

# Signal Processing and Machine Learning on Graphs

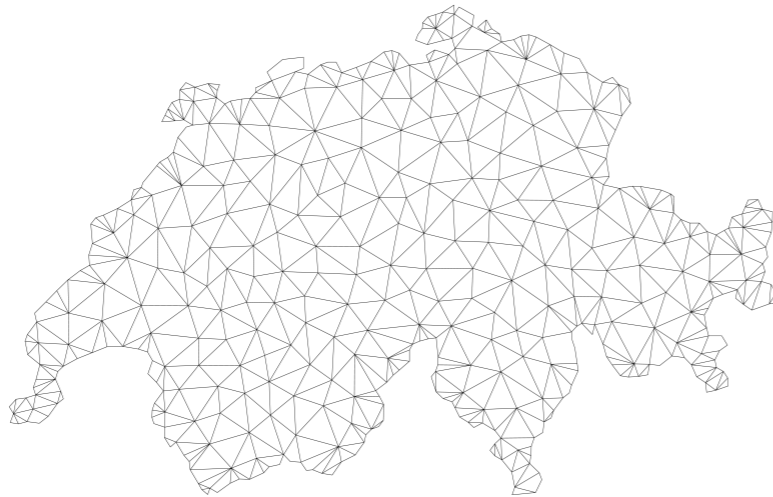
Xiaowen Dong

Department of Engineering Science  
University of Oxford

Summer School in Economic Networks  
Oxford, June 2019



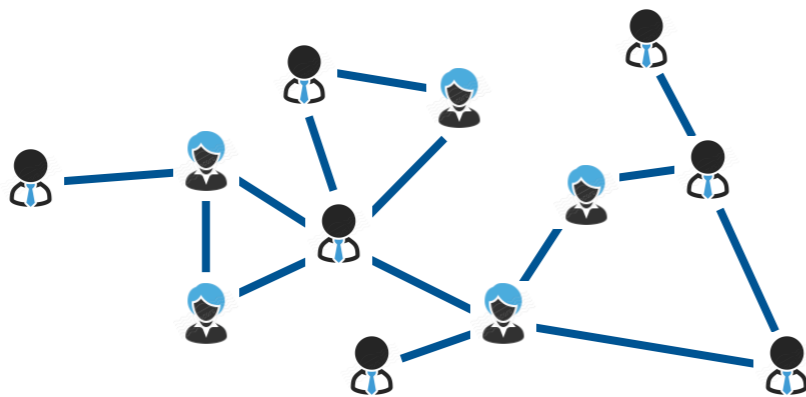
# Networks are everywhere



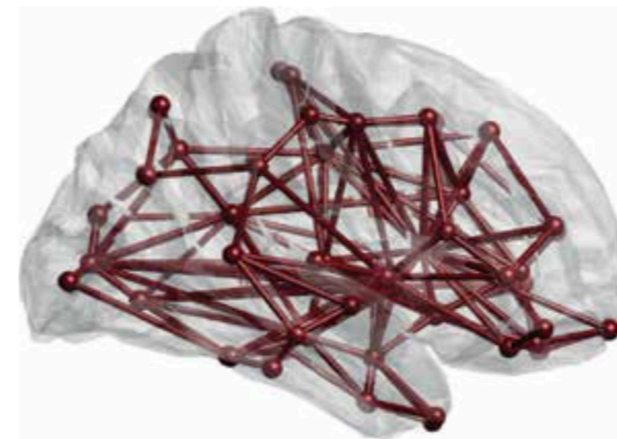
**geographical network**



**traffic network**

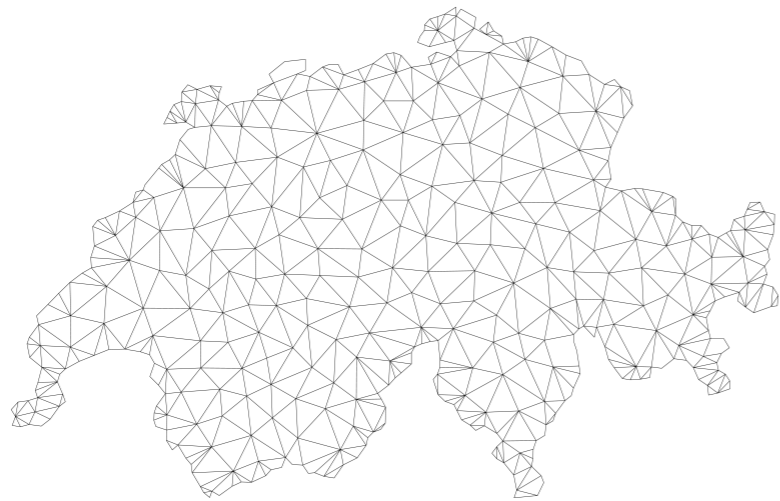


**social network**



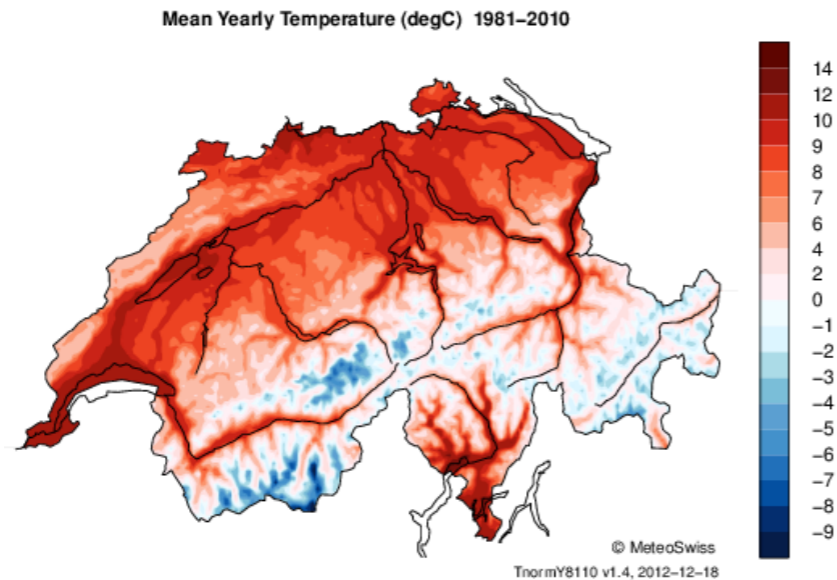
**brain network [Huang18]**

# Network-structured data are everywhere



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

# Network-structured data are everywhere



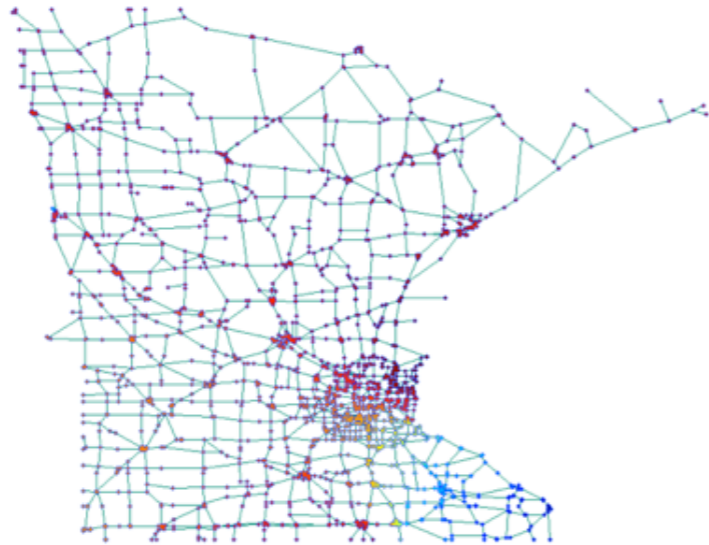
- vertices
  - geographical regions
- edges
  - geographical proximity between regions
- signal
  - temperature records in these regions

# Network-structured data are everywhere



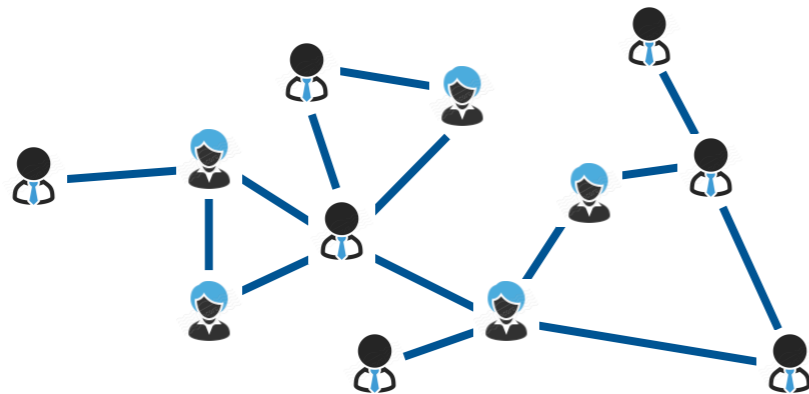
- vertices
  - road junctions
- edges
  - road network

# Network-structured data are everywhere



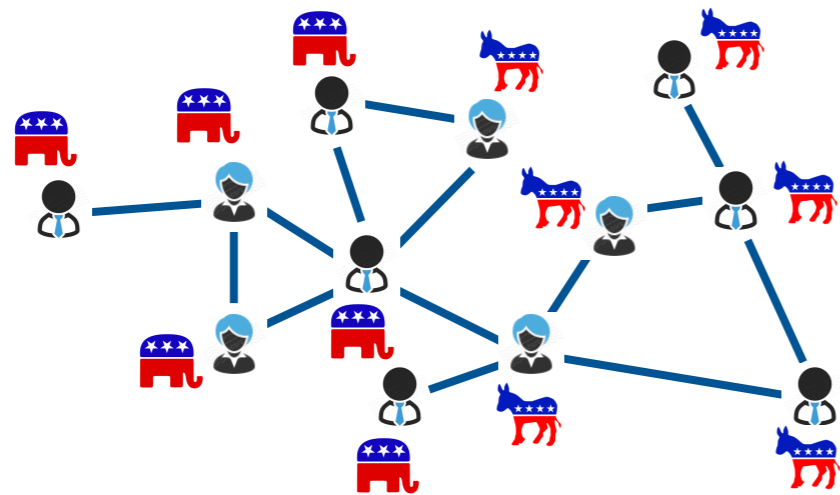
- vertices
  - road junctions
- edges
  - road network
- signal
  - traffic congestion at junctions

# Network-structured data are everywhere



- vertices
  - individuals
- edges
  - friendship between individuals

# Network-structured data are everywhere



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

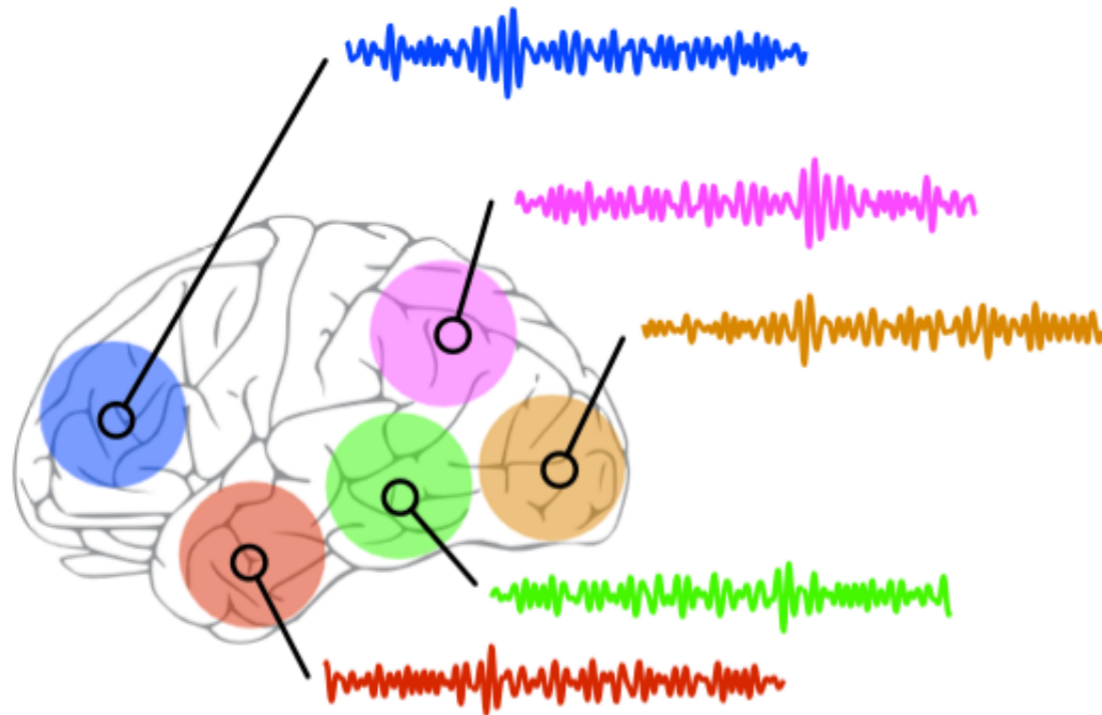


# Network-structured data are everywhere



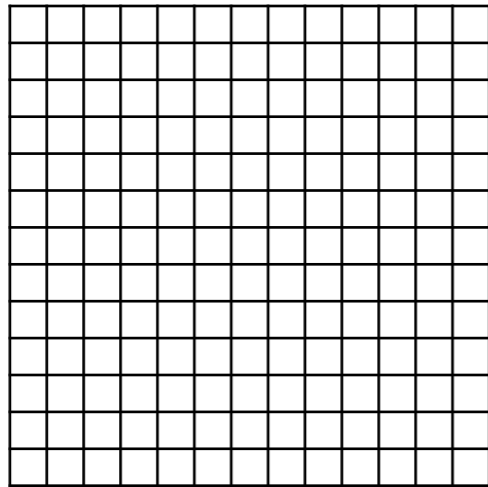
- vertices
  - brain regions
- edges
  - structural connectivity between brain regions

# Network-structured data are everywhere



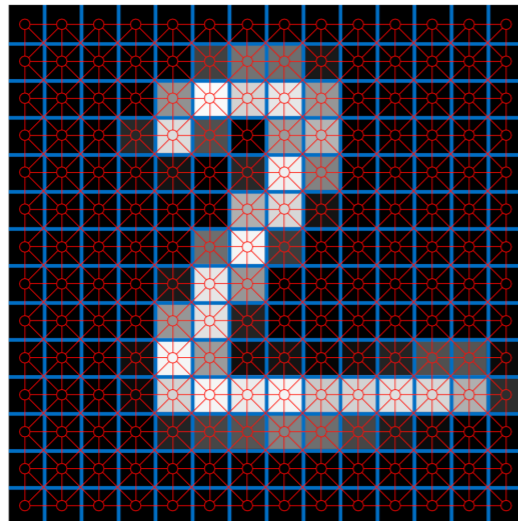
- vertices
  - brain regions
- edges
  - structural connectivity between brain regions
- signal
  - blood-oxygen-level-dependent (BOLD) time series

# Network-structured data are everywhere



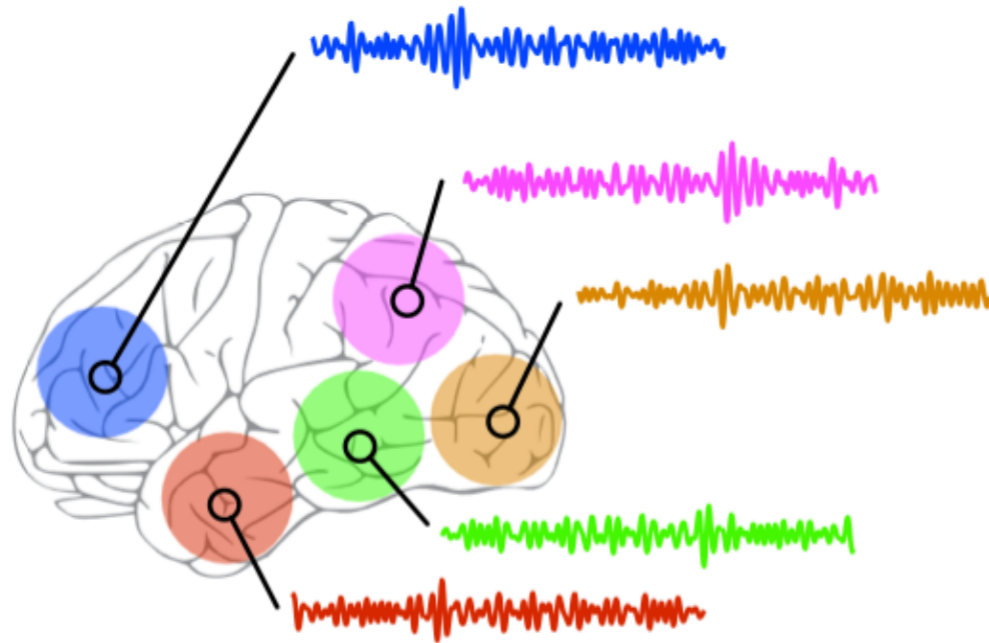
- vertices
  - pixels
- edges
  - spatial proximity between pixels

# Network-structured data are everywhere



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

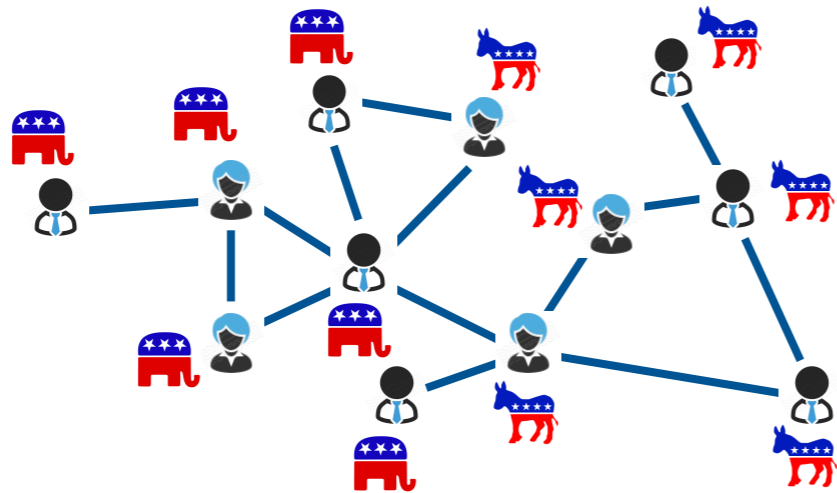
# Learning with network-structured data



medical condition?  
no medical condition?

**network-wise classification**

# Learning with network-structured data



democratic?  
republican?

**vertex-wise classification**

# Learning with network-structured data



**denoising and inpainting (inspired by image processing)**

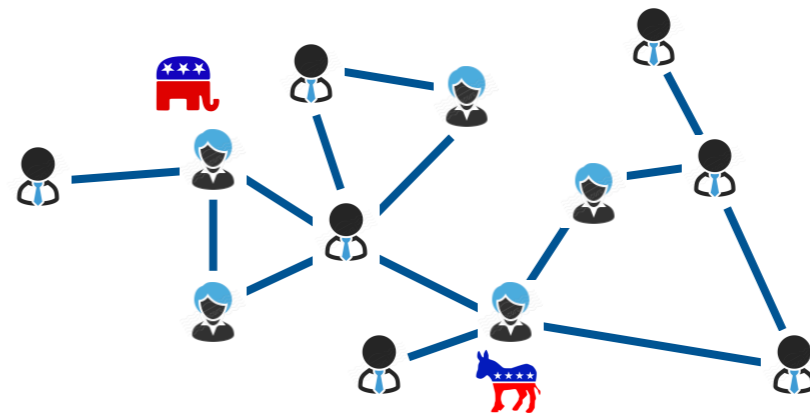
# Learning with network-structured data



**denoising and inpainting (inspired by image processing)**

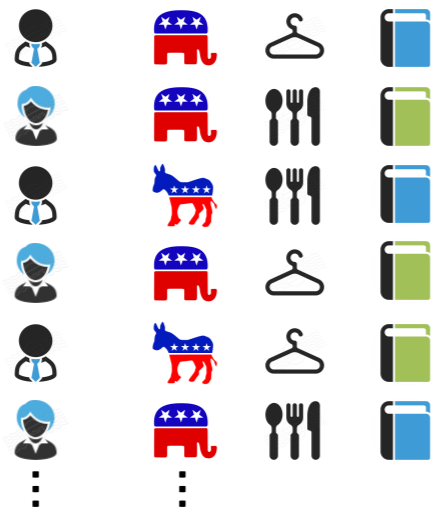


# Learning with network-structured data



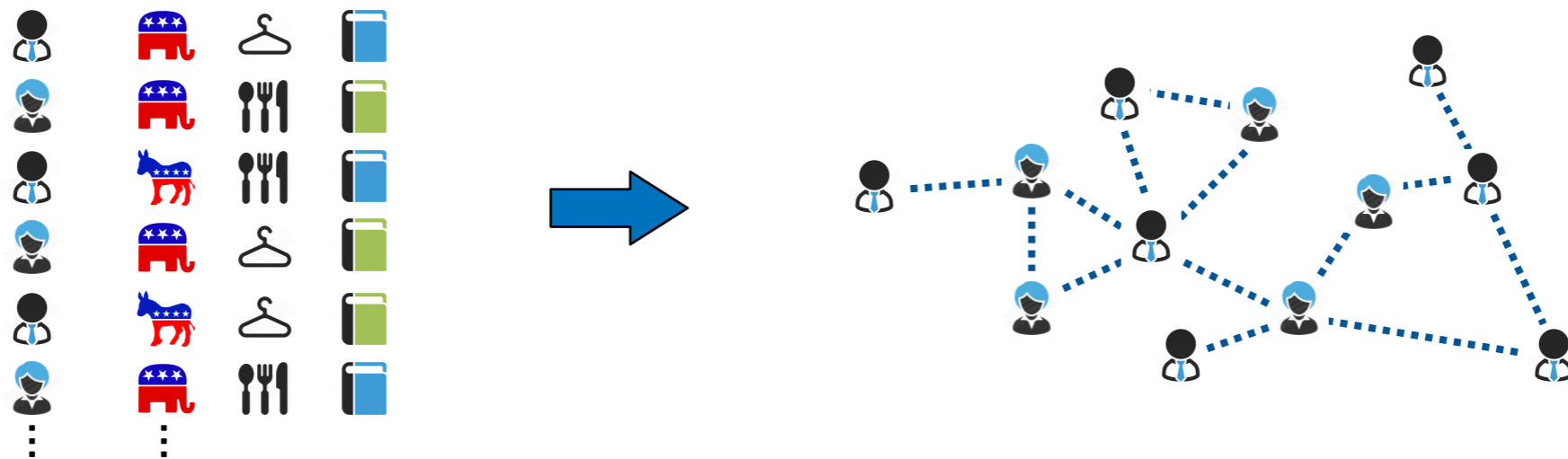
denoising and inpainting (inspired by image processing)

# Learning with network-structured data



inferring network structure from data

# Learning with network-structured data



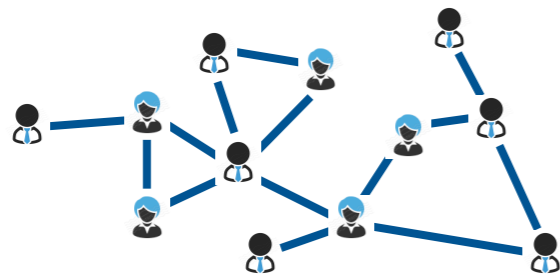
inferring network structure from data

# How to incorporate network into learning?

- Straightforward approach: embed the network into a Euclidean space

# How to incorporate network into learning?

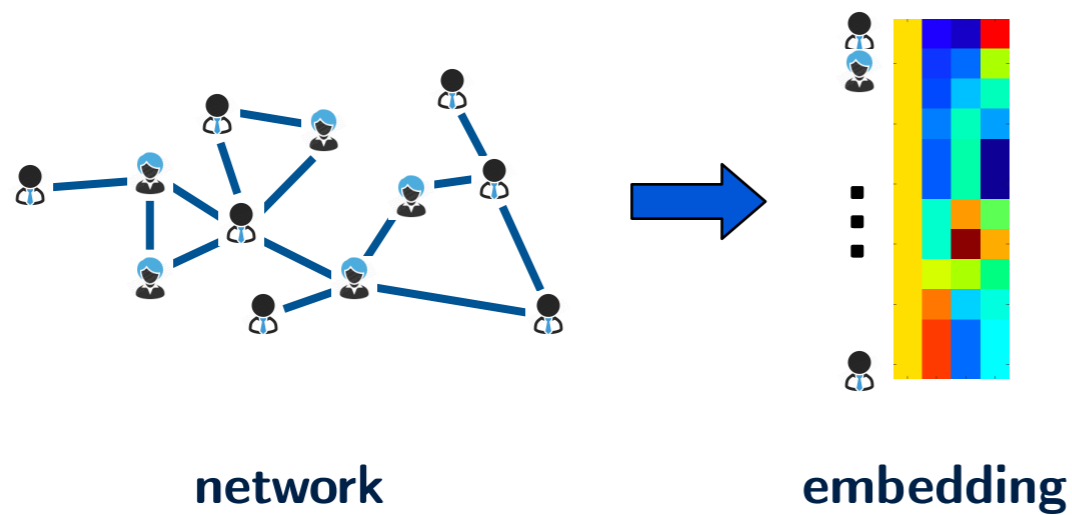
- Straightforward approach: embed the network into a Euclidean space



network

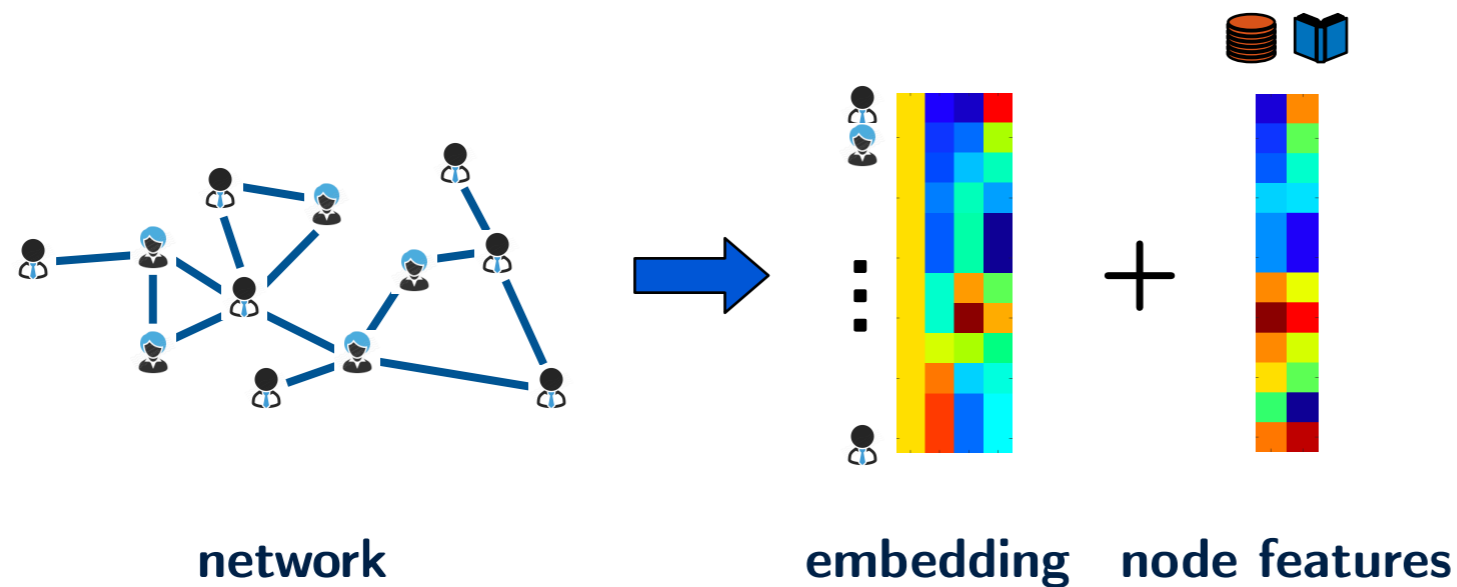
# How to incorporate network into learning?

- Straightforward approach: embed the network into a Euclidean space



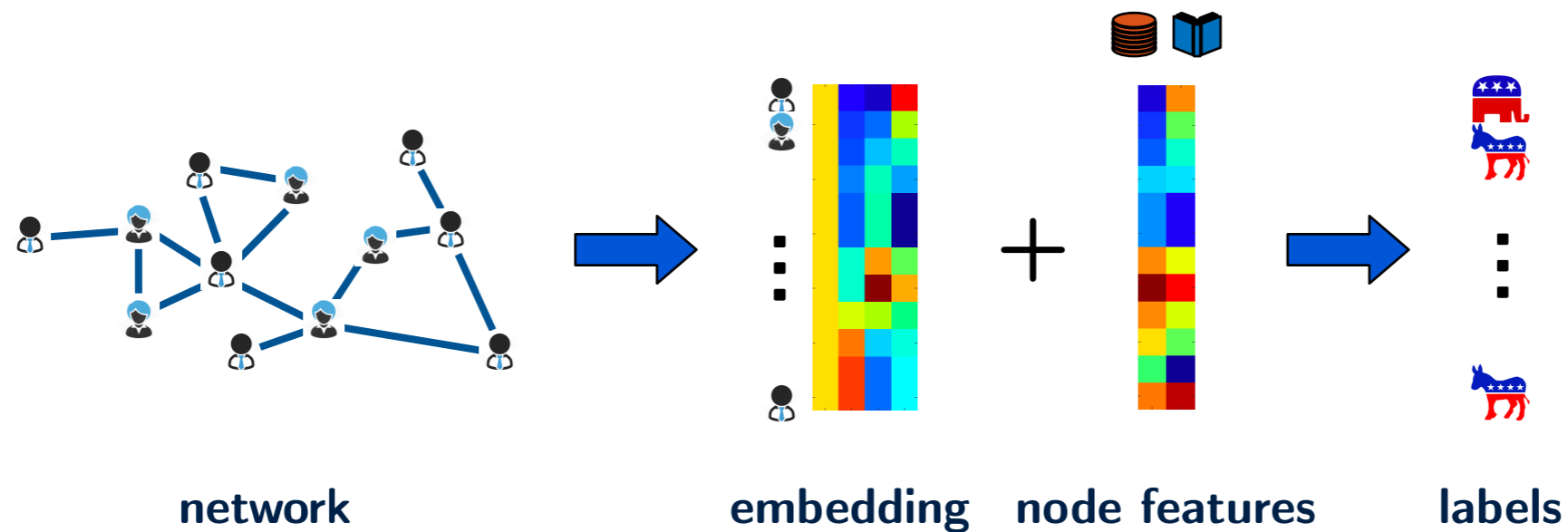
# How to incorporate network into learning?

- Straightforward approach: embed the network into a Euclidean space



# How to incorporate network into learning?

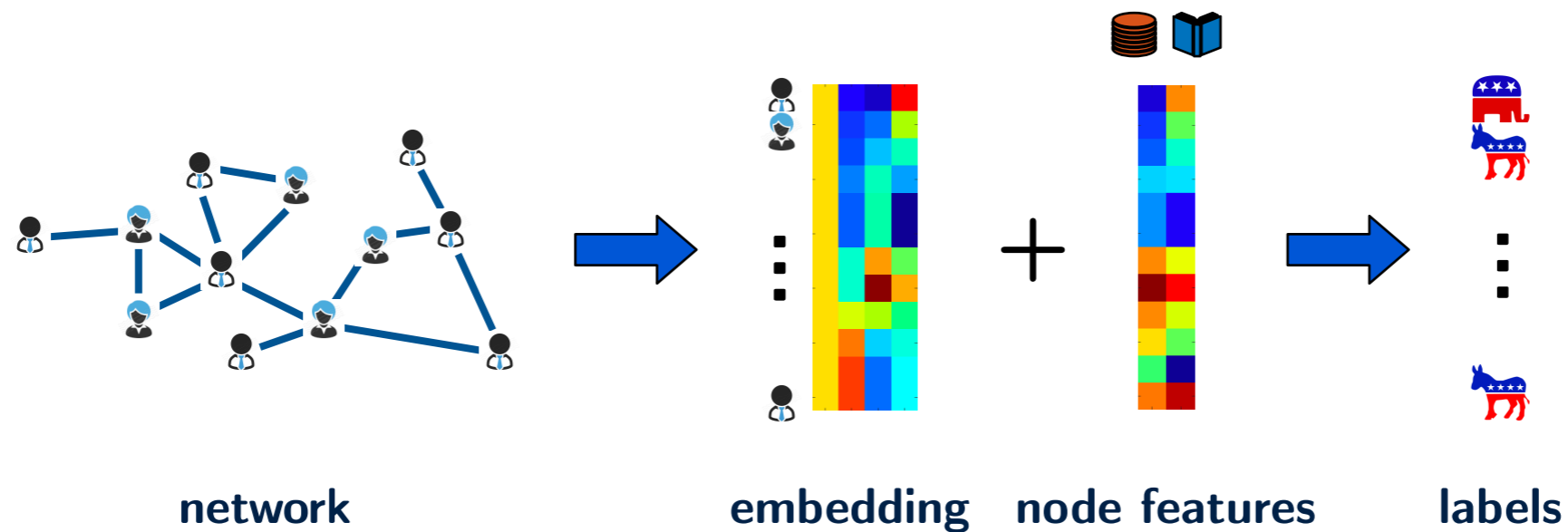
- Straightforward approach: embed the network into a Euclidean space





# How to incorporate network into learning?

- Straightforward approach: embed the network into a Euclidean space



- embedding of network structure leads to information loss
- need for new models & tools that directly incorporate structure in data analysis

# Outline

- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Connection with literature
- Research topics inspired by GSP
- Applications

# Outline

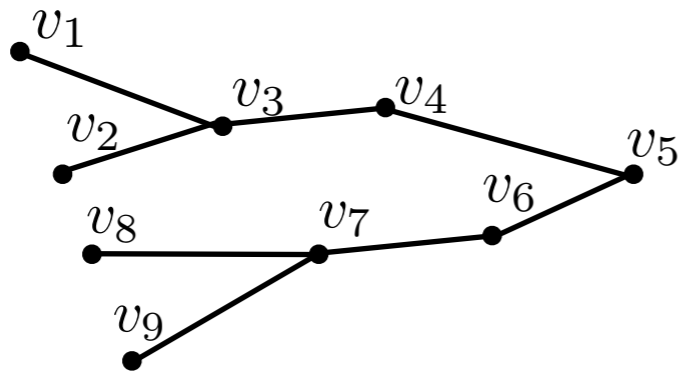
- Graph signal processing (GSP): Basic concepts
- Spectral filtering: Basic tools of GSP
- Connection with literature
- Research topics inspired by GSP
- Applications

# Graph signal processing

- Network-structured data can be represented by graph signals

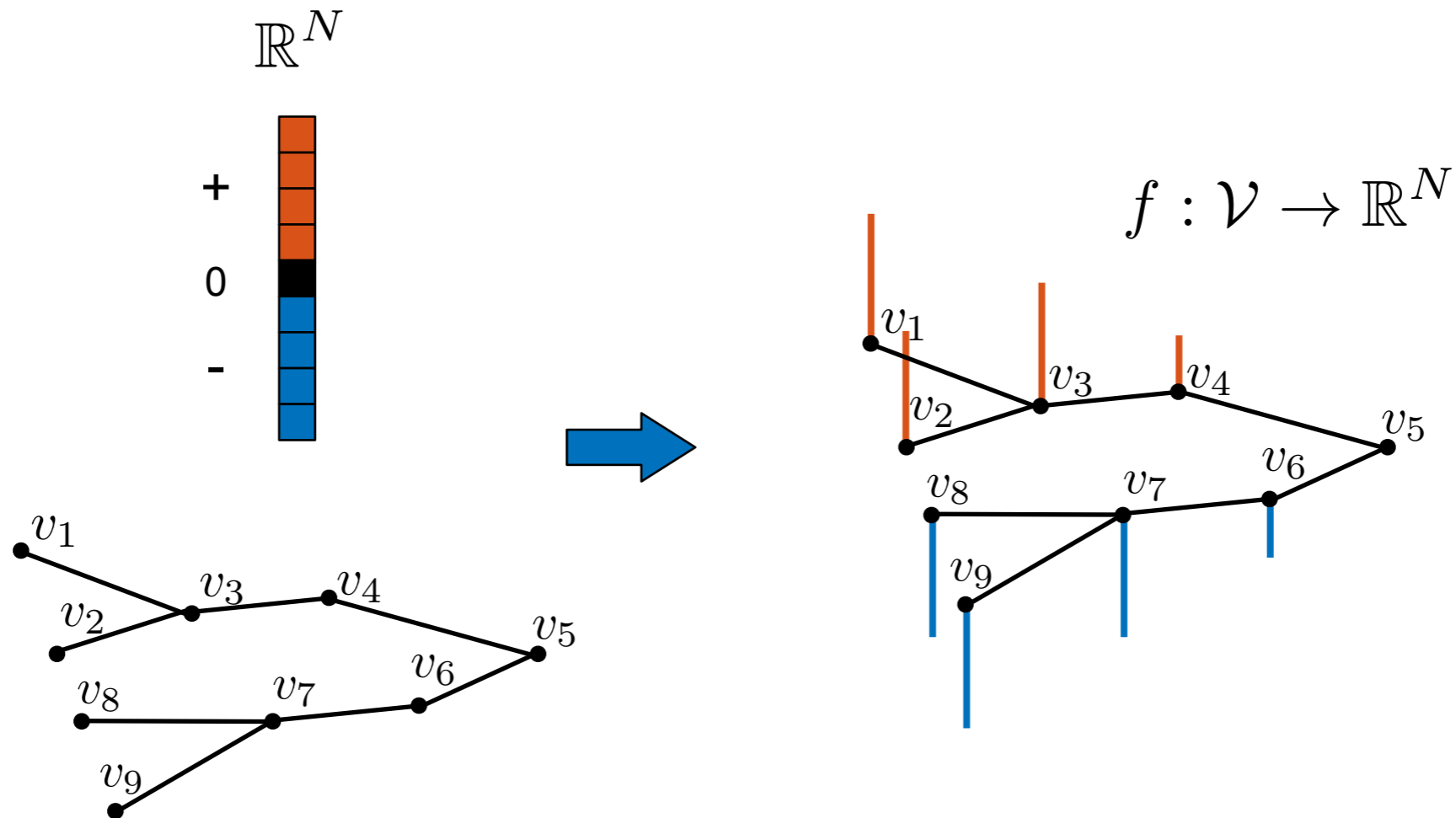
# Graph signal processing

- Network-structured data can be represented by graph signals



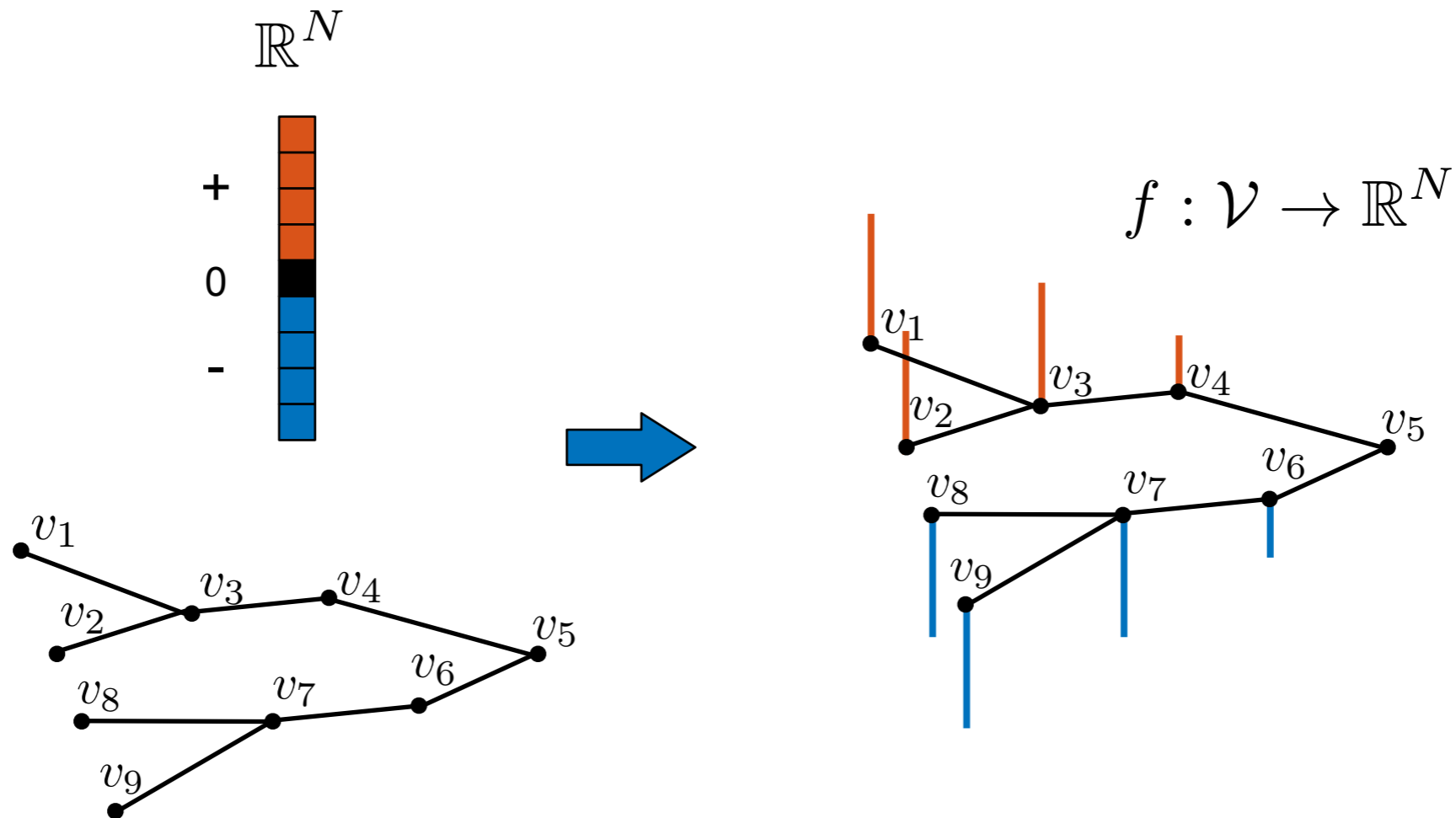
# Graph signal processing

- Network-structured data can be represented by graph signals



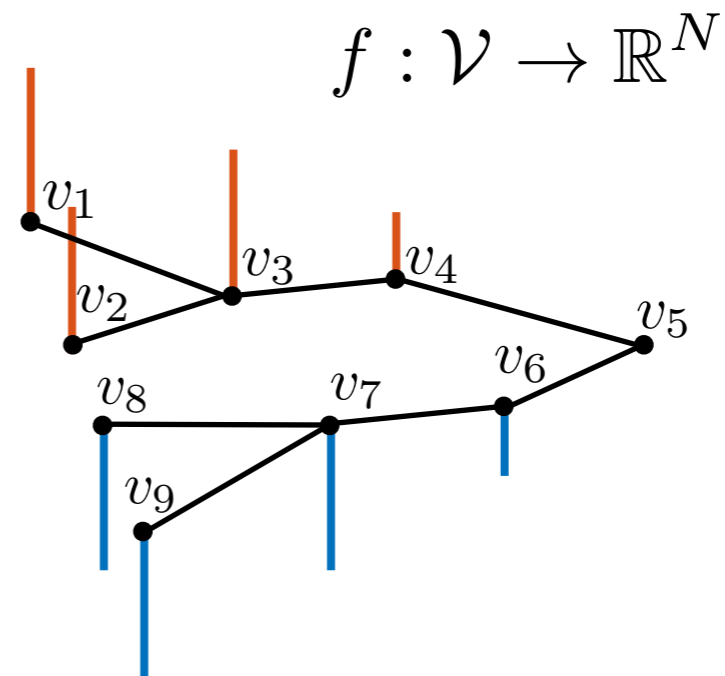
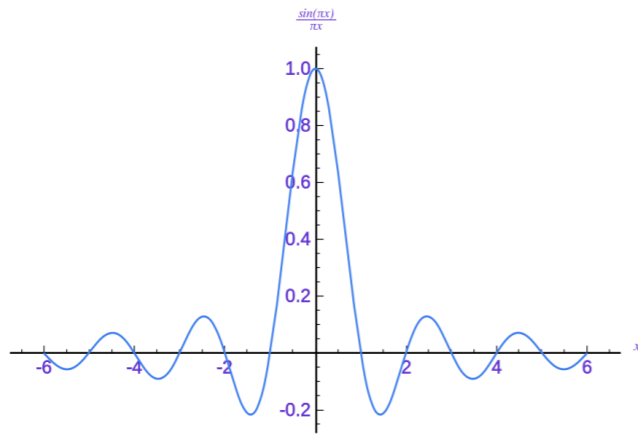
# Graph signal processing

- Network-structured data can be represented by graph signals



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

# Graph signal processing

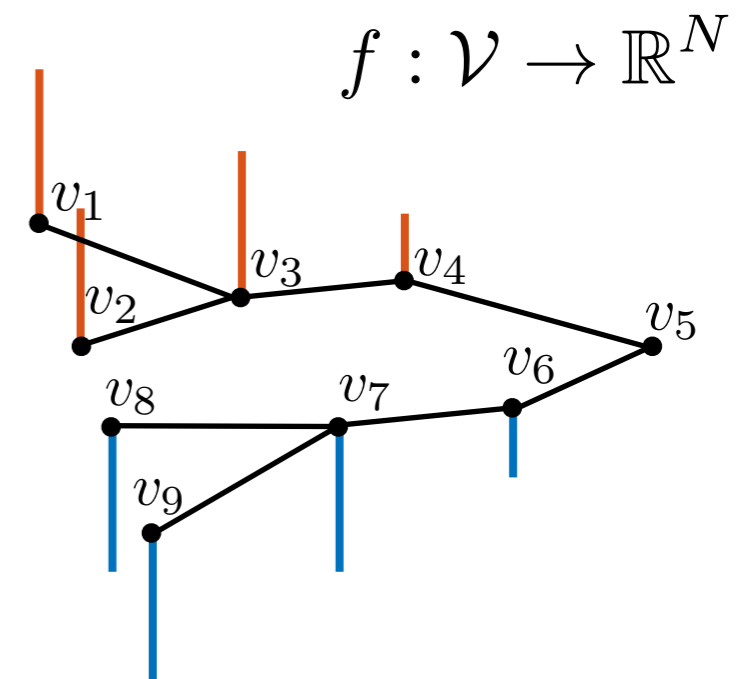


**how to generalise classical signal processing tools  
on irregular domains such as graphs?**



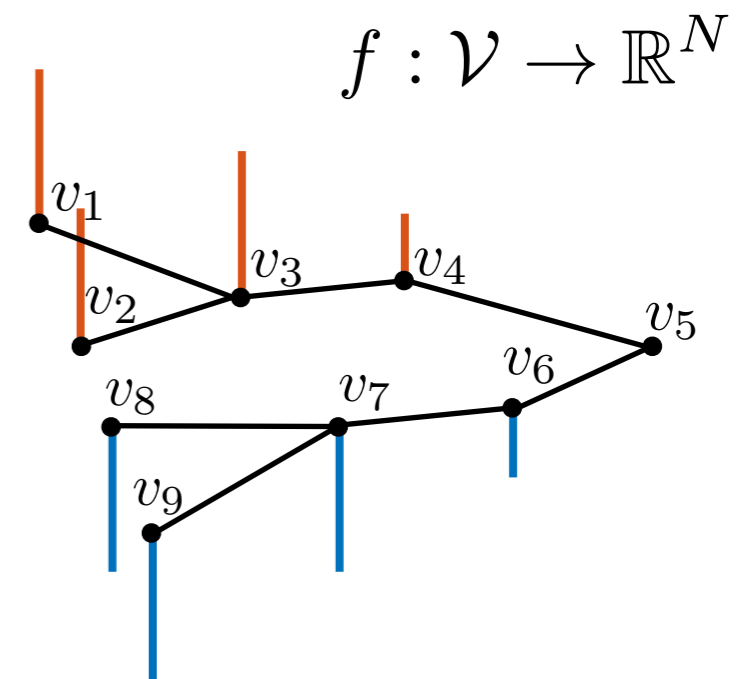
# Graph signal processing

- 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



# Two paradigms

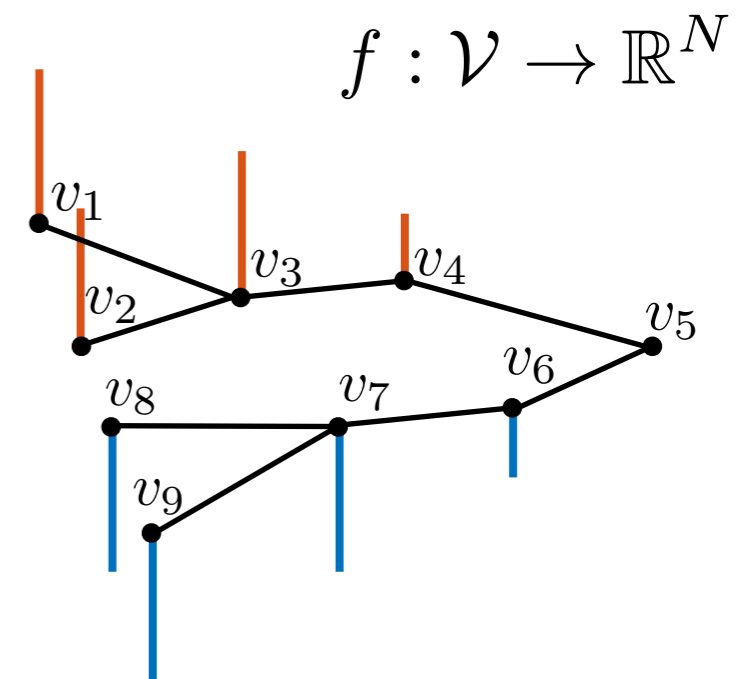
- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs



# Two paradigms

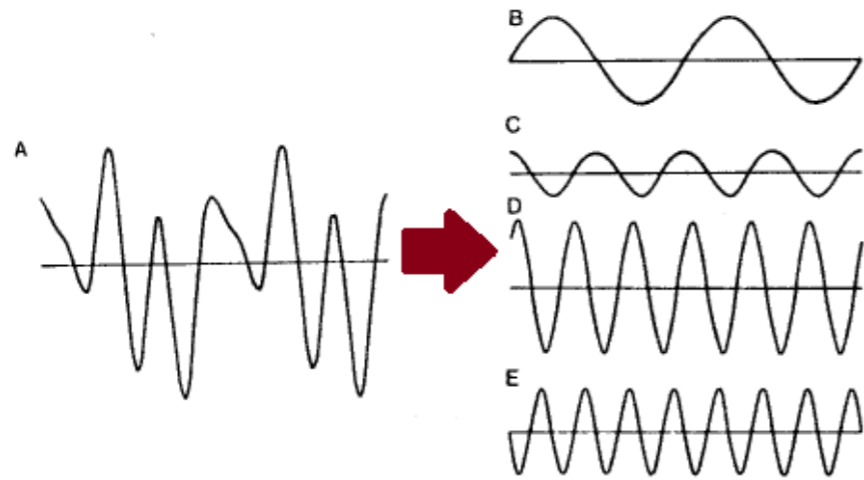
- Main GSP approaches can be categorised into two families:
  - vertex (spatial) domain designs
  - frequency (graph spectral) domain designs

**important for analysis of signal properties**



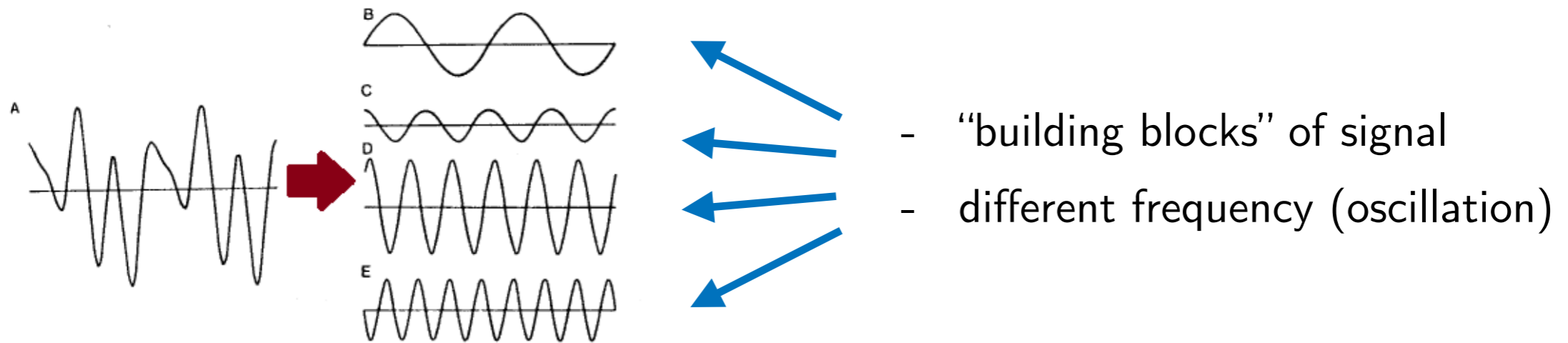
# Need for notion of frequency

- Classical Fourier transform provides frequency domain representation of signals



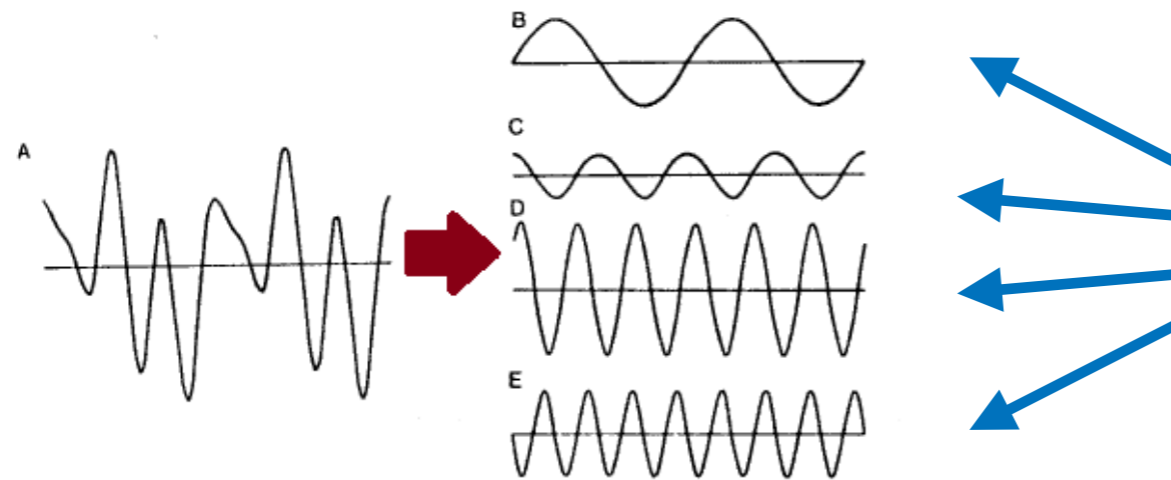
# Need for notion of frequency

- Classical Fourier transform provides frequency domain representation of signals



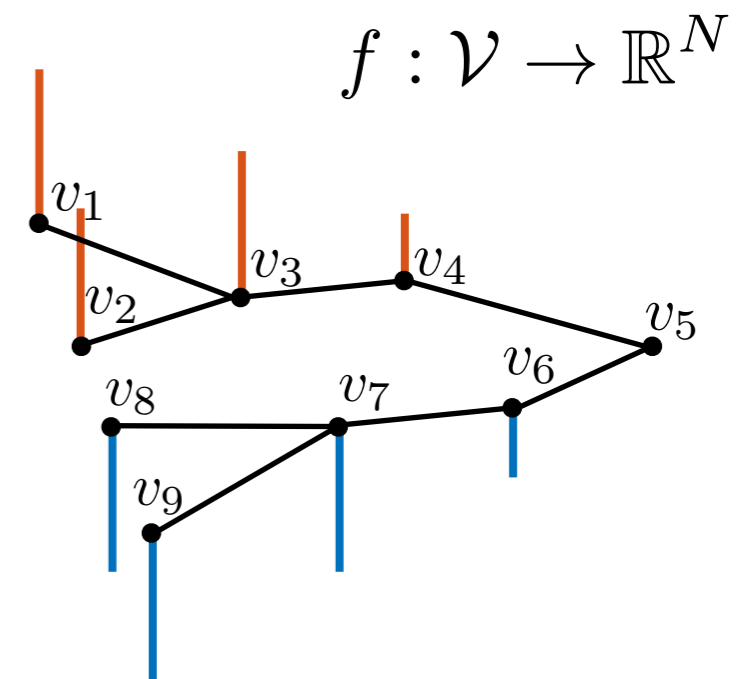
# Need for notion of frequency

- Classical Fourier transform provides frequency domain representation of signals



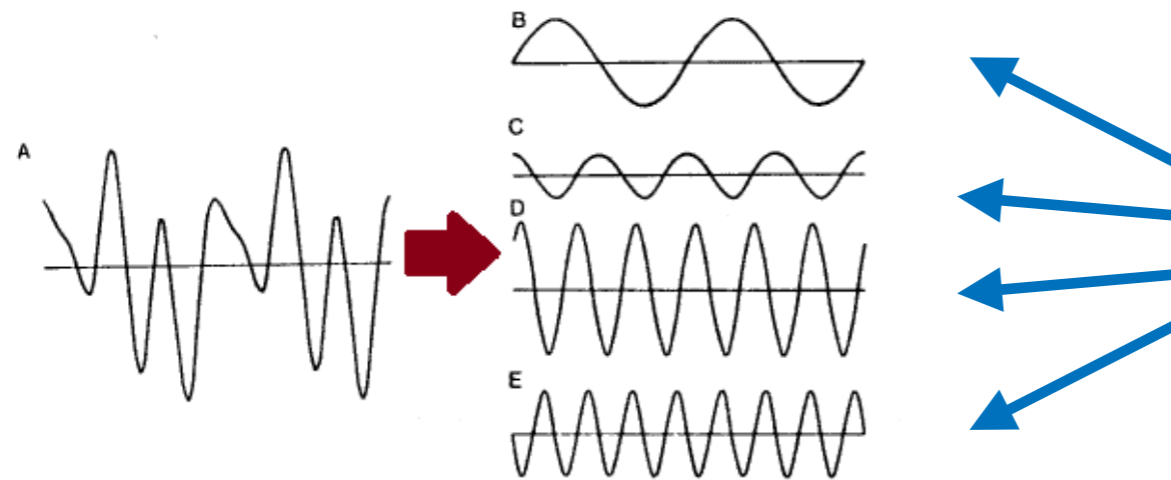
- “building blocks” of signal
- different frequency (oscillation)

- What about a notion of frequency for graph signals?



# Need for notion of frequency

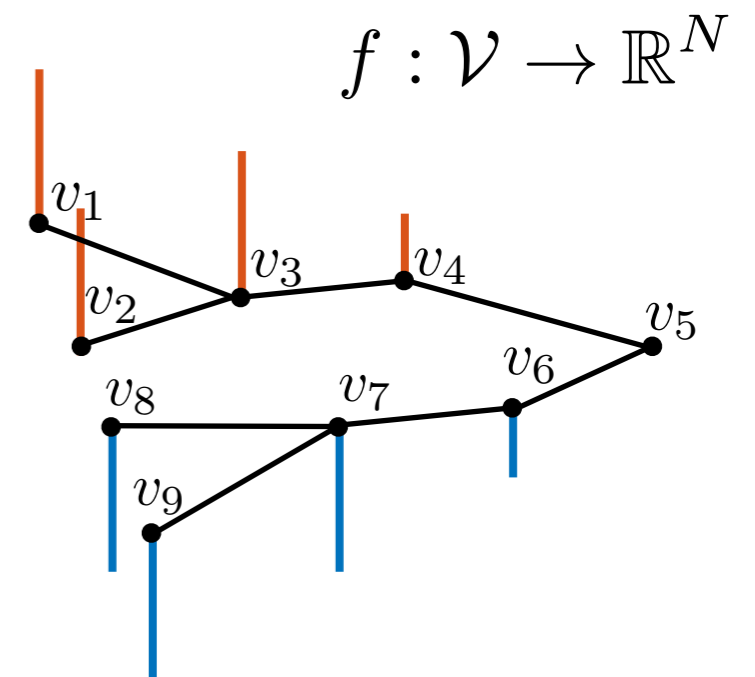
- Classical Fourier transform provides frequency domain representation of signals



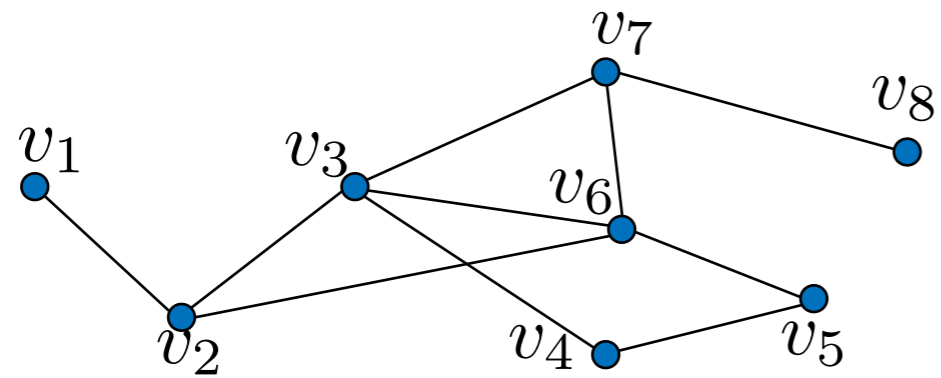
- “building blocks” of signal
- different frequency (oscillation)

- What about a notion of frequency for graph signals?

**we need the graph Laplacian matrix**



# Graph Laplacian



weighted and undirected graph:

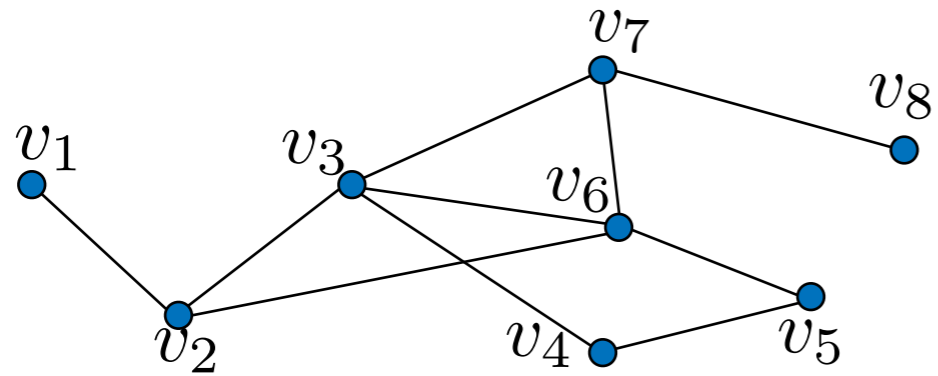
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

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

$W$



# Graph Laplacian



weighted and undirected graph:

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

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

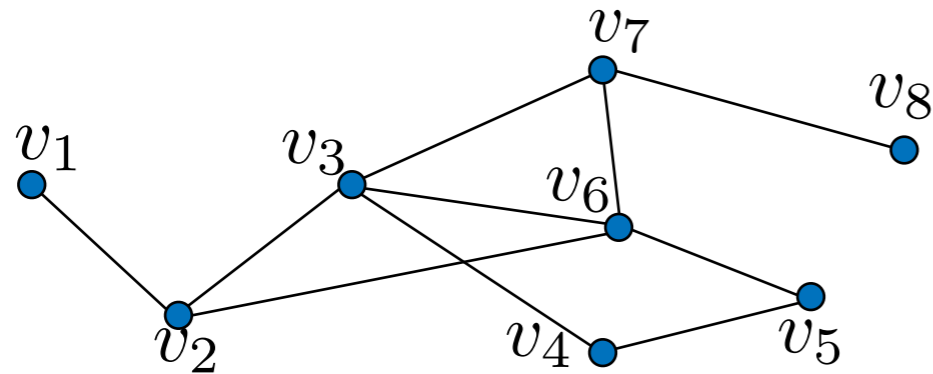
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$D$

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

$W$

# Graph Laplacian



weighted and undirected graph:

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

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

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

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

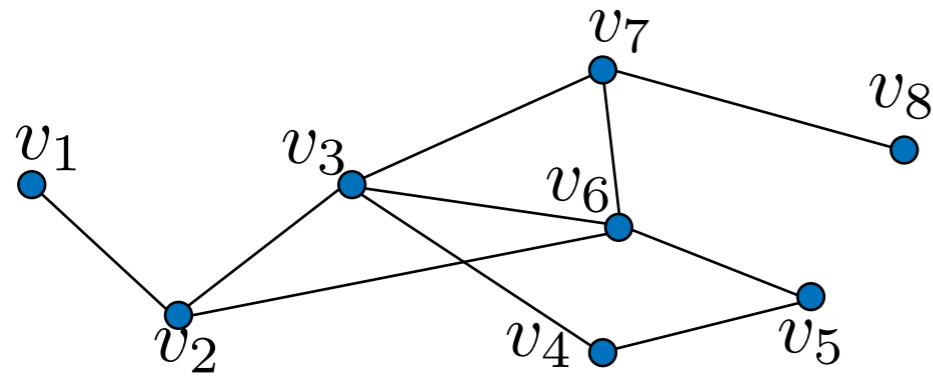
$D$

$W$

$L$

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

# Graph Laplacian



weighted and undirected graph:

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

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

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

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

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

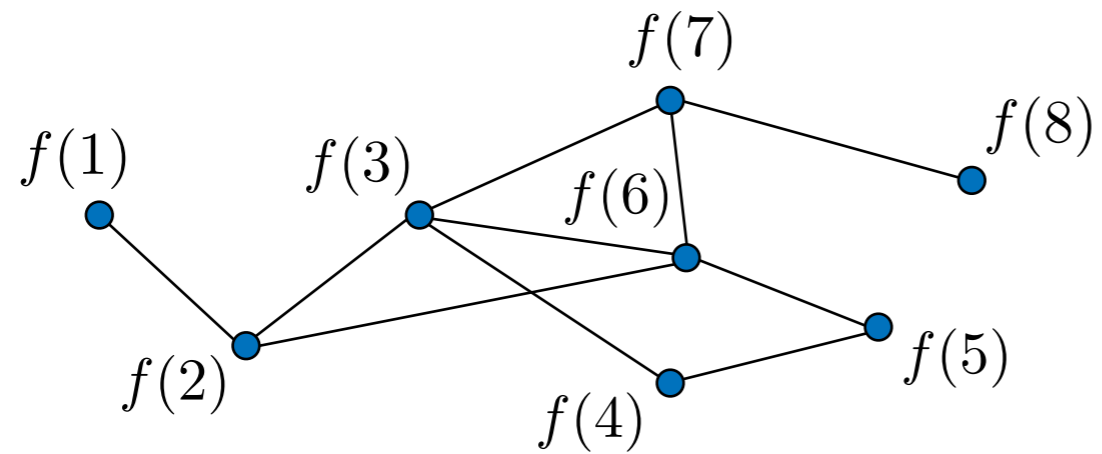
$D$

$W$

$L$

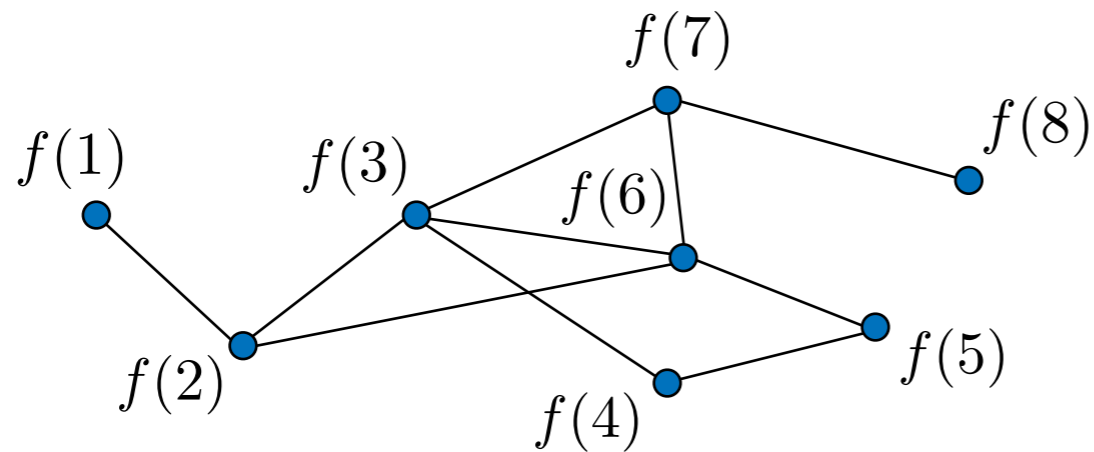
- symmetric
- off-diagonal entries non-positive
- rows sum up to zero

# Graph Laplacian



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

# Graph Laplacian

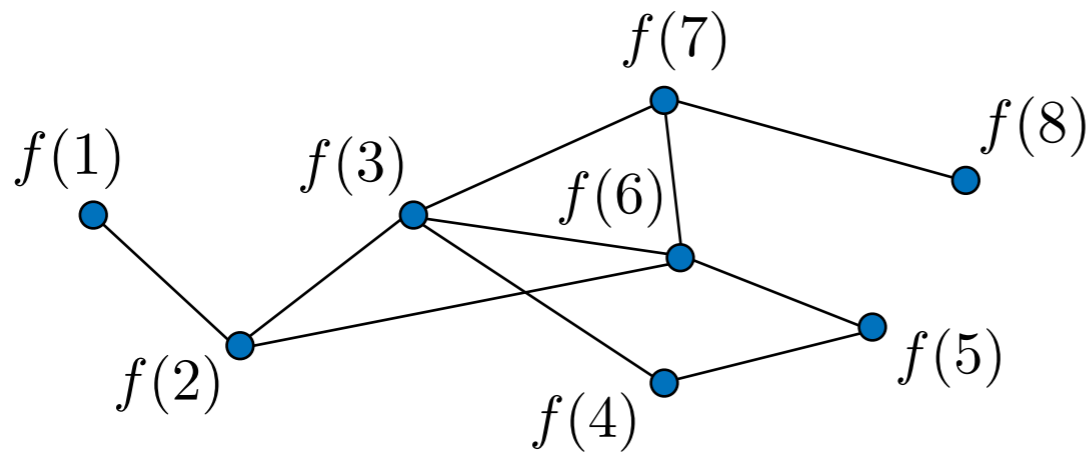


graph signal  $f : \mathcal{V} \rightarrow \mathbb{R}^N$

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

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

# Graph Laplacian



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

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

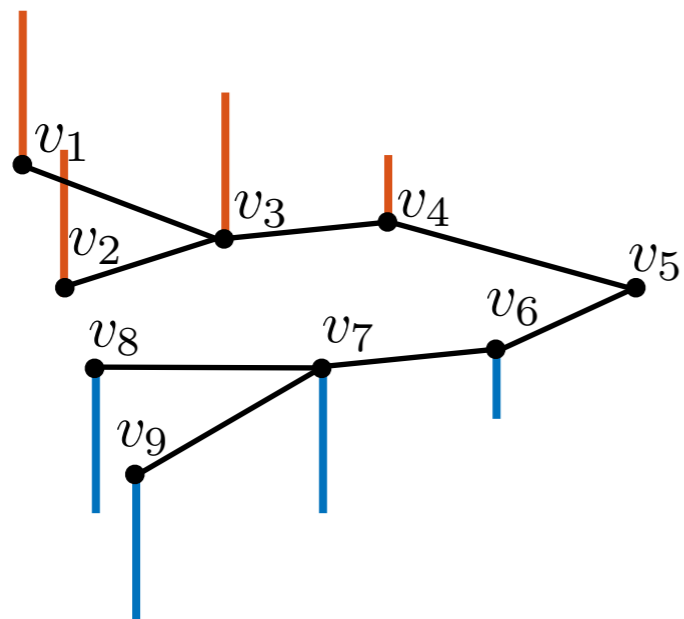
$$\begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}^T \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 &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

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

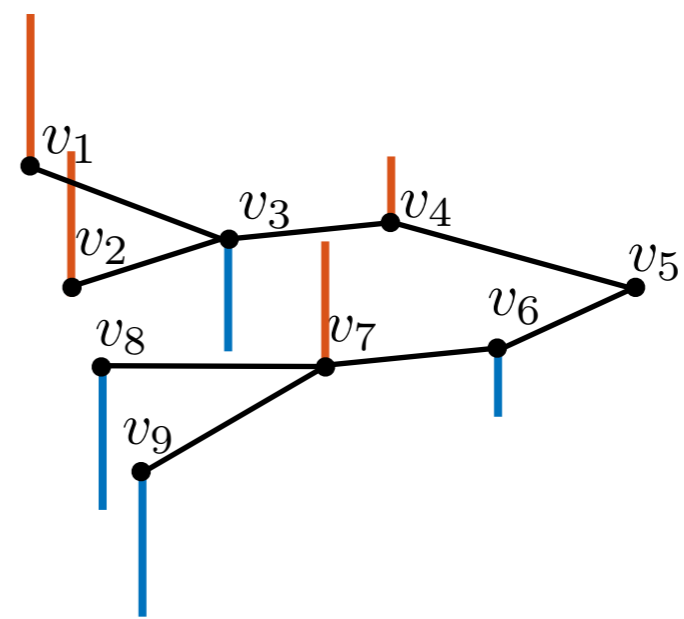
$$f^T Lf = \frac{1}{2} \sum_{i,j=1}^N W_{ij} (f(i) - f(j))^2$$

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

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

$\chi \qquad \qquad \Lambda \qquad \qquad \chi^T$



# Graph Laplacian

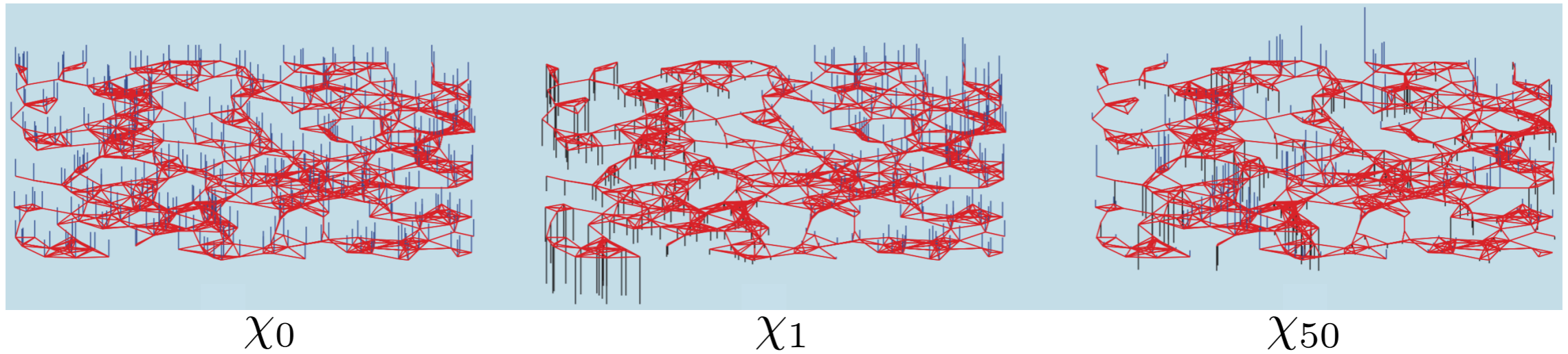
- $L$  has a complete set of orthonormal eigenvectors:  $L = \chi \Lambda \chi^T$

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

$\chi \qquad \qquad \Lambda \qquad \qquad \chi^T$

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

# Graph Fourier transform

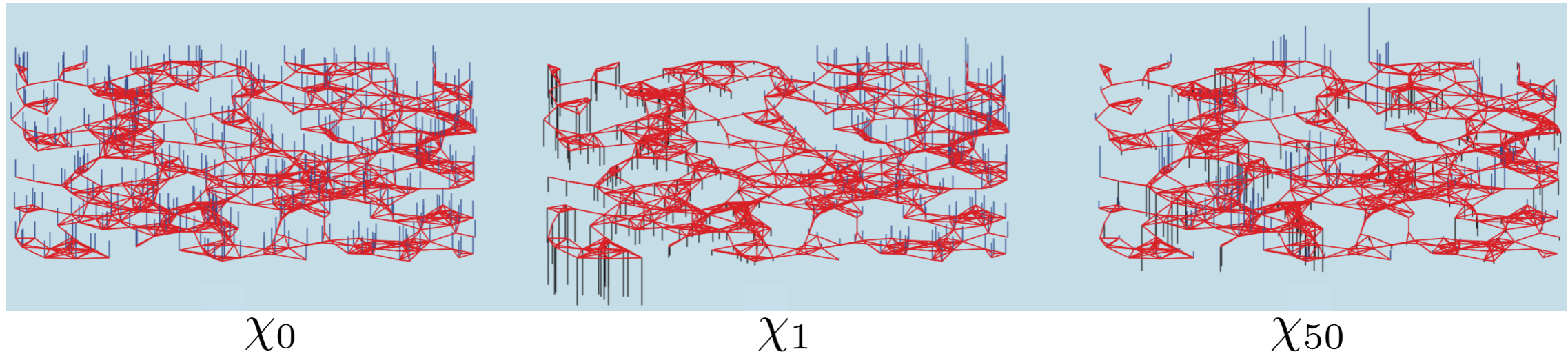


$\chi_0$

$\chi_1$

$\chi_{50}$

# Graph Fourier transform



low frequency

high frequency

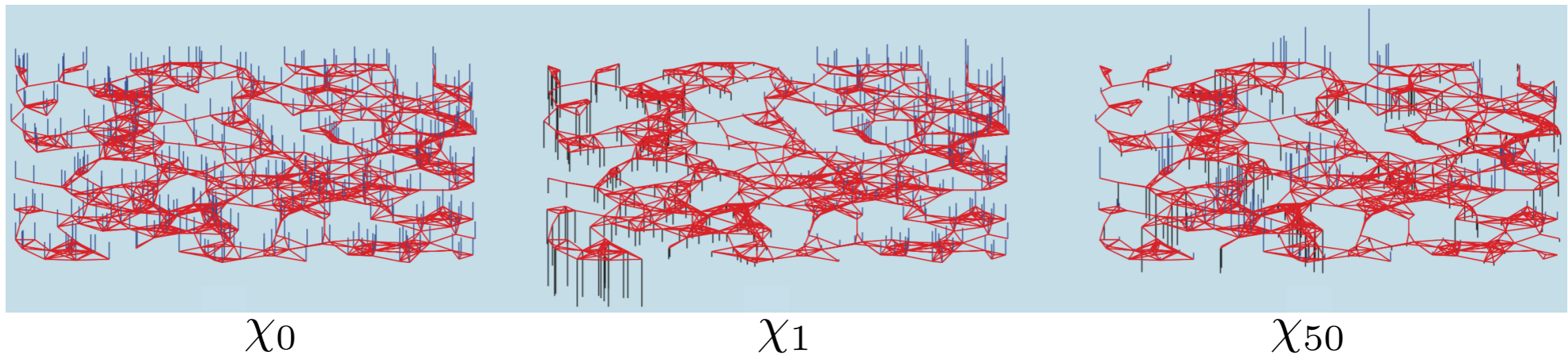
$$\chi_0^T L \chi_0 = \lambda_0 = 0$$

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

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

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

# Graph Fourier transform



low frequency

high frequency

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

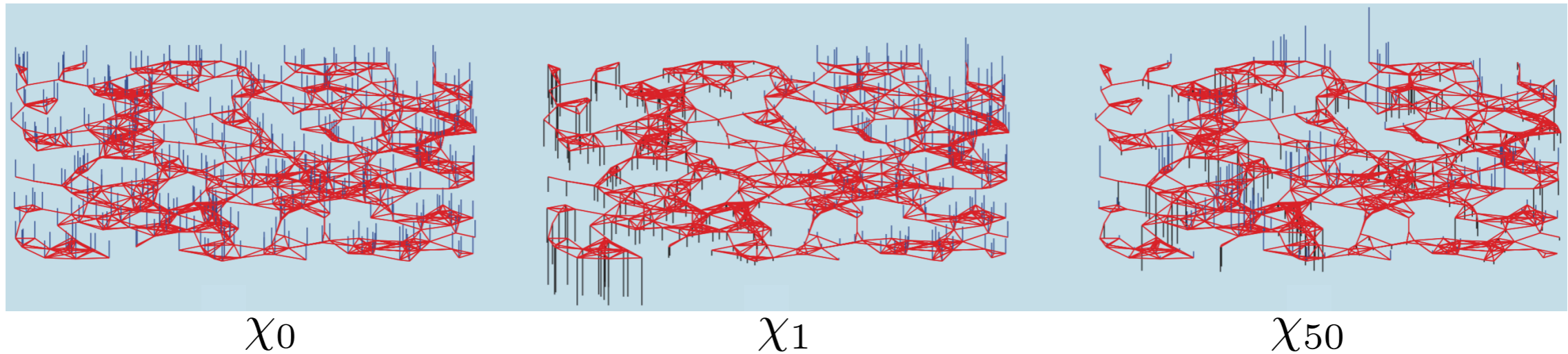
$$\chi_{50}^T L \chi_{50} = \lambda_{50}$$

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

graph Fourier transform:  
[Hammond11]

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

# Graph Fourier transform



low frequency

high frequency

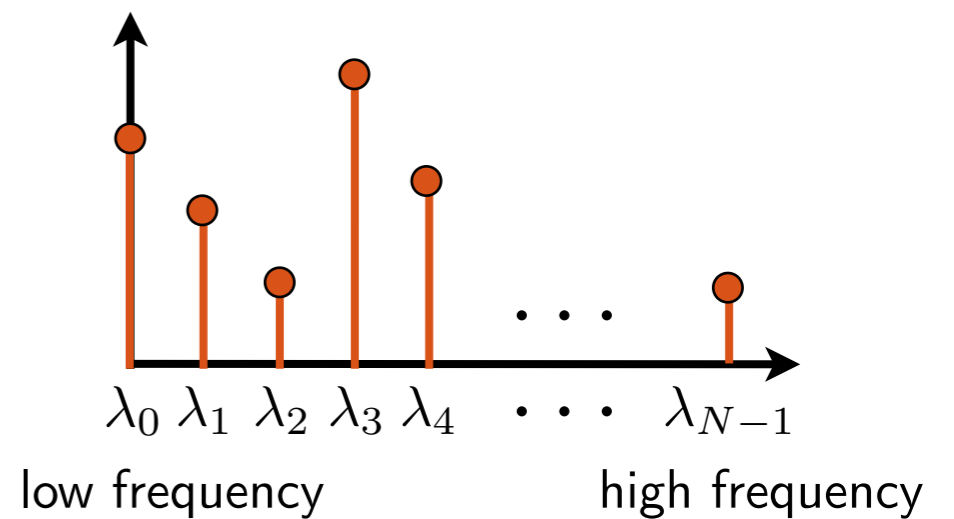
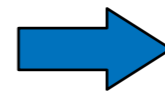
$$\chi_0^T L \chi_0 = \lambda_0 = 0$$

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

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

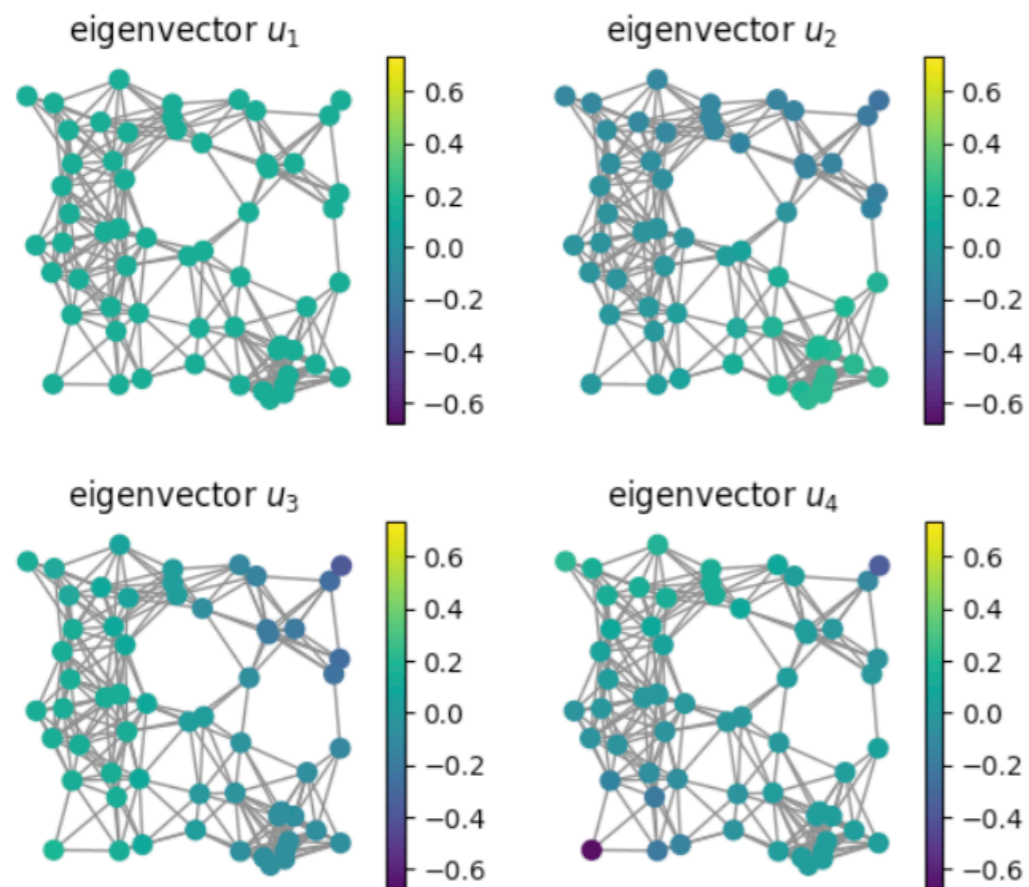
graph Fourier transform:  
[Hammond11]

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



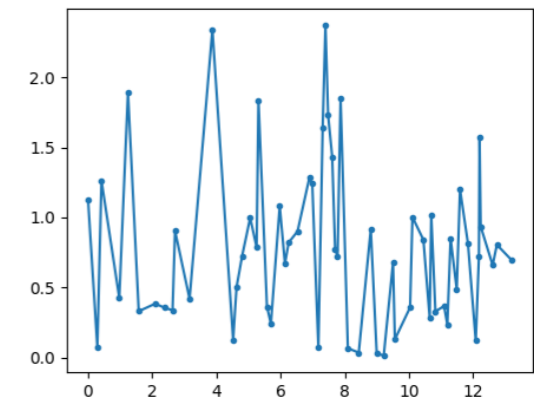
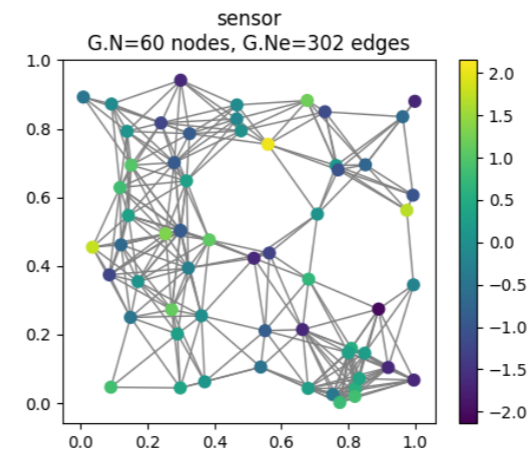
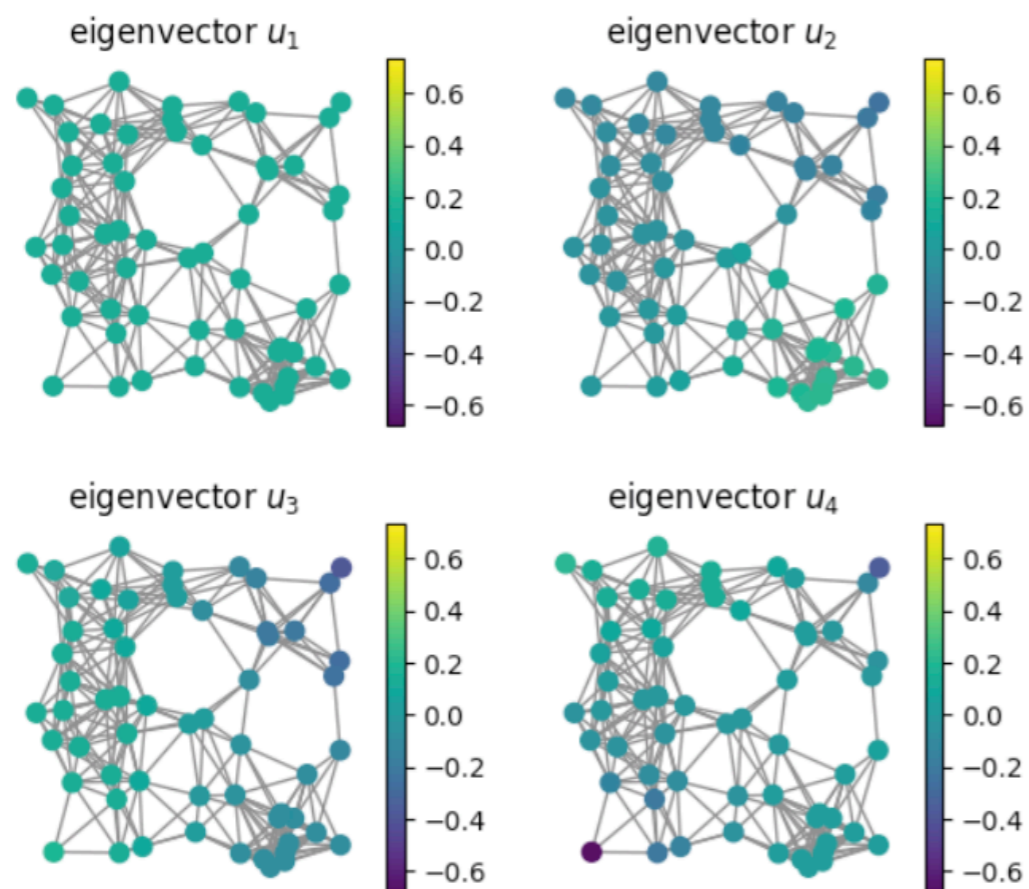
# Graph Fourier transform

- Example on a simple graph



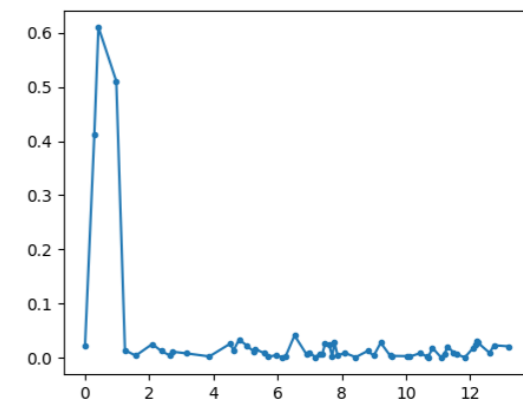
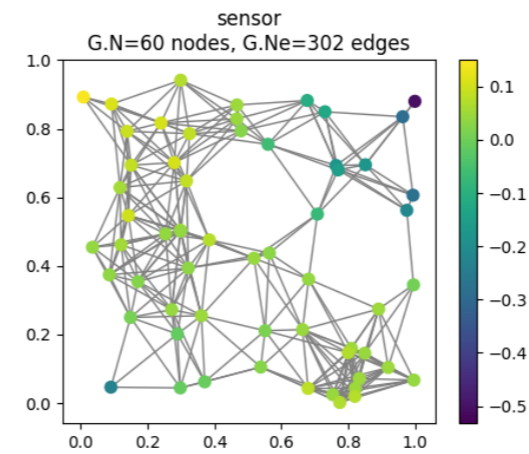
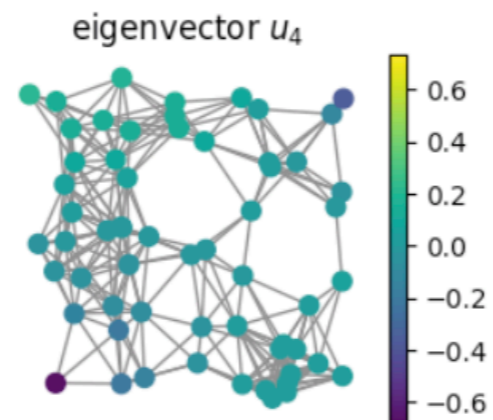
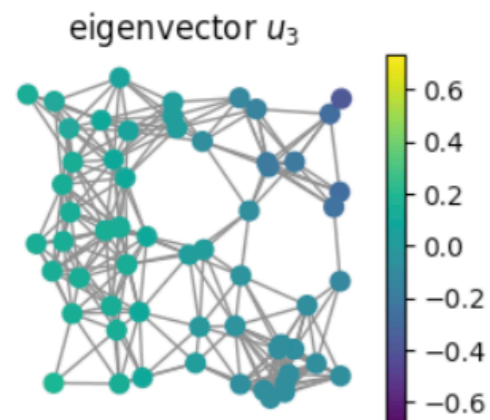
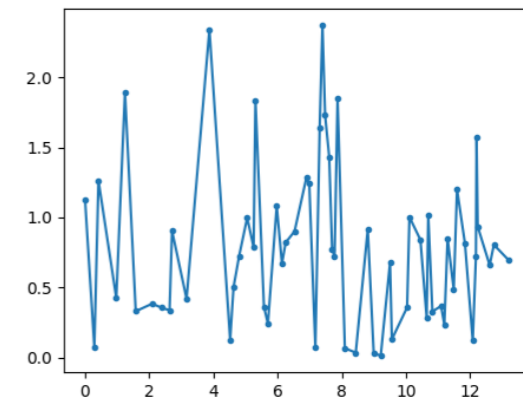
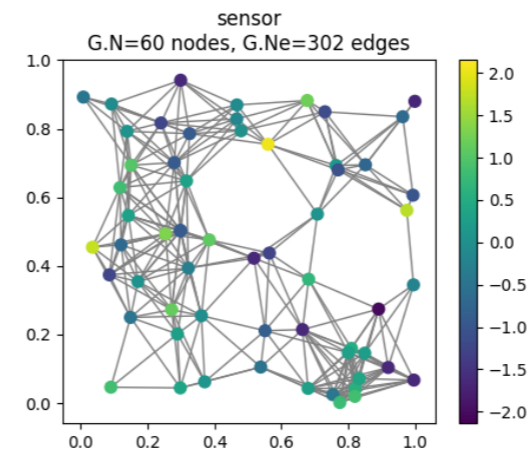
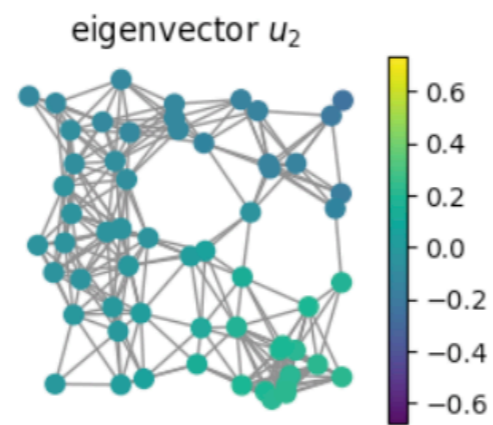
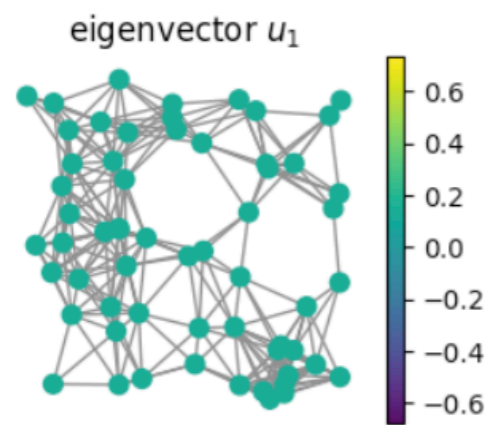
# Graph Fourier transform

- Example on a simple graph



# Graph Fourier transform

- Example on a simple graph





# Outline

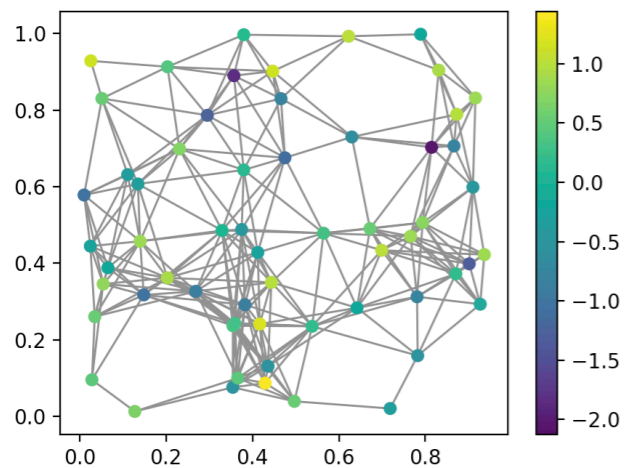
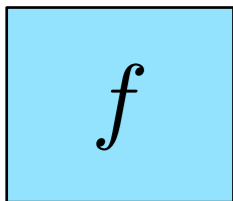
- Graph signal processing (GSP): Basic concepts
- Filtering of graph signals: Basic tool of GSP
- Connection with literature
- Research topics inspired by GSP
- Applications

# Filtering of graph signals

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

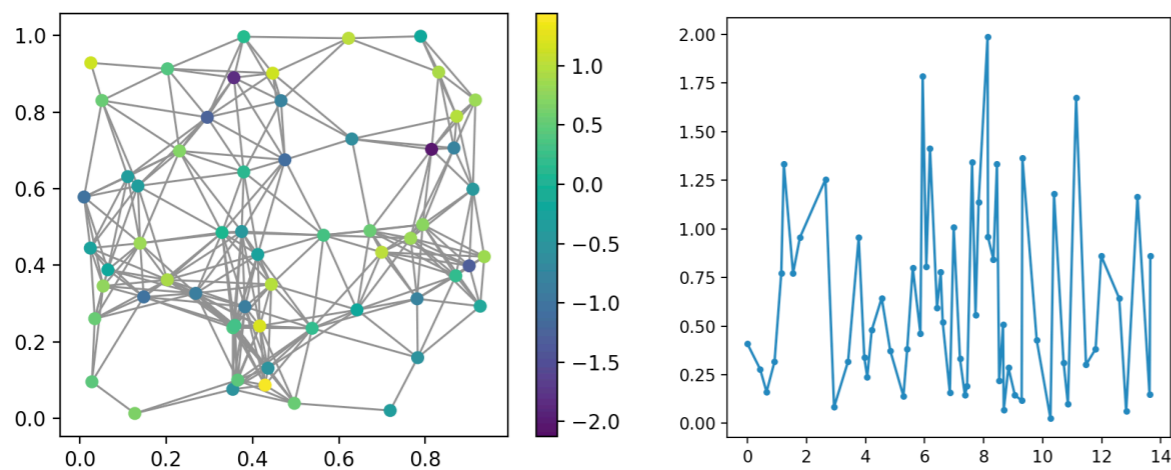
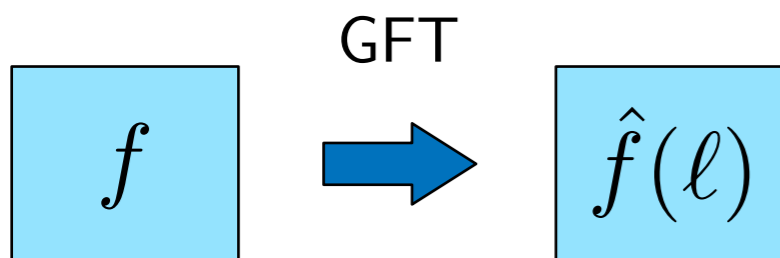
# Filtering of graph signals

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



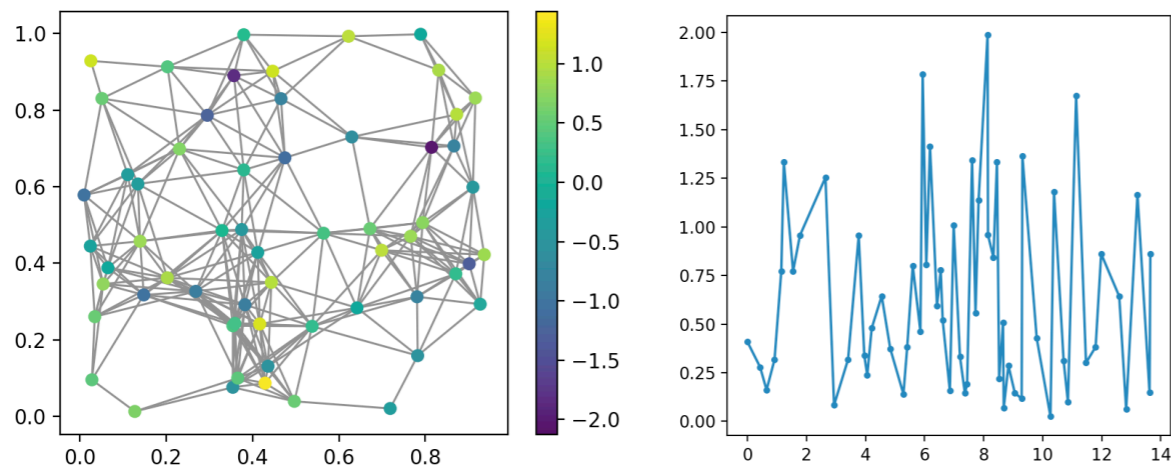
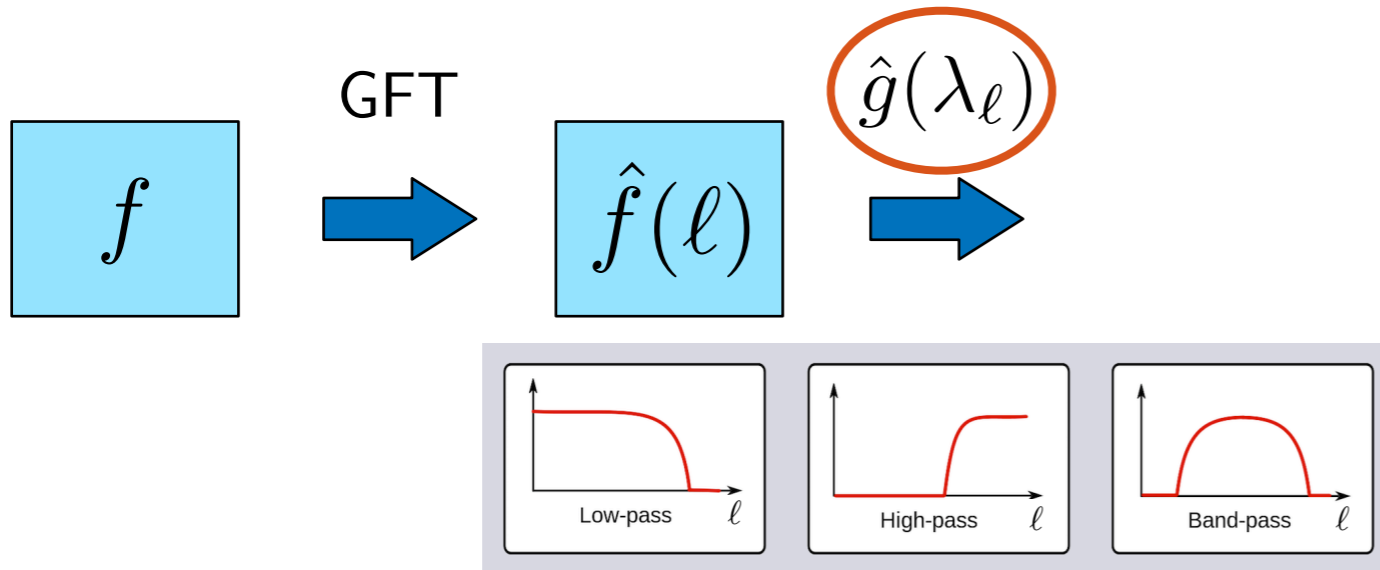
# Filtering of graph signals

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



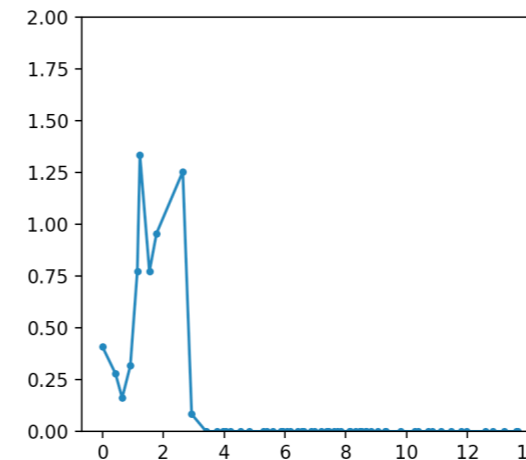
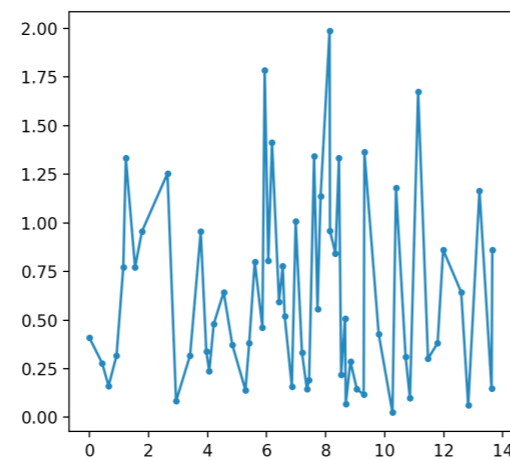
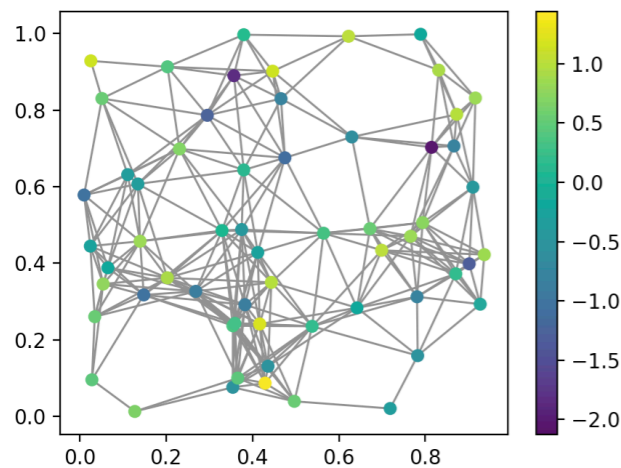
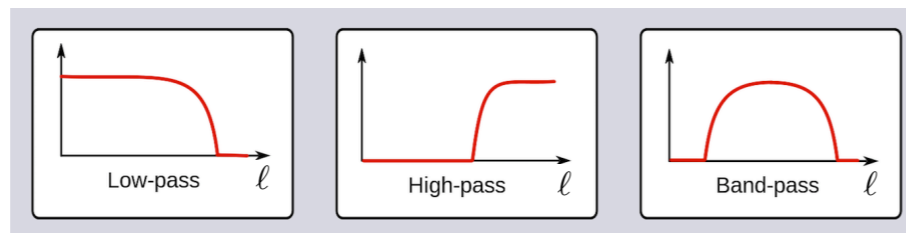
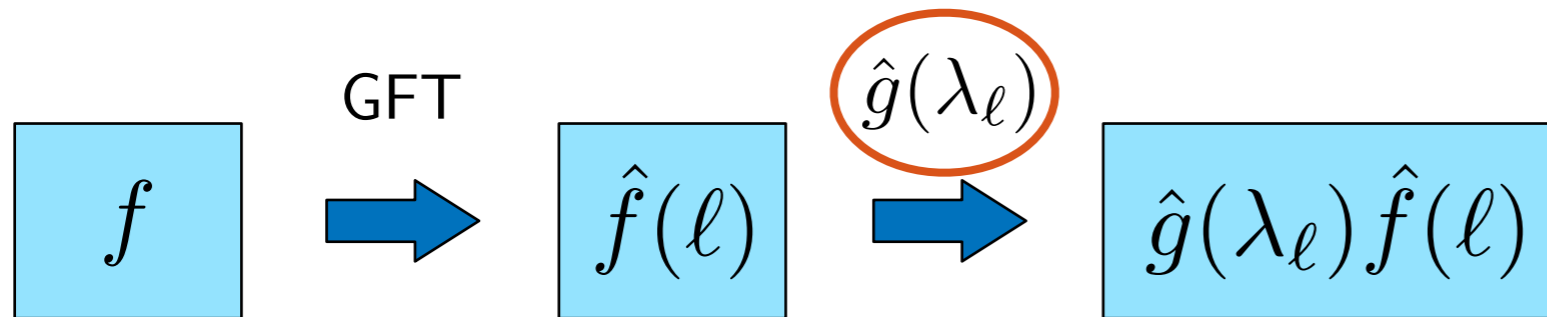
# Filtering of graph signals

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



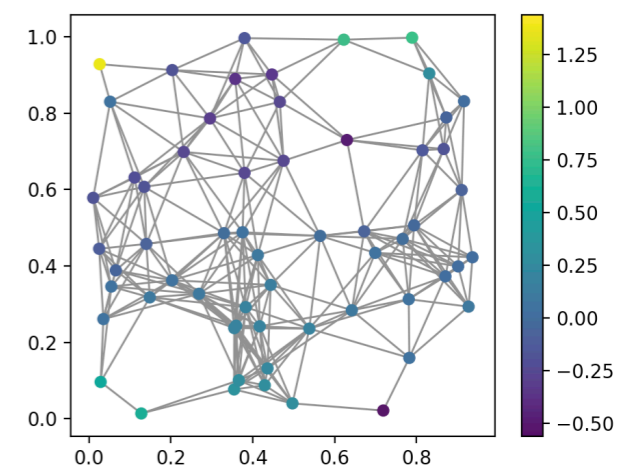
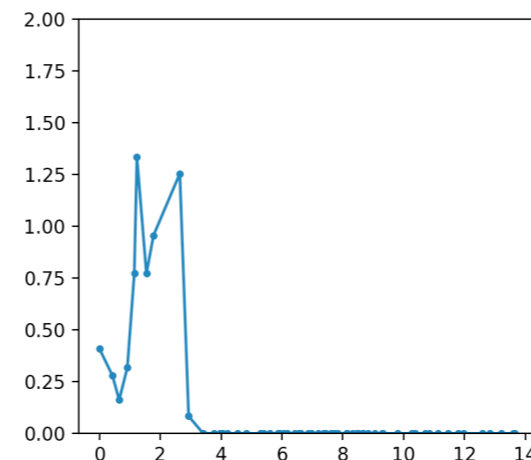
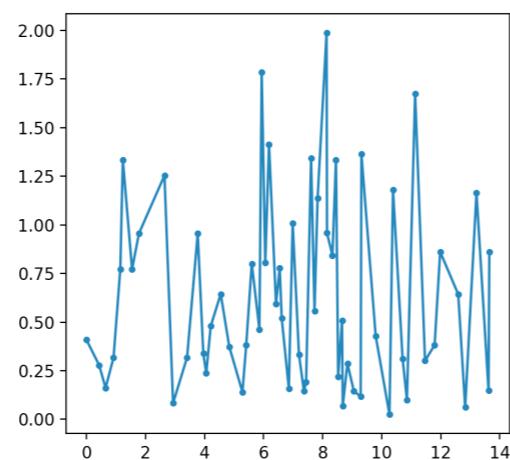
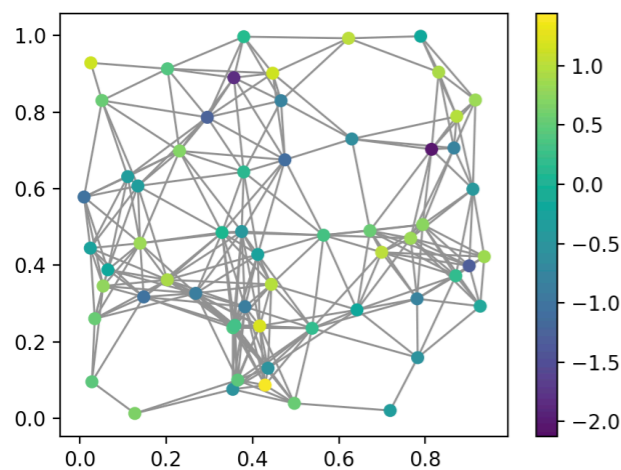
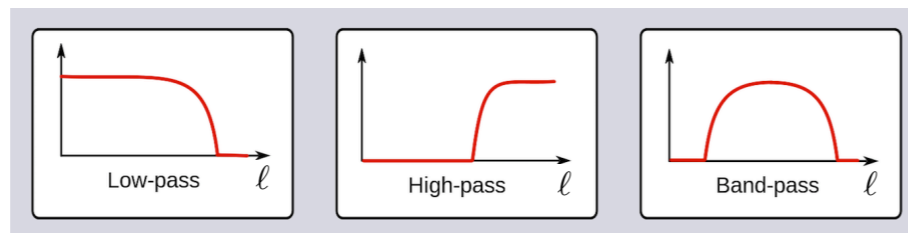
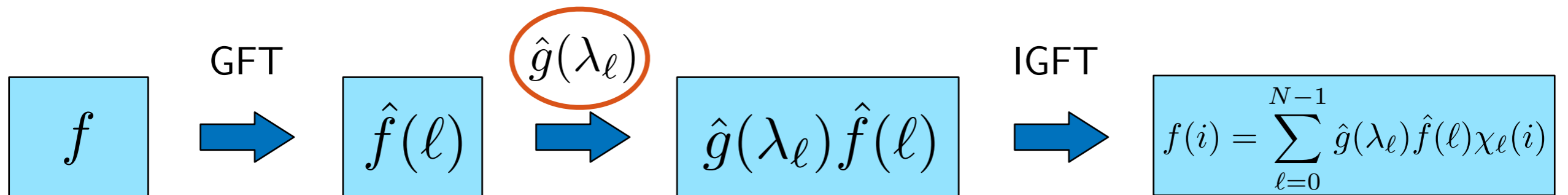
# Filtering of graph signals

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



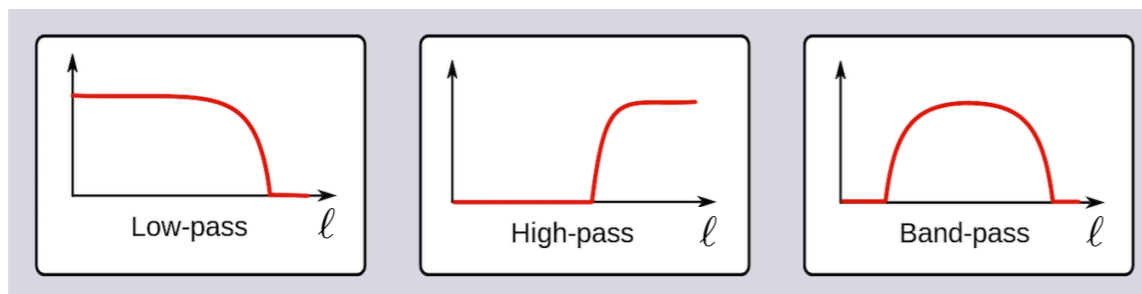
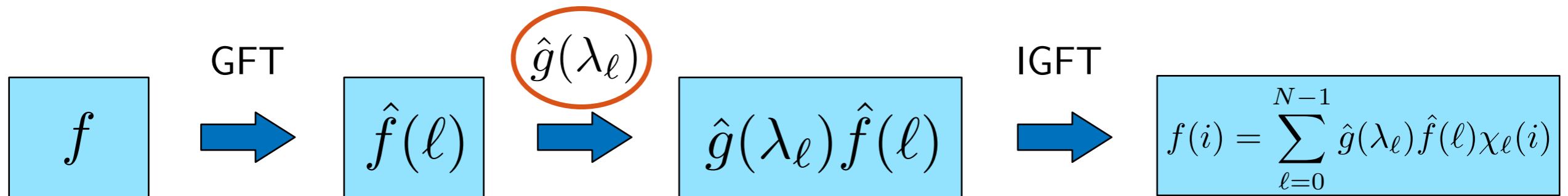
# Filtering of graph signals

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



# Filtering of graph signals

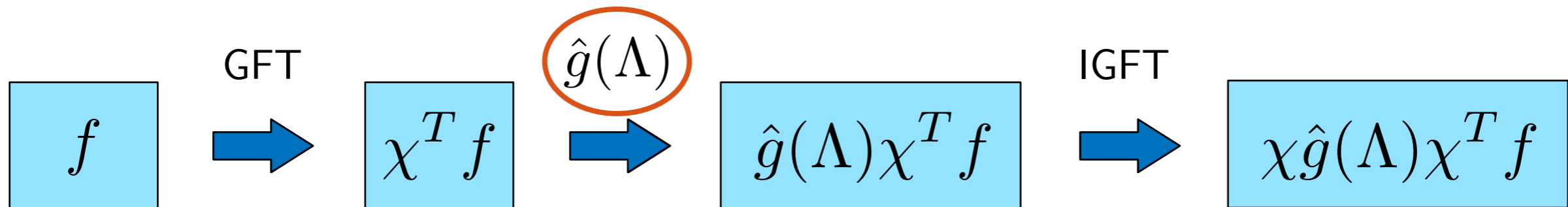
$$\text{GFT: } \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^N \chi_\ell^*(i) f(i) \quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$





# Filtering of graph signals

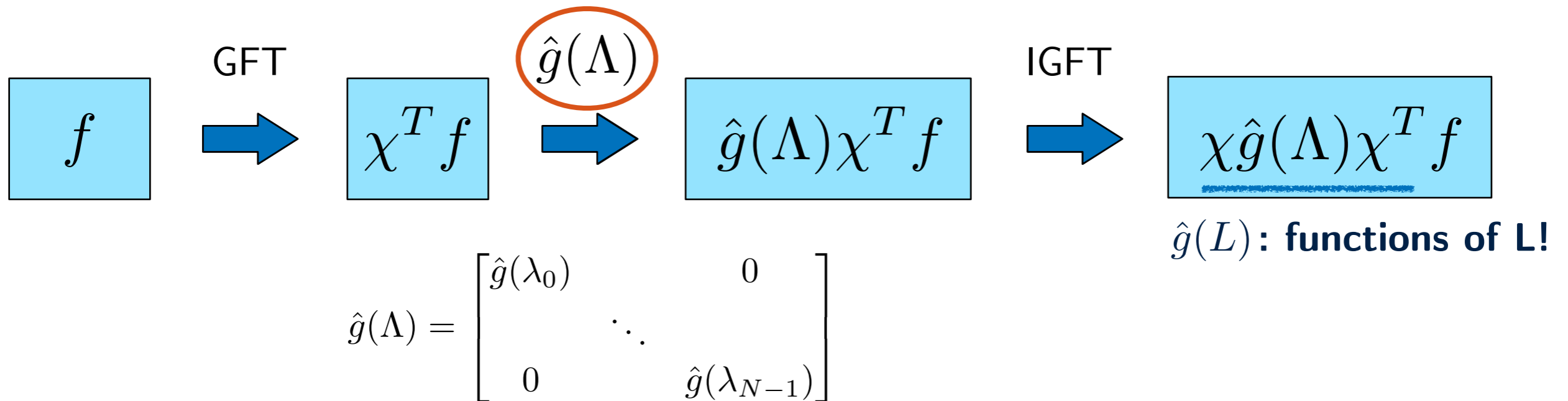
$$\text{GFT: } \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^N \chi_\ell^*(i) f(i) \quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$



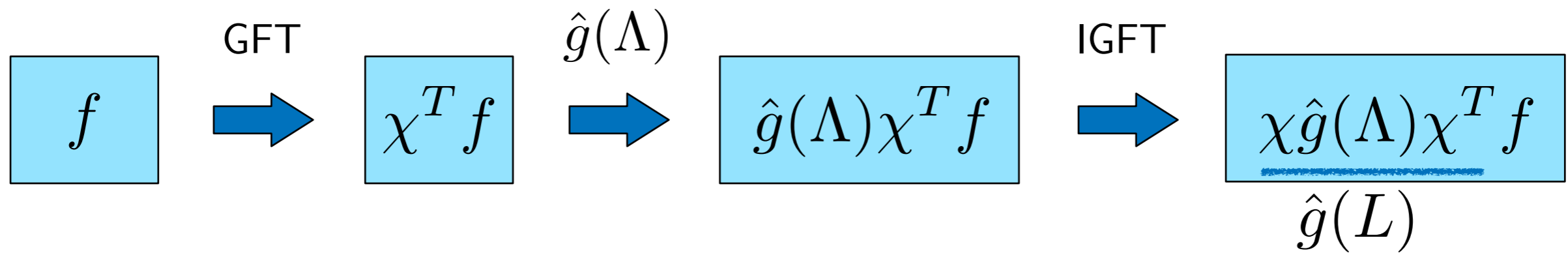
$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

# Filtering of graph signals

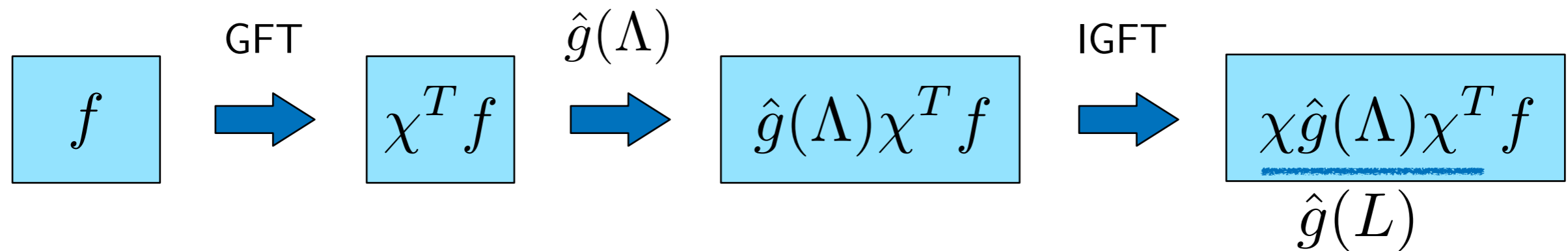
$$\text{GFT: } \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^N \chi_\ell^*(i) f(i) \quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$



# A practical example



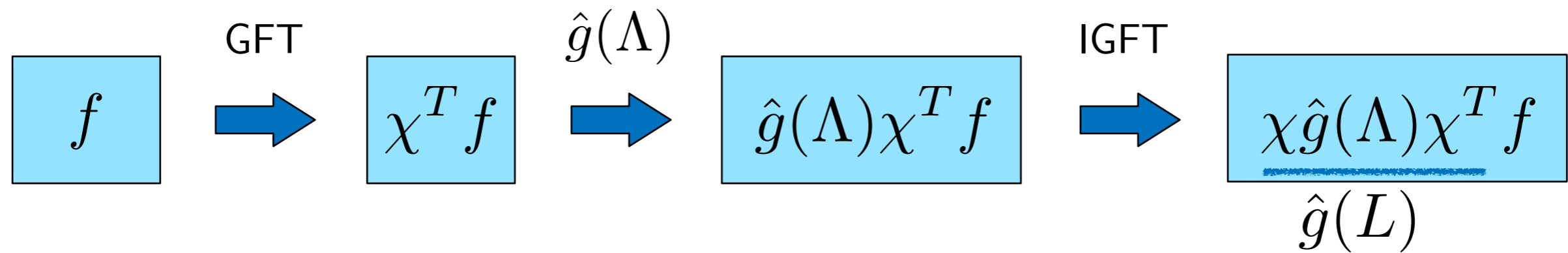
# A practical example



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

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

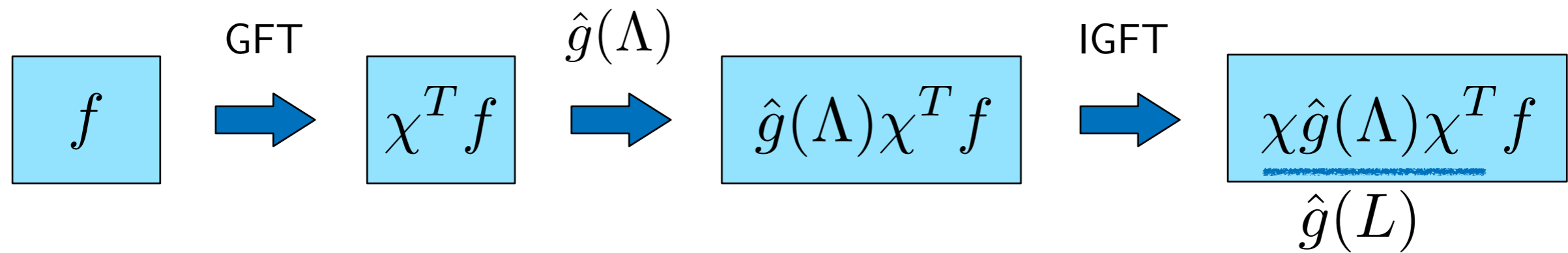
# A practical example



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

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

# A practical example

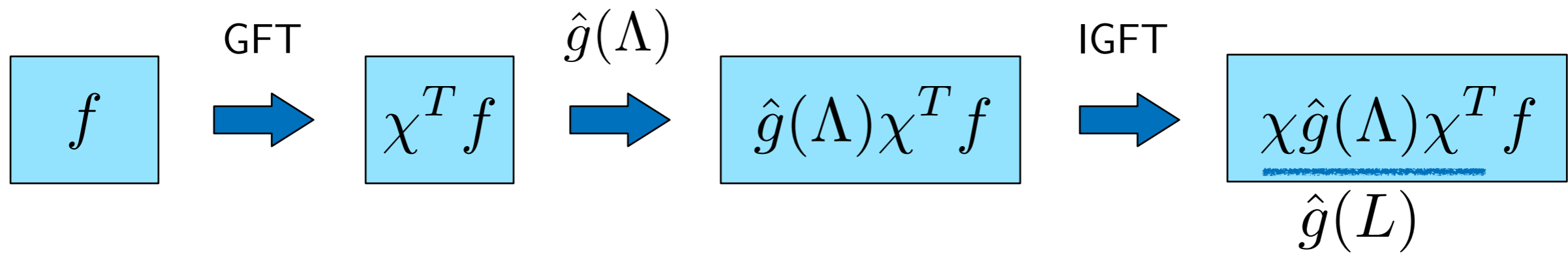


problem: we observe a noisy graph signal  $f = y_0 + \eta$  and wish to recover  $y_0$

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

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

# A practical example



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

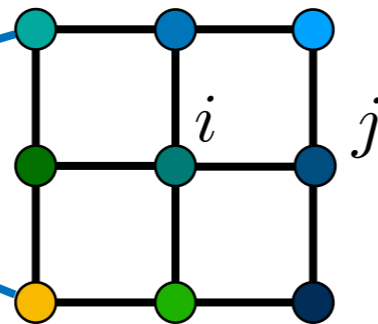
$$y^* = \arg \min_y \{ \underbrace{\|y - f\|_2^2}_{\text{data fitting term}} + \underbrace{\gamma y^T L y}_{\text{"smoothness" assumption}} \}$$

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

**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)

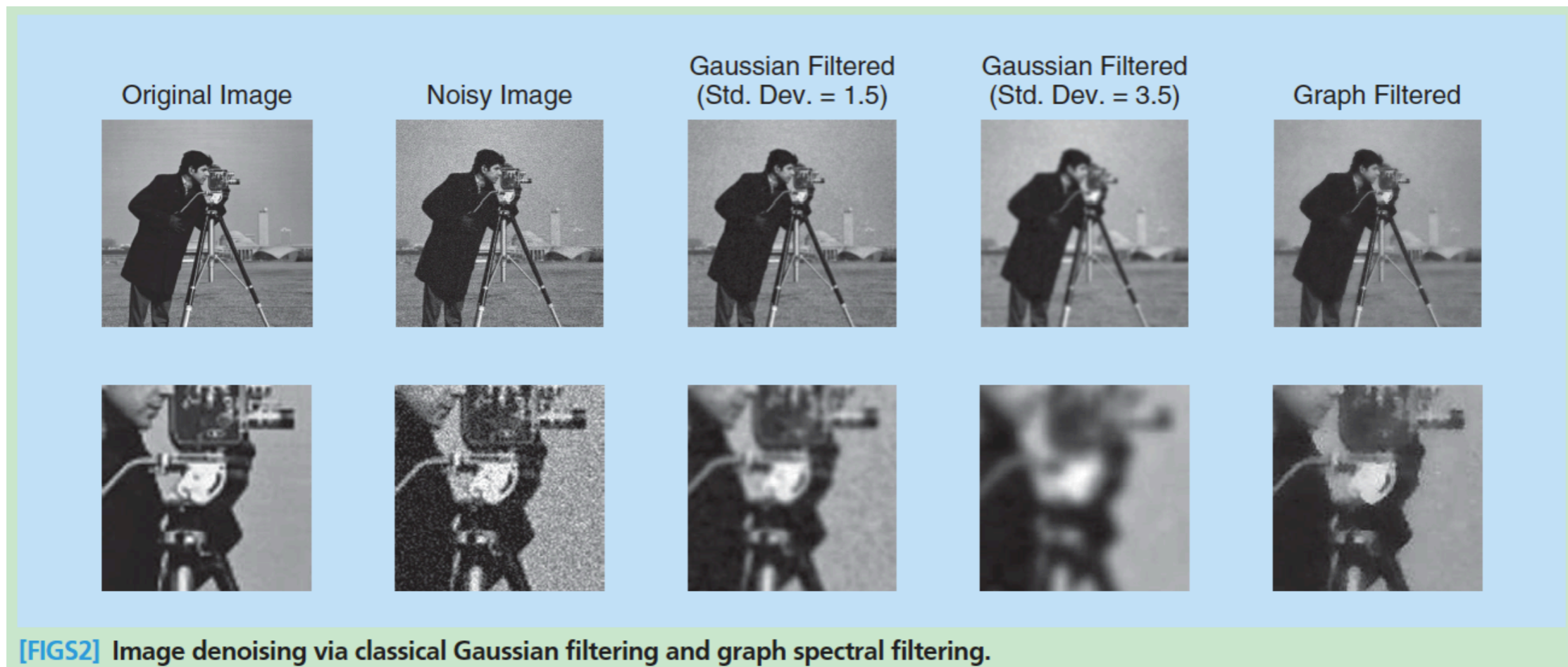


$$w_{ij} = \frac{1}{|f(i) - f(j)|}$$

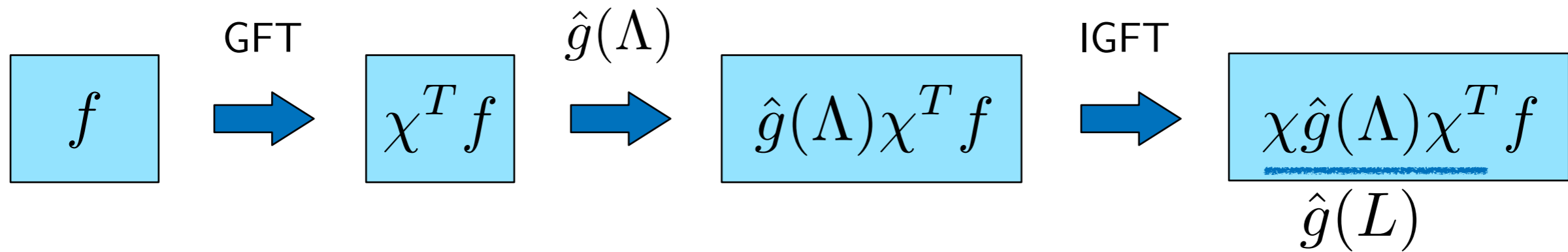


# A practical example

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



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

# Outline

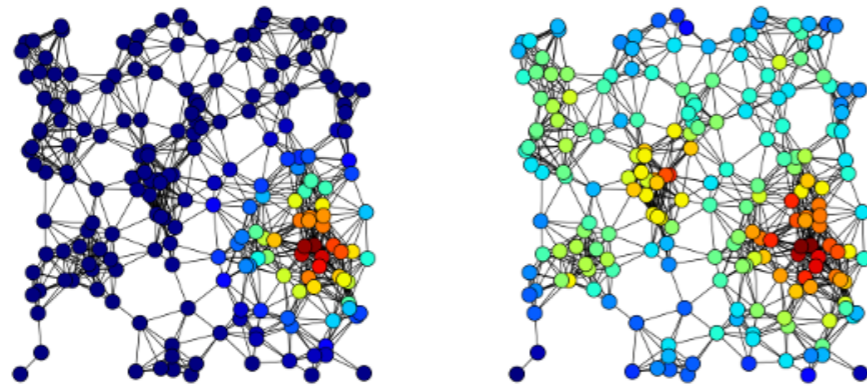
- Graph signal processing (GSP): Basic concepts
- Filtering of graph signals: Basic tool of GSP
- Connection with literature
- Research topics inspired by GSP
- Applications

# GSP and the literature

there is a rich literature about data analysis and learning on graphs

# GSP and the literature

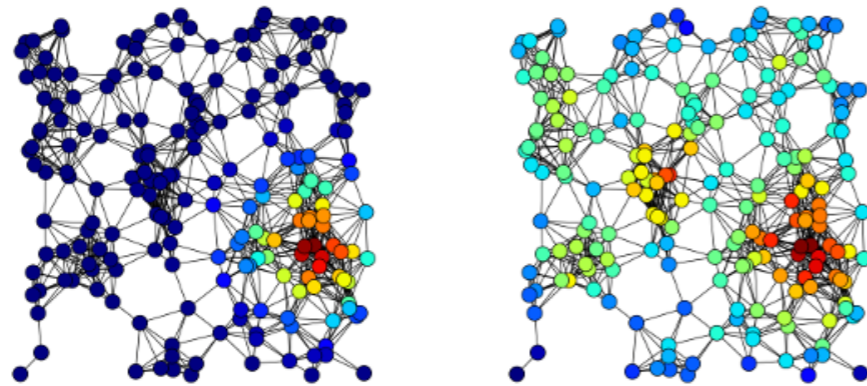
there is a rich literature about data analysis and learning on graphs



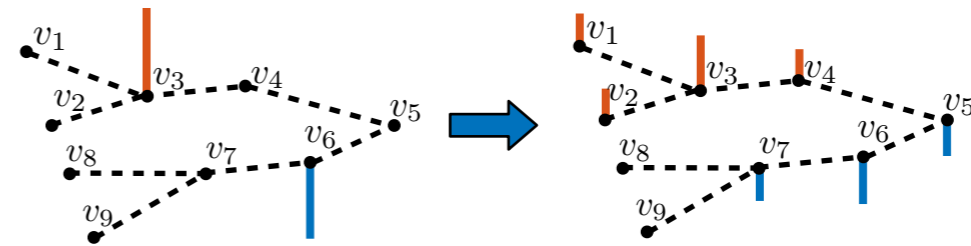
network science

# GSP and the literature

there is a rich literature about data analysis and learning on graphs



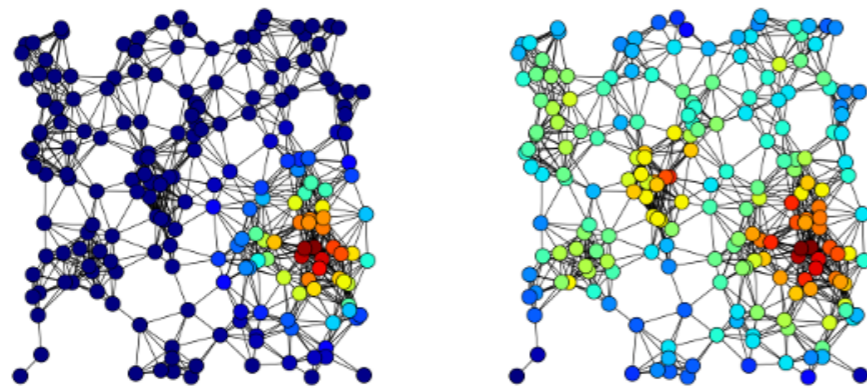
network science



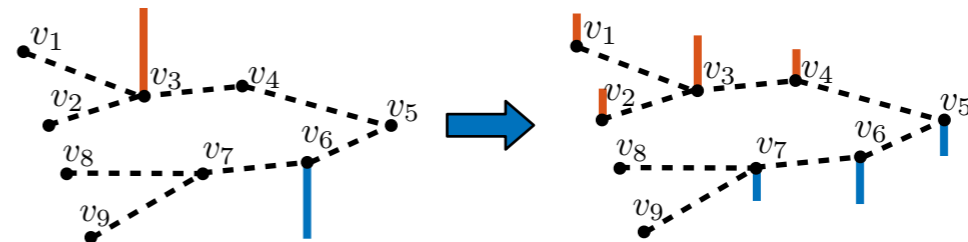
diffusion on graphs

# GSP and the literature

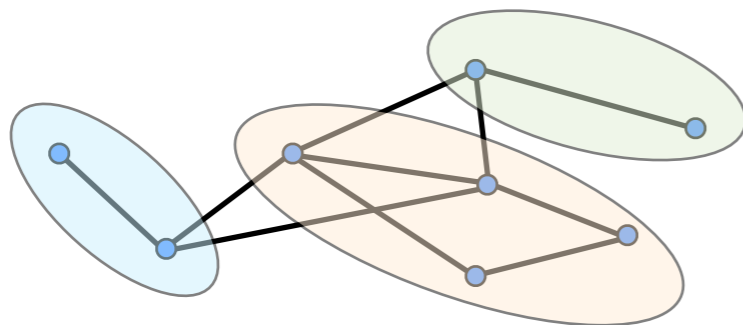
there is a rich literature about data analysis and learning on graphs



network science



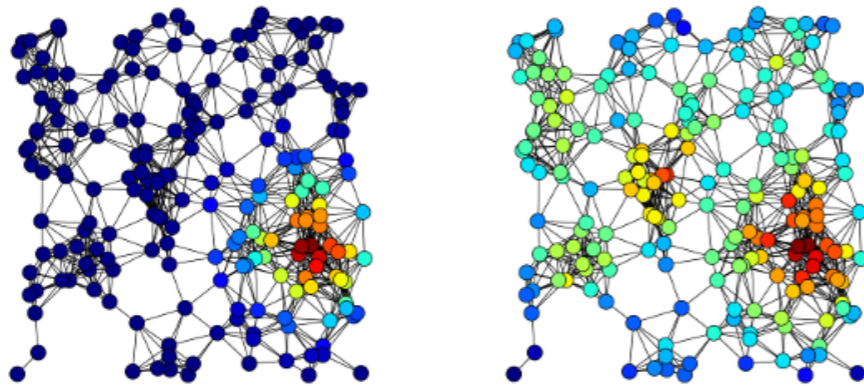
diffusion on graphs



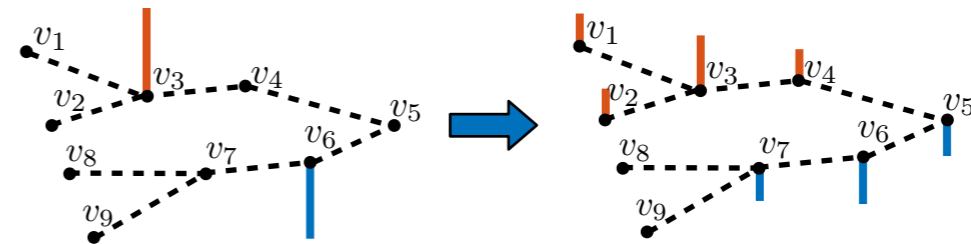
unsupervised learning (dimensionality reduction, clustering)

# GSP and the literature

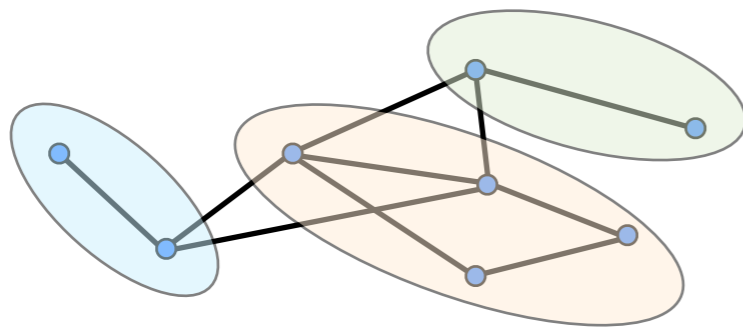
there is a rich literature about data analysis and learning on graphs



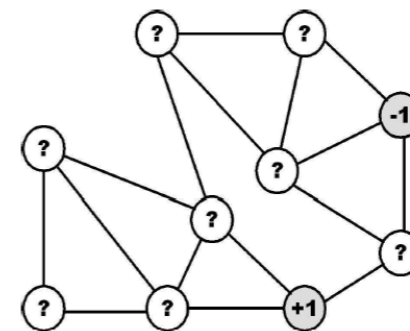
network science



diffusion on graphs



unsupervised learning (dimensionality reduction, clustering)

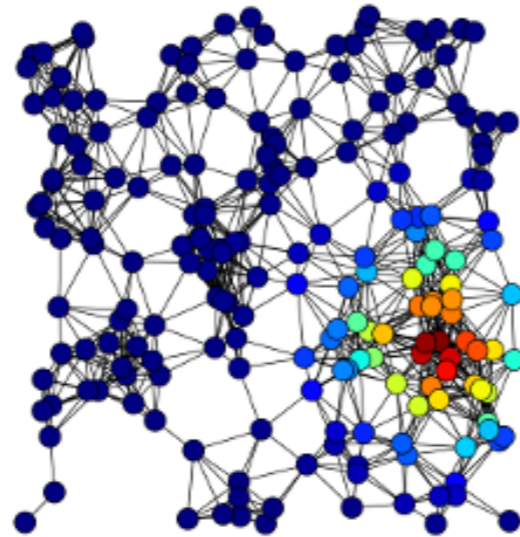


semi-supervised learning



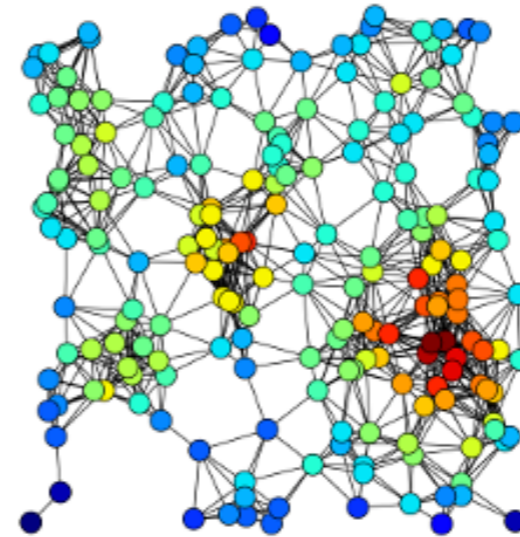
# Network centrality

eigenvector centrality



$$Wx = \lambda_{\max}x$$

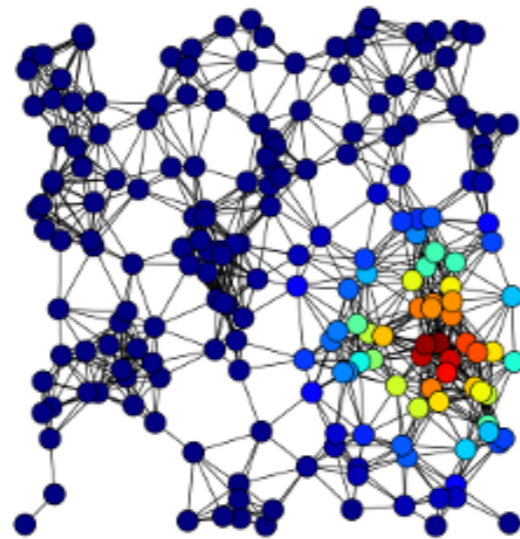
degree centrality



$$d = [d(v_1), \dots, d(v_N)]$$

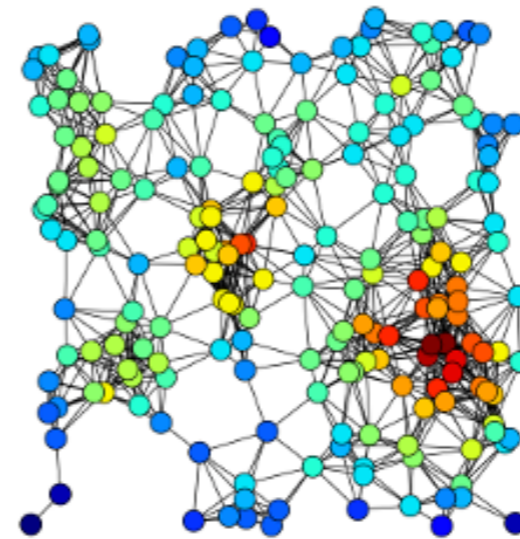
# Network centrality

eigenvector centrality



$$Wx = \lambda_{\max}x$$

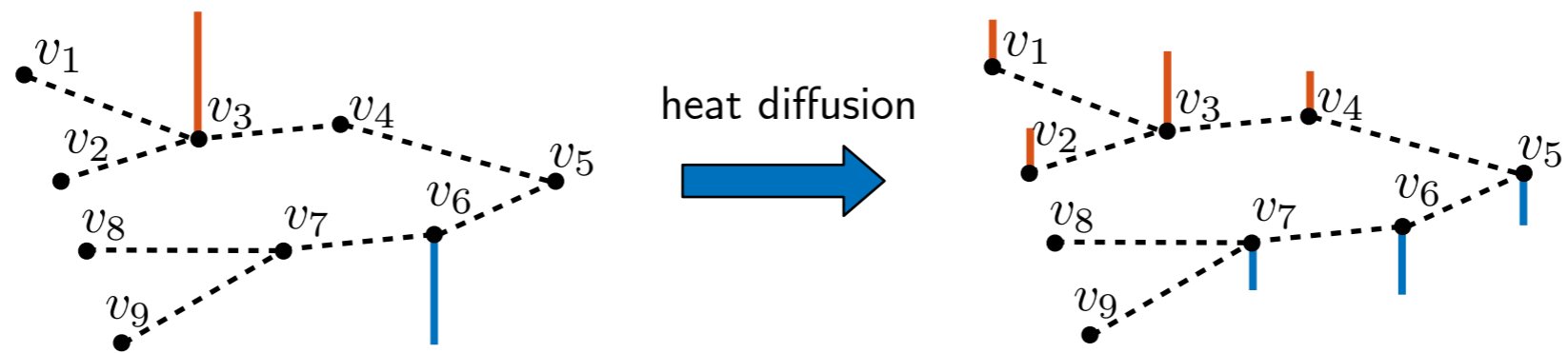
degree centrality



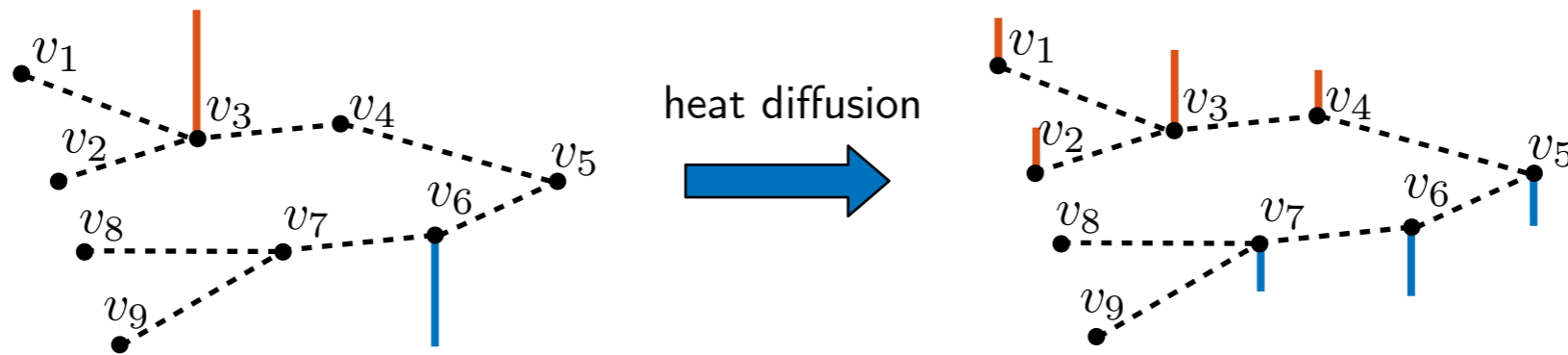
$$d = [d(v_1), \dots, d(v_N)]$$

- Google's PageRank is a variant of eigenvector centrality
- eigenvectors of  $W$  can also be used to provide a frequency interpretation for graph signals

# Diffusion on graphs



# Diffusion on graphs

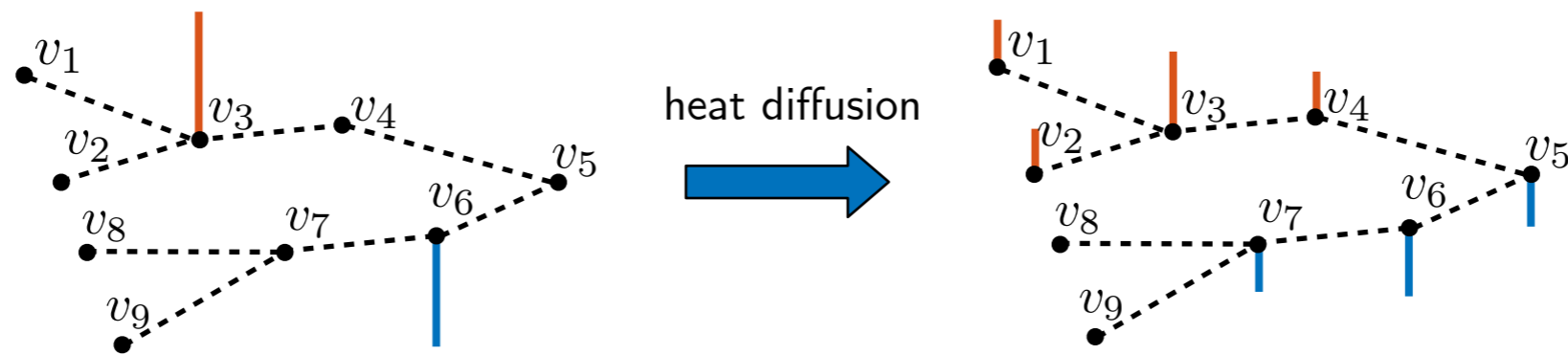


$$\frac{\partial x}{\partial \tau} - Lx = 0$$
$$x(v, 0) = x_0(v)$$



$$x(v, \tau) = e^{-\tau L} x_0(v)$$

# Diffusion on graphs



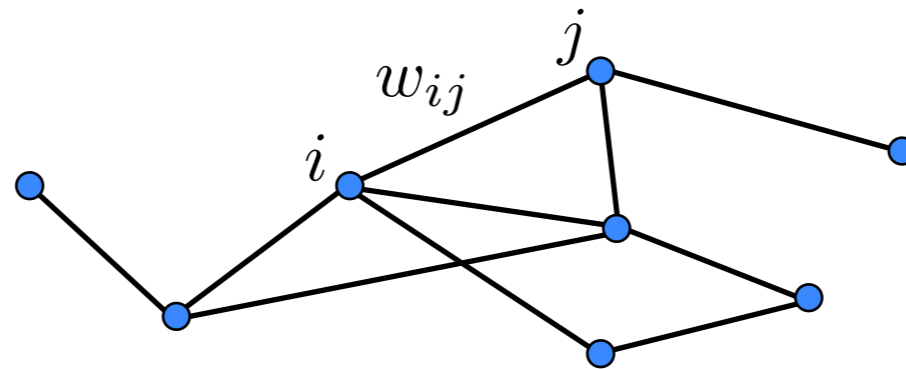
$$\frac{\partial x}{\partial \tau} - Lx = 0$$
$$x(v, 0) = x_0(v)$$



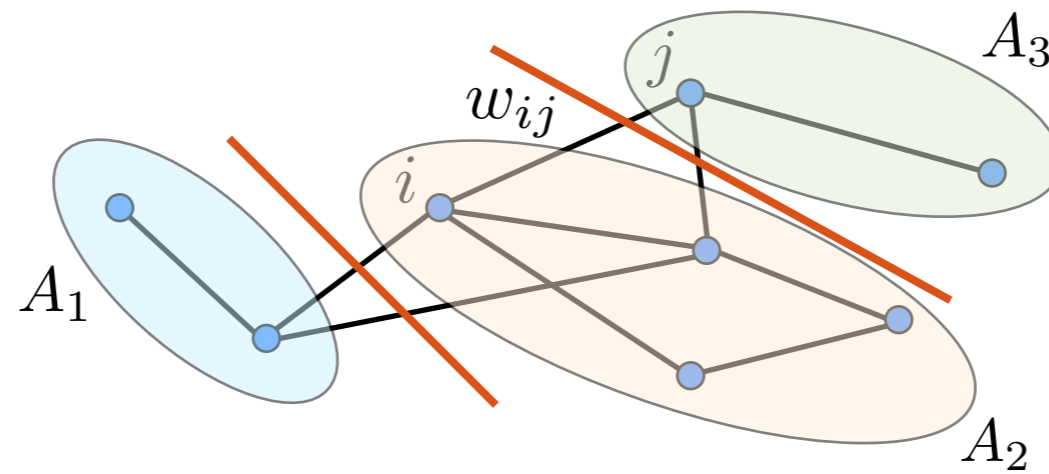
$$x(v, \tau) = e^{-\tau L} x_0(v)$$

- heat diffusion on graphs is a typical physical process on graphs
- other possibilities exist (e.g., random walk on graphs)
- many have an interpretation of filtering on graphs

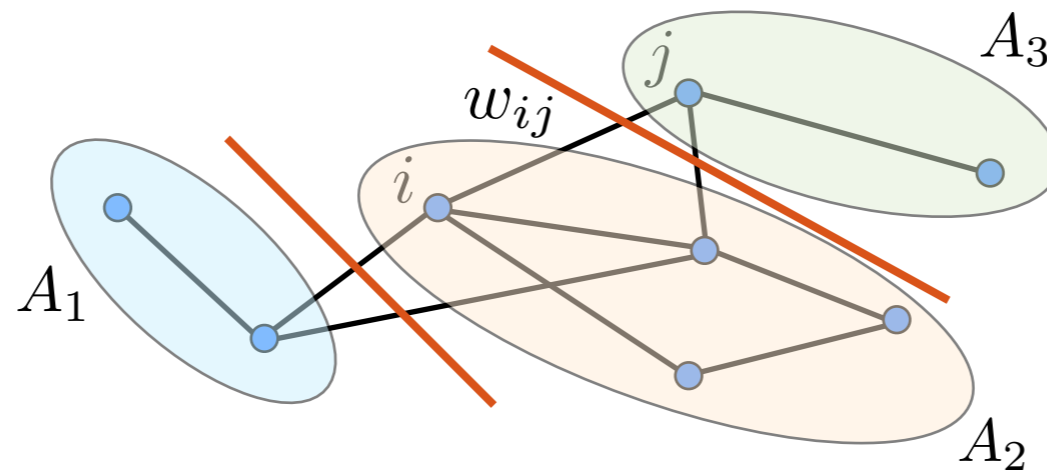
# Graph clustering (community detection)



# Graph clustering (community detection)



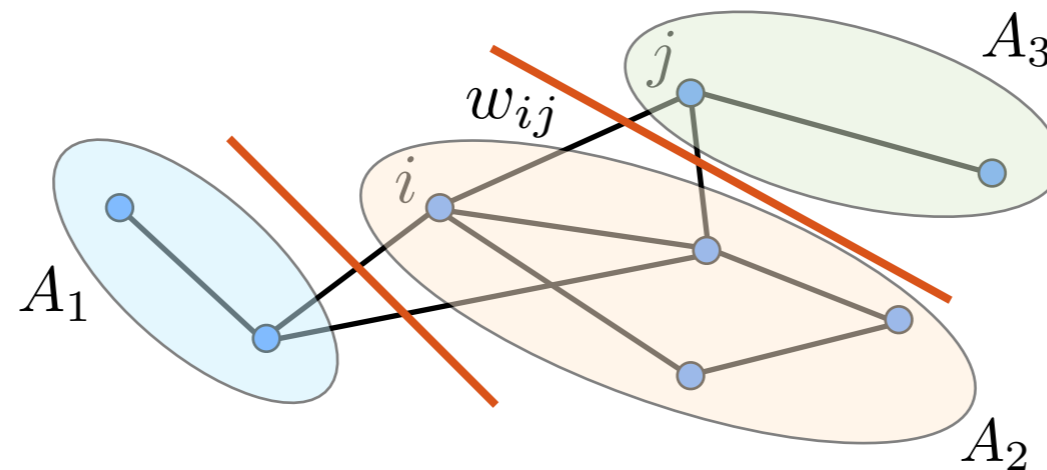
# Graph clustering (community detection)



$$NCut(A_1, \dots, A_k) = \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \overline{A_i})}{vol(A_i)}$$



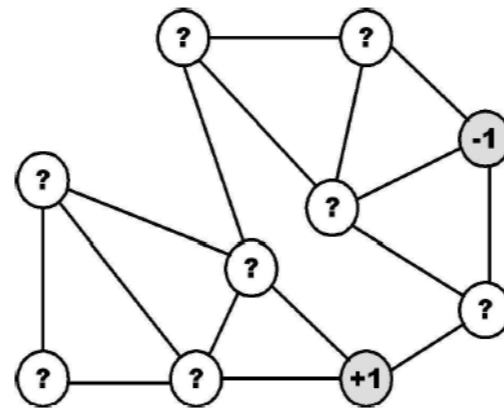
# Graph clustering (community detection)



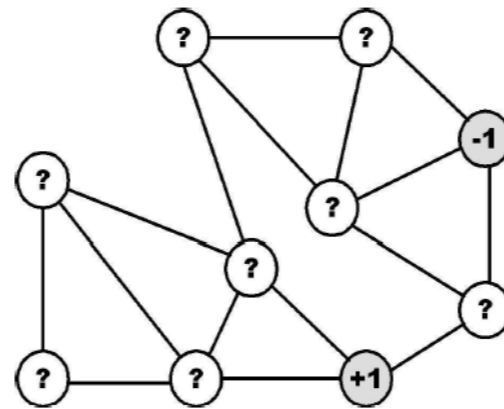
$$NCut(A_1, \dots, A_k) = \frac{1}{2} \sum_{i=1}^k \frac{W(A_i, \bar{A}_i)}{vol(A_i)}$$

- first  $k$  eigenvectors of graph Laplacian minimise the graph cut
- eigenvectors of graph Laplacian enable a Fourier-like analysis for graph signals

# Semi-supervised learning



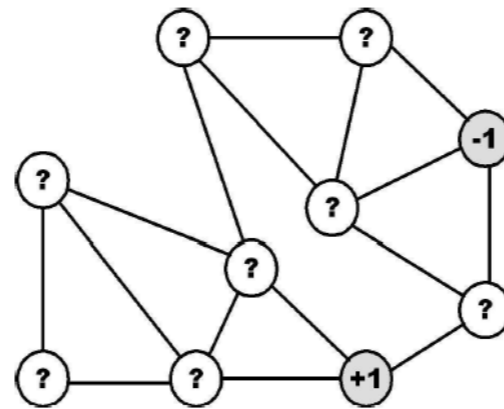
# Semi-supervised learning



$$y : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ +1 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}$$

$$\min_{x \in \mathbb{R}^N} \|y - x\|_2^2 + \alpha x^T L x,$$

# Semi-supervised learning



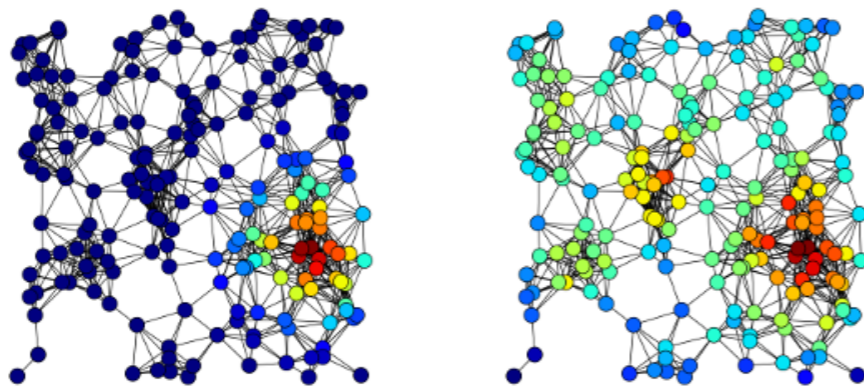
$$y : \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ +1 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}$$

$$\min_{x \in \mathbb{R}^N} \|y - x\|_2^2 + \alpha x^T L x,$$

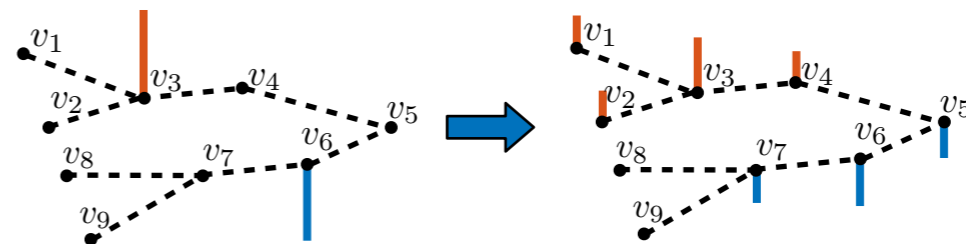
- learning by assuming smoothness of predicted labels
- this is equivalent to a denoising problem for graph signal  $y$

# GSP and the literature

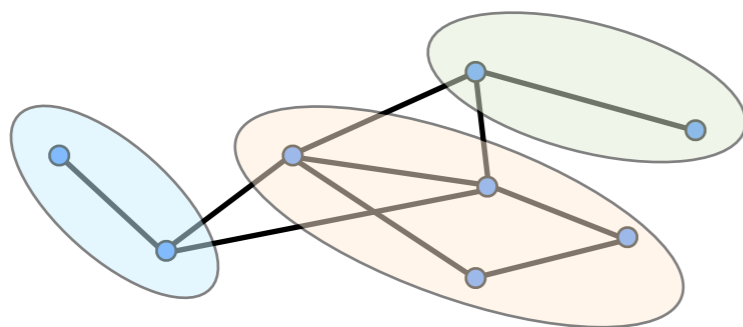
centrality, diffused information, cluster membership, node labels (and node features in general) can ALL be viewed as graph signals



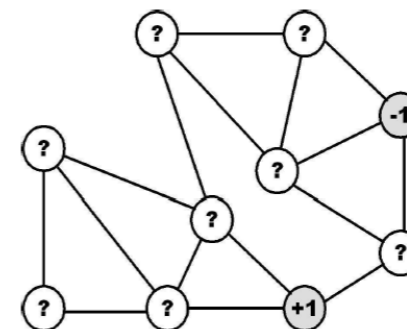
network science



diffusion on graphs



unsupervised learning (dimensionality reduction, clustering)

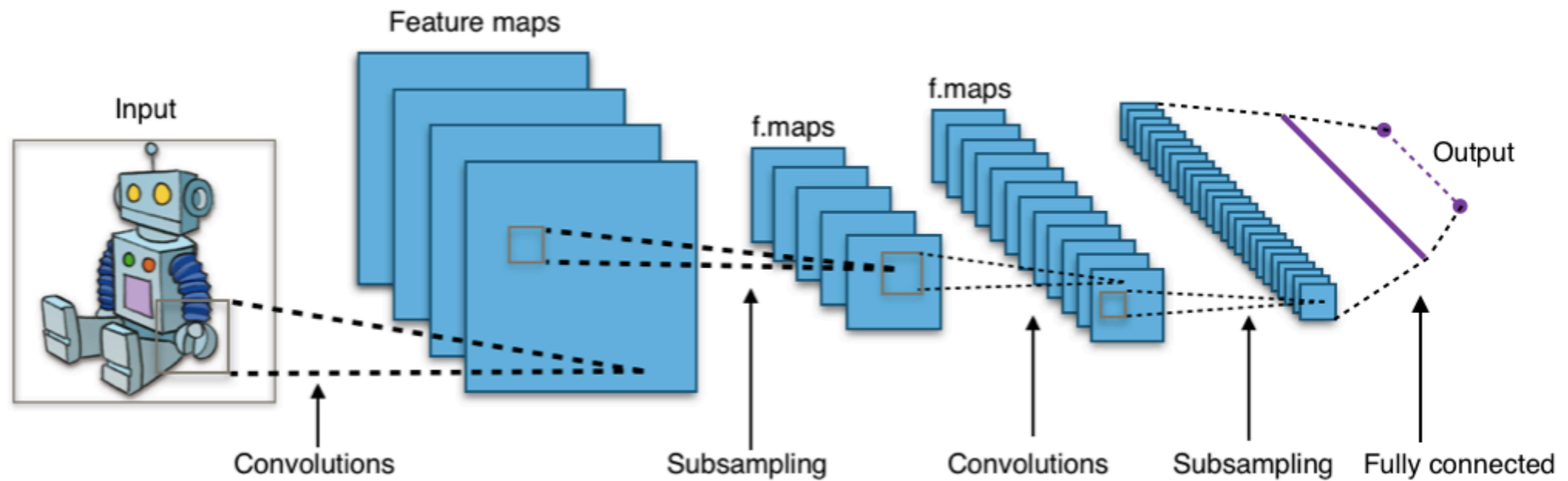


semi-supervised learning

# Outline

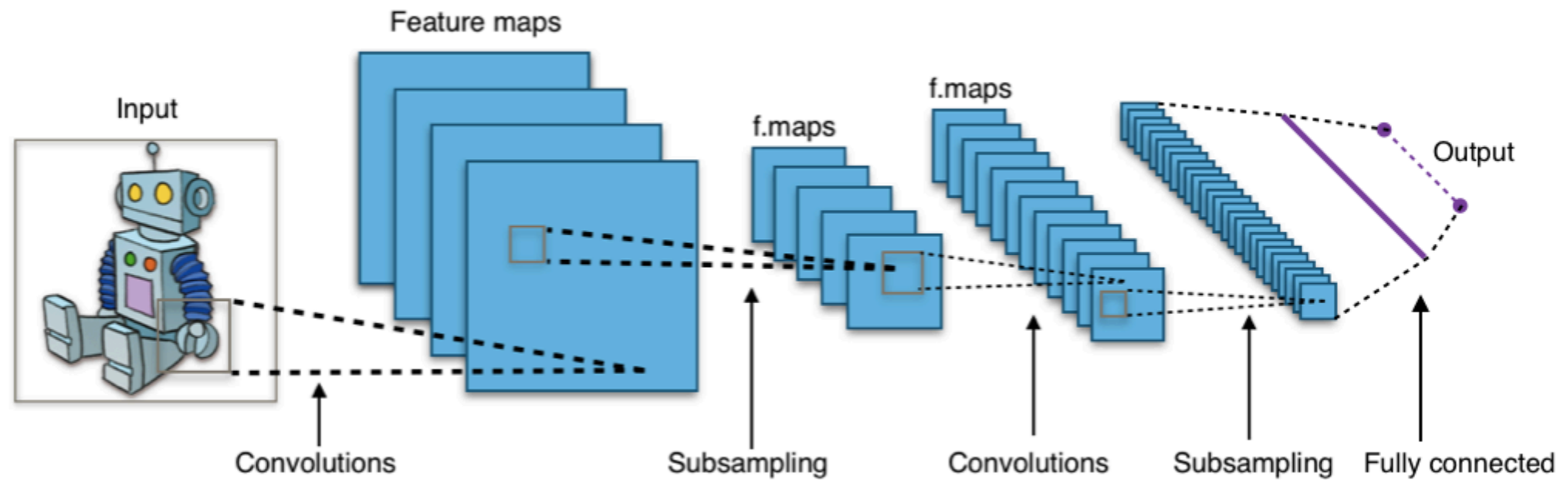
- Graph signal processing (GSP): Basic concepts
- Filtering of graph signals: Basic tool of GSP
- Connection with literature
- Research topics inspired by GSP
- Applications

# Deep learning on graphs

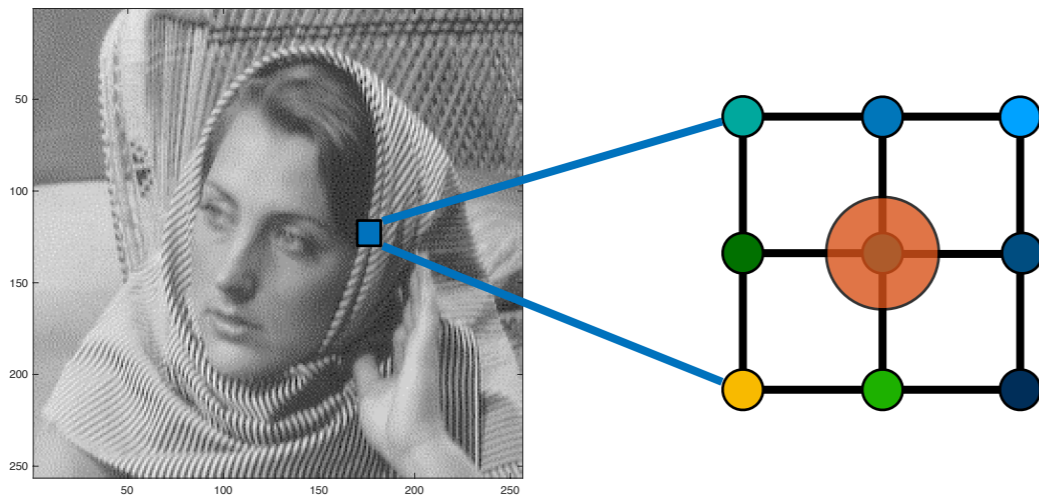


[https://en.wikipedia.org/wiki/File:Typical\\_cnn.png](https://en.wikipedia.org/wiki/File:Typical_cnn.png)

# Deep learning on graphs

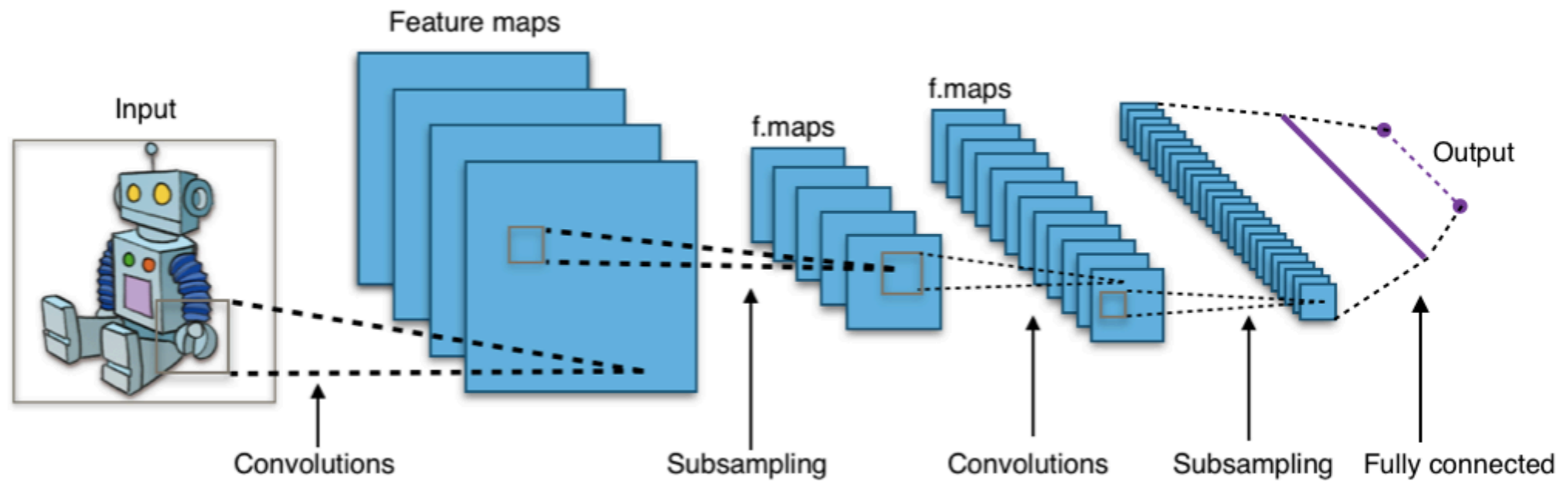


[https://en.wikipedia.org/wiki/File:Typical\\_cnn.png](https://en.wikipedia.org/wiki/File:Typical_cnn.png)

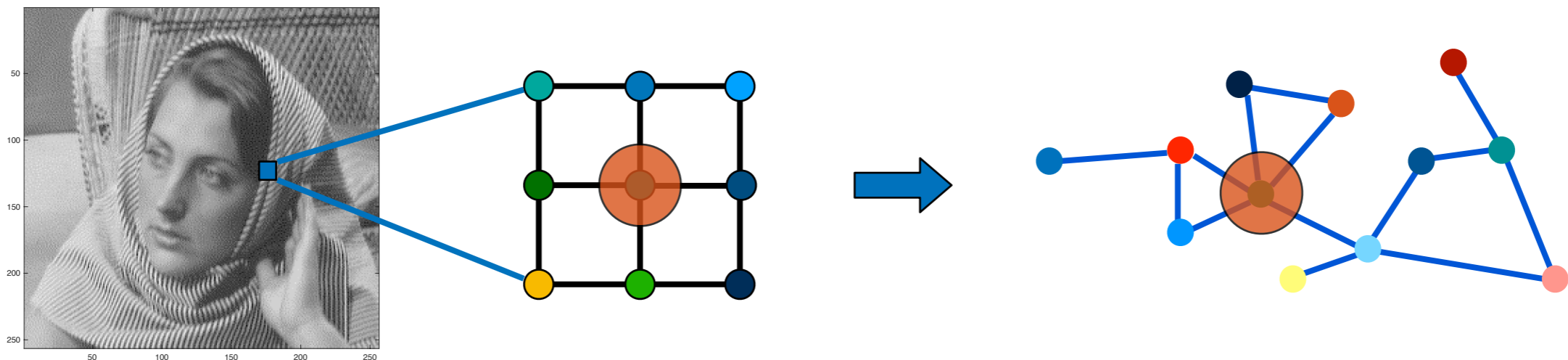




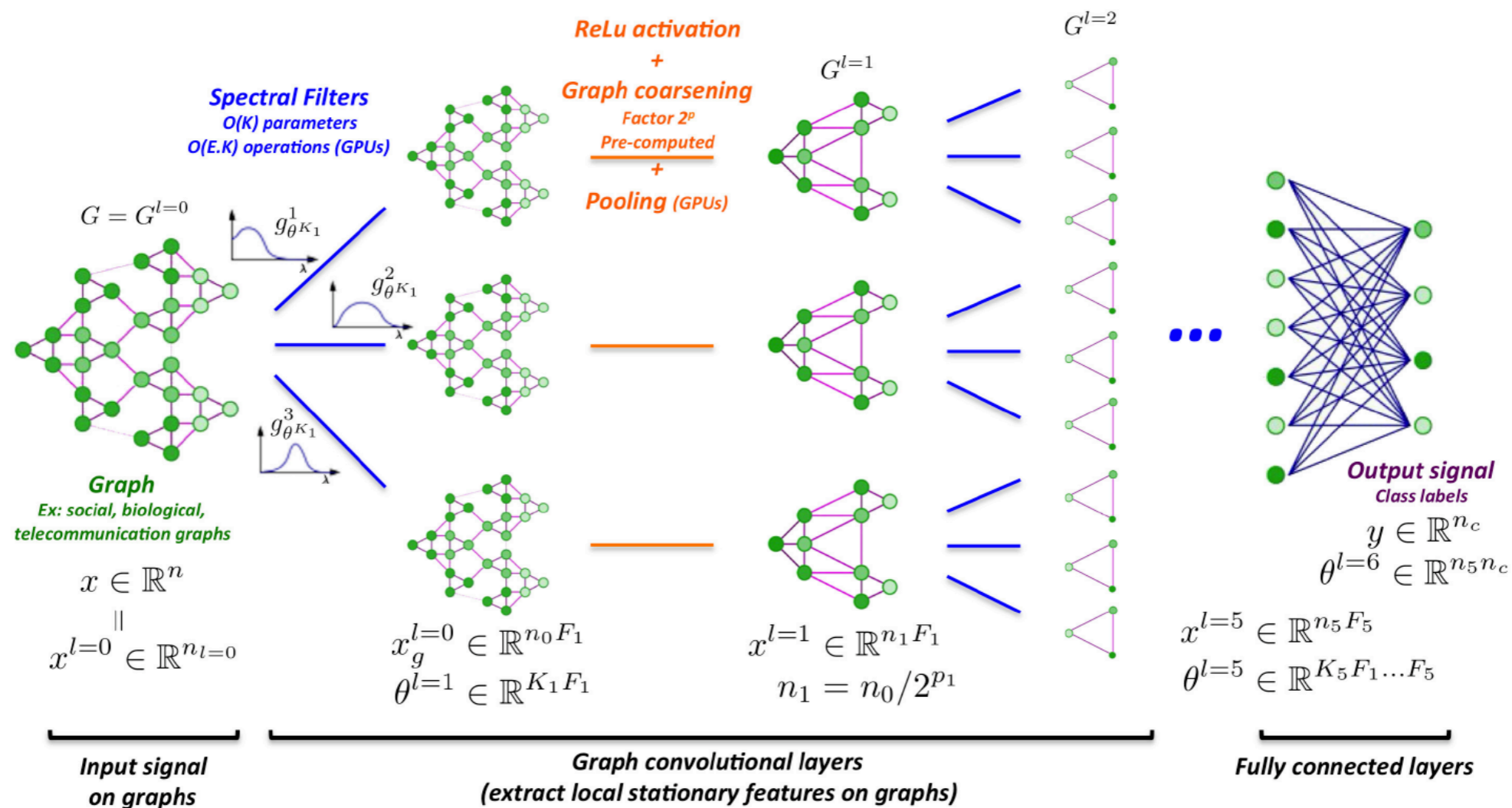
# Deep learning on graphs



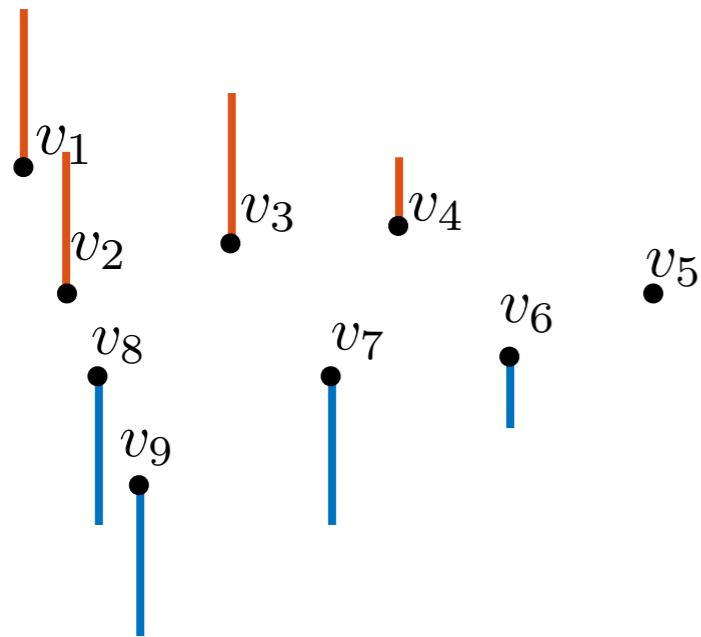
[https://en.wikipedia.org/wiki/File:Typical\\_cnn.png](https://en.wikipedia.org/wiki/File:Typical_cnn.png)



# Deep learning on graphs

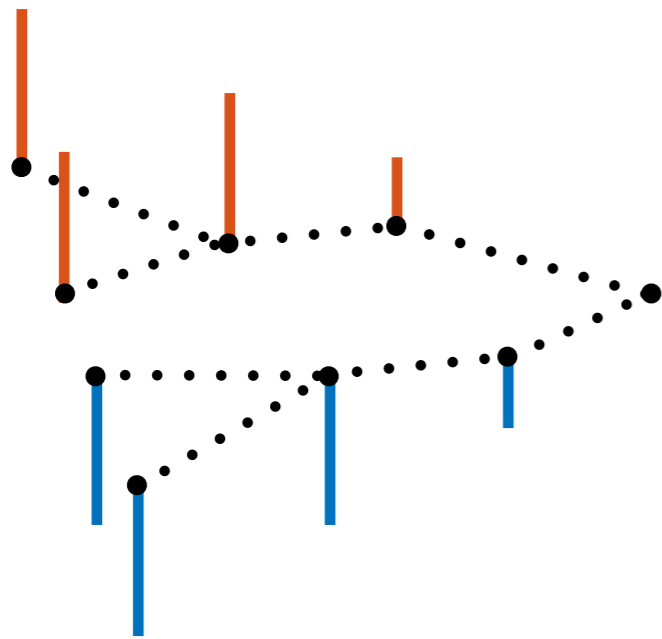


# Inferring graph structure from data

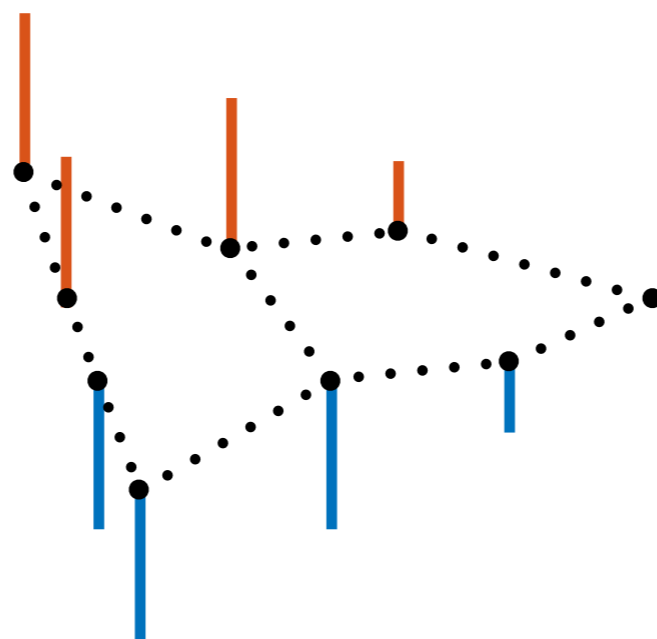


# Inferring graph structure from data

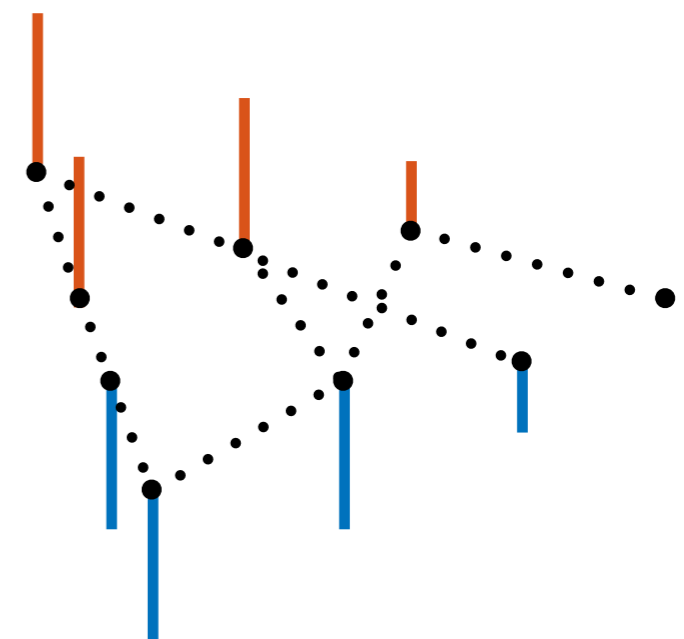
$\mathcal{G}_1$



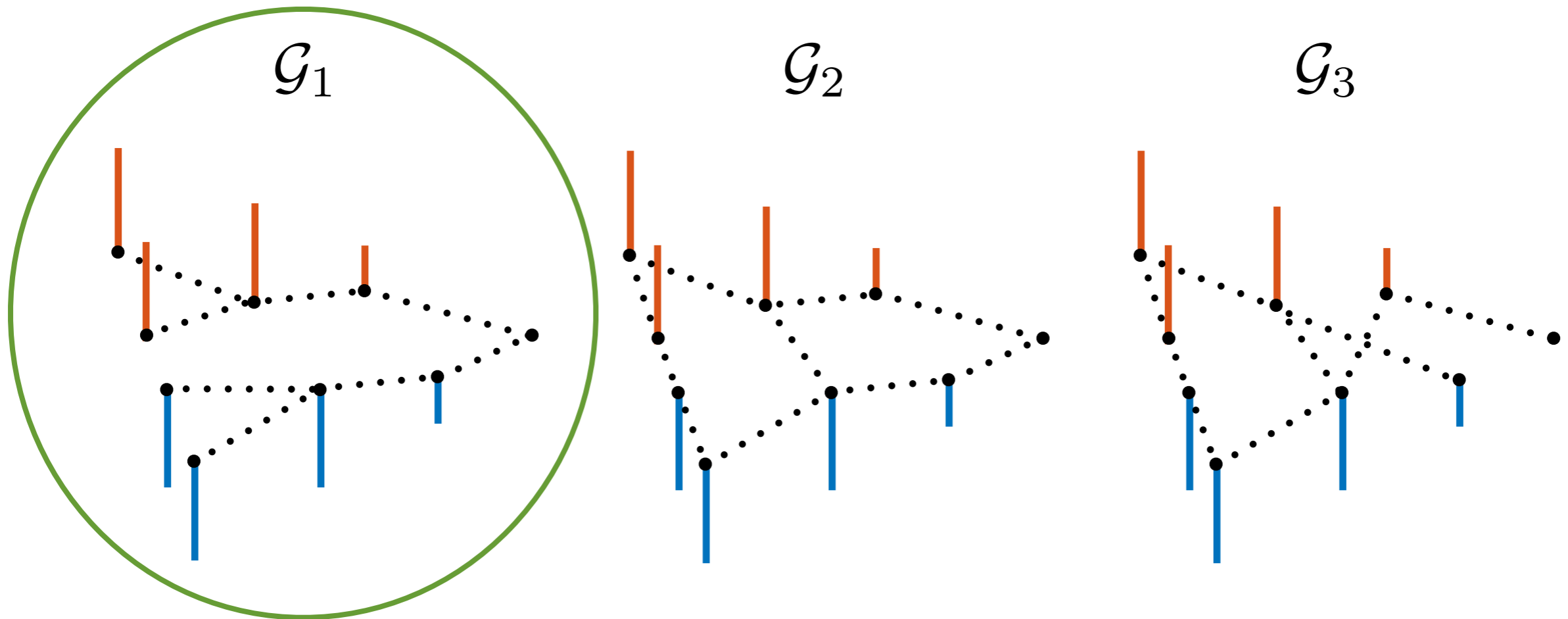
$\mathcal{G}_2$



$\mathcal{G}_3$



# Inferring graph structure from data



**idea: choose the structure that enforces certain signal characteristics**

# Outline

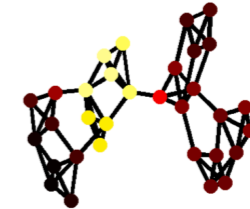
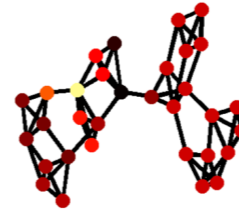
- Graph signal processing (GSP): Basic concepts
- Filtering of graph signals: Basic tool of GSP
- Connection with literature
- Research topics inspired by GSP
- Applications

# Application I: Community detection

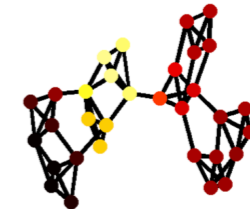
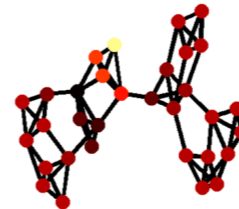
spectral graph wavelets  
at different scales:

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

NODE  
A:



NODE  
B:



CORR.  
COEF.:

-0.50

0.97

**small scale**

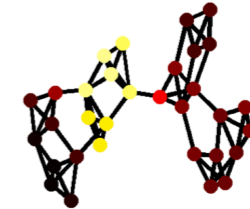
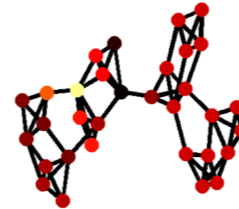
**large scale**

# Application I: Community detection

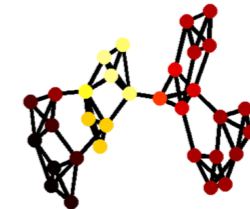
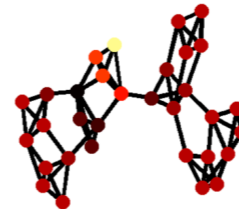
spectral graph wavelets  
at different scales:

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

NODE  
A:



NODE  
B:



CORR.  
COEF.:

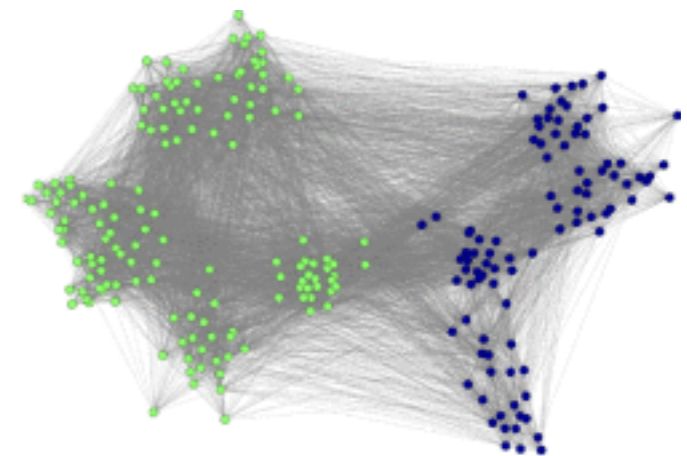
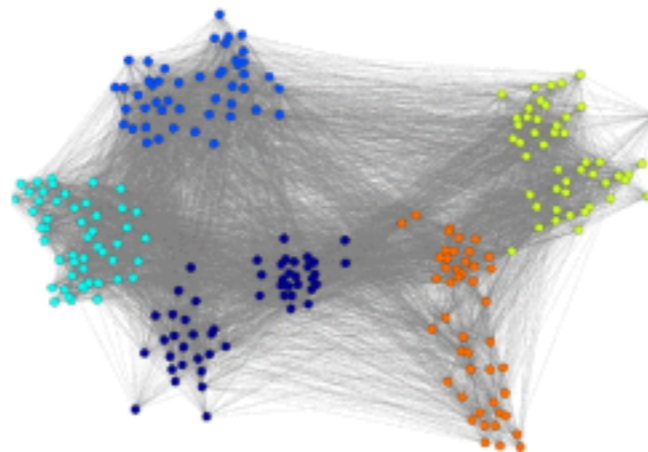
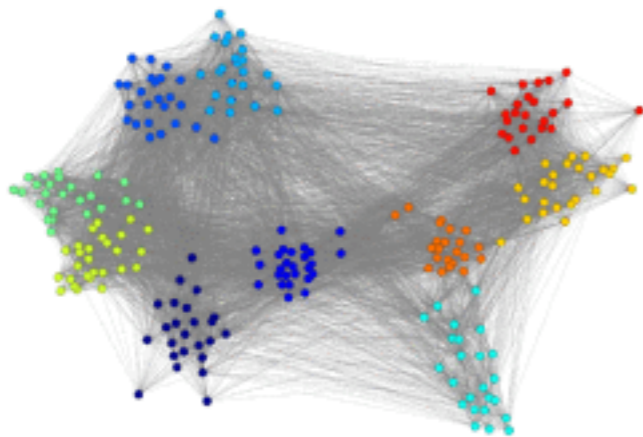
-0.50

0.97

**small scale**

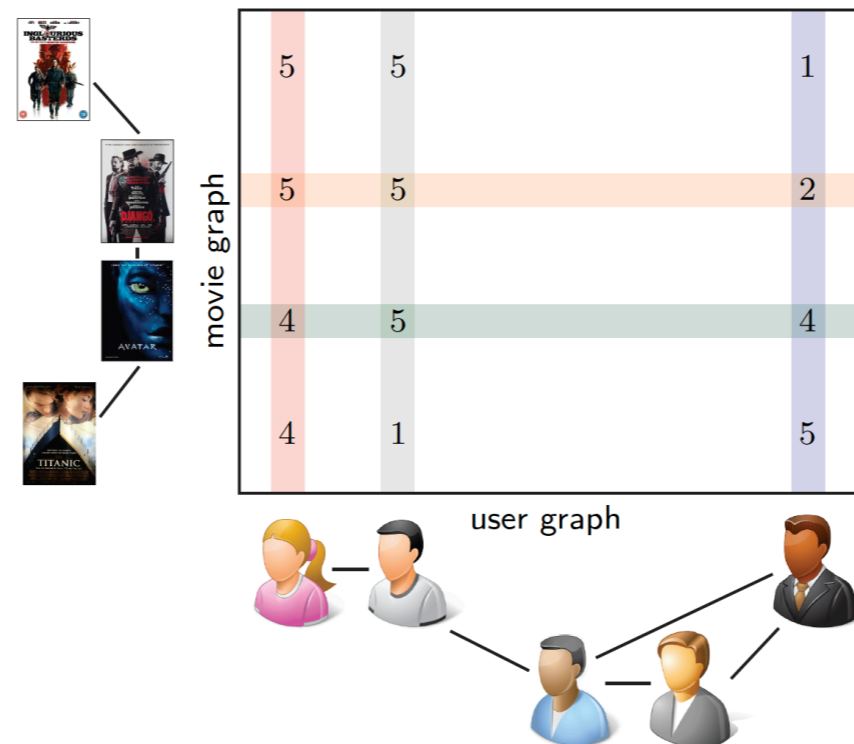
**large scale**

multi-scale community  
detection:

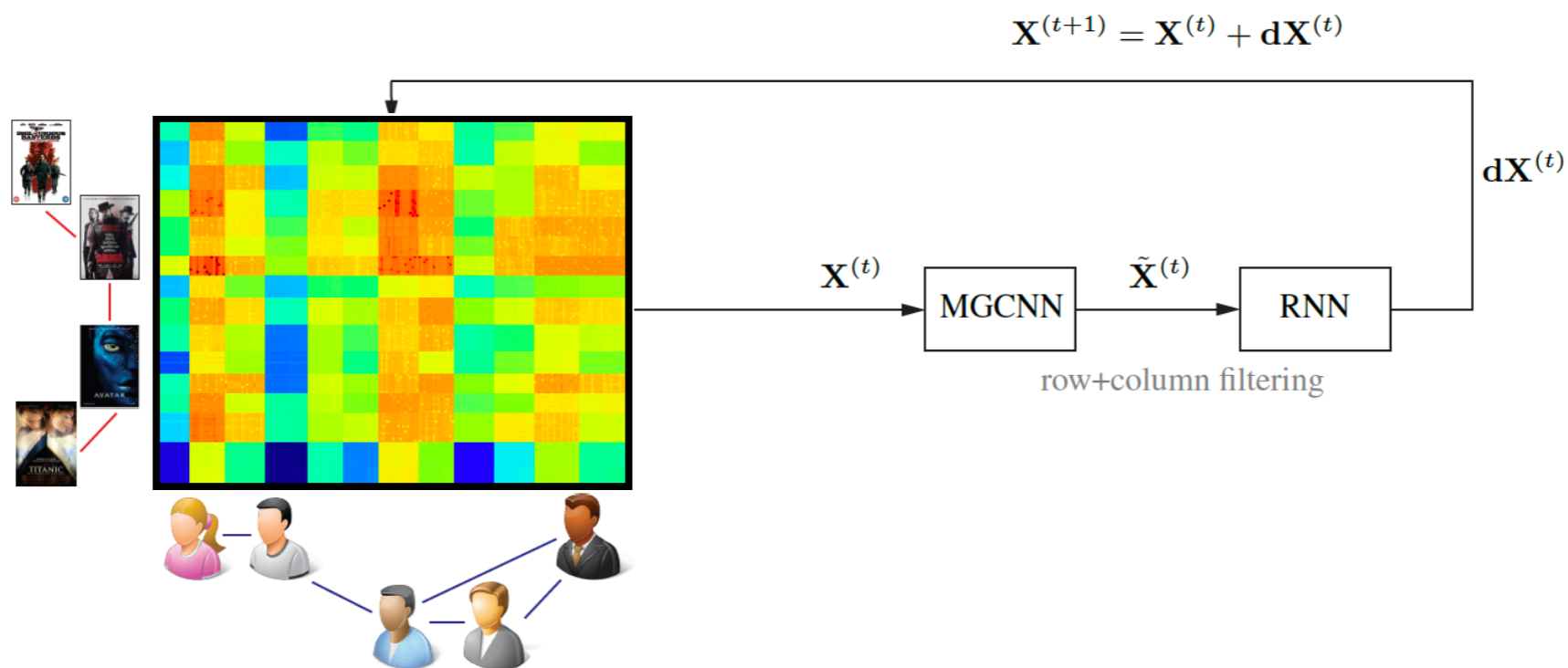




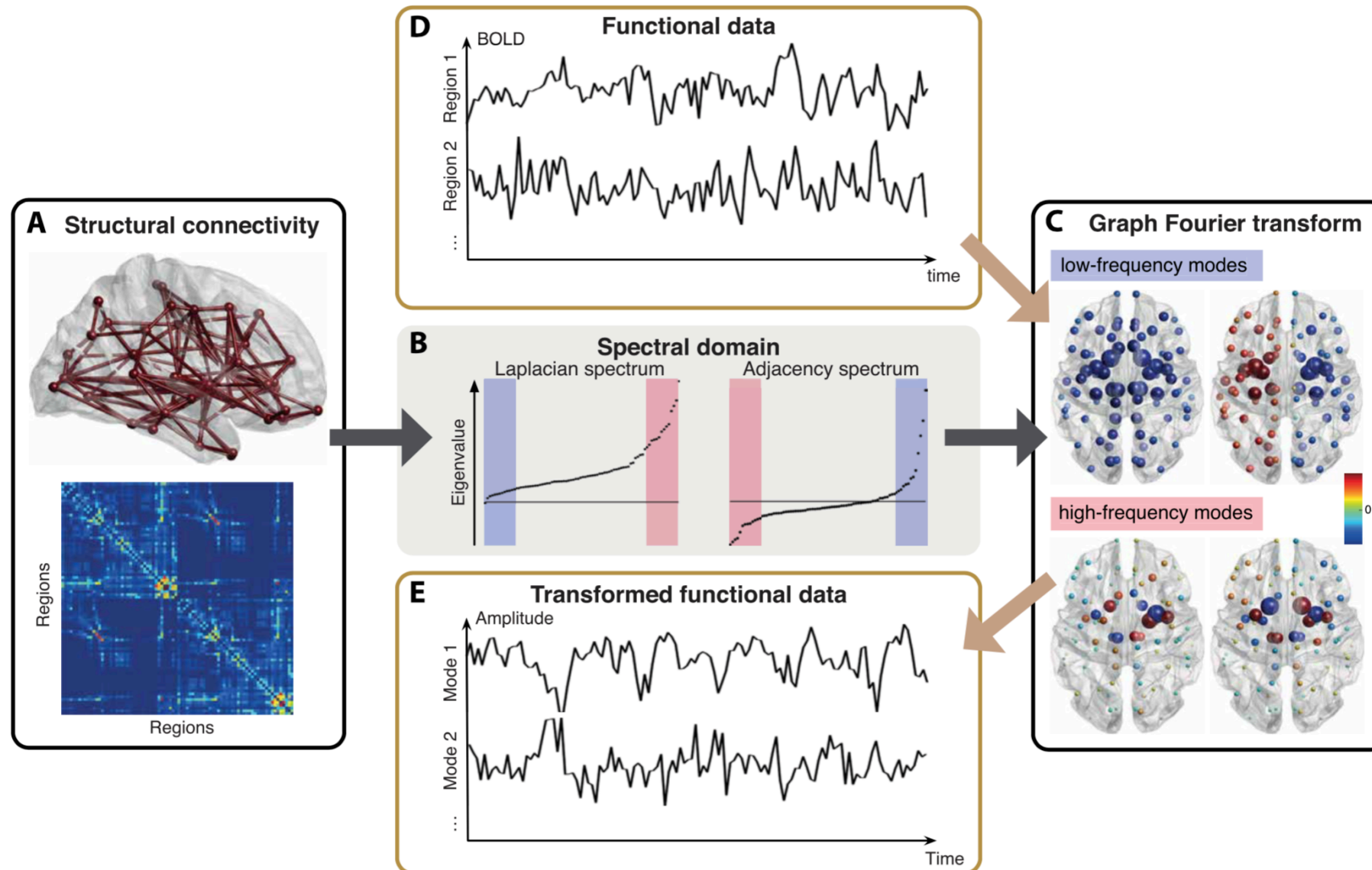
# Application II: Recommender systems



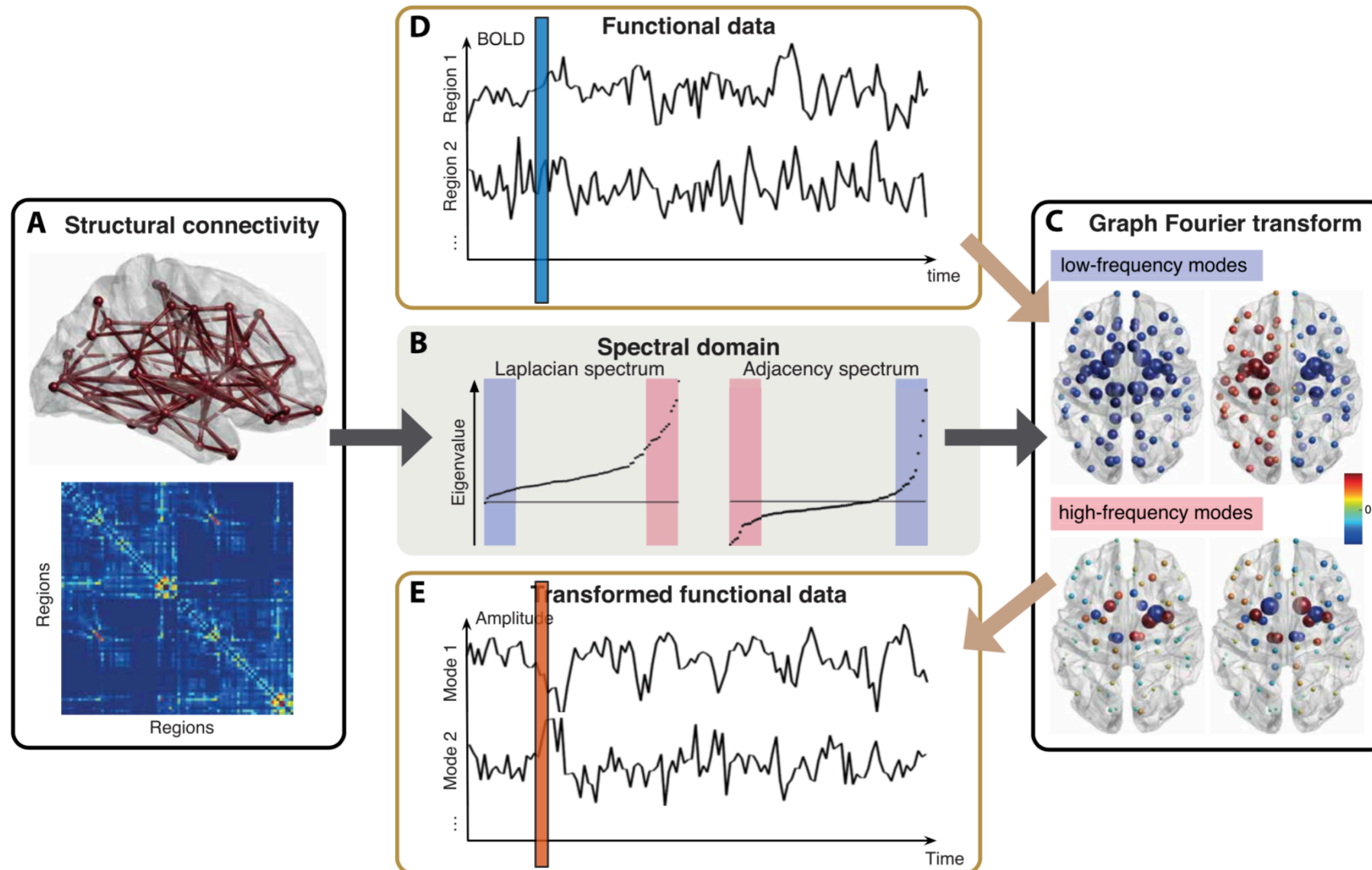
# Application II: Recommender systems



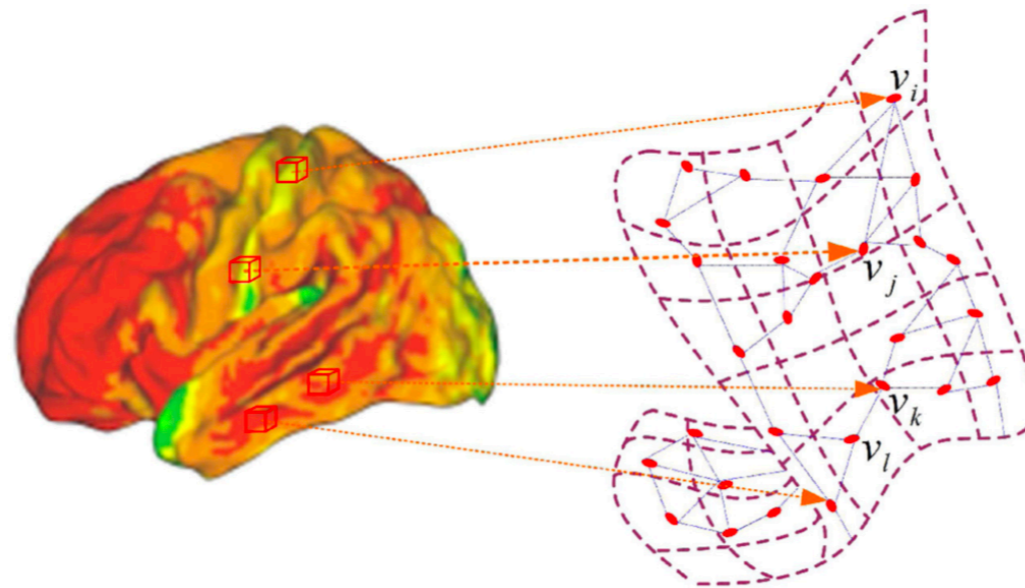
# Application III: Functional brain imaging



# Application III: Functional brain imaging

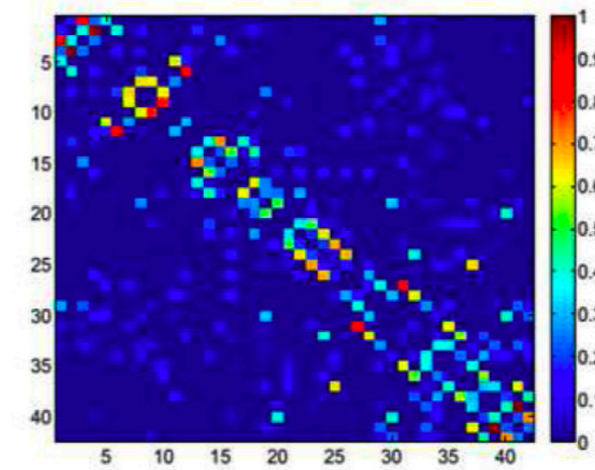
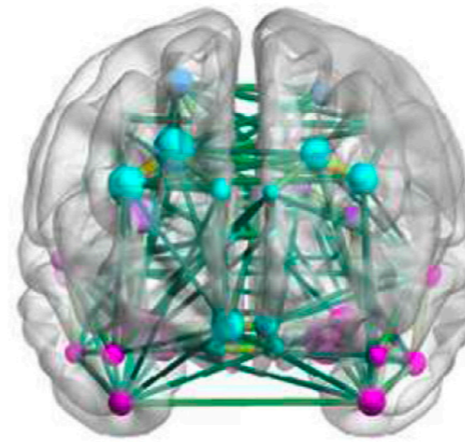
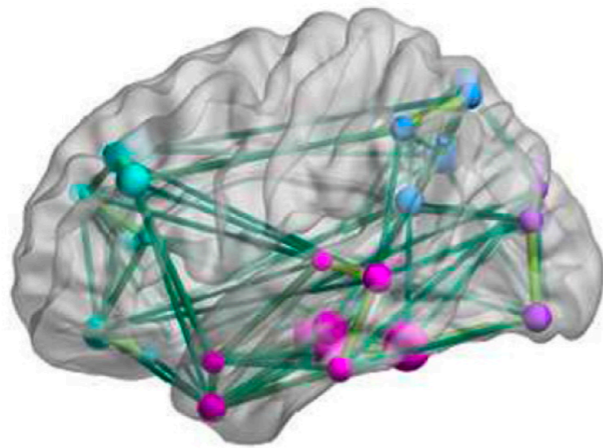


# Application IV: Inferring brain connectivity

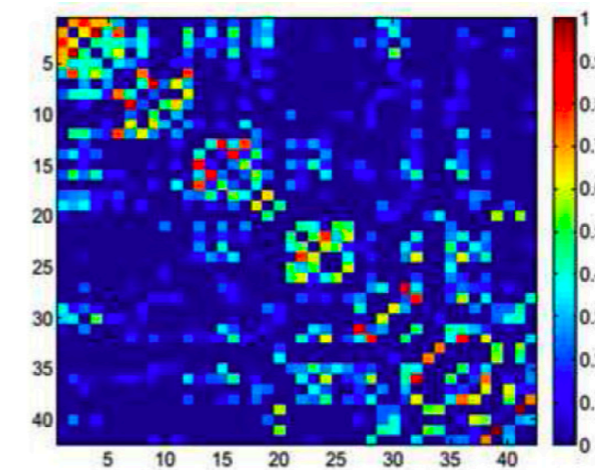
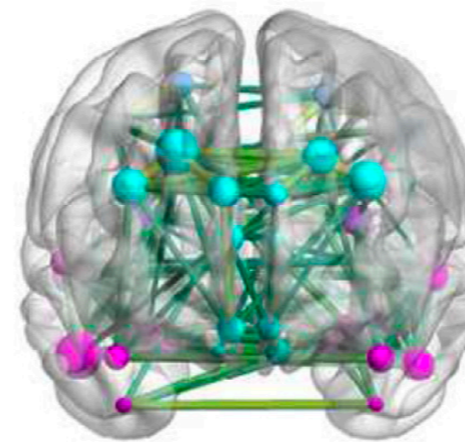
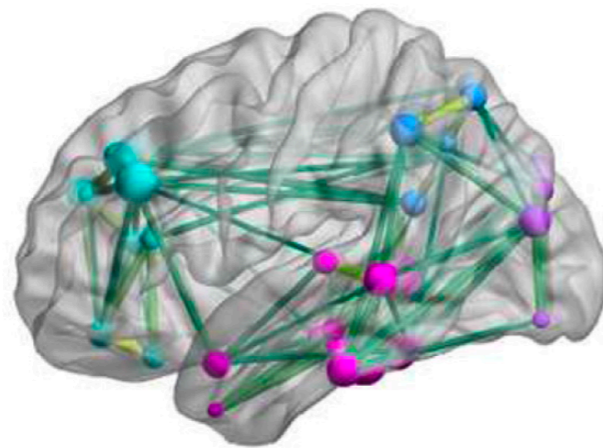


# Application IV: Inferring brain connectivity

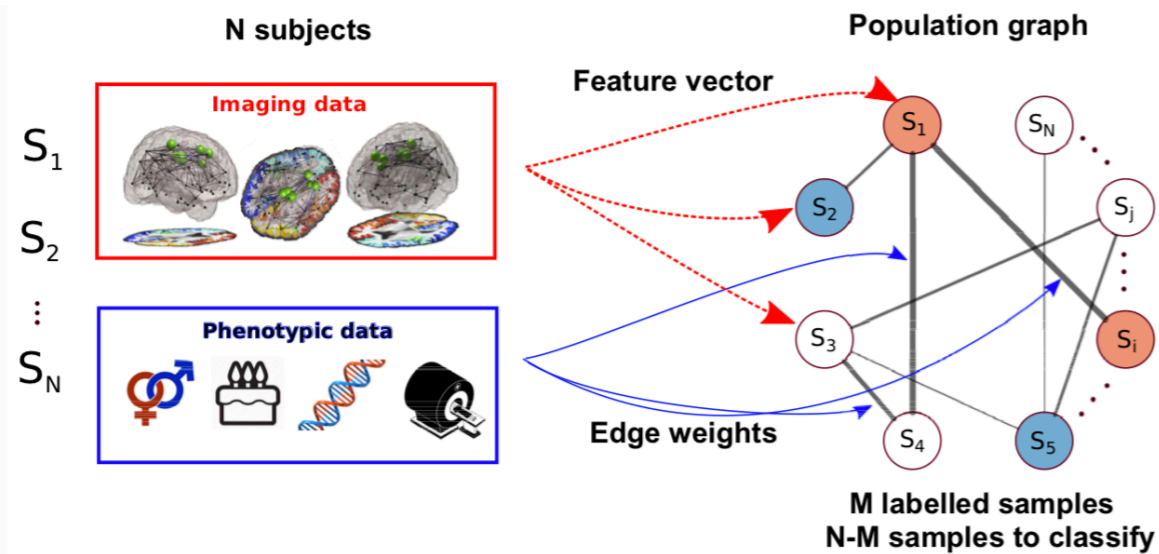
Alzheimer's disease



normal control

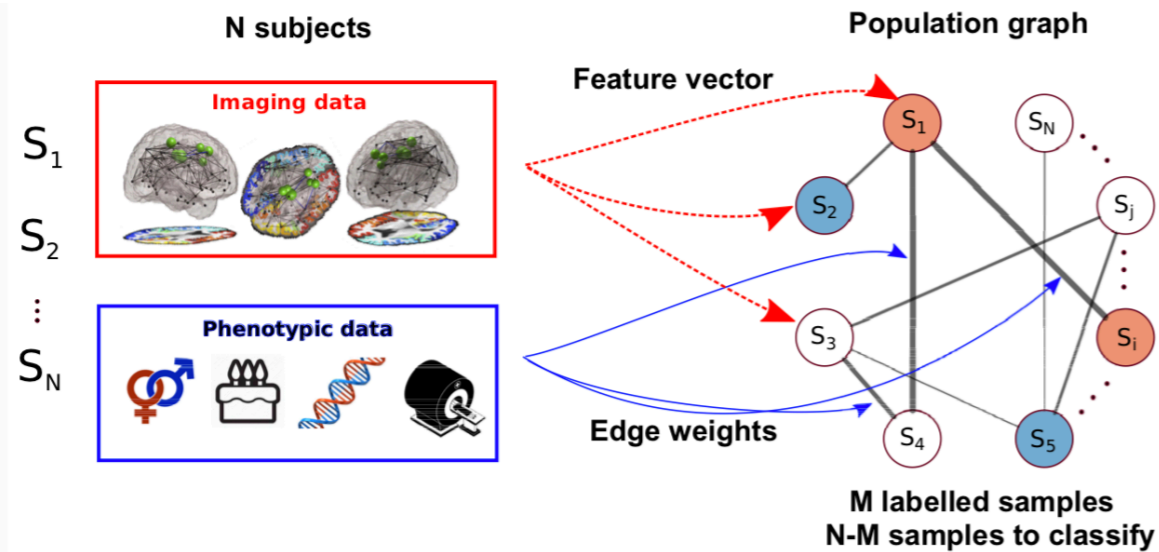


# Application V: Disease classification



# Application V: Disease classification

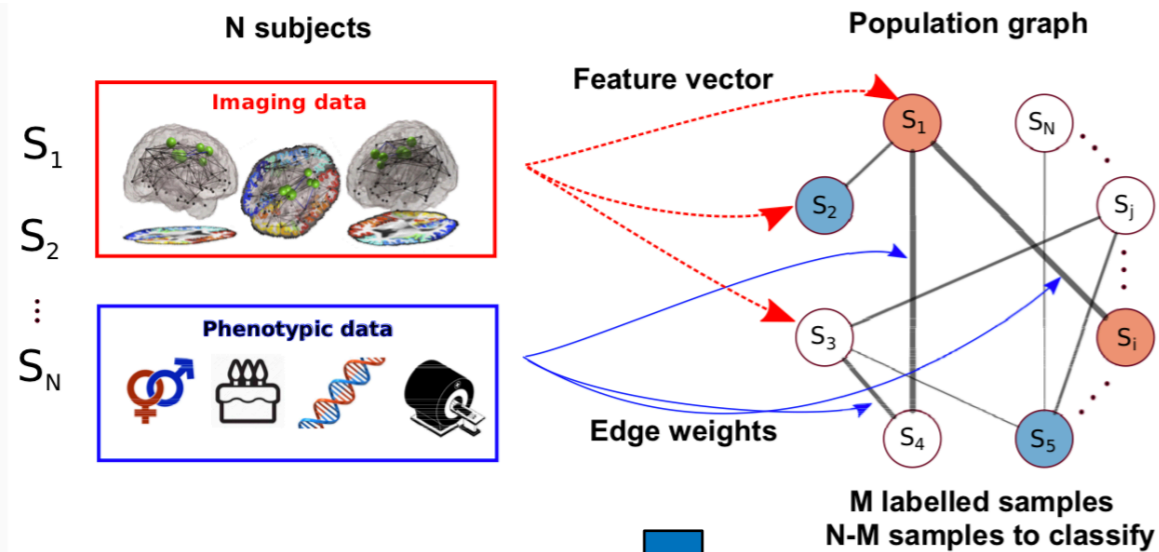
features extracted from brain analysis





# Application V: Disease classification

features extracted from brain analysis

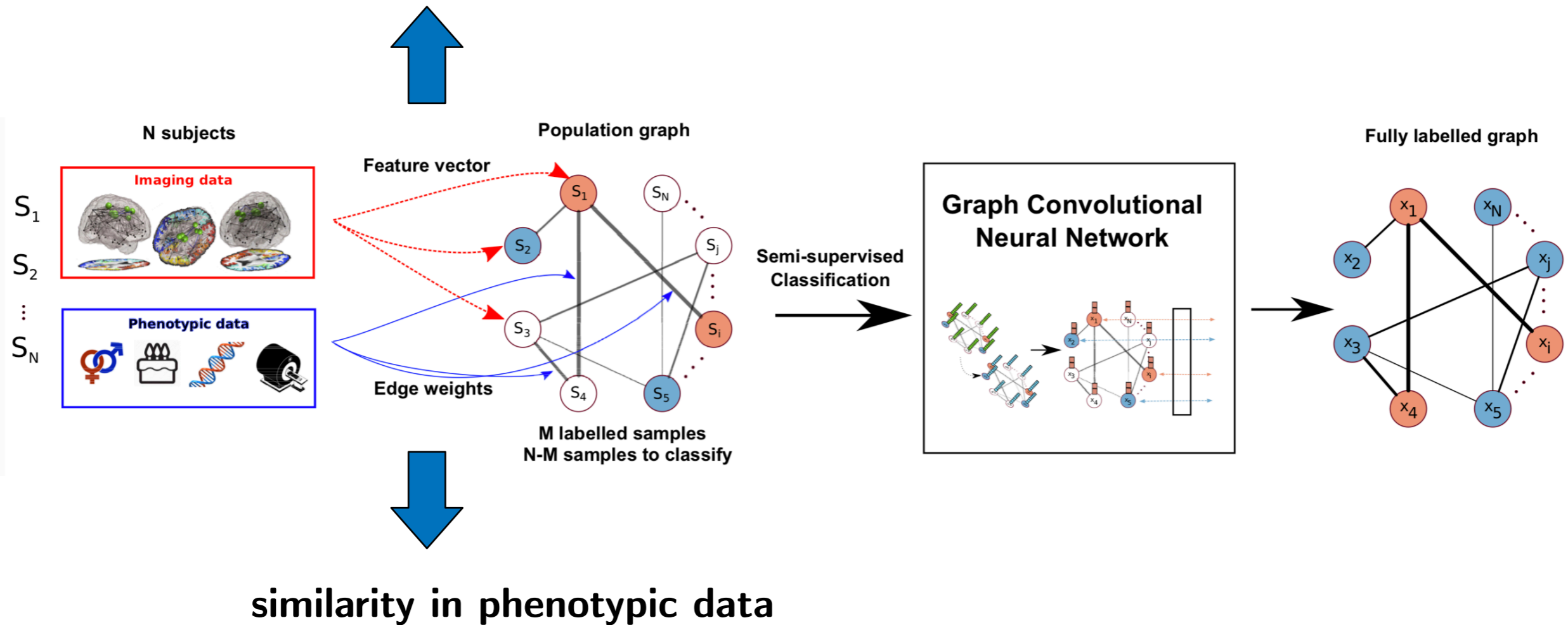


similarity in phenotypic data



# Application V: Disease classification

features extracted from brain analysis




# Papers & Resources

- Tutorial/overview papers:

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

## The Emerging Field of Signal Processing on Graphs



Adaptation and Learning over Complex Networks

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

1853-0889/18/01 000013EE © 2018 IEEE. This journal is part of the IEEE Signal Processing Magazine. For more information on this journal, please refer to the journal website at [http://www.ieee.org/publications\\_standards/publications\\_standards\\_info.html](http://www.ieee.org/publications_standards/publications_standards_info.html).

INVITED PAPER

## 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, Fellow IEEE, PASCAL FROSSARD, Fellow IEEE, JELJENA KOVAČEVIĆ, Fellow IEEE, JOÏF M. F. MOURA, 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

**1. 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 fluorescent markers, 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.

Typical graphs that are used to represent common real-world data include Erdős-Rényi graphs, ring graphs, random geometric graphs, small-world graphs, power-law graphs, nearest-neighbor graphs, scale-free graphs, and many others. These model networks with random connections (Erdős-Rényi graphs), networks of brain neurons (small-world graphs), social networks (scale-free graphs), and others.

As in classical signal processing, graph signals can have properties, such as smoothness, that need to be appropriately defined. They can also be represented via basic atoms and can have a spectral representation. In particular, the graph Fourier transform allows us to develop the intuition gathered in the classical setting and extend it to graphs; we can talk about the notions of frequency and bandlimitedness,

Graphs offer the ability to model such data and complex interactions among them. For example, users on Twitter can be modeled as nodes while their friend connections can be modeled as edges. This paper explores adding attributes to such nodes and modeling those attributes on graphs; for example, year of graduation in a social network, temperature in a given city on a given day in a weather network, etc. Doing so requires us to extend classical signal processing concepts and tools such as Fourier transforms, filtering, and frequency response to data residing on graphs. It also leads us to tackle complex tasks such as sampling in a principled way. The field that gathers all these questions under a common umbrella is graph signal processing (GSP) [2], [3].

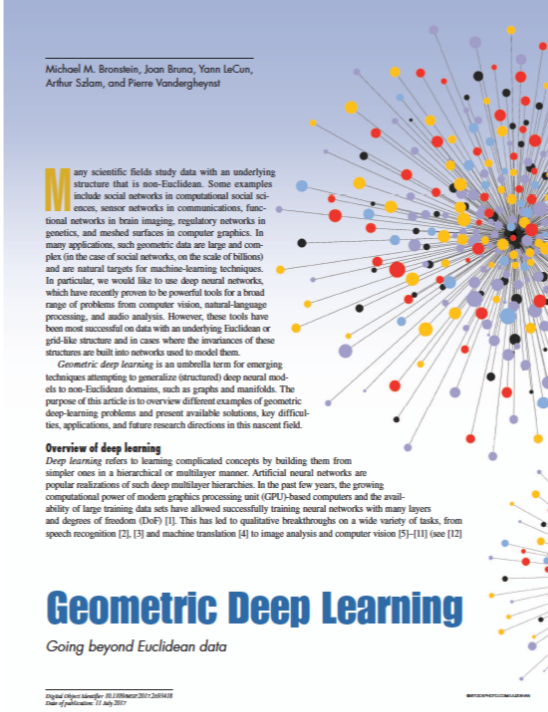
While the precise definition of a graph signal will be given later in the paper, let us assume for now that a graph signal is a set of values residing on a set of nodes. These nodes are connected via (possibly weighted) edges. As in classical signal processing, such signals can stem from a variety of domains; unlike in classical signal processing, however, the underlying graphs can tell a fair amount about those signals through their structure. Different types of graphs model different types of networks that these nodes represent.

Typical graphs that are used to represent common real-world data include Erdős-Rényi graphs, ring graphs, random geometric graphs, small-world graphs, power-law graphs, nearest-neighbor graphs, scale-free graphs, and many others. These model networks with random connections (Erdős-Rényi graphs), networks of brain neurons (small-world graphs), social networks (scale-free graphs), and others.

As in classical signal processing, graph signals can have properties, such as smoothness, that need to be appropriately defined. They can also be represented via basic atoms and can have a spectral representation. In particular, the graph Fourier transform allows us to develop the intuition gathered in the classical setting and extend it to graphs; we can talk about the notions of frequency and bandlimitedness,

1853-0889/18/01 000013EE © 2018 IEEE. This journal is part of the IEEE Signal Processing Magazine. For more information on this journal, please refer to the journal website at [http://www.ieee.org/publications\\_standards/publications\\_standards\\_info.html](http://www.ieee.org/publications_standards/publications_standards_info.html).

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



## Geometric Deep Learning

Going beyond Euclidean data

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.

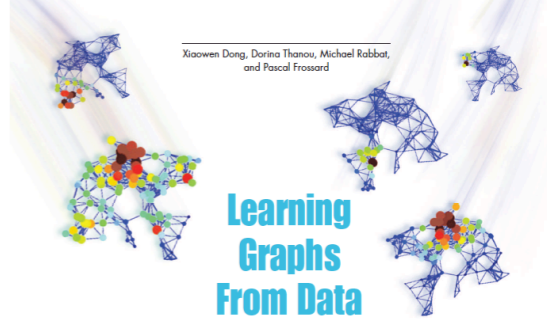
**Geometric deep learning** is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available 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 hierarchies. 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]

1853-0889/17/000170017EE © 2017 IEEE. This journal is part of the IEEE Signal Processing Magazine. For more information on this journal, please refer to the journal website at [http://www.ieee.org/publications\\_standards/publications\\_standards\\_info.html](http://www.ieee.org/publications_standards/publications_standards_info.html).

Xiaowen Dong, Darina Thanou, Michael Rabat, and Pascal Frossard



## Learning Graphs From Data

A signal representation perspective

The construction of a meaningful graph topology plays a crucial role in the effective representation, processing, analysis, and visualization of structured data. When a natural choice of the graph is not readily available from the data sets, it is thus desirable to infer or learn a graph topology from the data. In this article, we survey solutions to the problem of graph learning, including classical viewpoints from statistics and physics, and more recent approaches that adopt a graph signal processing (GSP) perspective. We further emphasize the conceptual similarities and differences between classical and GSP-based graph-inference methods and highlight the potential advantage of the latter in a number of theoretical and practical scenarios. We conclude with several open issues and challenges that are keys to the design of future signal processing and machine-learning algorithms for learning graphs from data.

**Introduction**

Modern data analysis and processing tasks typically involve large sets of structured data, where the structure carries critical information about the nature of the data. One can find numerous examples of such data sets in a wide diversity of application domains, including transportation networks, social networks, computer networks, and brain networks. Typically, graphs are used as mathematical tools to describe the structure of such data. They provide a flexible way of representing the relationship between data entities. In the past decade, numerous signal processing and machine-learning algorithms have been introduced for analyzing structured data on a priori known graphs [1]–[3]. However, there are often settings where the graph is not readily available, and the structure of the data has to be estimated to permit the effective representation, processing, analysis, or visualization of the data. In this case, a crucial task is to infer a graph topology that describes the characteristics of the data observations, hence capturing the underlying relationship between these entities.

Consider as an example in brain signal analysis: suppose we are given blood-oxygen-level-dependent (BOLD) signals, i.e., time series extracted from functional magnetic resonance imaging data that reflect the activities of different regions of the brain. An area of significant interest in neuroscience is the inference of functional connectivity, i.e., to capture the relationship between brain regions that correlate or synchronize given a certain condition of a patient, which may help reveal underpinnings of some neurodegenerative diseases (see Figure 1). This leads to the problem of inferring a graph structure, given the multivariate BOLD time series data.

Formally, the problem of graph learning is the following: given  $M$  observations on  $N$  variables or data entities represented in a data matrix  $X \in \mathbb{R}^{M \times N}$ , and given some prior knowledge (e.g., distribution, data model, and so on) about

1853-0889/18/01 000013EE © 2018 IEEE. This journal is part of the IEEE Signal Processing Magazine. For more information on this journal, please refer to the journal website at [http://www.ieee.org/publications\\_standards/publications\\_standards\\_info.html](http://www.ieee.org/publications_standards/publications_standards_info.html).

- More available at: <http://web.media.mit.edu/~xdong/resource.html>