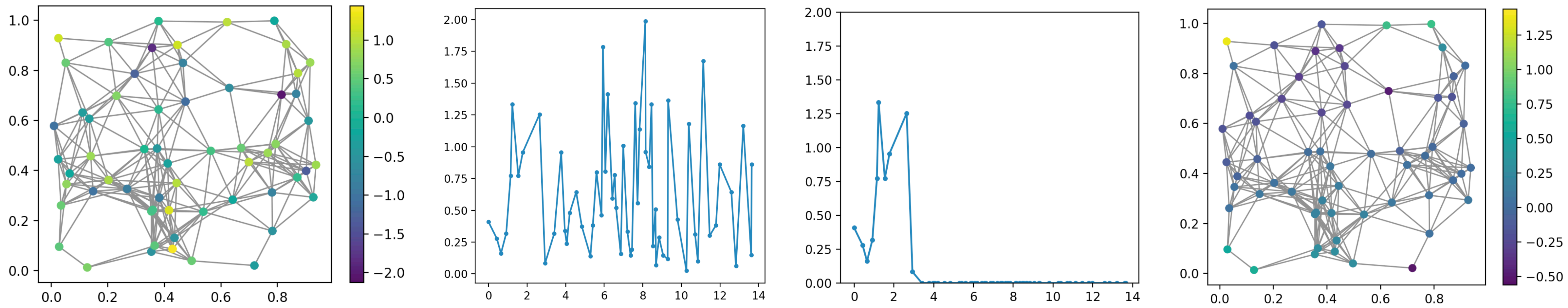


Graph spectral filtering

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

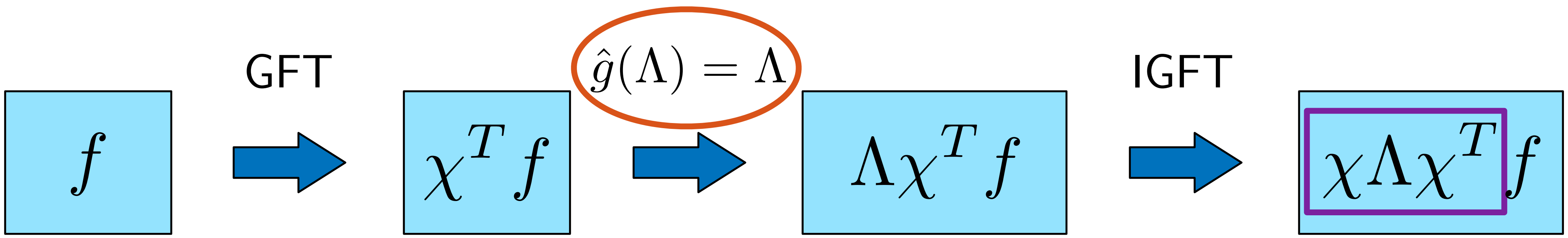




Graph Laplacian revisited

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

The Laplacian L is a difference operator: $Lf = \chi \Lambda \chi^T f$

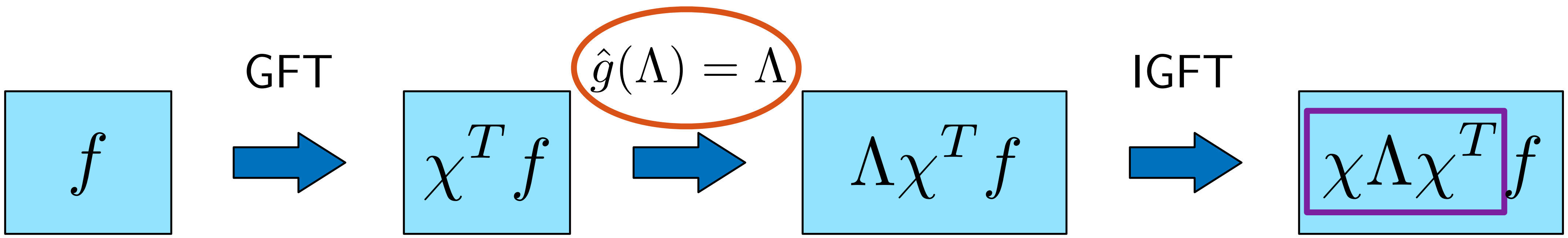


$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}$$

Graph Laplacian revisited

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

The Laplacian L is a difference operator: $Lf = \chi\Lambda\chi^T f$



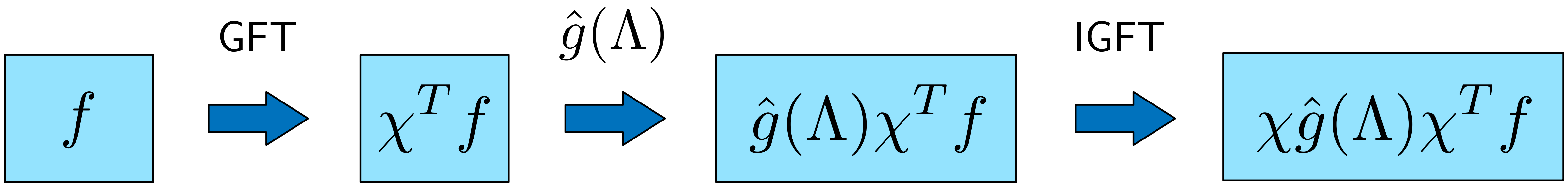
$$\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}$$

The Laplacian operator filters the signal in the spectral domain by its eigenvalues!

The Laplacian quadratic form: $f^T L f = ||L^{\frac{1}{2}} f||_2 = ||\chi\Lambda^{\frac{1}{2}}\chi^T f||_2$

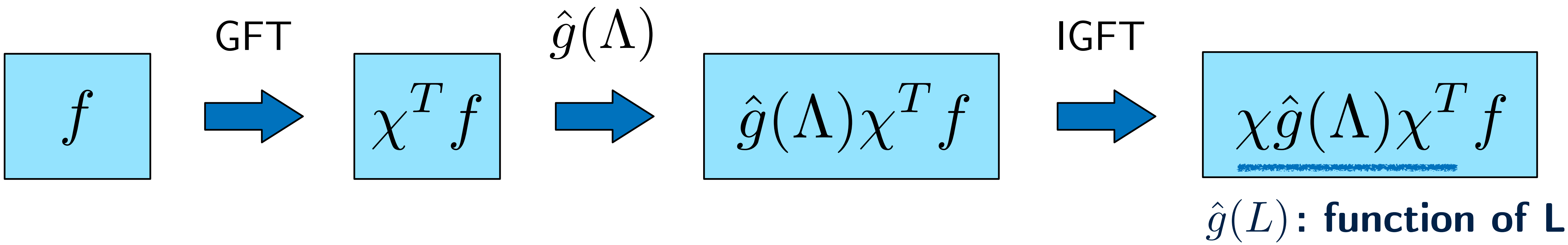
Graph transform/dictionary design

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



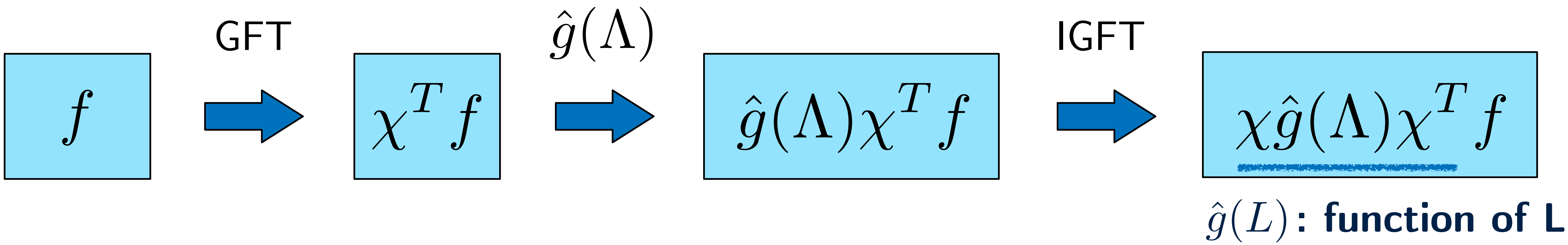
Graph transform/dictionary design

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



Graph transform/dictionary design

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

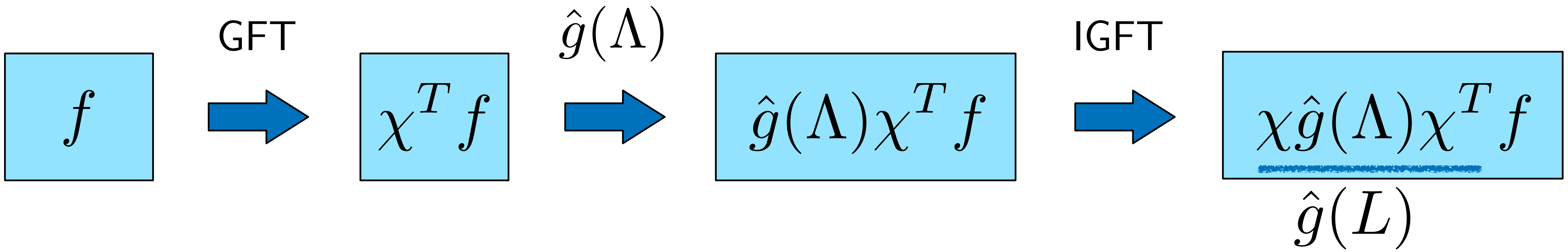


- Important properties can be achieved by properly defining $\hat{g}(L)$, such as localisation of atoms

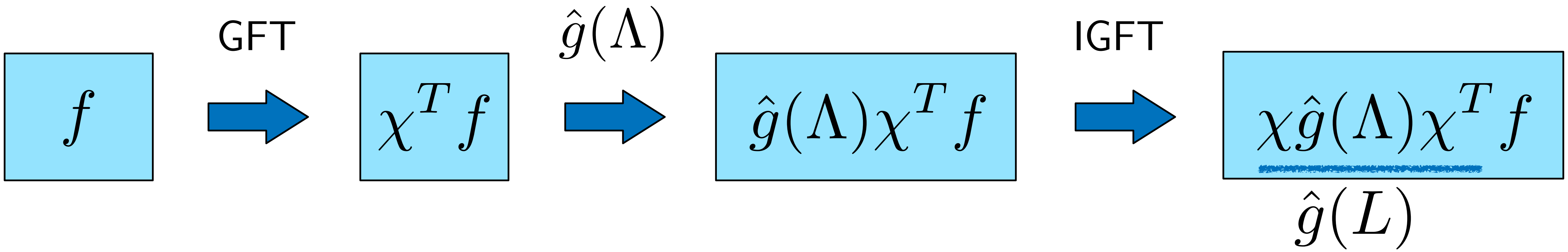
- Closely related to kernels and regularisation on graphs

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

A practical example



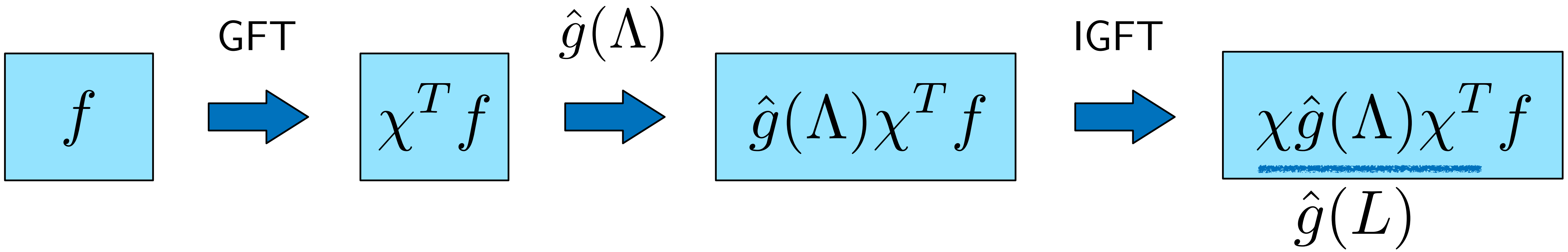
A practical example



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



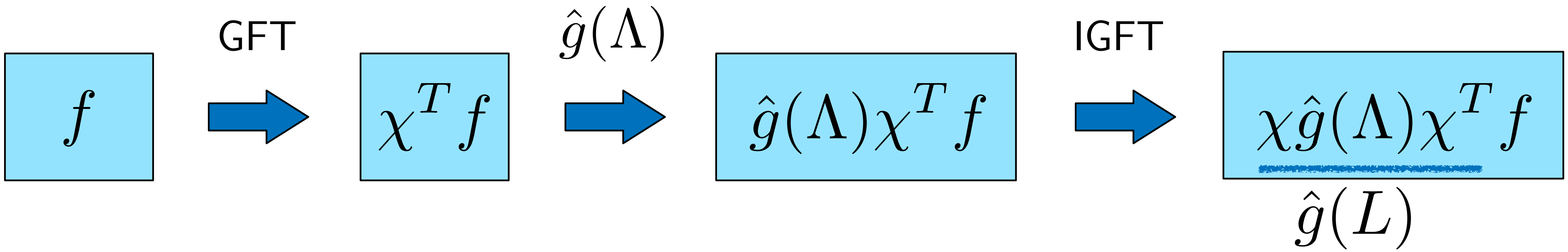
problem: we observe a noisy graph signal $f = y_0 + \eta$ and wish to recover y_0

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

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

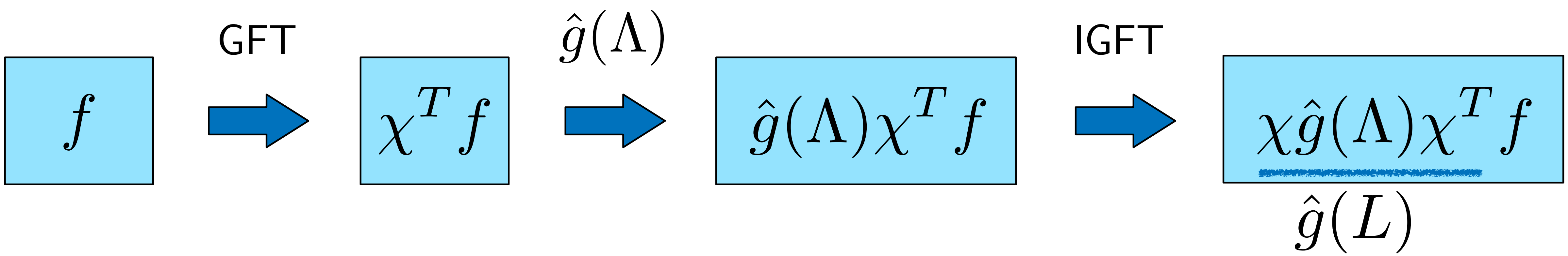
$$y^* = \arg\min_y \{ ||y - f||_2^2 + \gamma y^T L y \}$$

data fitting term: $||y - f||_2^2$

"smoothness" assumption: $\gamma y^T L y$

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

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: $||y - f||_2^2$

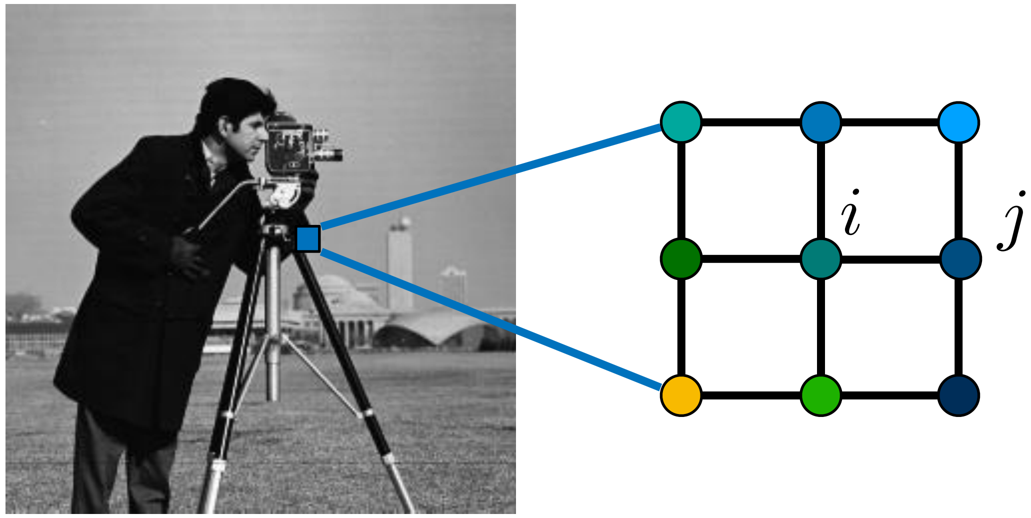
"smoothness" assumption: $\gamma y^T L y$

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

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

A practical example

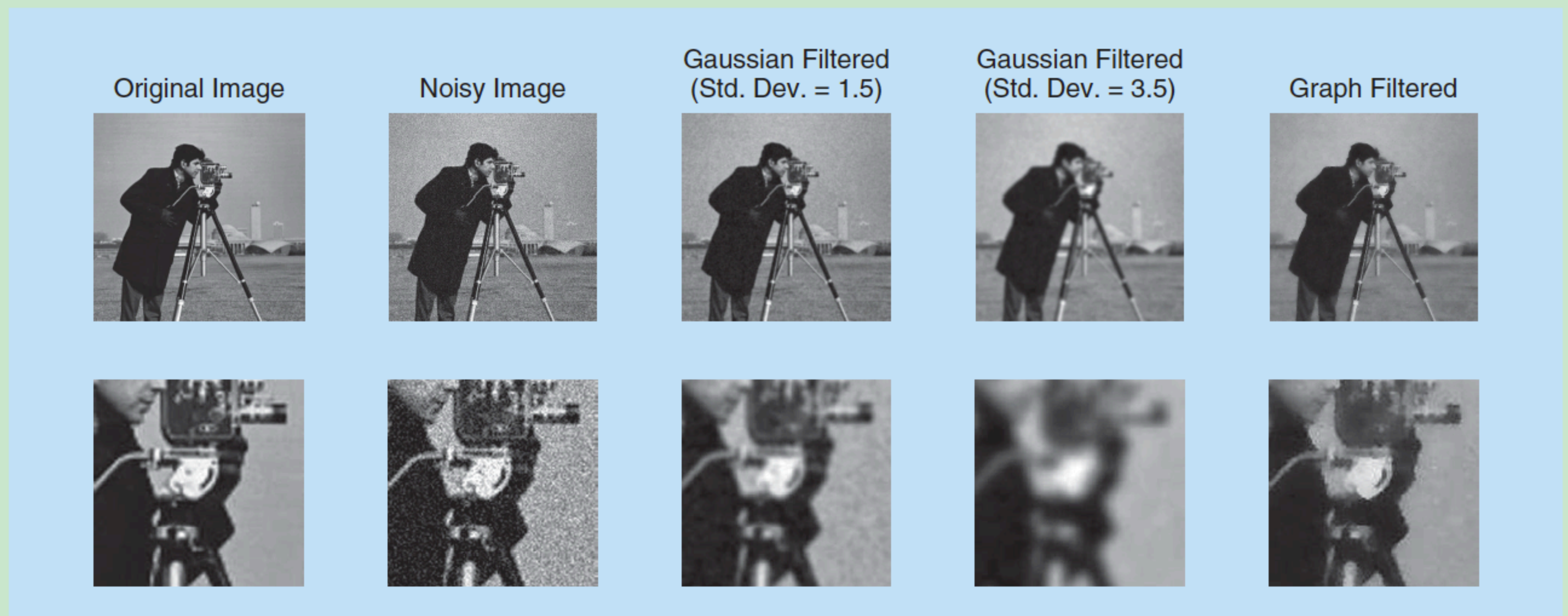
- noisy image as observed noisy graph signal
- regular grid graph (weights inversely proportional to pixel value difference)



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

A practical example

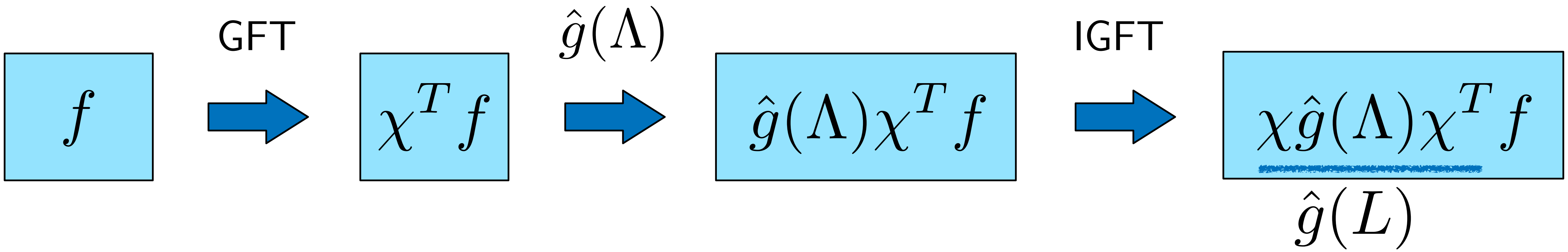
- noisy image as observed noisy graph signal
- regular grid graph (weights inversely proportional to pixel value difference)



[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.

More filtering operations



low-pass filters: $\hat{g}(L) = (I + \gamma L)^{-1} = \chi(I + \gamma\Lambda)^{-1}\chi^T$

window kernel: windowed graph Fourier transform

shifted and dilated band-pass filters: spectral graph wavelets $\hat{g}(sL)$

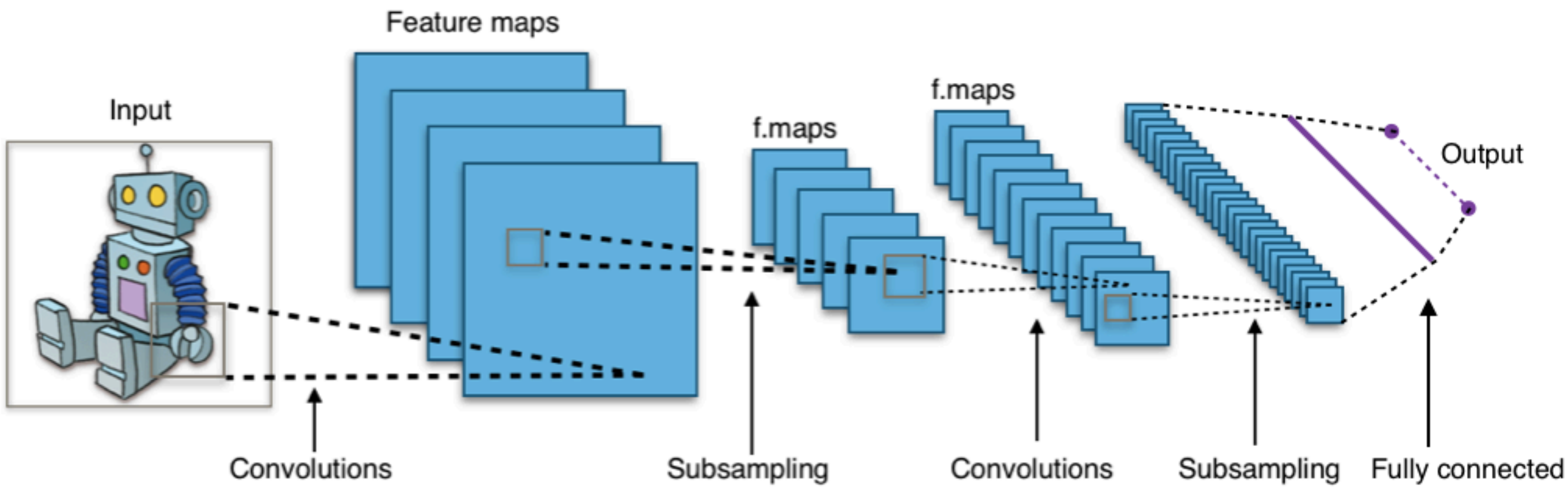
adapted kernels: learn values of $\hat{g}(L)$ directly from data

parametric polynomials: $\hat{g}_s(L) = \sum_{k=0}^{K} \alpha_{sk} L^k = \chi(\sum_{k=0}^{K} \alpha_{sk} \Lambda^k)\chi^T$

Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tool of GSP
- Graph neural networks (GNNs)
- Applications

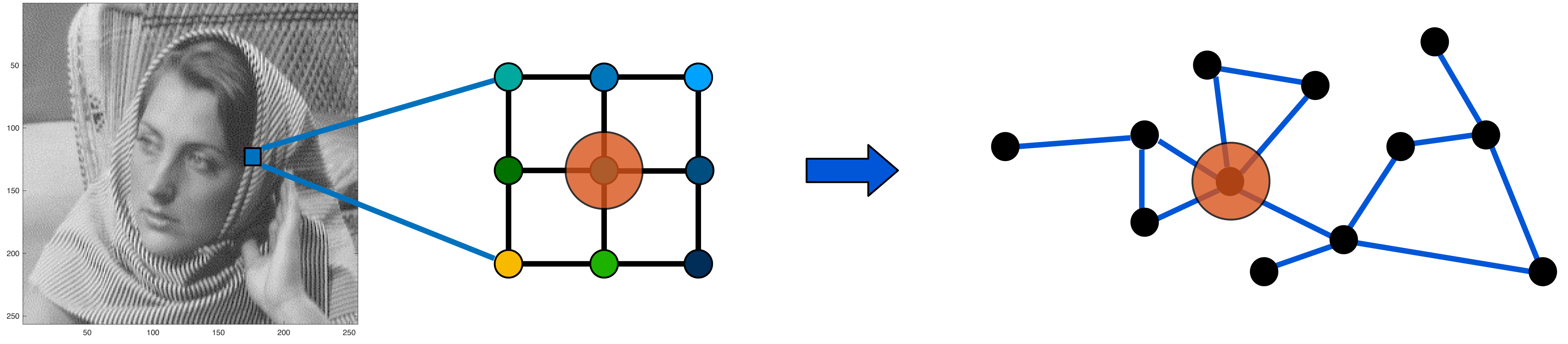
CNNs exploit structure within data



https://en.wikipedia.org/wiki/File:Typical_cnn.png

- checklist
  - **convolution:** translation equivariance
  - **localisation:** compact filters (independent of sample dimension)
  - **multi-scale:** compositionality
  - **efficiency:** $\mathcal{O}(N)$ computational complexity

CNNs on graphs?



- checklist
  - **convolution:** how to do it on graphs?
  - **localisation:** what's the notion of locality?
  - **multi-scale:** how to down-sample on graphs?
  - **efficiency:** how to keep the computational complexity low?

Convolution on graphs

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

?

Convolution on graphs

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

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

Convolution on graphs

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

node domain

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

graph spectral domain

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

Convolution on graphs

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

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

Convolution on graphs

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

parametric filter as polynomial of eigenvalues

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

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

Convolution on graphs

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

parametric filter as polynomial of eigenvalues

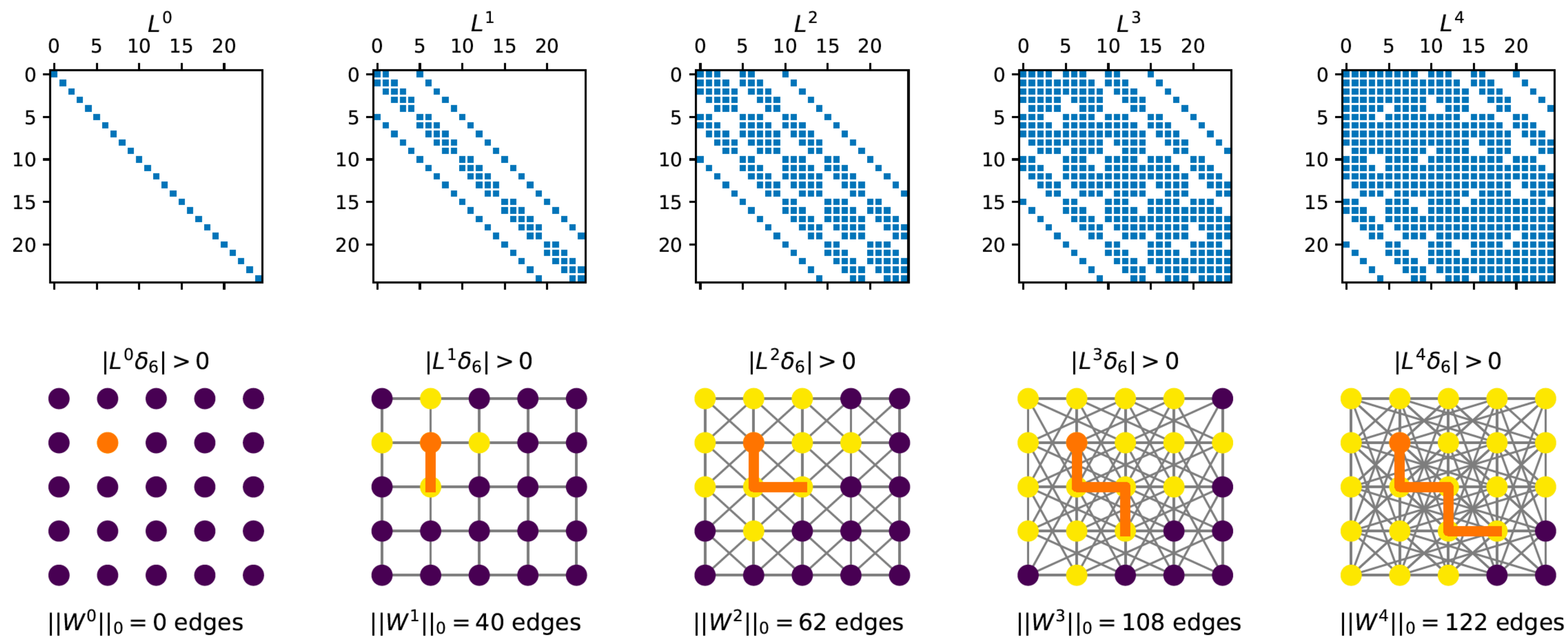
$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

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 $(L^K)_{ij} = 0$

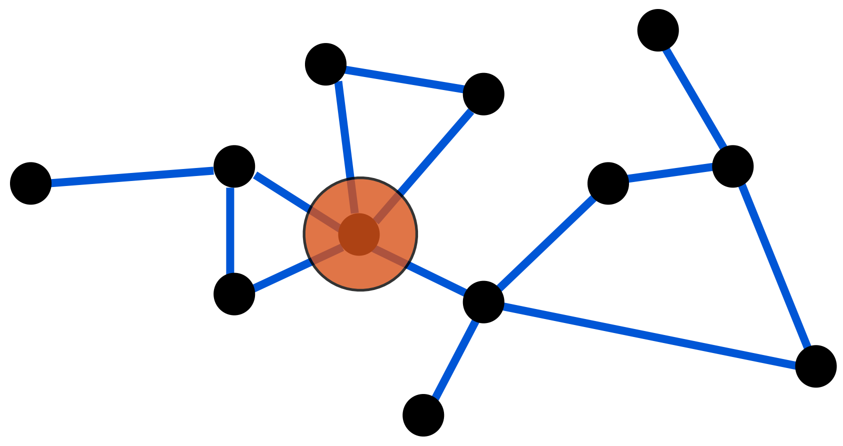
(slides by Michaël Defferrard)

Convolution on graphs

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

parametric filter as polynomial of eigenvalues

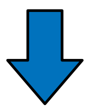
$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$



(localisation within *K*-hop neighbourhood)

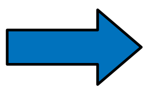
Convolution on graphs

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



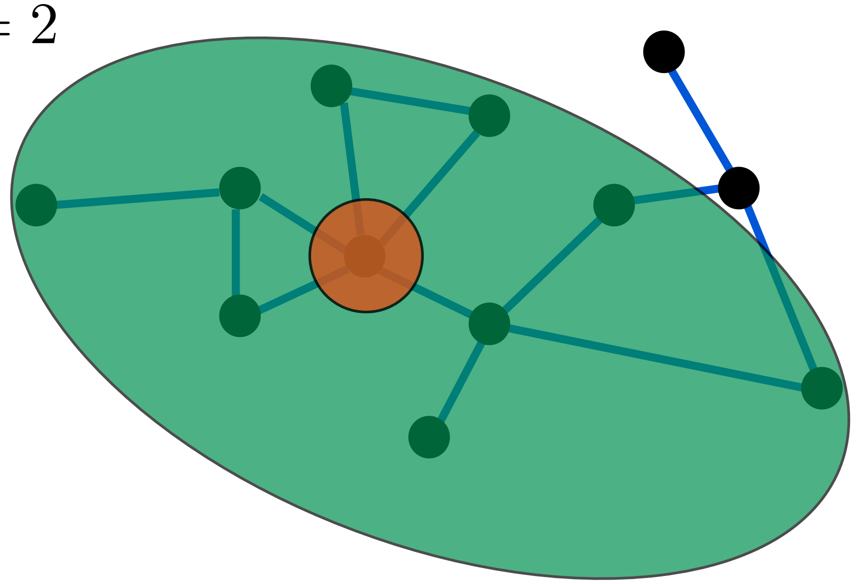
parametric filter as polynomial of eigenvalues

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



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

$K = 2$



(localisation within *K*-hop neighbourhood)

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

Convolution on graphs

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

parametric filter as polynomial of eigenvalues

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

- checklist
  - **convolution:** expressed in the graph spectral domain
  - **localisation:** within K-hop neighbourhood
  - **learning complexity:** $\mathcal{O}(K)$
  - **computational complexity:** $\mathcal{O}(K|\mathcal{E}|)$, no need for GFT

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

Convolution on graphs

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

fast and stable implementation by Chebyshev approximation

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j T_j(\tilde{L})$$ with a scaled Laplacian $$\tilde{L} = \frac{2}{\lambda_{N-1}} L - I$$

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

Convolution on graphs

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

⬇

fast and stable implementation by Chebyshev approximation

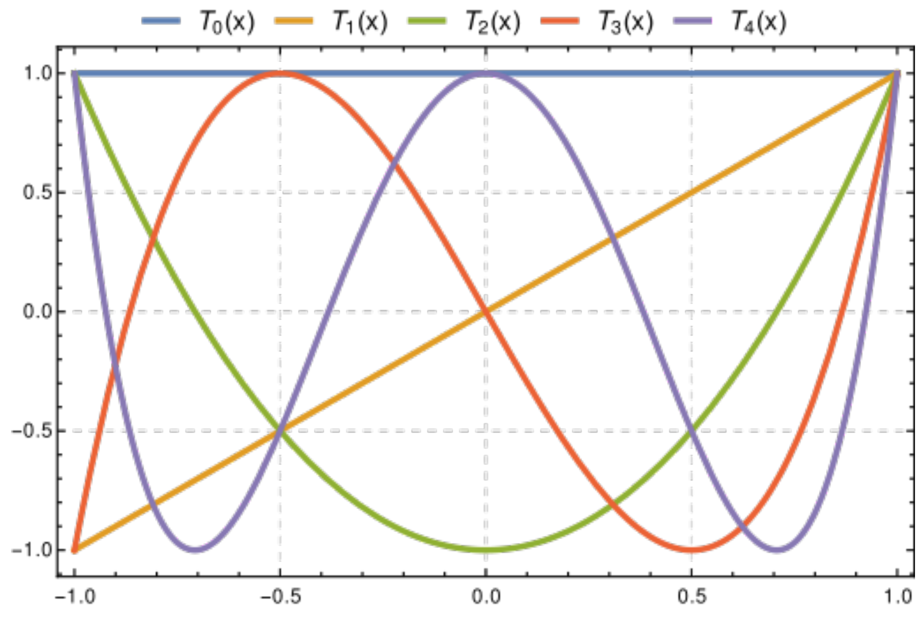
$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j T_j(\tilde{L})$$ with a scaled Laplacian $\tilde{L} = \frac{2}{\lambda_{N-1}} L - I$

$T_0(x) = 1$

$T_1(x) = x$

$T_k(x) = 2x T_{k-1}(x) - T_{k-2}(x)$

- recursively defined
- orthogonal basis for $L^2([-1,1], dy/\sqrt{1-y^2})$



Convolution on graphs

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

↓

fast and stable implementation by Chebyshev approximation

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j T_j(\tilde{L})$$ with a scaled Laplacian $$\tilde{L} = \frac{2}{\lambda_{N-1}} L - I$$

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_k(x) = 2x T_{k-1}(x) - T_{k-2}(x)$$

→

$$T_0(\tilde{L}) = I$$
$$T_1(\tilde{L}) = \tilde{L}$$
$$T_k(\tilde{L}) = 2\tilde{L}\, T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L})$$

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

Convolution on graphs

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

⬇

simplified Chebyshev approximation

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

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

Convolution on graphs

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

⬇

simplified Chebyshev approximation

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j T_j(\tilde{L}) \quad \xrightarrow{K=1} \quad \theta_0 I + \theta_1 (\tilde{L} - I)$$

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

Convolution on graphs

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

⬇

simplified Chebyshev approximation

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

$\xrightarrow{K=1}$ $\theta_0 I + \theta_1(\tilde{L} - I)$

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

(localisation within 1-hop neighbourhood)

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

Convolution on graphs

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

simplified Chebyshev approximation

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

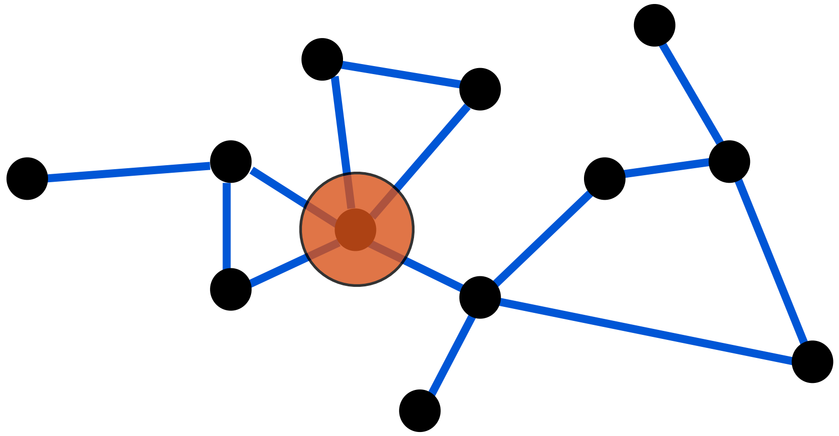
$K = 1$

$$\theta_0 I + \theta_1(\tilde{L} - I)$$

$\lambda_{N-1} \approx 2$

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

(localisation within 1-hop neighbourhood)



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

Convolution on graphs

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

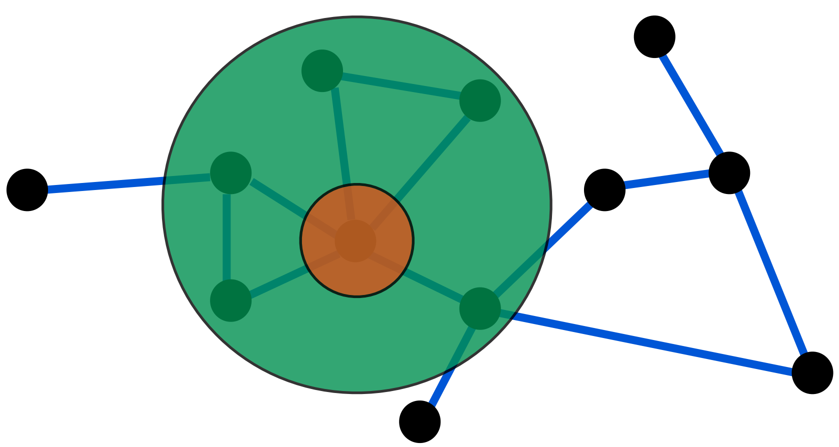
simplified Chebyshev approximation

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

$\xrightarrow{K=1}$ $\theta_0 I + \theta_1(\tilde{L} - I)$

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

(localisation within 1-hop neighbourhood)



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

Convolution on graphs

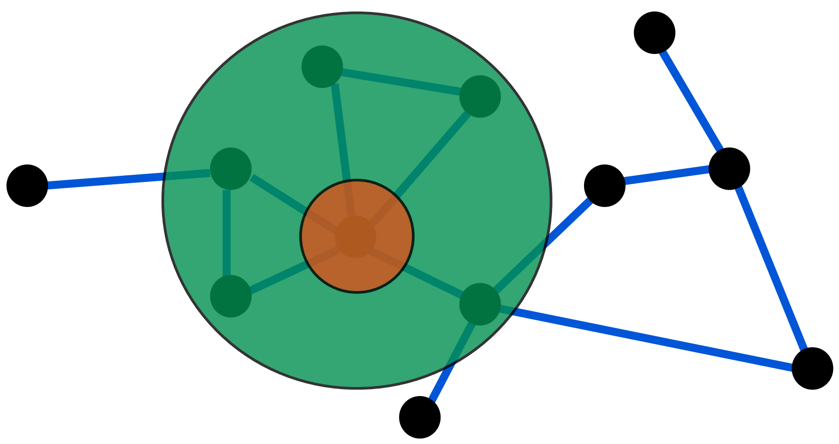
$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

simplified Chebyshev approximation

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

$K = 1$ → $\theta_0 I + \theta_1(\tilde{L} - I)$

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



(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.

Convolution on graphs

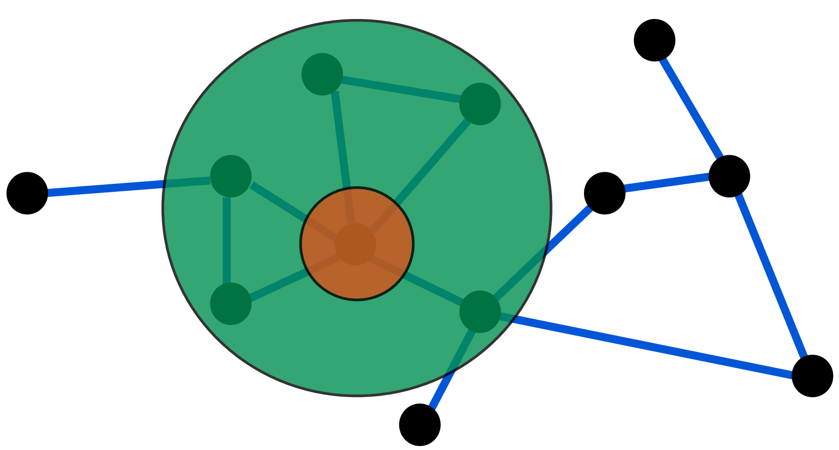
$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

simplified Chebyshev approximation

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

$K = 1$ $\Rightarrow$ $\theta_0 I + \theta_1(\tilde{L} - I)$

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



(localisation within 1-hop neighbourhood)

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

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.

Convolution on graphs

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

⬇

simplified Chebyshev approximation: too simple?

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

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

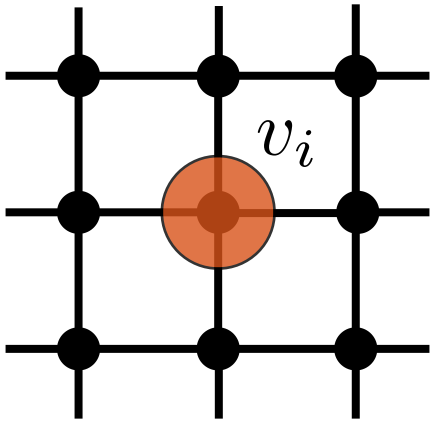
Convolution on graphs

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

simplified Chebyshev approximation: too simple?

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

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



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

Convolution on graphs

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

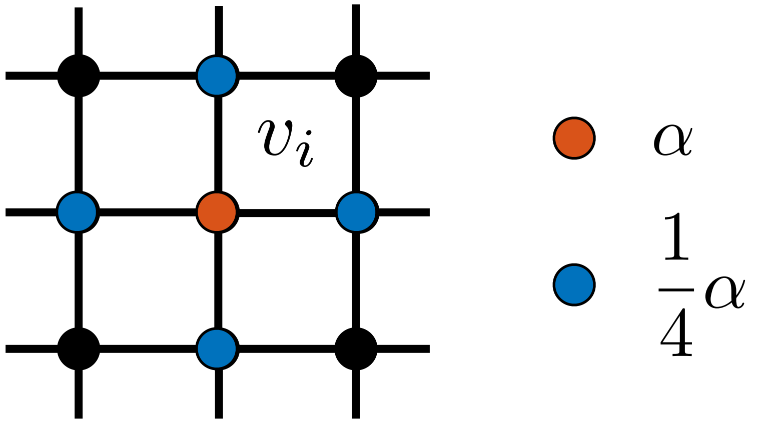
simplified Chebyshev approximation: too simple?

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

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

unitary edge weights

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



Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

- ...but can be implemented in the spatial (vertex) domain

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

Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

- ...but can be implemented in the spatial (vertex) domain

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

➡ simple averaging in Kipf and Welling 2017

Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

- ...but can be implemented in the spatial (vertex) domain

$$y = \hat{g}_\theta(L) f = \sum_{j=0}^{K} \theta_j T_j(\tilde{L}) f$$ ⇨ simple averaging in Kipf and Welling 2017

- Forward pass in spectral graph CNNs

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

Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

- ...but can be implemented in the spatial (vertex) domain

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

simple averaging in Kipf and Welling 2017

- Forward pass in spectral graph CNNs

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

Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

- ...but can be implemented in the spatial (vertex) domain

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

→ simple averaging in Kipf and Welling 2017

- Forward pass in spectral graph CNNs

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

Convolution on graphs - Summary

- Convolution is defined via the graph spectral domain...

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

- ...but can be implemented in the spatial (vertex) domain

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

→ simple averaging in Kipf and Welling 2017

- Forward pass in spectral graph CNNs

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

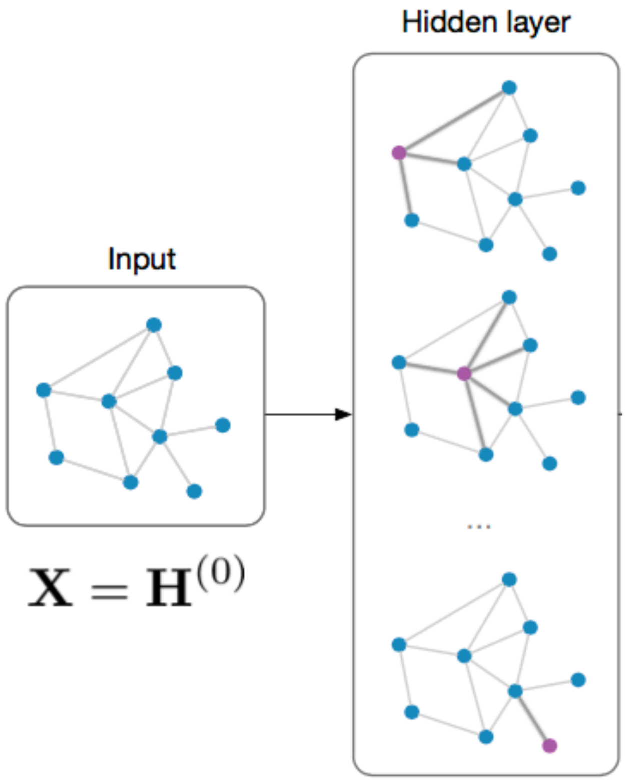
Graph convolutional networks (GCN)

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

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

Graph convolutional networks (GCN)

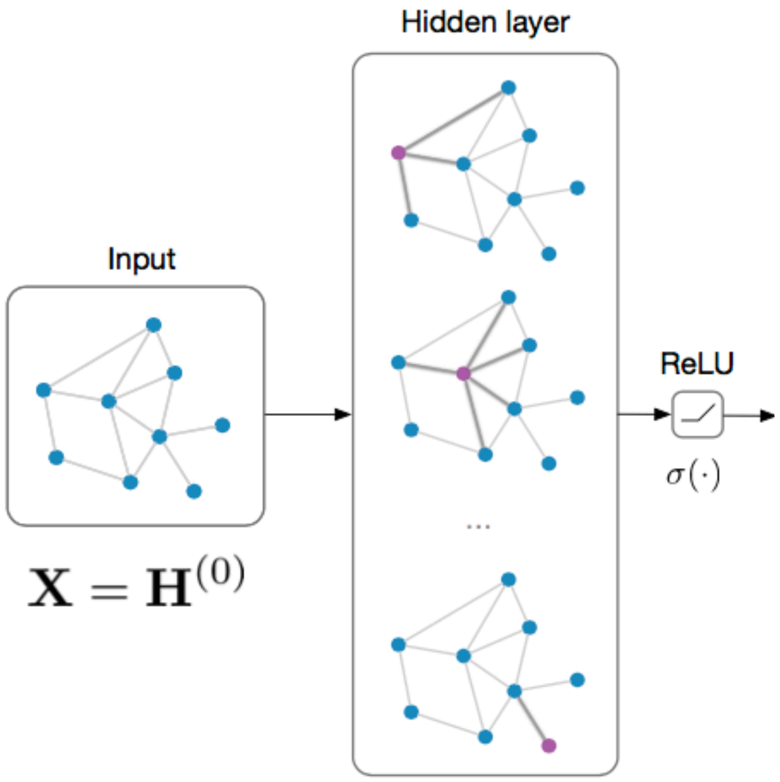
- Forward pass: $\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$



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

Graph convolutional networks (GCN)

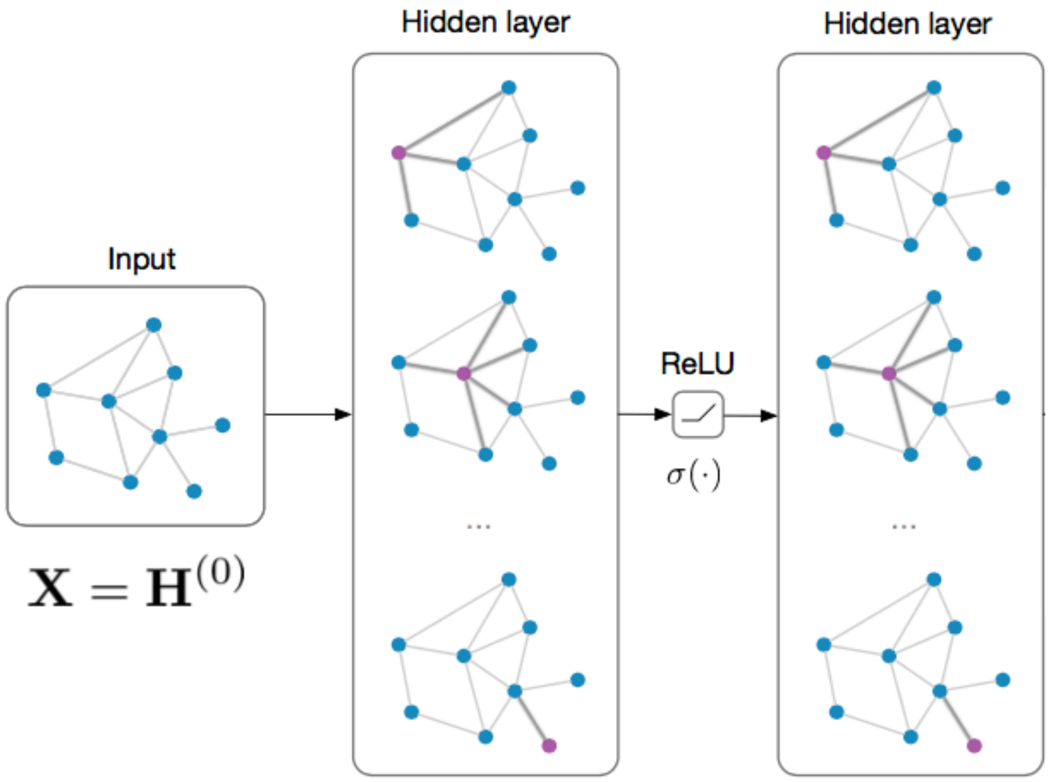
- Forward pass: $\hat{g}_{\theta^{(k+1)}}(L)\left(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\right)$



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

Graph convolutional networks (GCN)

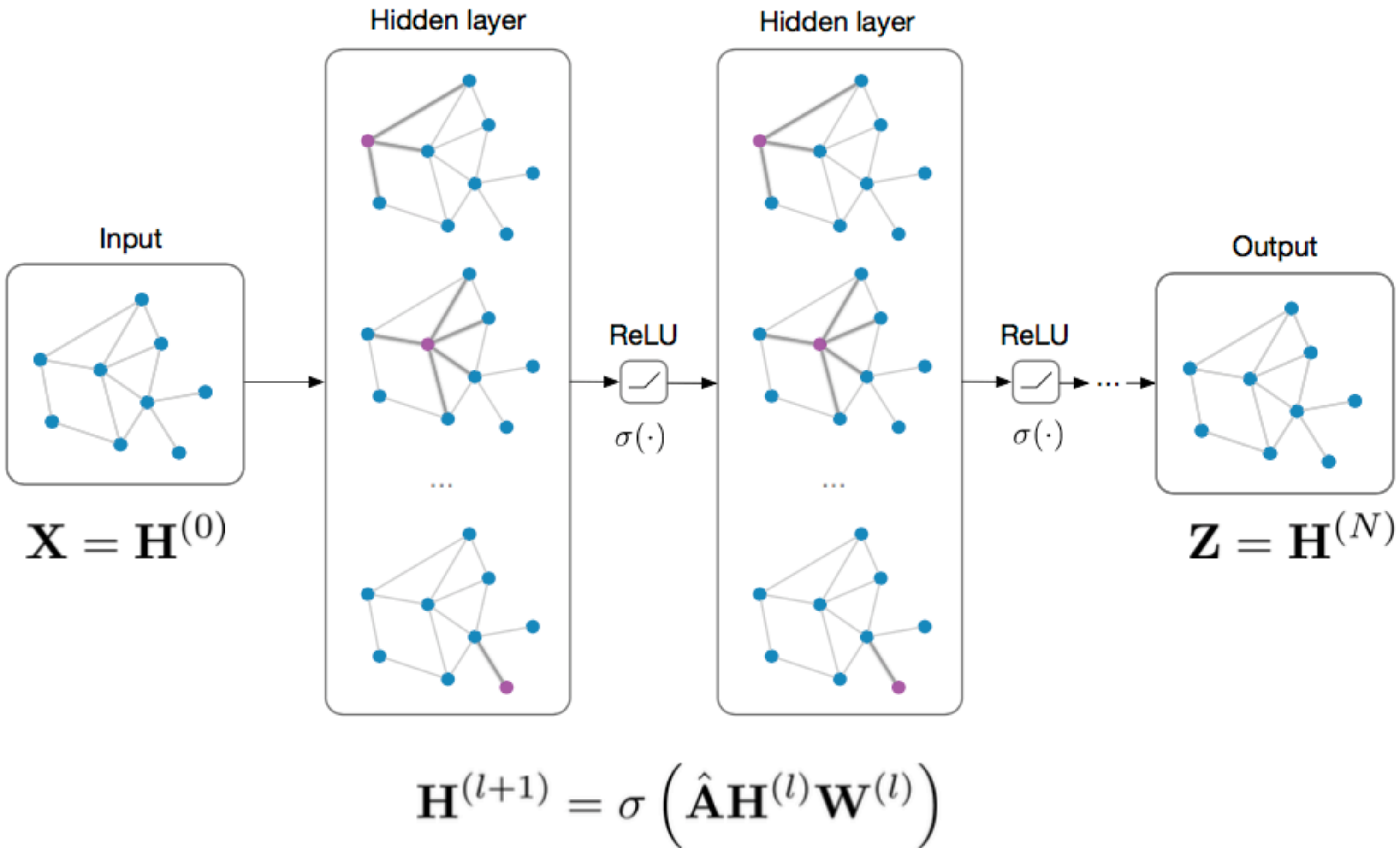
- Forward pass: $\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$



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

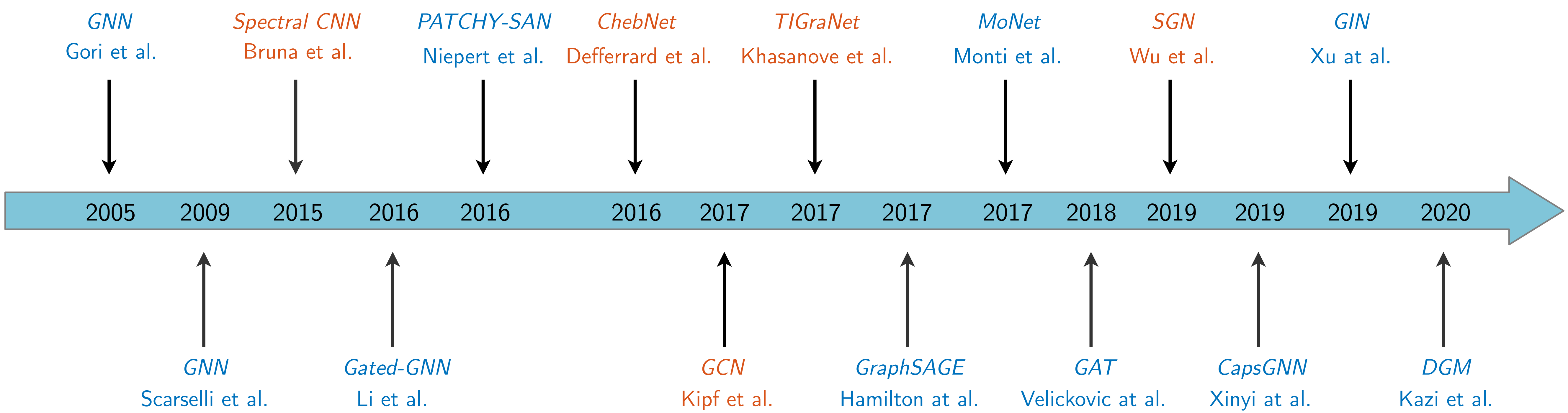
Graph convolutional networks (GCN)

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



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

Graph neural networks (GNNs)



spatial-based methods (message-passing, attention)

spectral-based methods (spectral filtering)

Balcilar et al., "Bridging the gap between spectral and spatial domains in graph neural networks," arXiv, 2020.

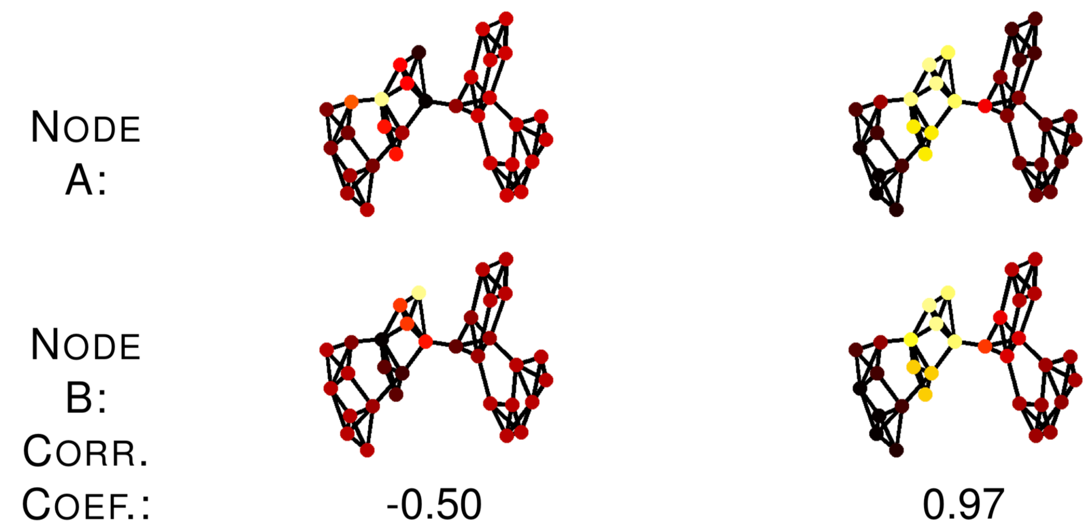
Outline

- Graph signal processing (GSP): Basic concepts
- Graph spectral filtering: Basic tool of GSP
- Graph neural networks (GNNs)
- Applications

Application I: Community detection

spectral graph wavelets
at different scales:

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



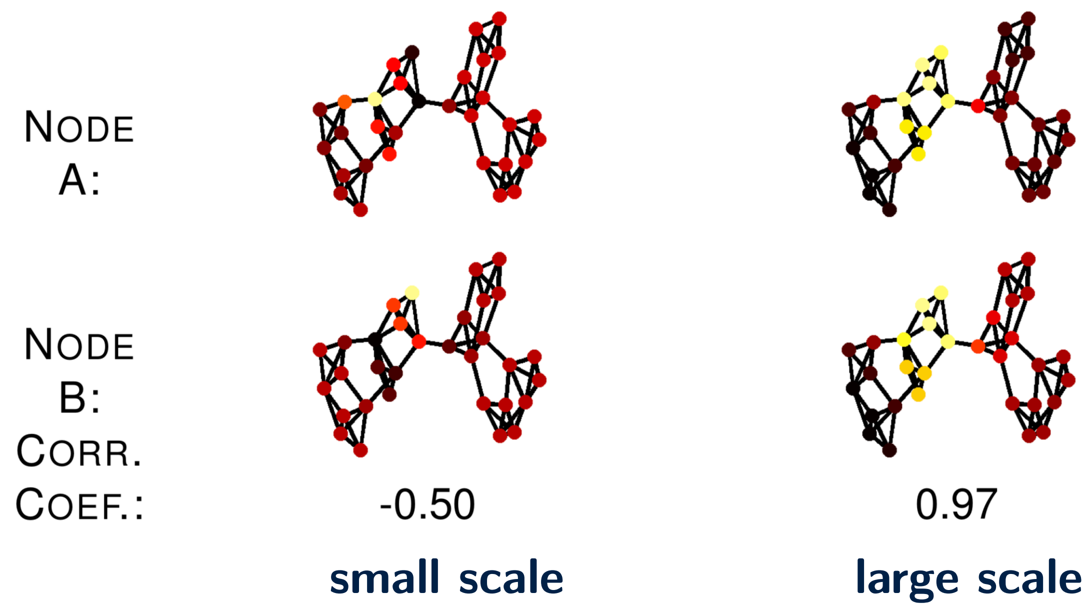
small scale **large scale**

Hammond et al., "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis, 2011.
Tremblay and Borgnat, "Graph wavelets for multiscale community mining," IEEE TSP, 2014.

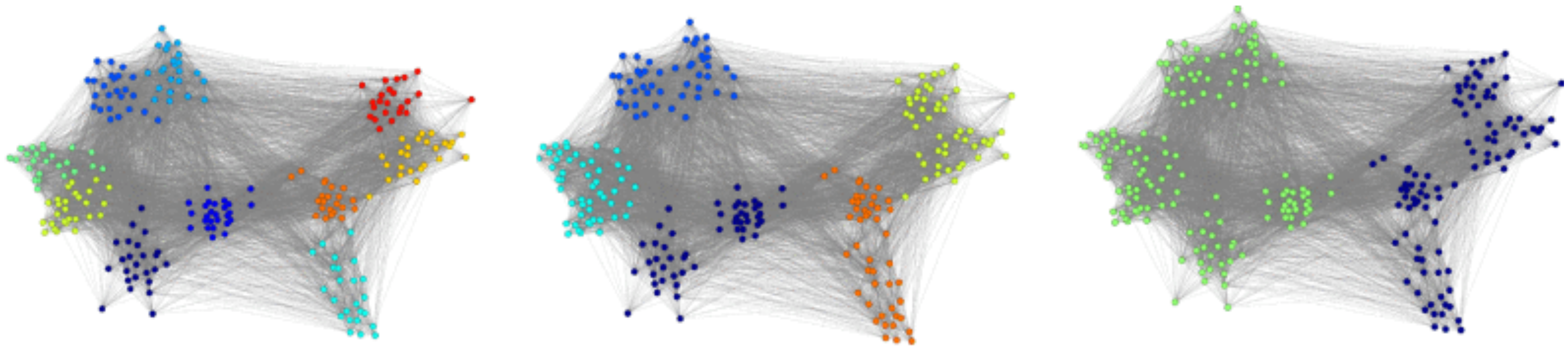
Application I: Community detection

spectral graph wavelets at different scales:

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

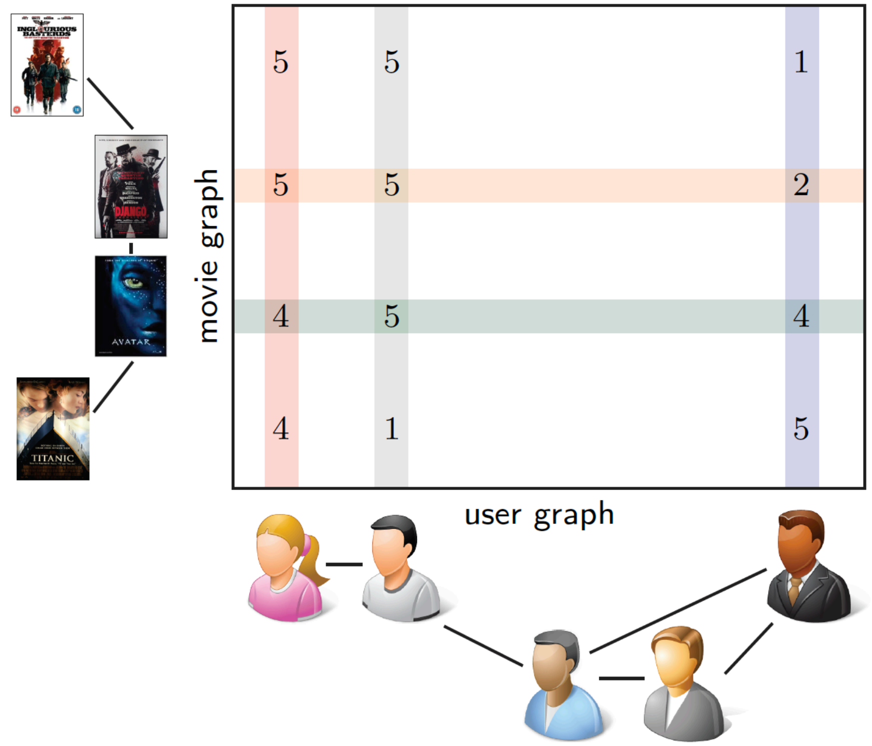


multi-scale community detection:



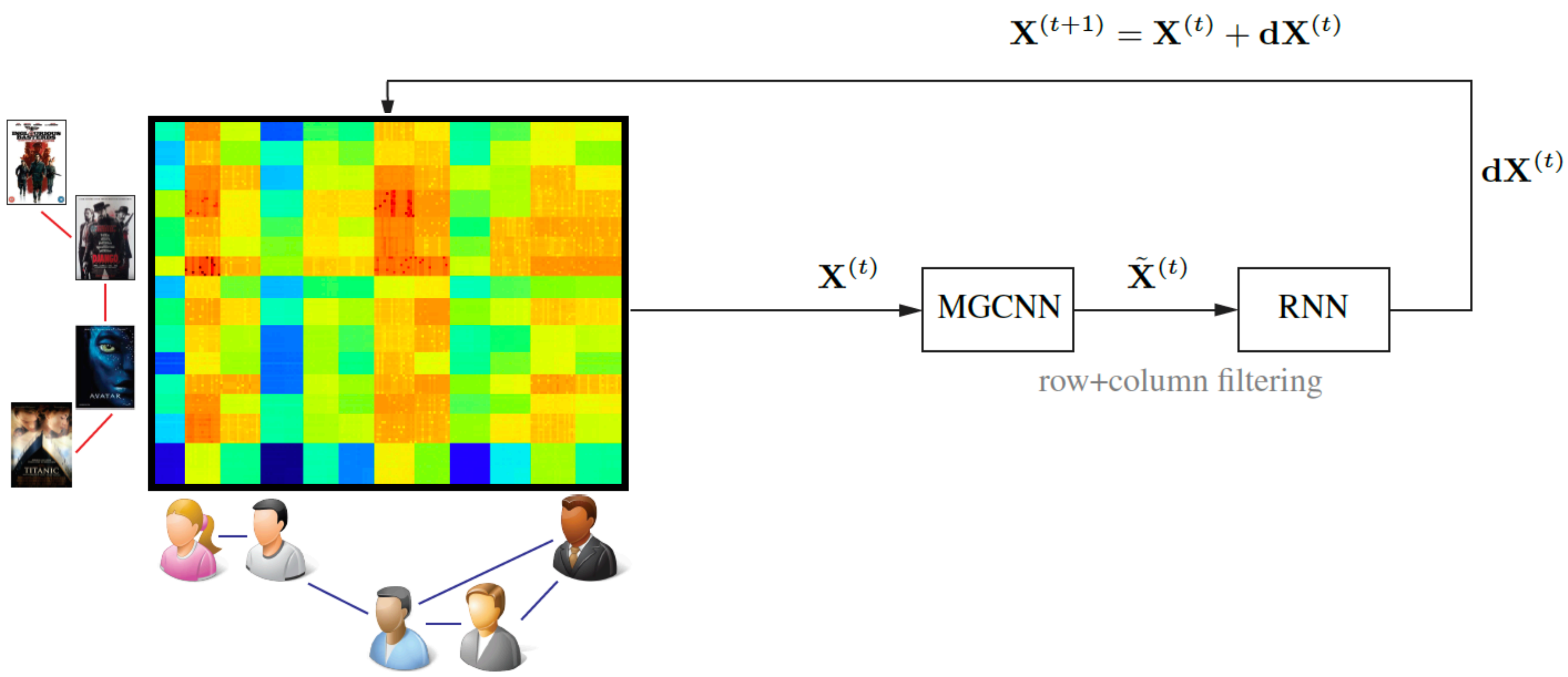
Hammond et al., "Wavelets on graphs via spectral graph theory," Applied and Computational Harmonic Analysis, 2011.
Tremblay and Borgnat, "Graph wavelets for multiscale community mining," IEEE TSP, 2014.

Application II: Recommender systems



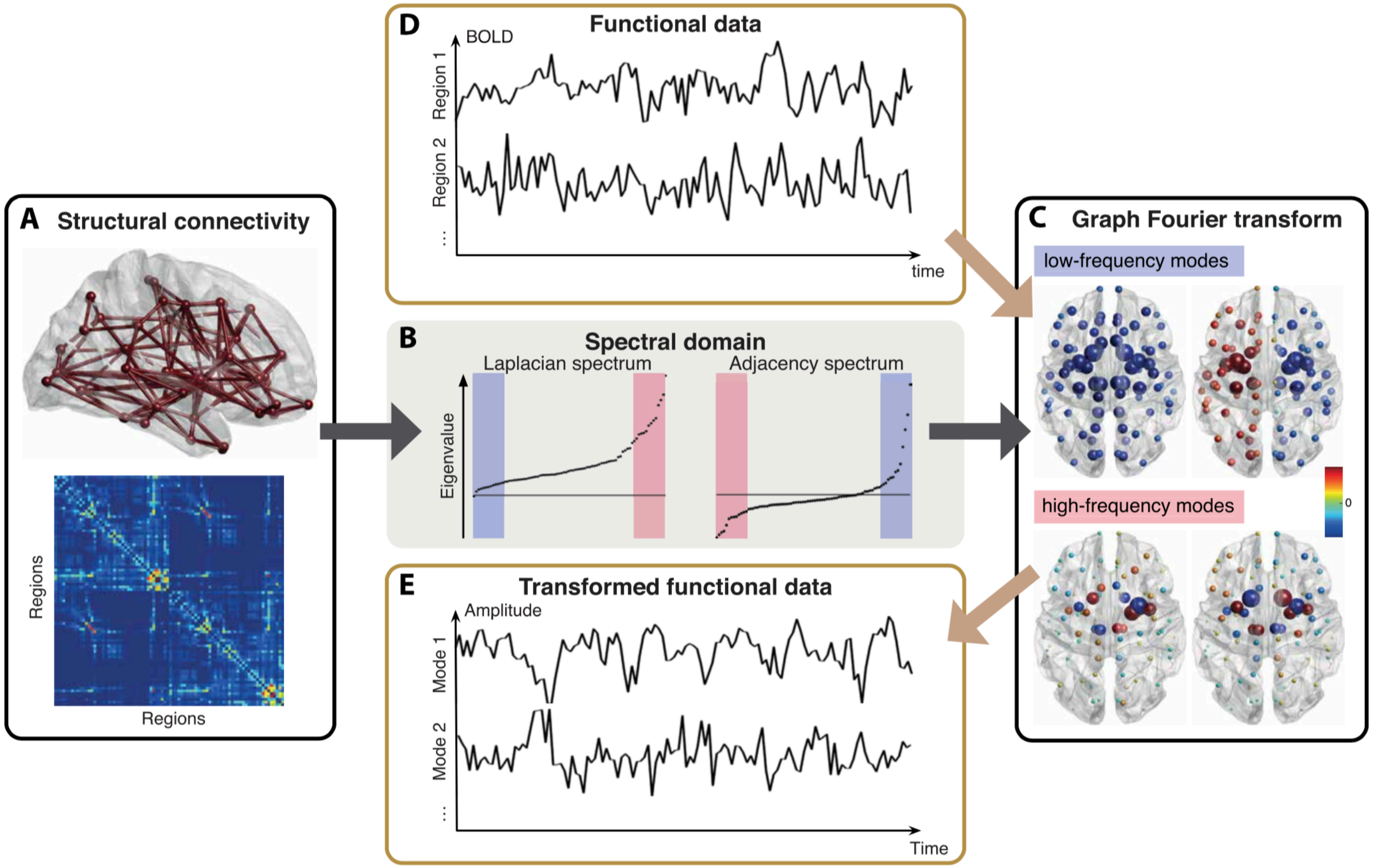
Monti et al., "Geometric matrix completion with recurrent multi-graph neural networks," NIPS, 2017.

Application II: Recommender systems



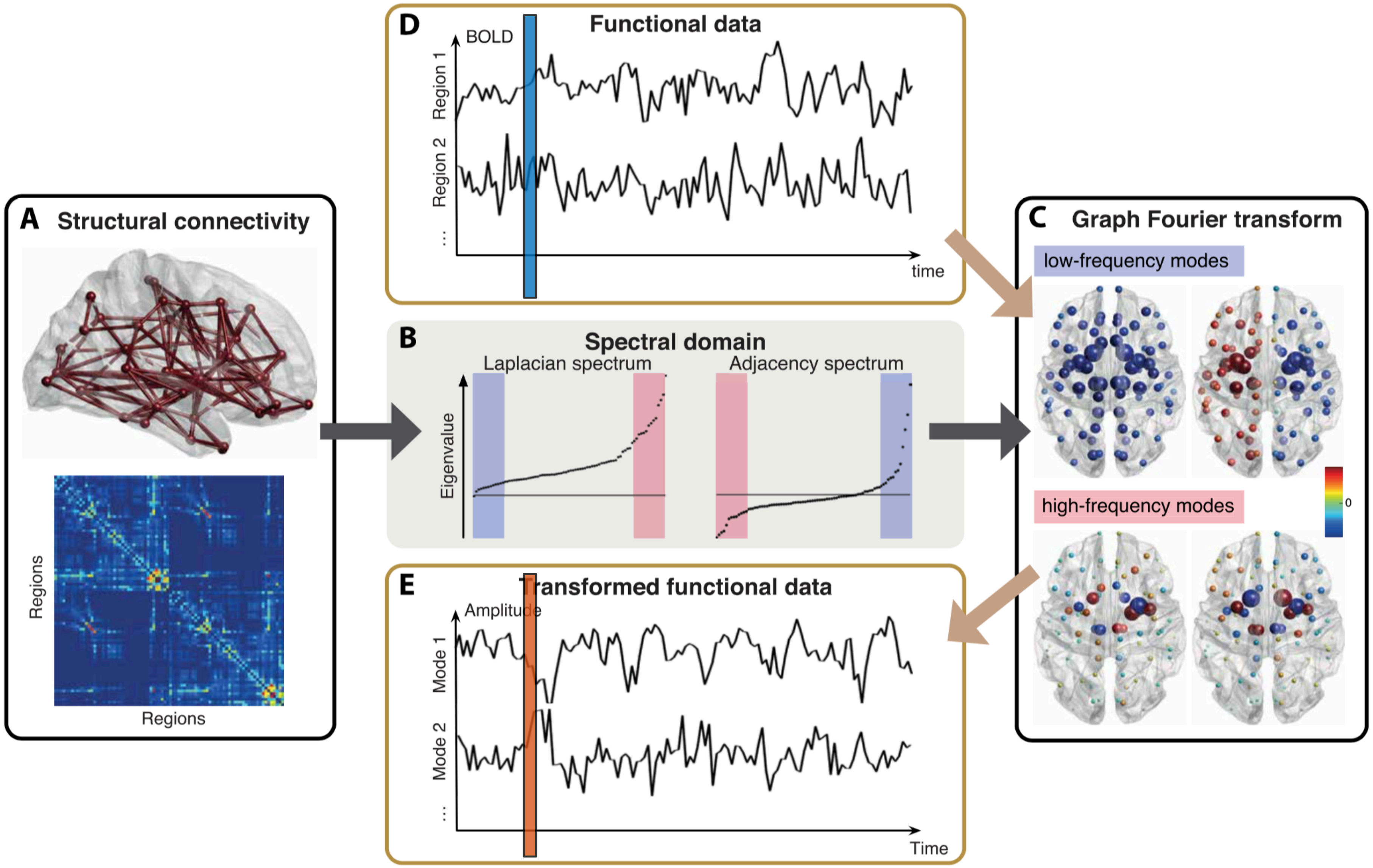
Monti et al., "Geometric matrix completion with recurrent multi-graph neural networks," NIPS, 2017.

Application III: Functional brain imaging



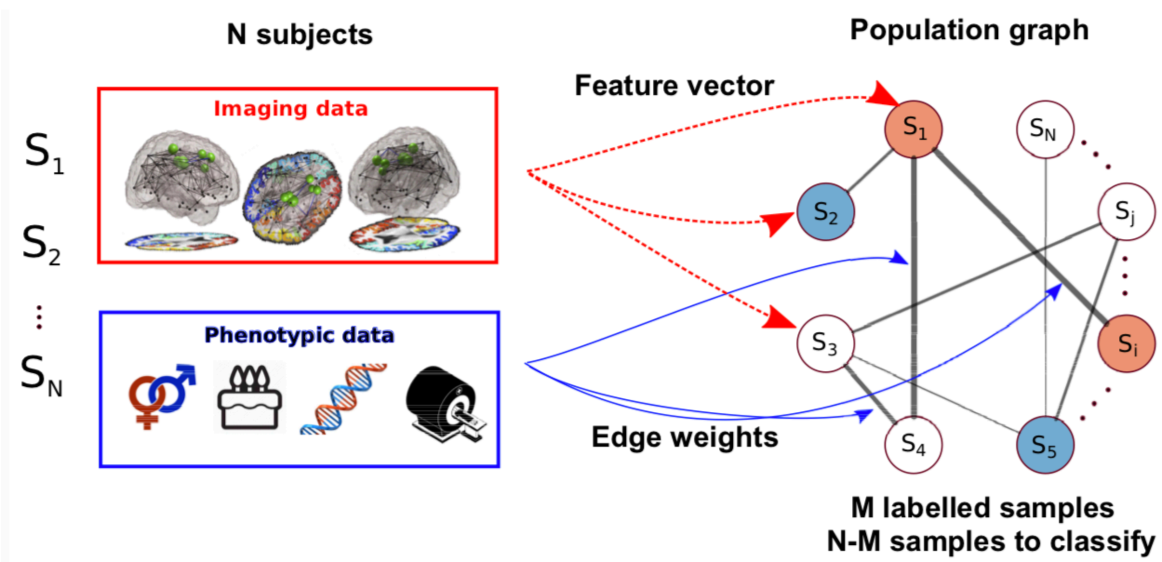
Huang et al., "A graph signal processing perspective on functional brain imaging," Proceedings of the IEEE, 2018.

Application III: Functional brain imaging



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

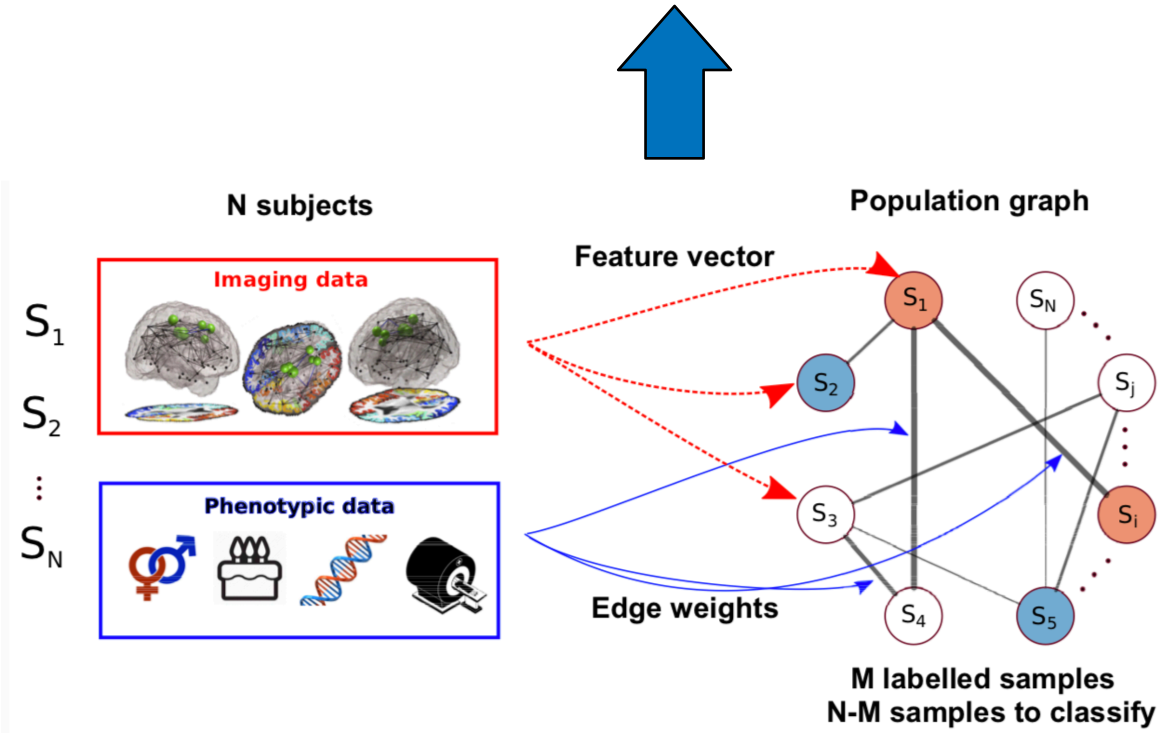
Application IV: Disease classification



Parisot et al., "Disease prediction using graph convolutional networks," Medical Image Analysis, 2018.

Application IV: Disease classification

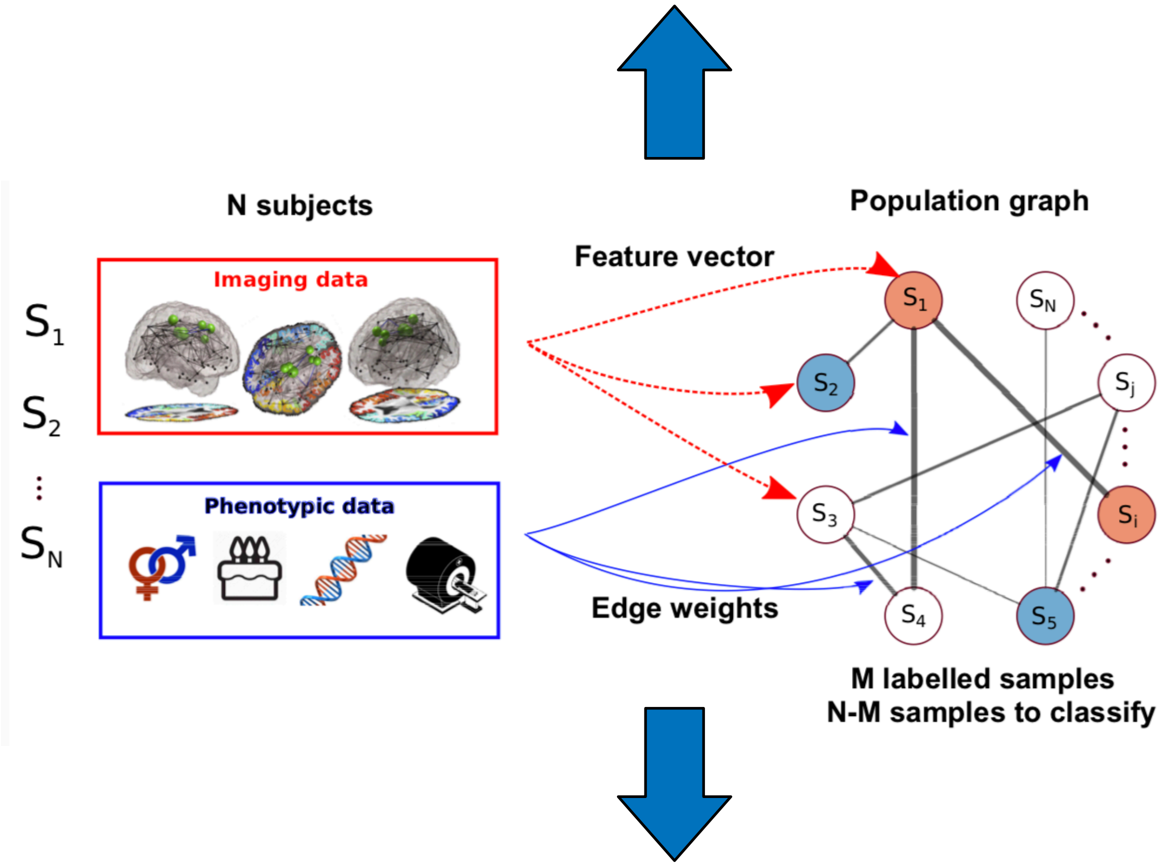
features extracted from brain analysis



Parisot et al., "Disease prediction using graph convolutional networks," Medical Image Analysis, 2018.

Application IV: Disease classification

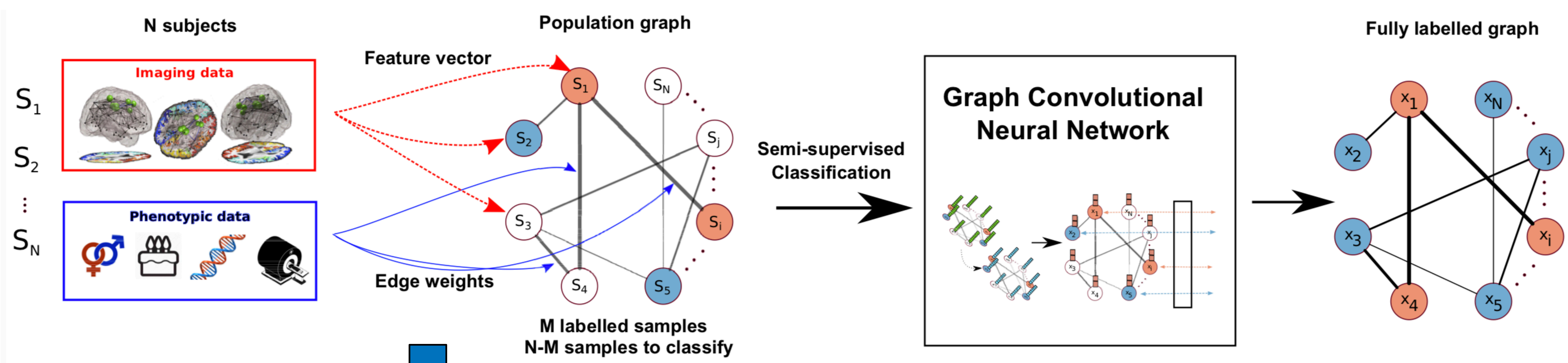
features extracted from brain analysis



similarity in phenotypic data

Application IV: Disease classification

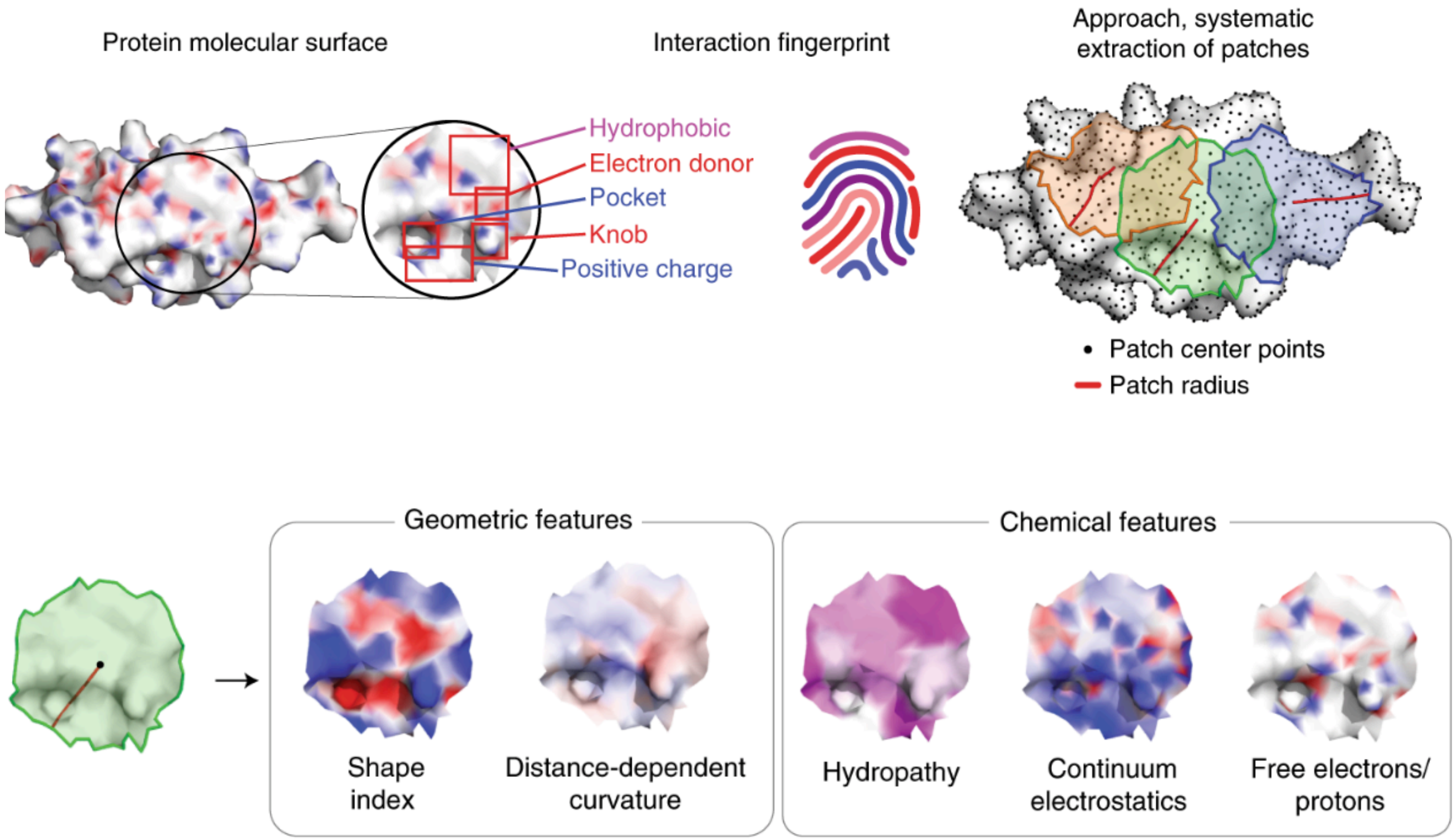
features extracted from brain analysis



similarity in phenotypic data

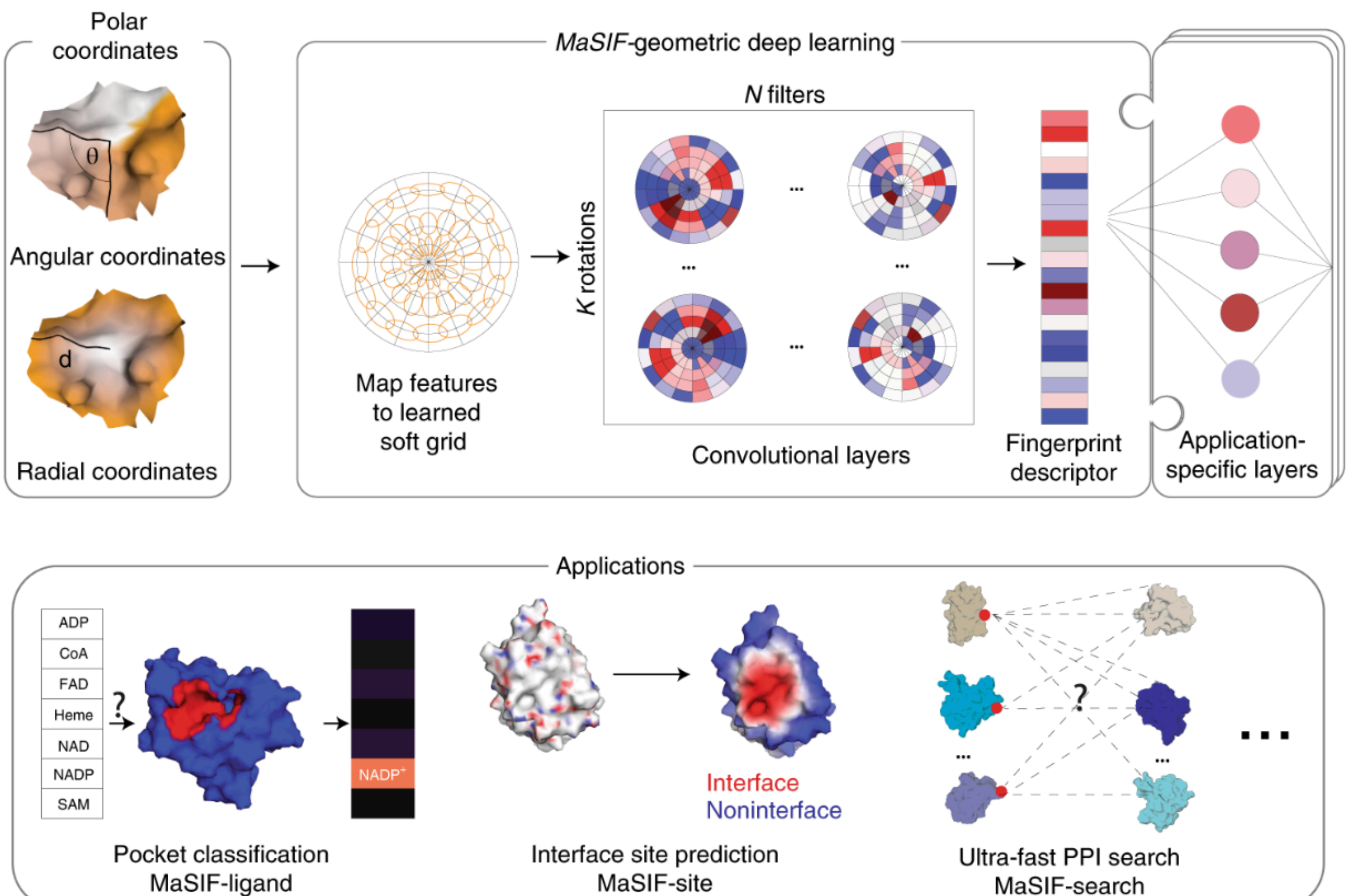
Parisot et al., "Disease prediction using graph convolutional networks," Medical Image Analysis, 2018.

Application V: Protein-protein interaction



Gainza et al., "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning," Nature Methods, 2020.

Application V: Protein-protein interaction

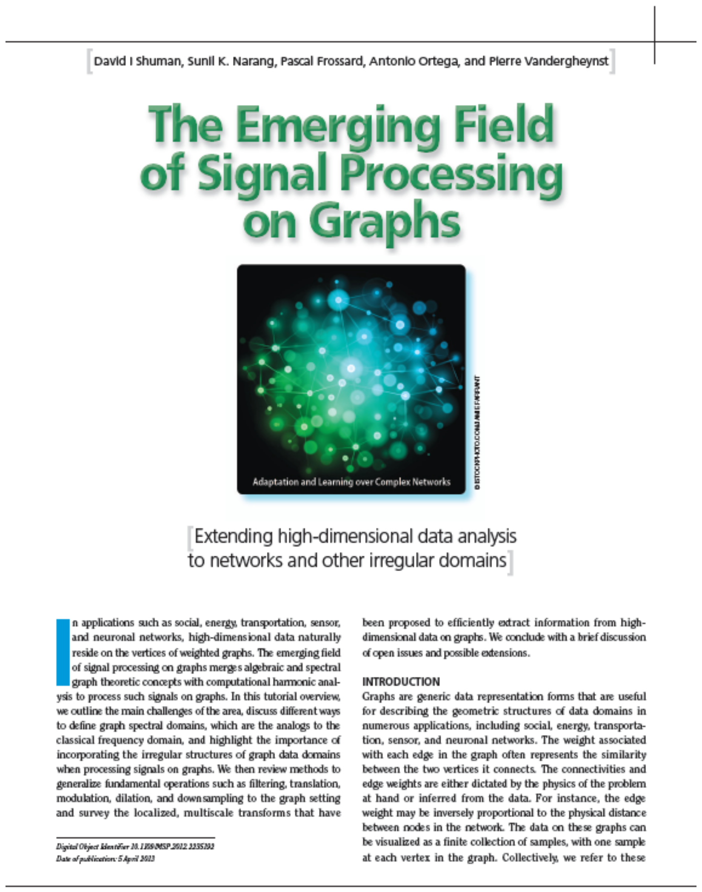


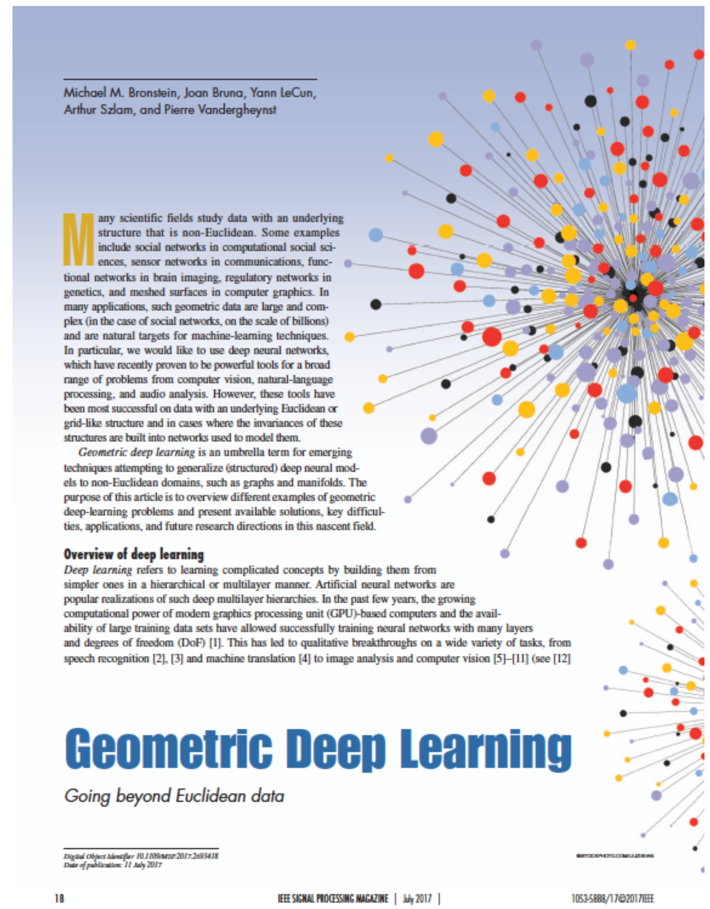
Gainza et al., "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning," Nature Methods, 2020.

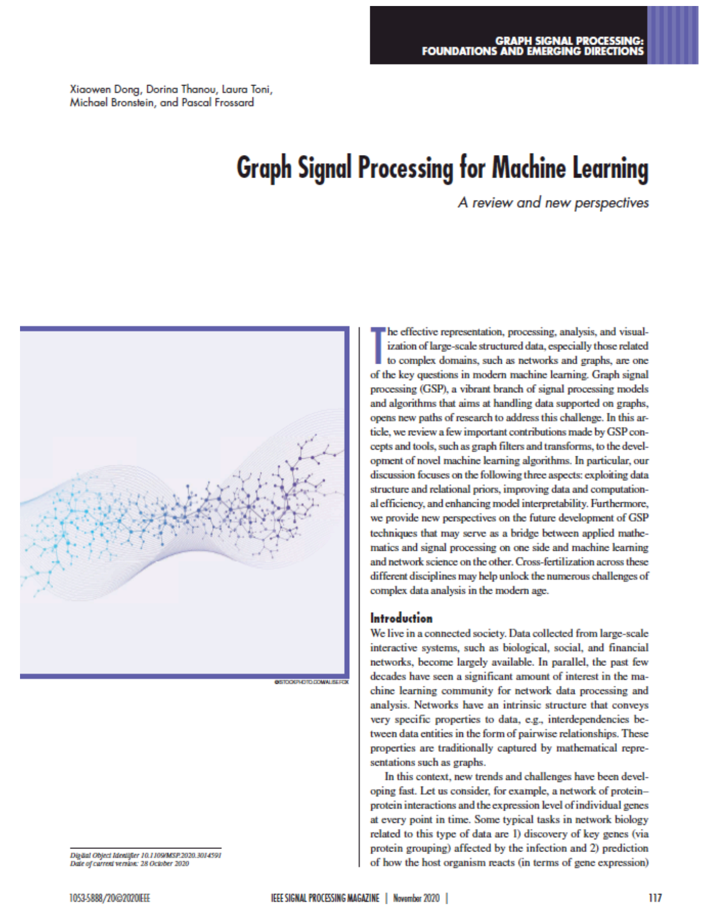
What's next?

- Mathematical models for graph-structured data
  - global and local smoothness & regularity
  - underlying physical processes
- Robustness & generalisation analysis
  - how robust is the model to topological change
  - how can the trained model be generalised to unseen graph
- Probabilistic interpretation
  - connection to Bayesian inference
  - Gaussian processes on graphs
- Learning graphs form data
  - dynamic graph construction
  - graph generative models
- Fast implementation

Papers & Resources







- http://www.robots.ox.ac.uk/~xdong/resource.html
- https://towardsdatascience.com/graph-deep-learning/home
- https://graphml.substack.com