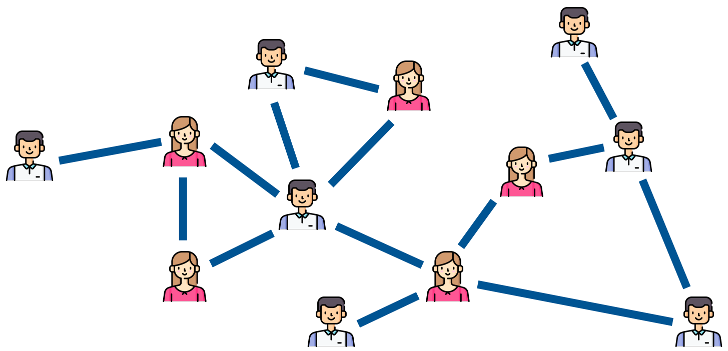
Recent advances in learning with graphs

Xiaowen Dong

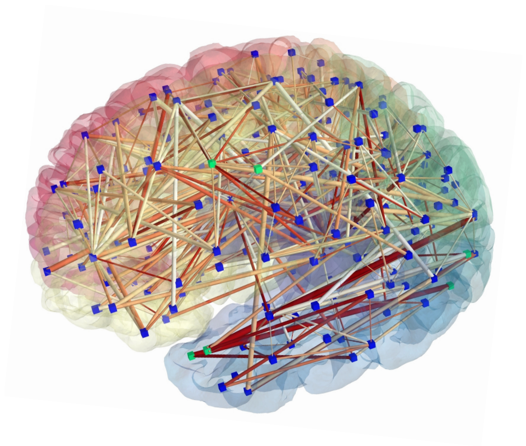
Department of Engineering Science
University of Oxford

Networks are pervasive

traffic network



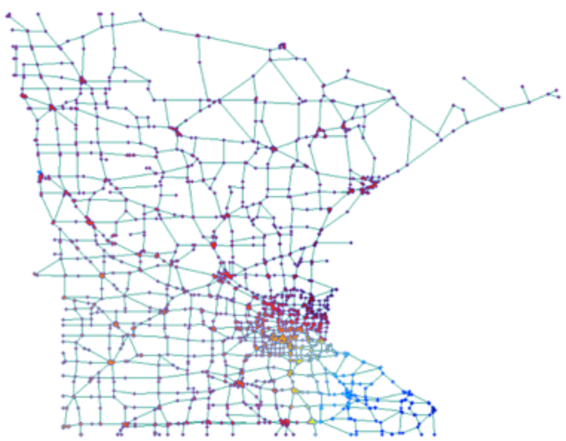
social network



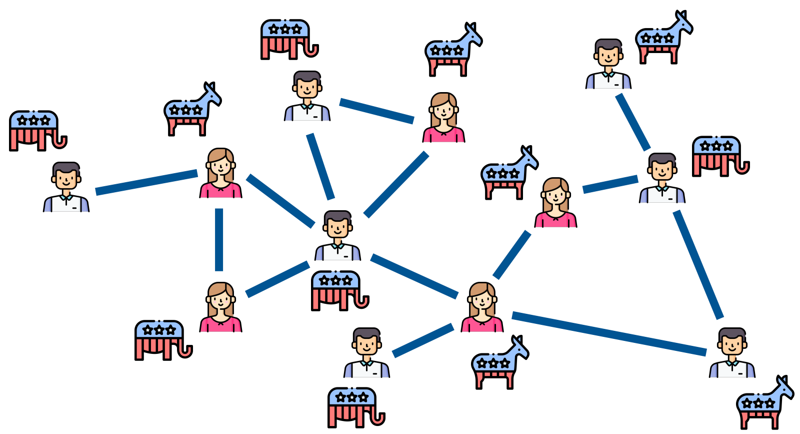
brain network

networks are mathematically represented by graphs

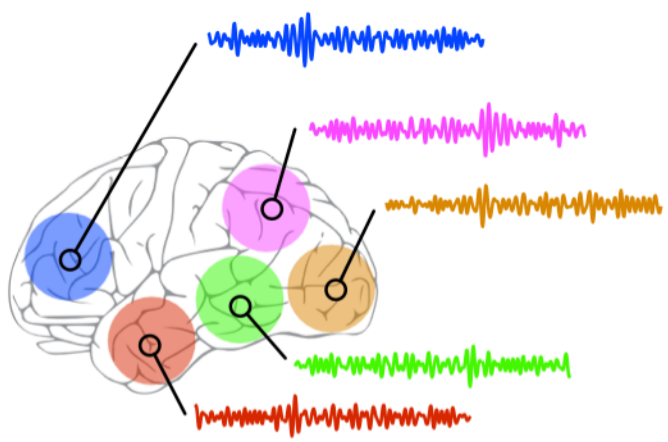
Data collected in networks are pervasive



congestion in road junctions



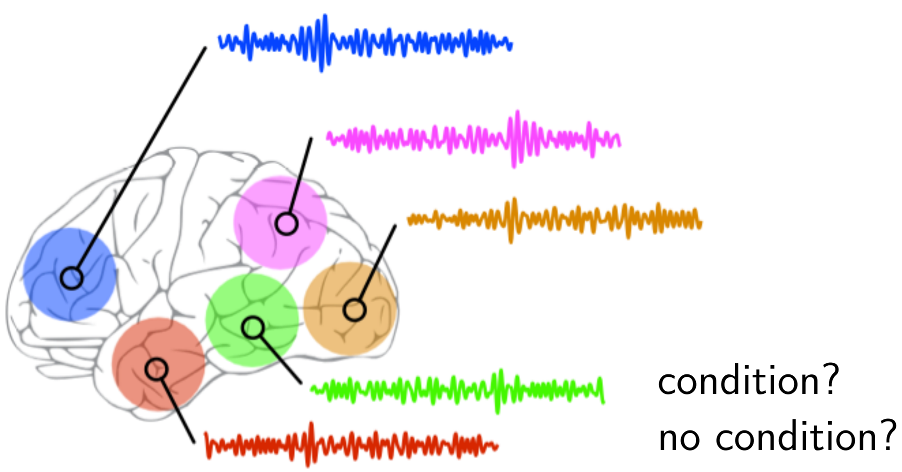
preferences of individuals



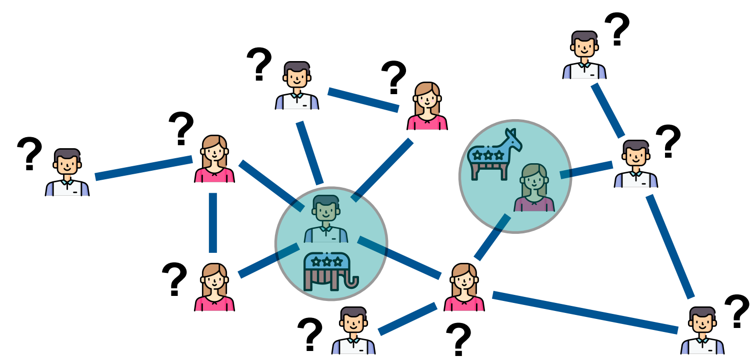
activities in brain regions

from graphs to graph-structured data

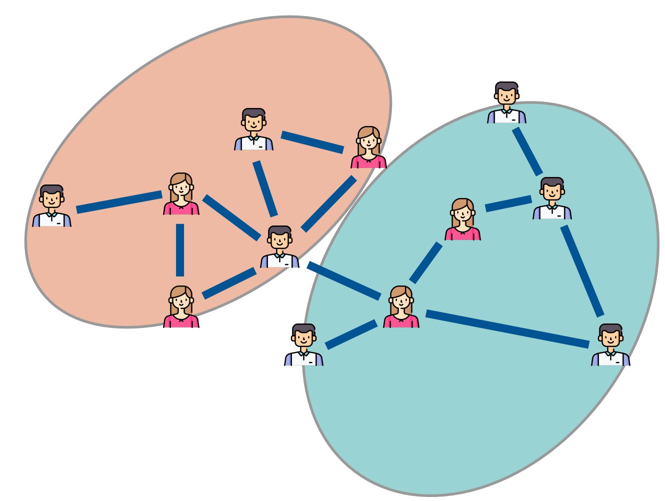
Learning with graph-structured data



graph-level classification
(supervised)



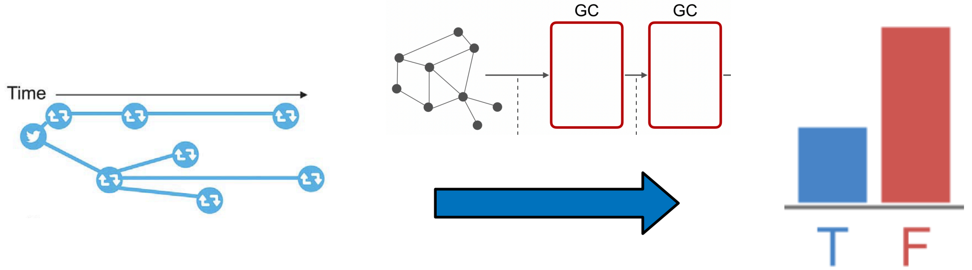
node-level classification
(semi-supervised)



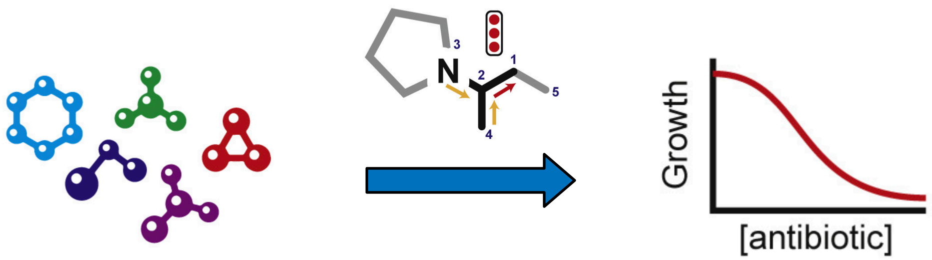
graph clustering
(unsupervised)

Exciting possibilities enabled by graph ML

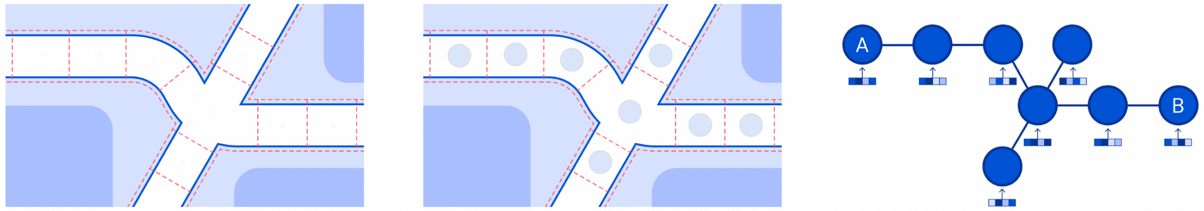
fake news detection

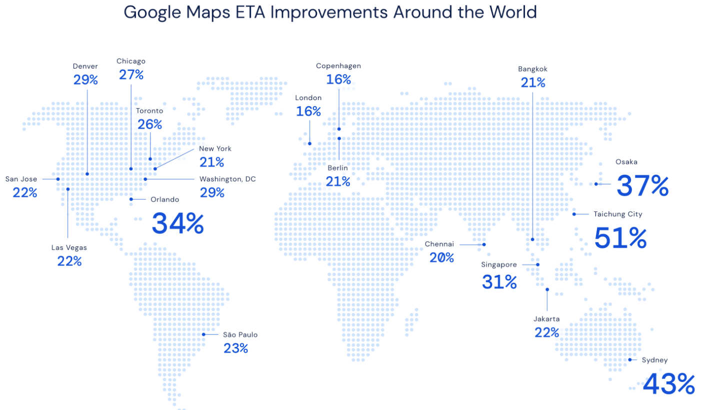


drug discovery



traffic prediction



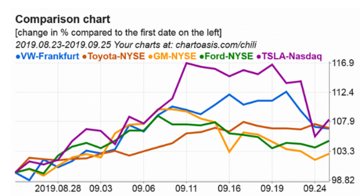


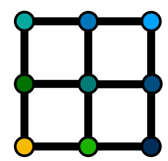
Monti et al., "Fake news detection on social media using geometric deep learning," ICLR Workshop, 2019.
Stokes et al., "A deep learning approach to antibiotic discovery," Cell, 2020.
Derrow-Pinion et al., "ETA prediction with graph neural networks in Google Maps," CIKM, 2021.

Classical ML vs Graph ML

Classical ML

regular domain
(real line, 2D grid)





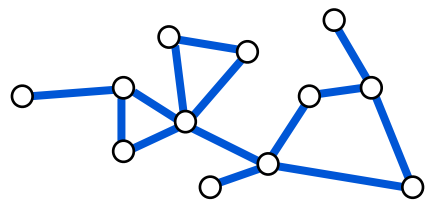
$f(X)$

time series forecasting

image classification

Graph ML

irregular domain
(graphs)



$\mathcal{G}$

X

$f(\mathcal{G}, X)$

node classification

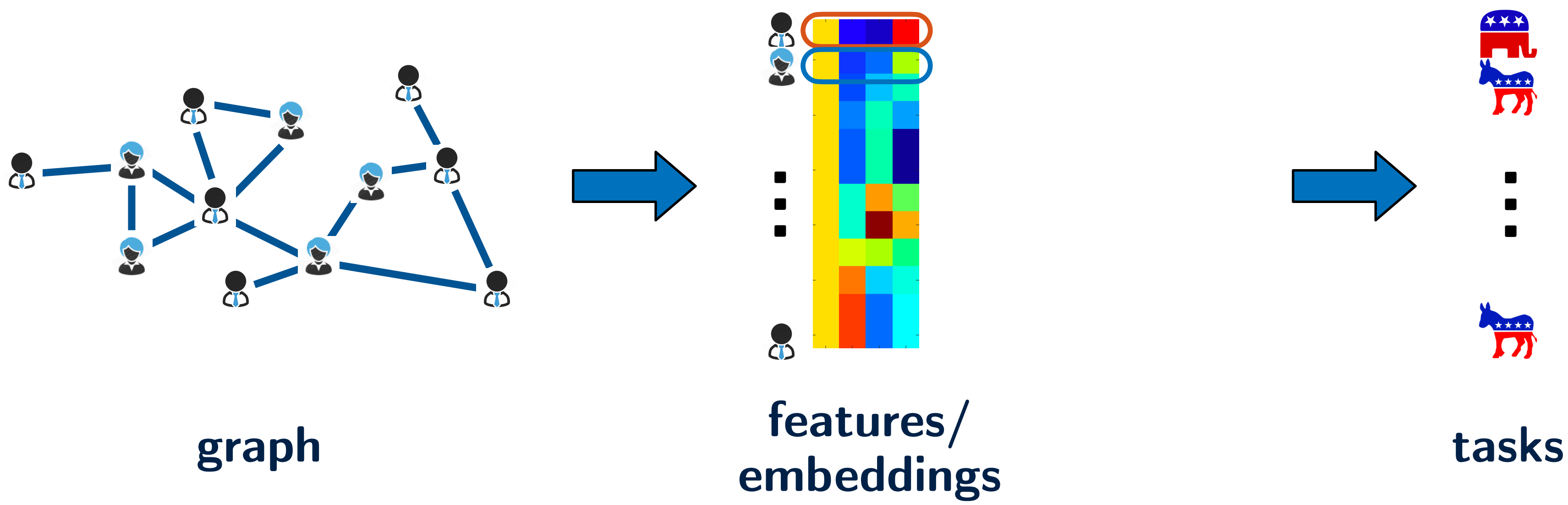
link prediction

graph classification

graph clustering

How to incorporate graphs into learning?

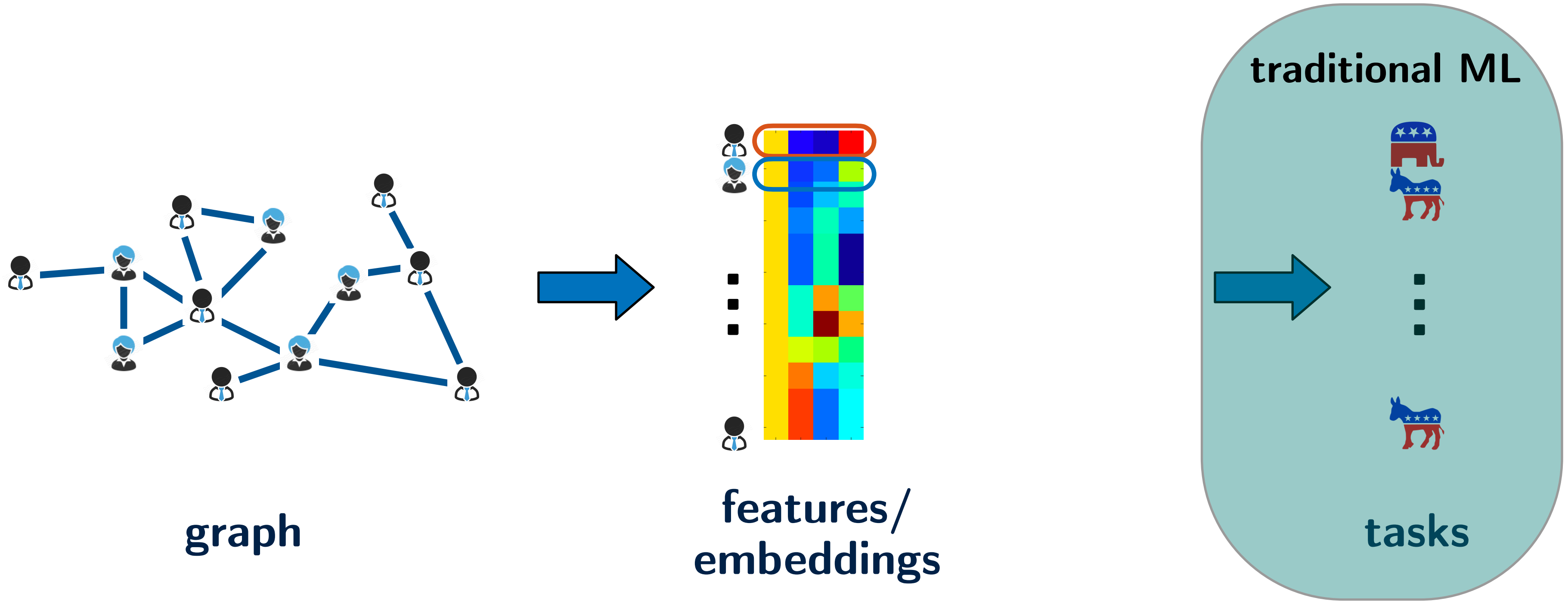
- Traditional machine learning on graphs



- Limitations
  - hand-crafted features or optimised embeddings, often focused on graph structure

How to incorporate graphs into learning?

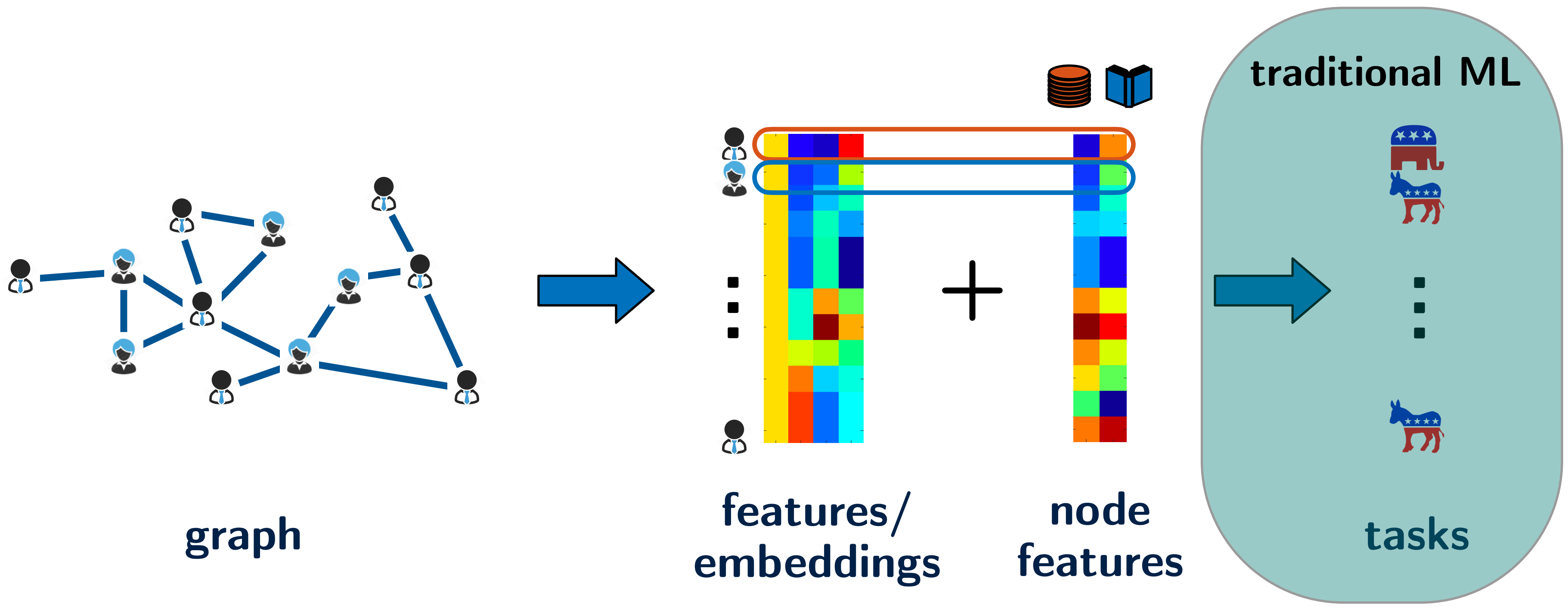
- Traditional machine learning on graphs



- Limitations
  - hand-crafted features or optimised embeddings, often focused on graph structure
  - respect notion of "closeness" in the graph, but do not adapt to downstream tasks

How to incorporate graphs into learning?

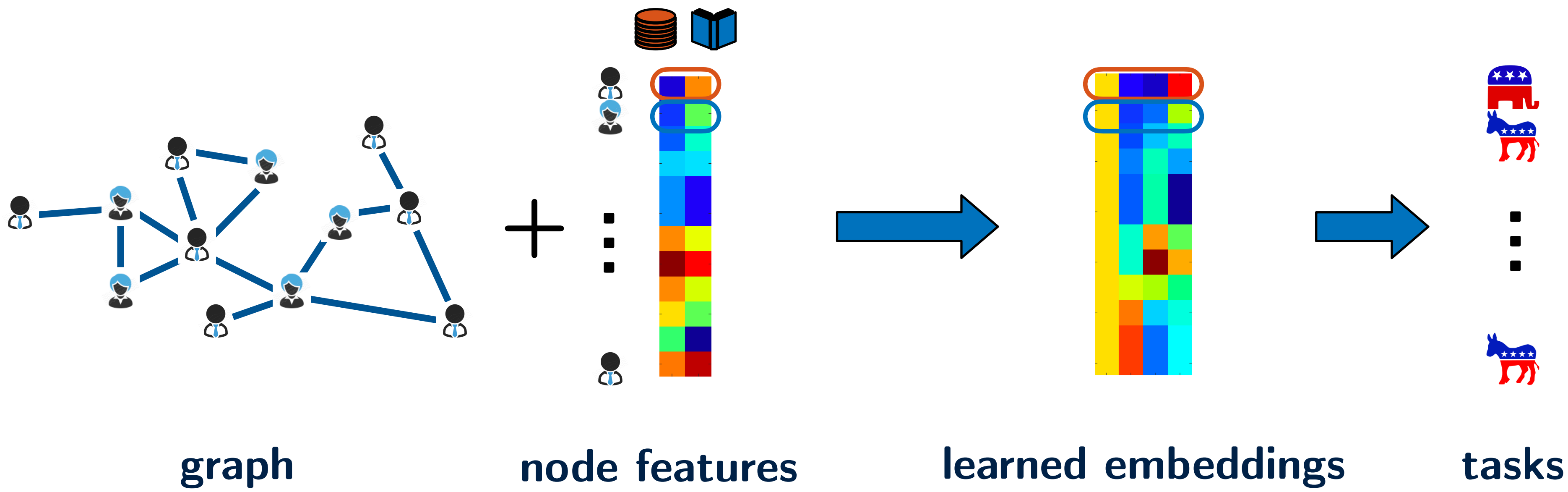
- Traditional machine learning on graphs



- Limitations
  - hand-crafted features or optimised embeddings, often focused on graph structure
  - respect notion of “closeness” in the graph, but do not adapt to downstream tasks
  - can incorporate additional node features, but in a mechanical way

How to incorporate graphs into learning?

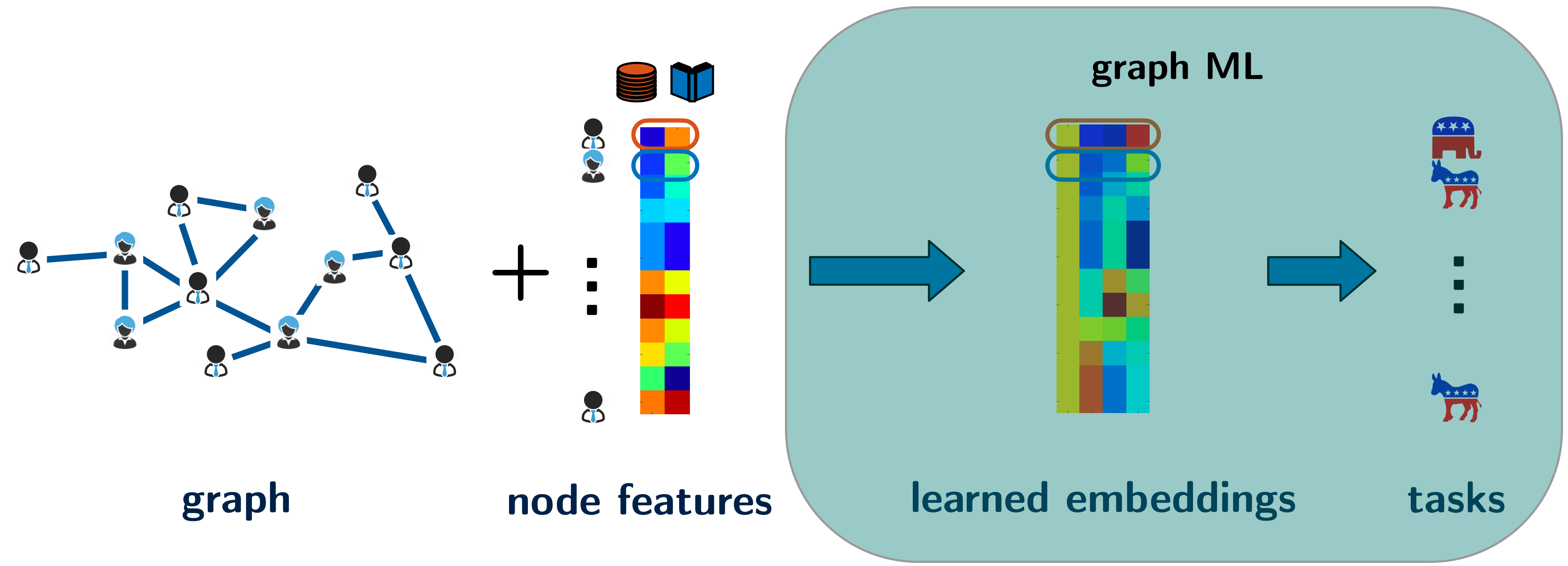
- Graph machine learning



- Advantages
  - naturally combine graph structure and node features in analysis and learning
    - new tools: graph signal processing, graph neural networks

How to incorporate graphs into learning?

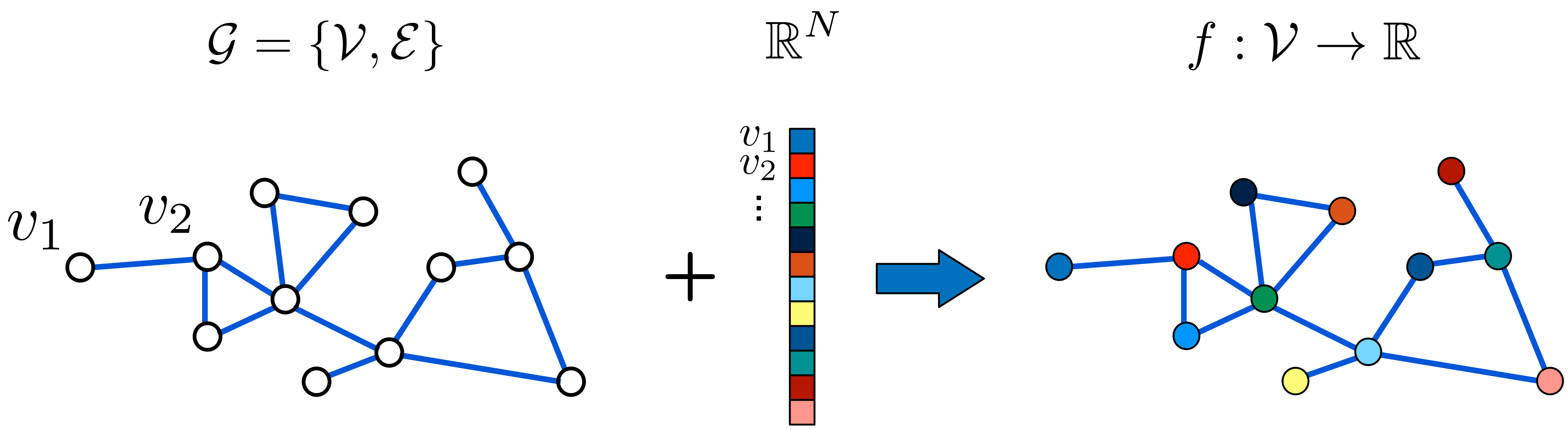
- Graph machine learning



- Advantages
  - naturally combine graph structure and node features in analysis and learning
    - new tools: graph signal processing, graph neural networks
  - embeddings can adapt to downstream tasks and be trained in end-to-end fashion
  - offers more flexibility and enables "deeper" architectures and embeddings

Graph signal processing

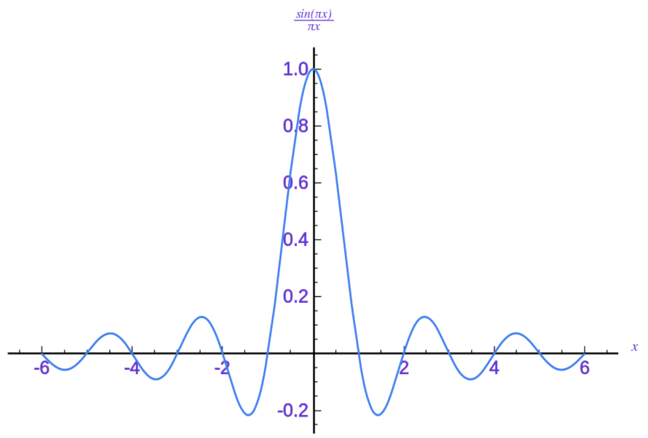
- Graph-structured data can be represented by graph signals



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

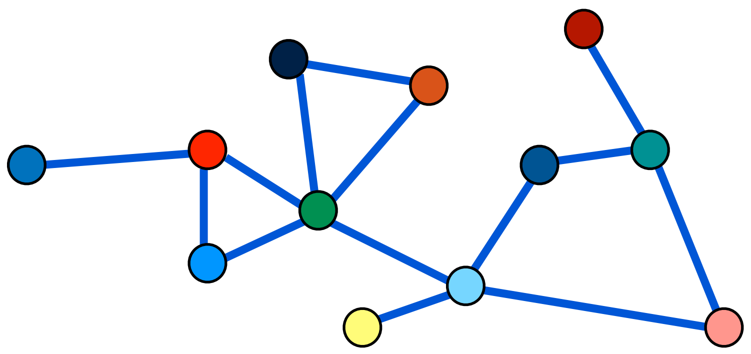
Graph signal processing

1D signal



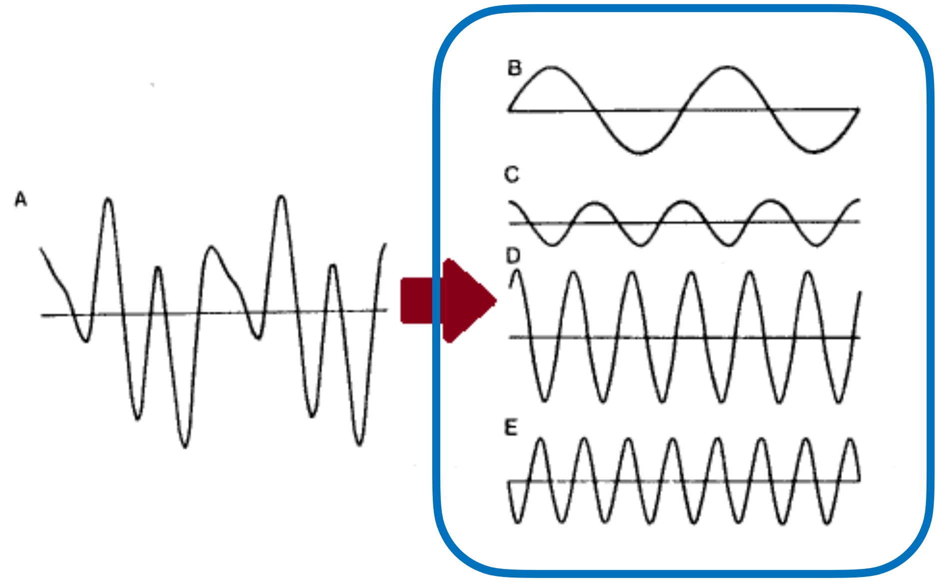
2D signal

graph signal



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

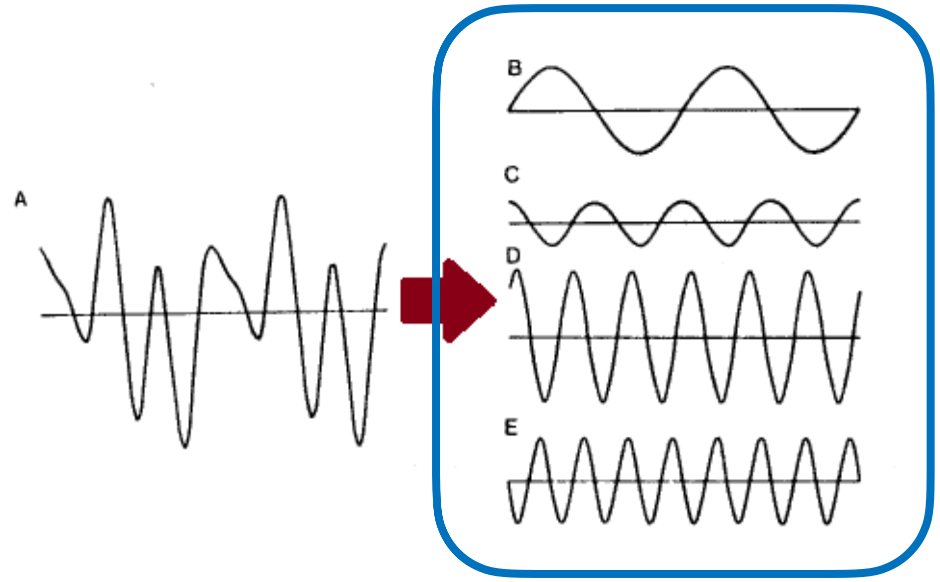
Graph signal processing



classical signal processing

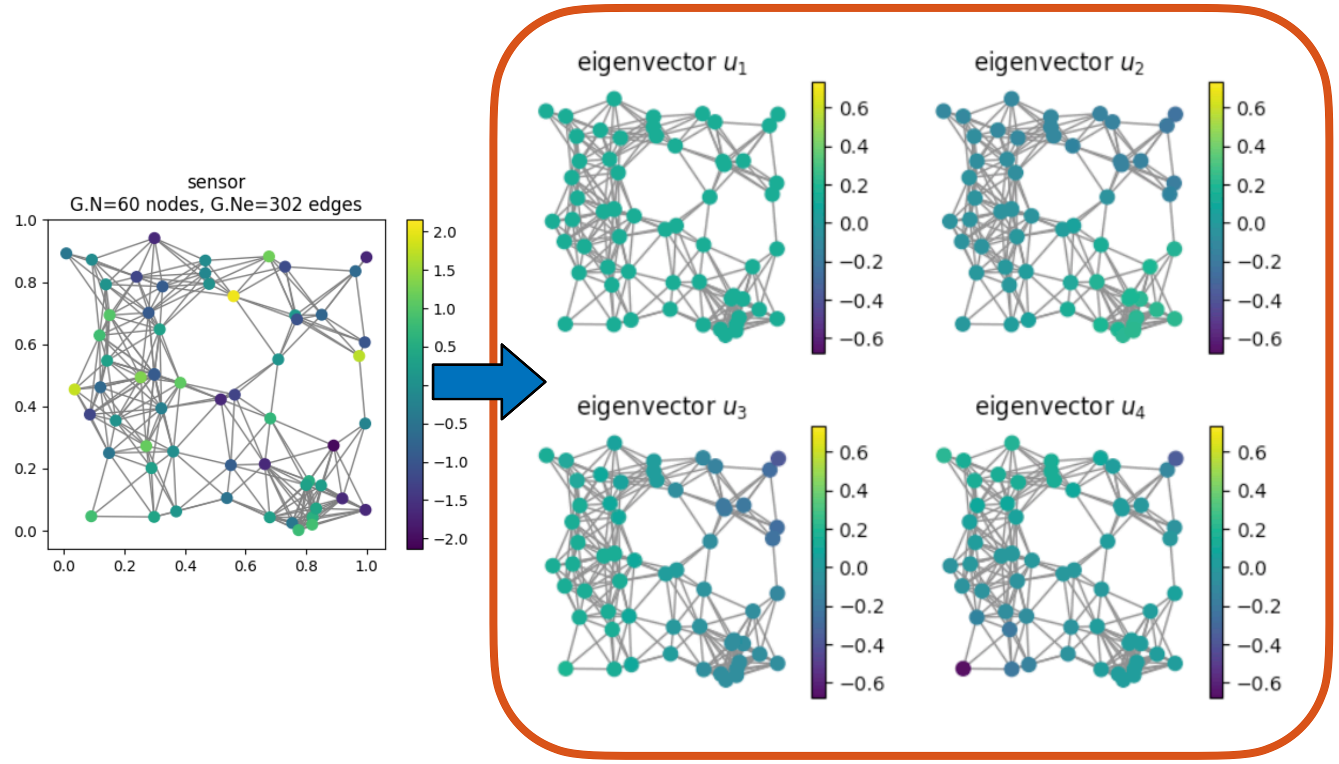
- complex exponentials provide "building blocks" of 1D signal (different oscillations or frequencies)
- leads to **Fourier transform**
- enables convolution and filtering

Graph signal processing



classical signal processing

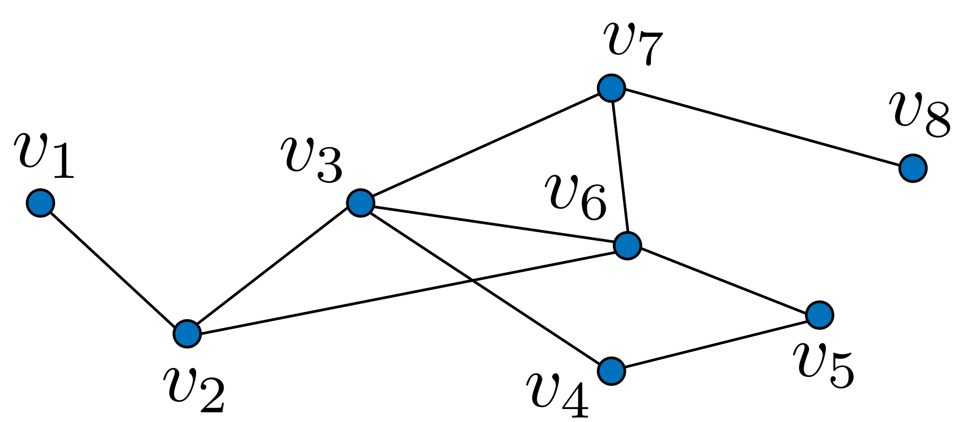
- complex exponentials provide "building blocks" of 1D signal (different oscillations or frequencies)
- leads to **Fourier transform**
- enables convolution and filtering



graph signal processing

- Laplacian eigenvectors provide "building blocks" of graph signal (different oscillation or frequencies)
- leads to **graph Fourier transform**
- enables **convolution** and **filtering** on graphs

Graphs and graph Laplacian



weighted and undirected graph:

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

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

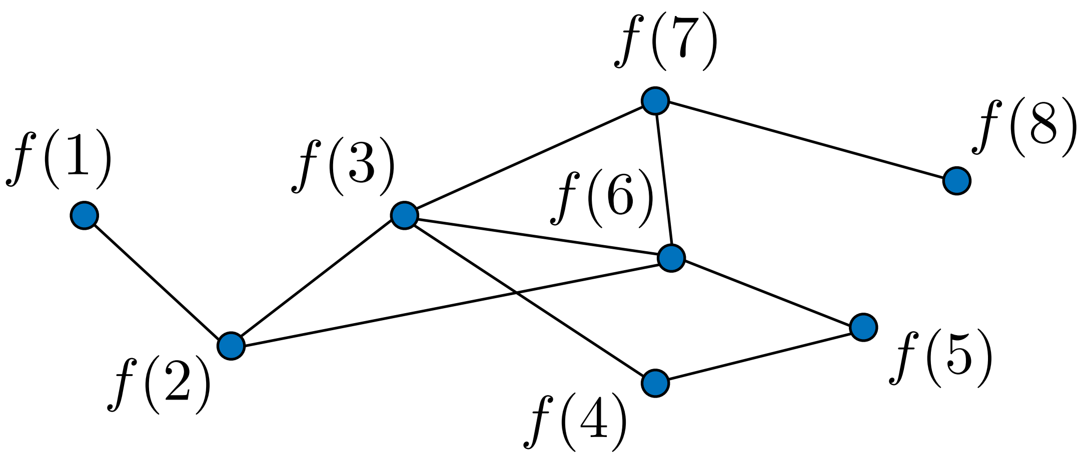
$$L = D - W$$ **equivalent to G!**

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

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

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

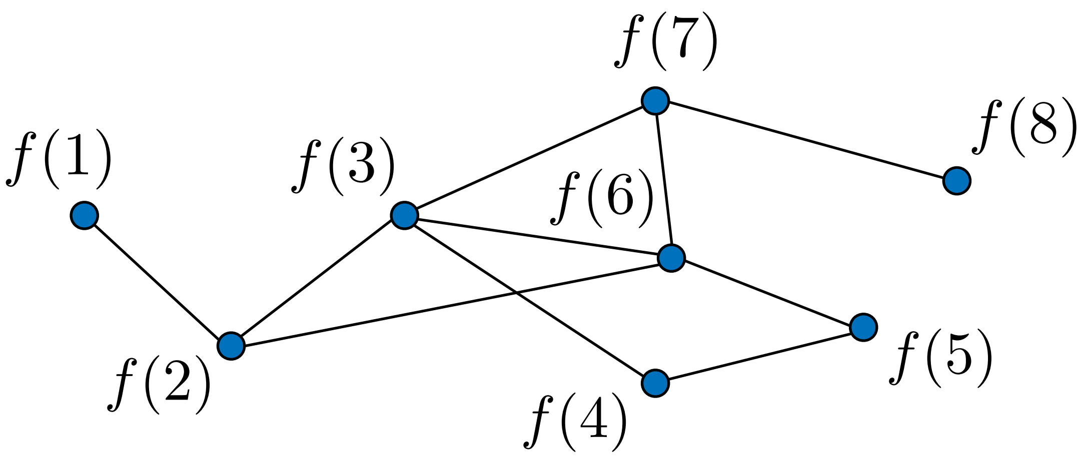
Graph Laplacian



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

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

Graph Laplacian



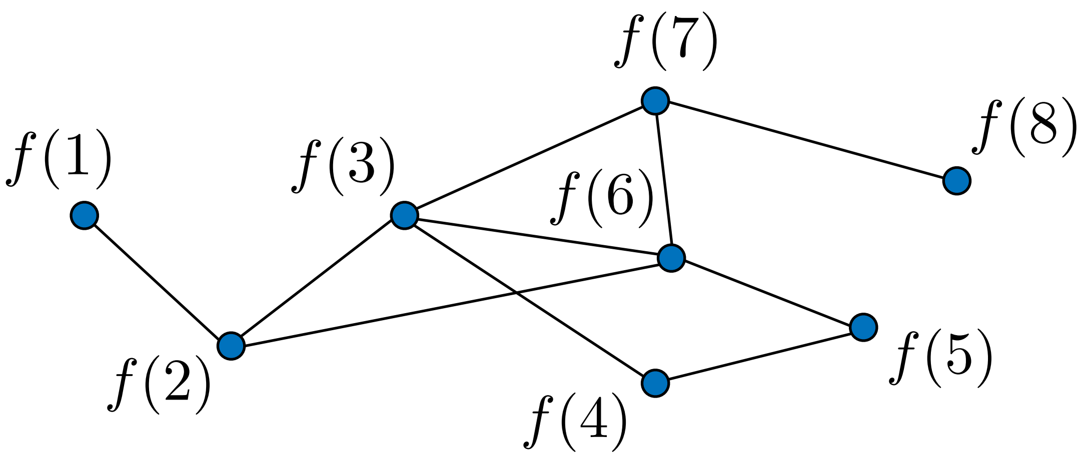
graph signal $f : \mathcal{V} \to \mathbb{R}$

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

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

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

Graph Laplacian



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

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

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

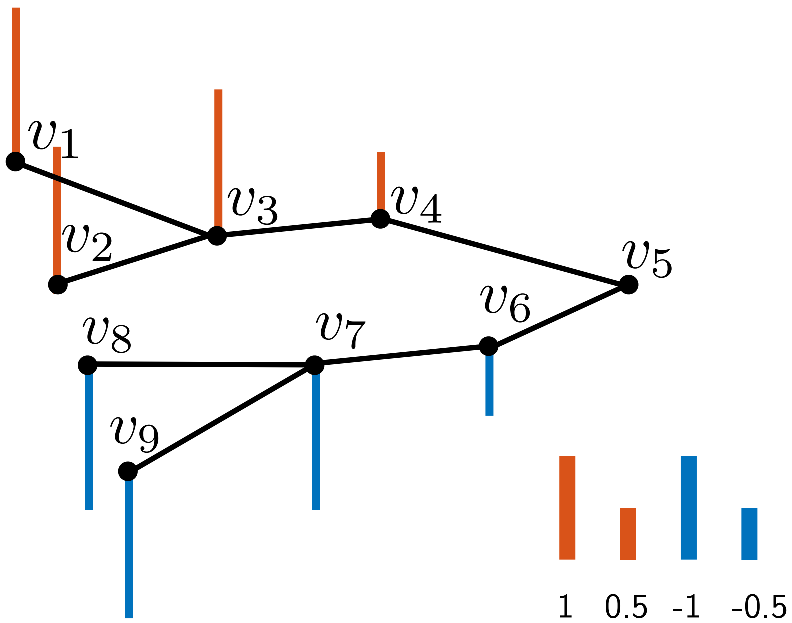
$$Lf(i) = \sum_{j=1}^{N} W_{ij}(f(i) - f(j))$$

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

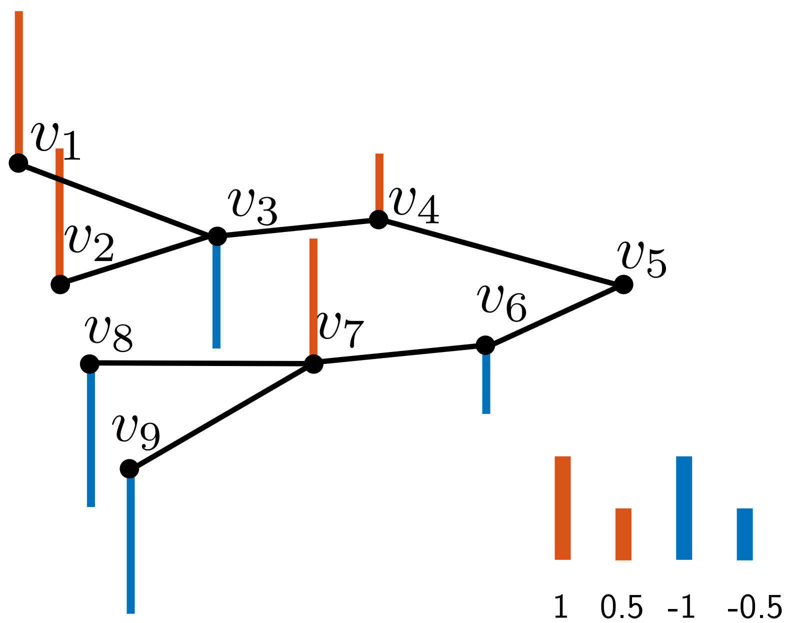
a measure of "smoothness"

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

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

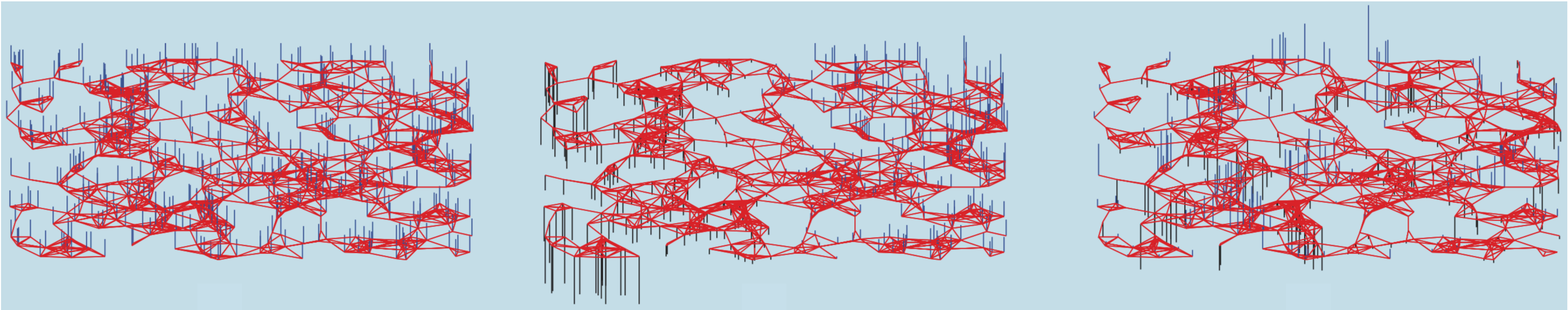
Graph Laplacian

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

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

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

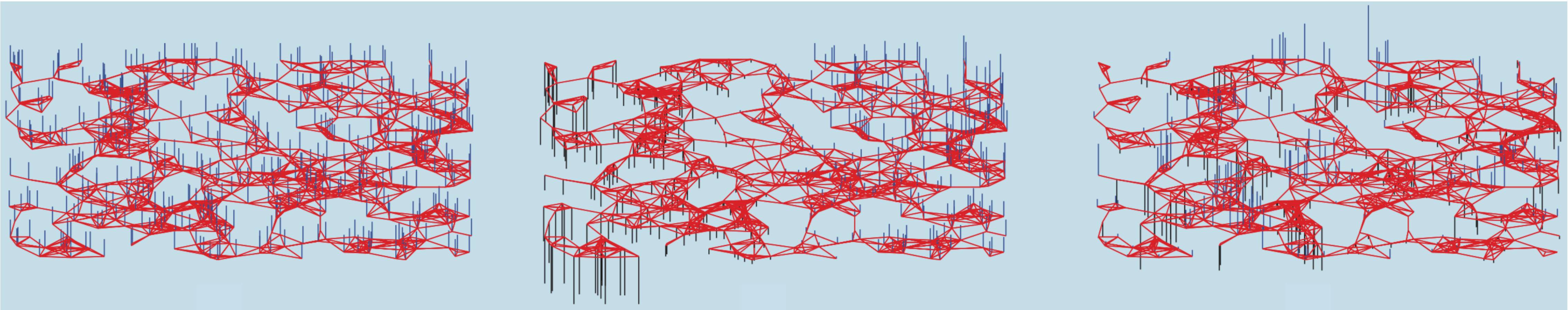
Graph Fourier transform



χ_0 χ_1 χ_{50}

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

Graph Fourier transform



low frequency → high frequency

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

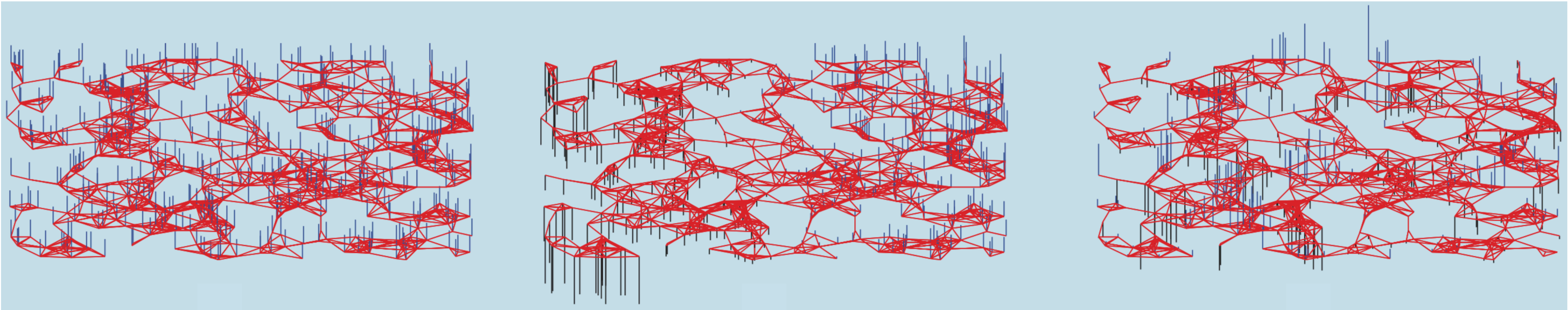
$$\chi_0^T L \chi_0 = \lambda_0 = 0$$

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

- 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

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

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

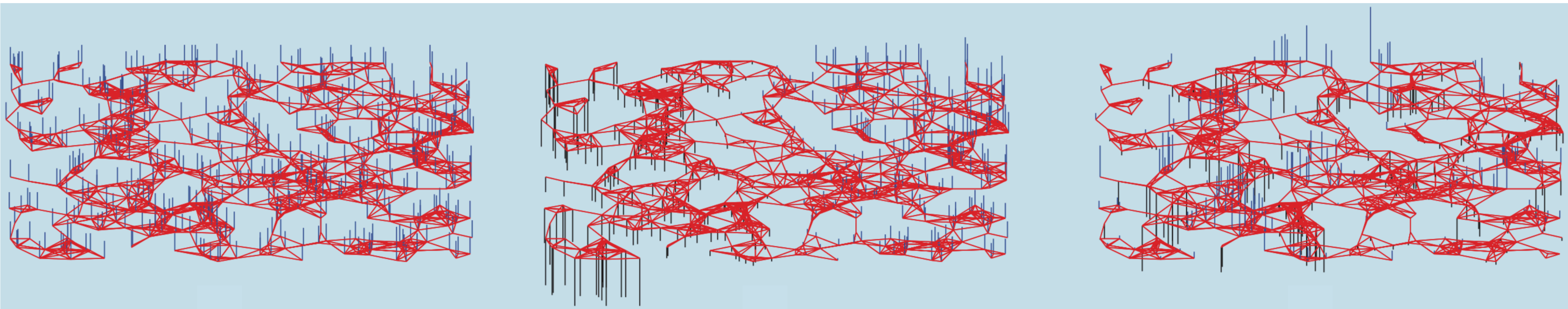
$\chi_{50}^T L \chi_{50} = \lambda_{50}$

graph Fourier transform:

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

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

Graph Fourier transform



χ_0 χ_1 χ_{50}

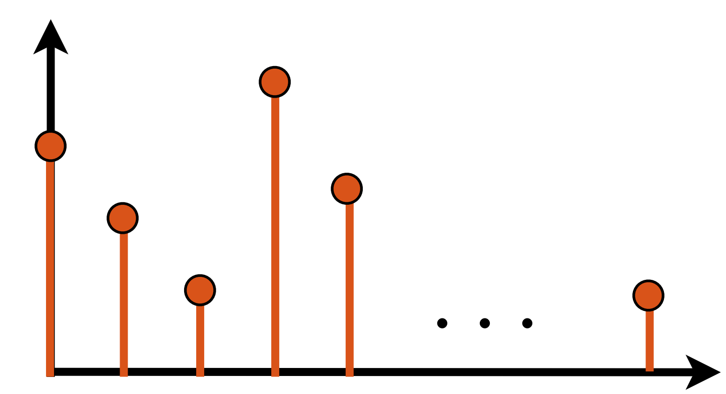
low frequency → high frequency

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

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

graph Fourier transform:

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



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

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

Graph Fourier transform

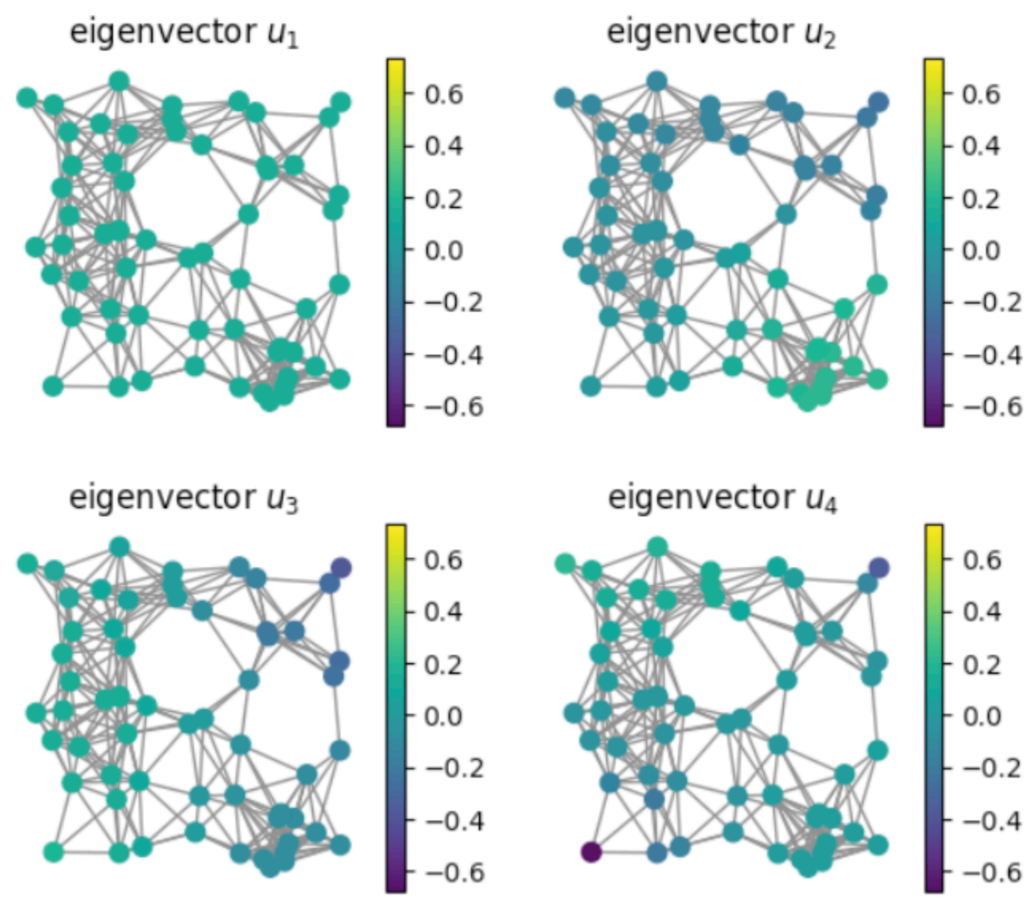
- The Laplacian L admits the following eigendecomposition: $L\chi_\ell = \lambda_\ell \chi_\ell$

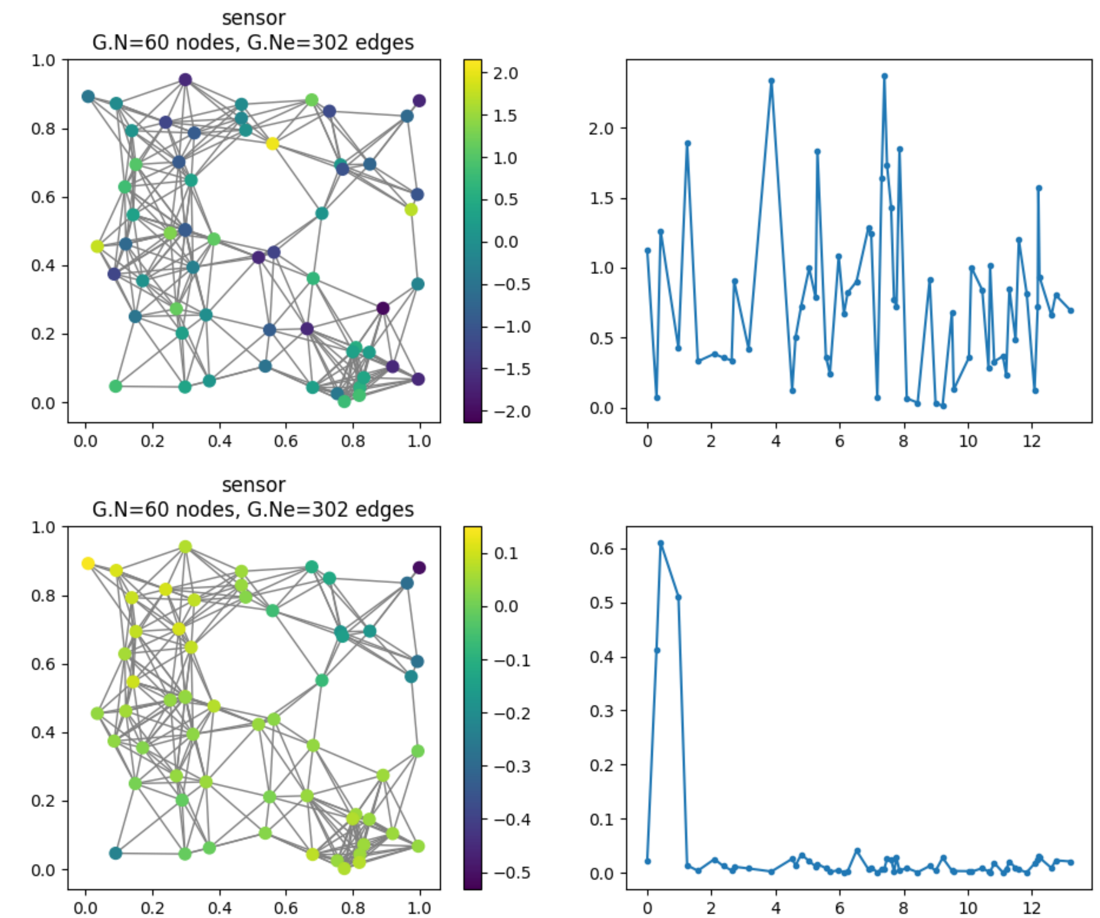
one-dimensional Laplace operator: $-\nabla^2$	graph Laplacian: L
eigenfunctions: $e^{j\omega x}$	eigenvectors: χ_ℓ
	$f : V \to \mathbb{R}^N$
Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$	Graph FT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$
$f(x)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

Example on synthetic signals

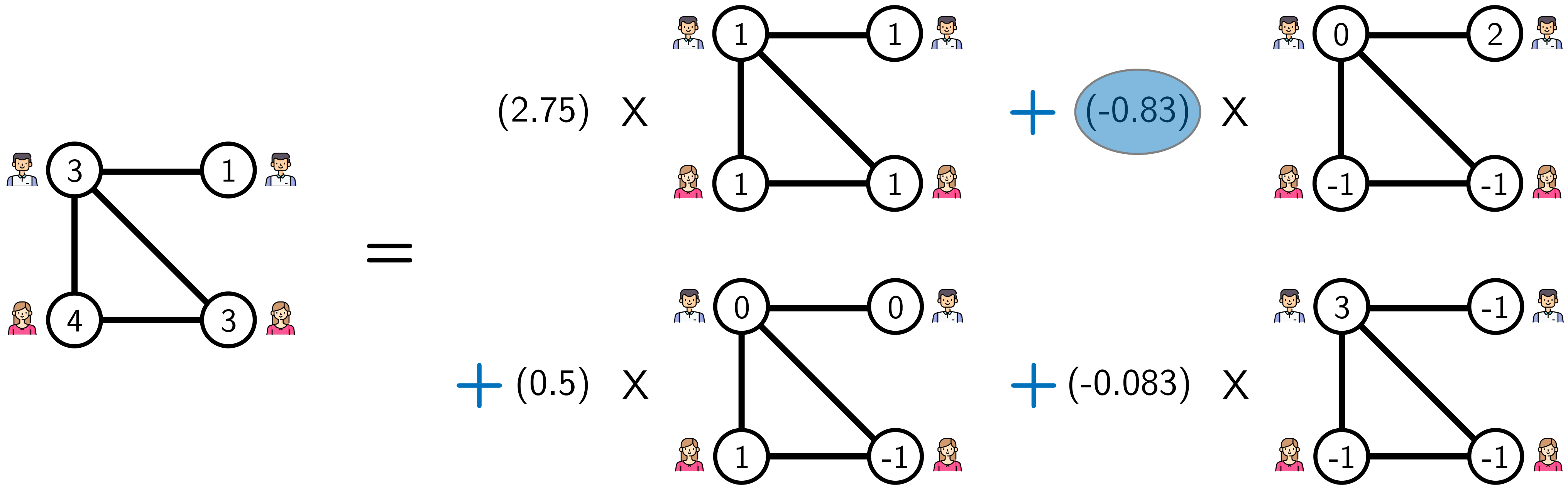
GFT:

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

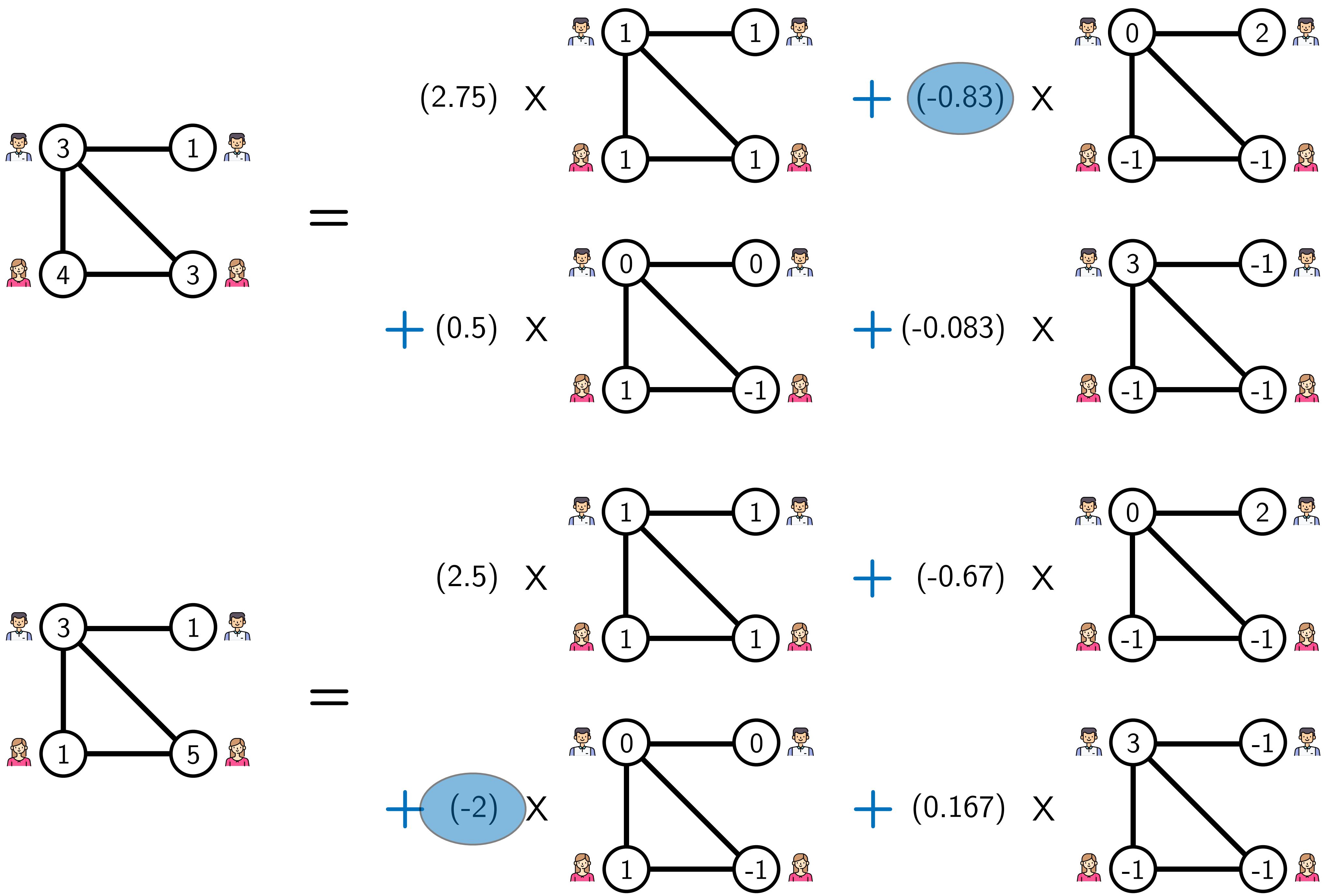




Example on movie ratings



Example on movie ratings

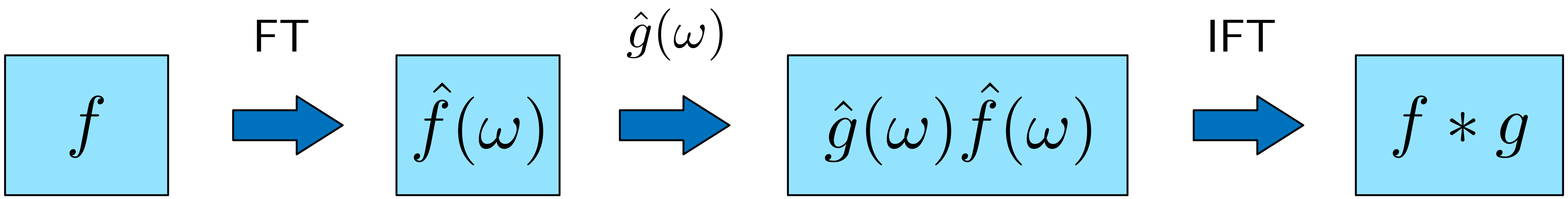


Classical frequency filtering

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

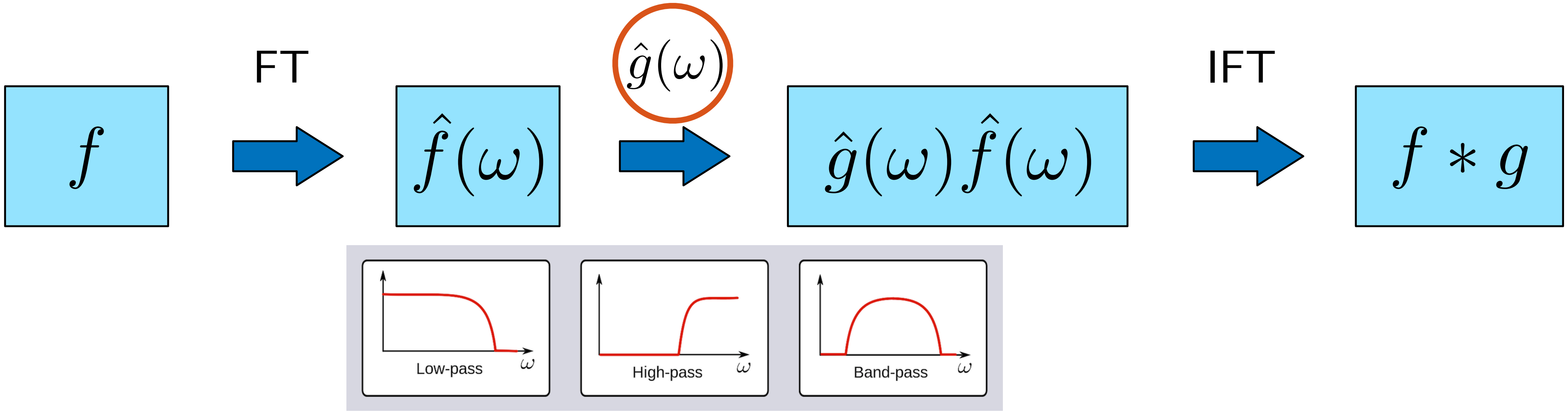
Classical frequency filtering

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



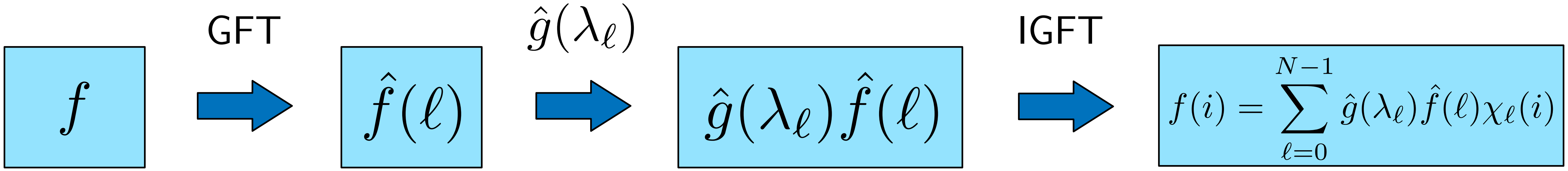
Classical frequency filtering

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



Graph spectral filtering

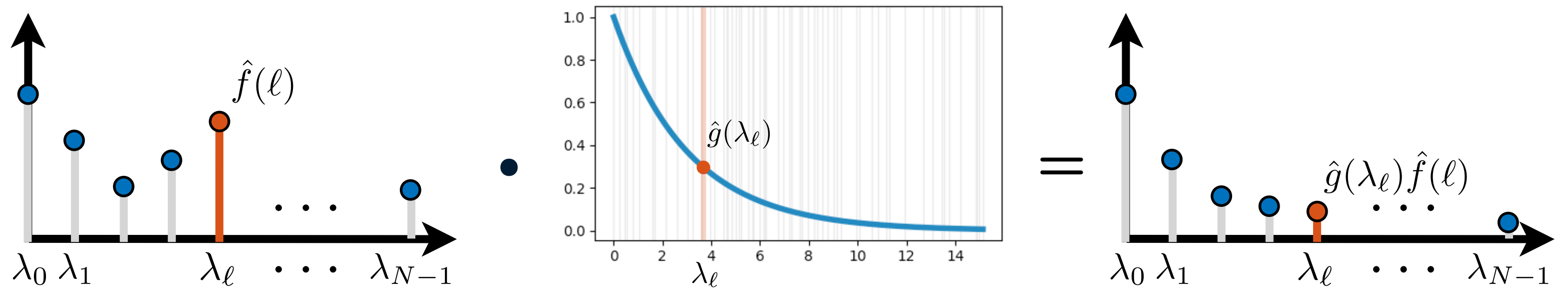
GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$



Graph spectral filtering

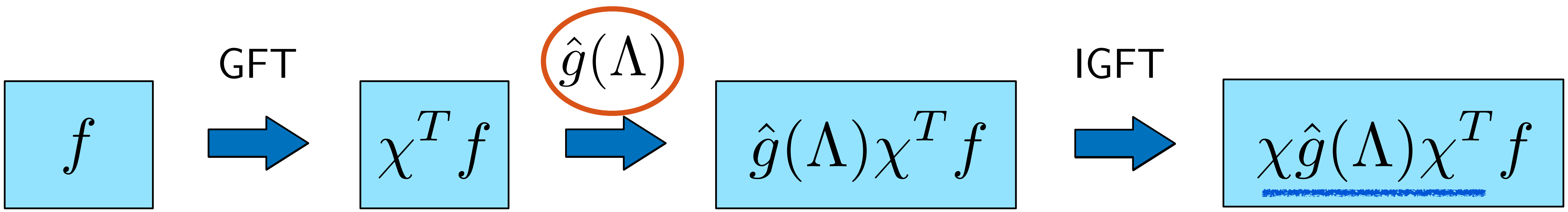
$$\text{GFT:} \quad \hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i) \qquad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

f → 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 spectral filtering

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



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

$\hat{g}(L)$**: function of L!**

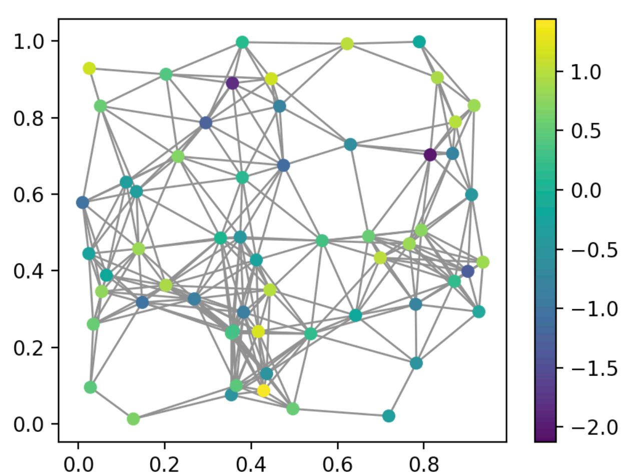
Graph spectral filtering

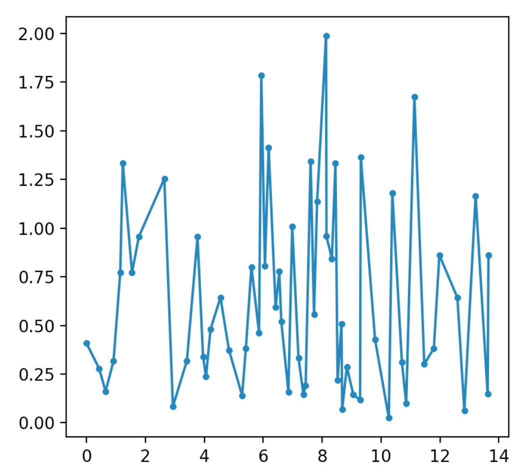
GFT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^{N} \chi_\ell^*(i) f(i)$ $\quad f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell)\chi_\ell(i)$

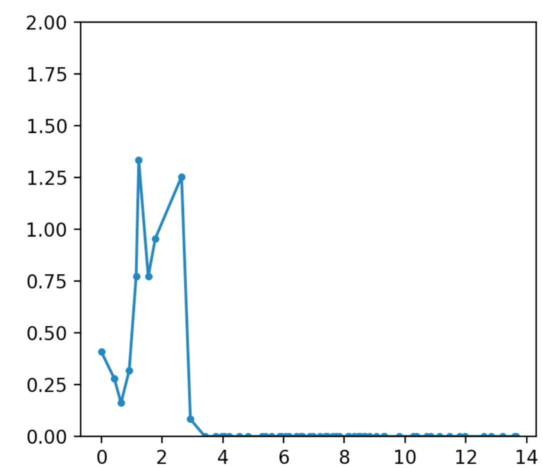
f → GFT → $\chi^T f$ → $\hat{g}(\Lambda)$ → $\hat{g}(\Lambda)\chi^T f$ → IGFT → $\chi\hat{g}(\Lambda)\chi^T f$

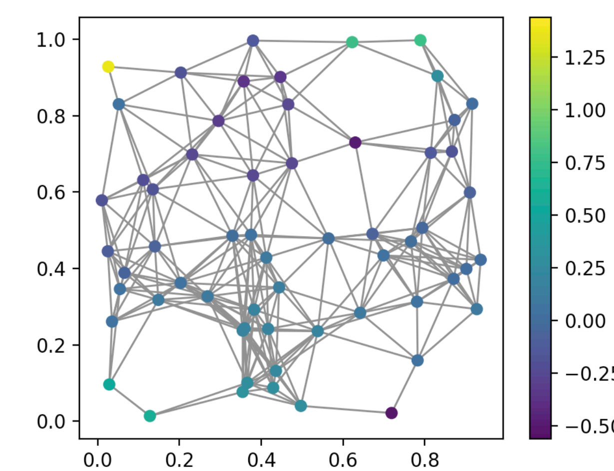
$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

$\hat{g}(L)$: **function of L!**

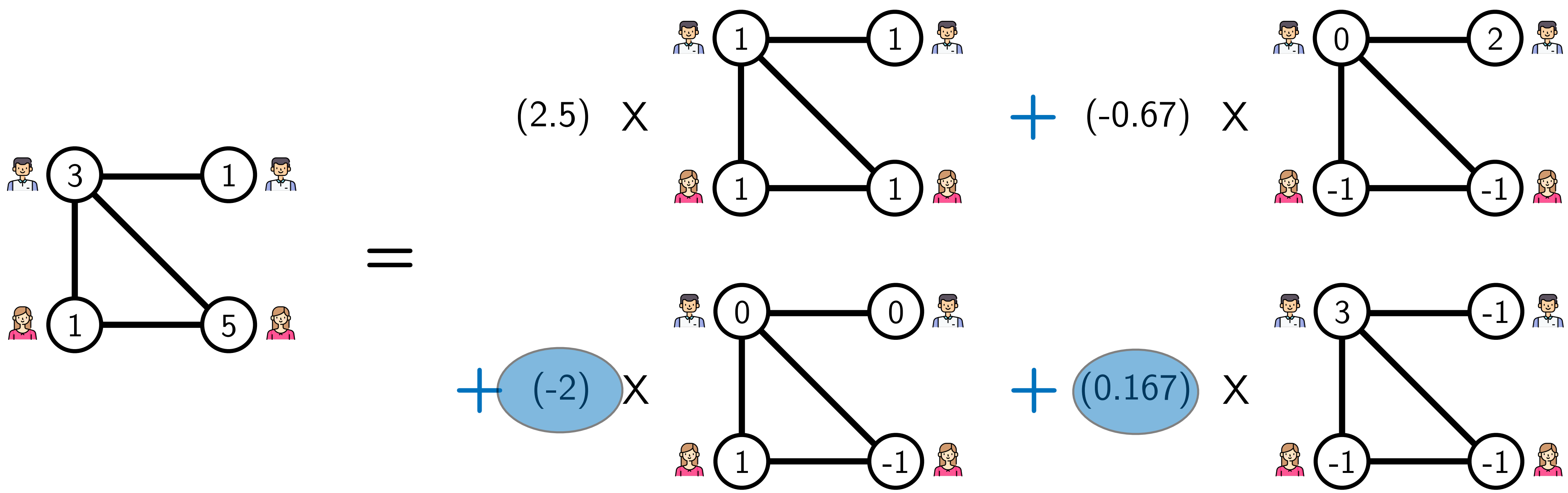




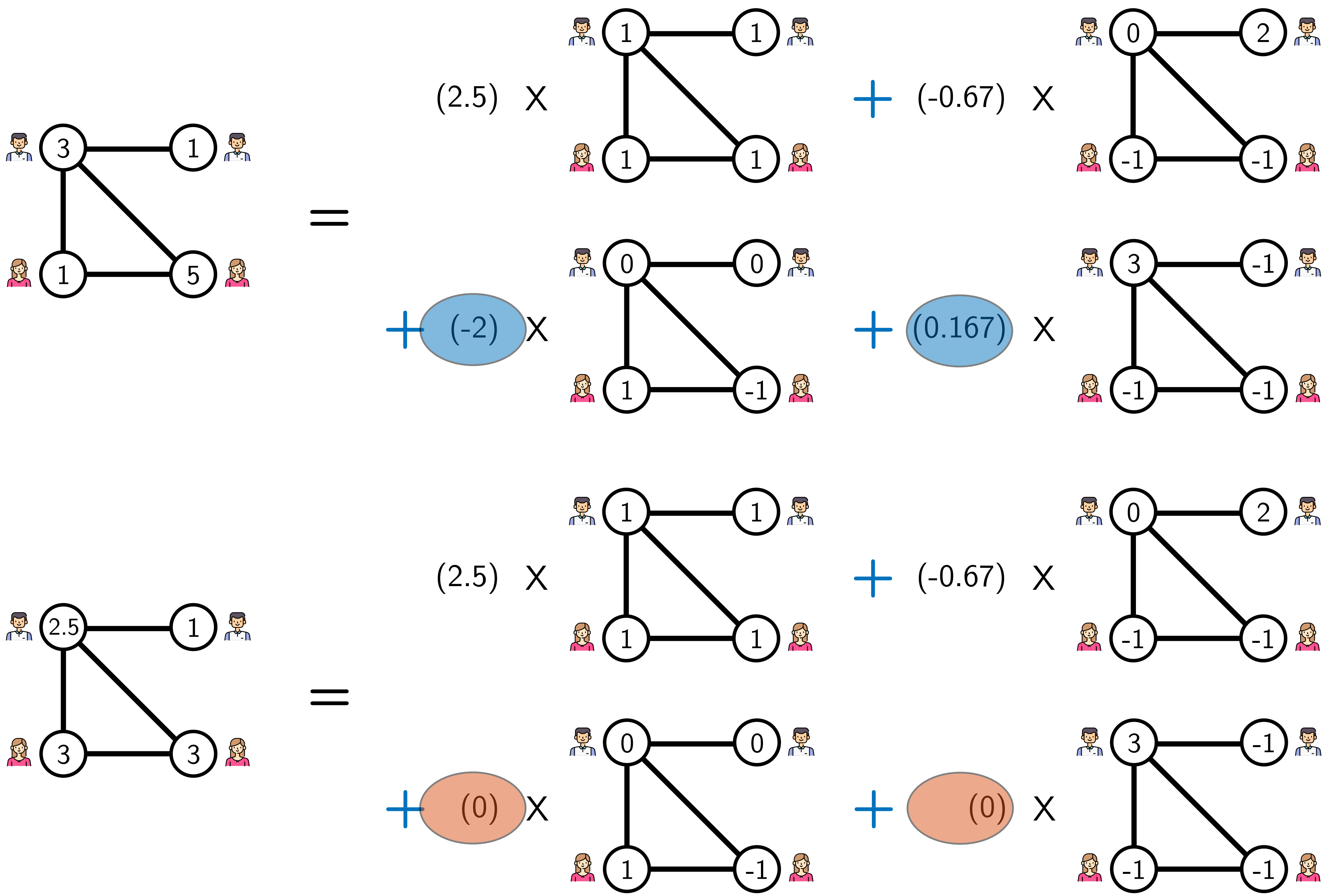




Example on movie ratings



Example on movie ratings



Graph spectral filtering

- Filters can be designed as functions of graph Laplacian

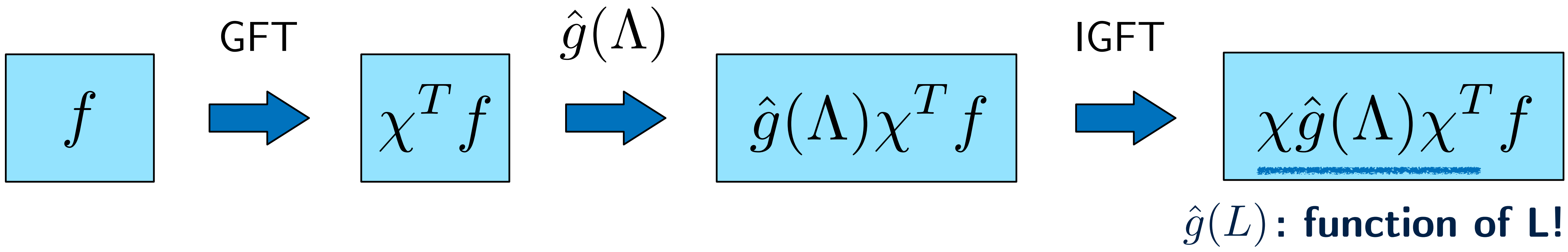
$$f \xrightarrow{\text{GFT}} \chi^T f \xrightarrow{\hat{g}(\Lambda)} \hat{g}(\Lambda)\chi^T f \xrightarrow{\text{IGFT}} \underline{\chi\hat{g}(\Lambda)\chi^T} f$$

$\hat{g}(L)$ **: function of L!**

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

Graph spectral filtering

- Filters can be designed as functions of graph Laplacian



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

- Closely related to kernels and regularisation on graphs

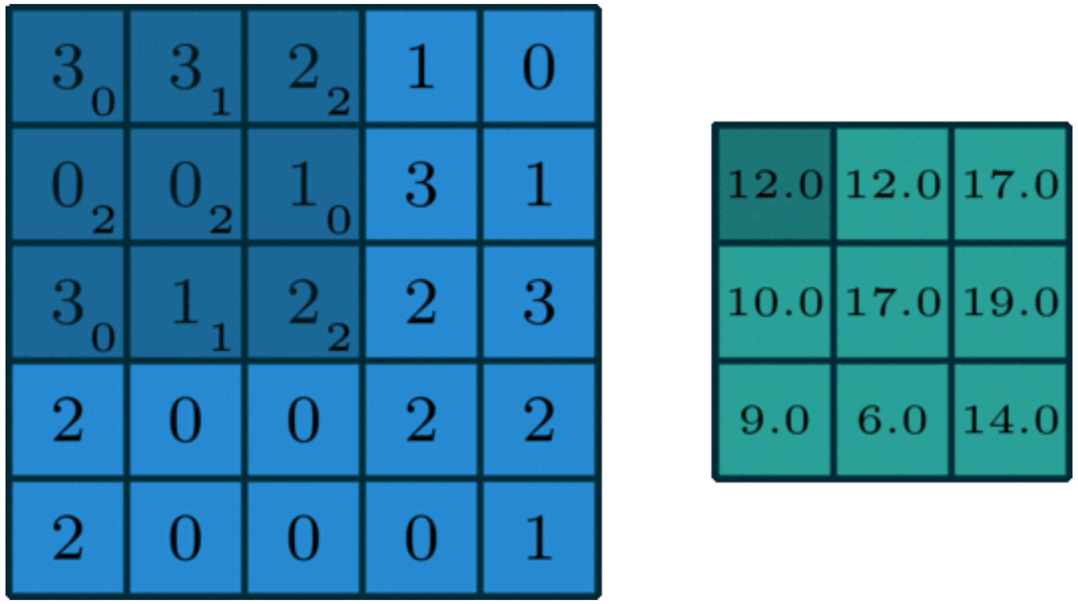
Smola and Kondor, "Kernels and regularization on graphs," COLT, 2003.

Convolution on graphs

classical convolution

time domain

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



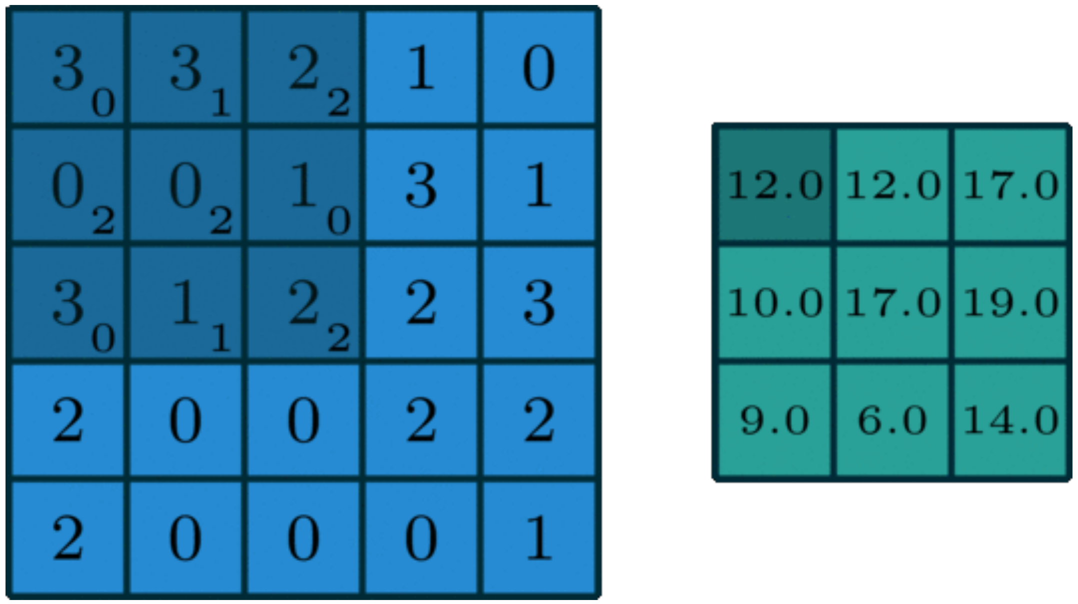
Dumoulin and Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

Convolution on graphs

classical convolution

time domain

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



Dumoulin and Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

Convolution on graphs

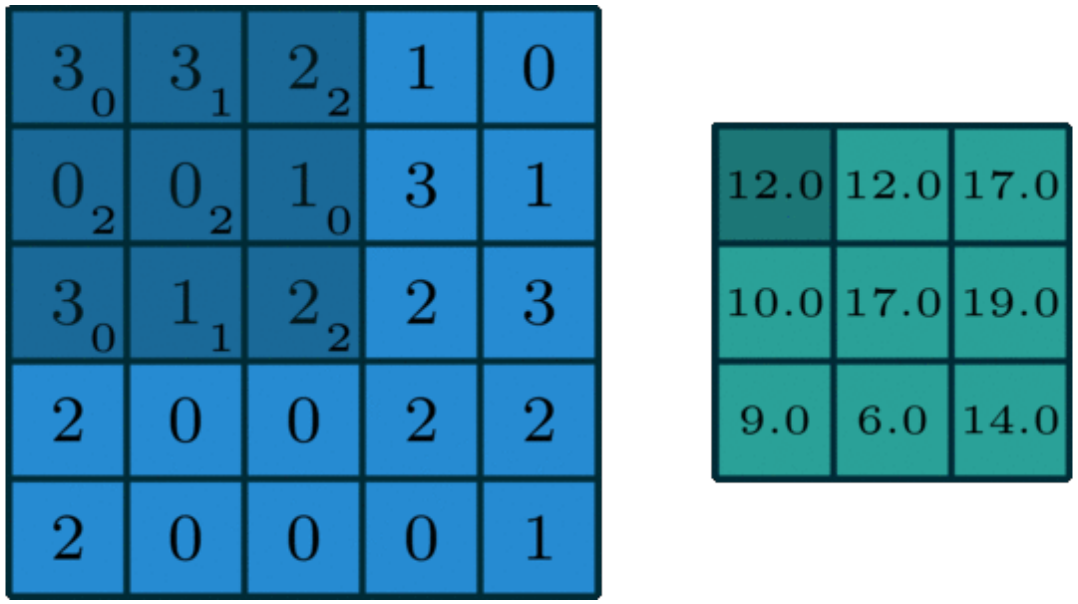
classical convolution

time domain

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

frequency domain

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



Dumoulin and Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

Convolution on graphs

classical convolution

convolution on graphs

time domain

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

frequency domain

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

graph spectral domain

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

Dumoulin and Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

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

spatial (node) domain

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

graph spectral domain

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

Dumoulin and Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

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

spatial (node) domain

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

convolution = filtering

graph spectral domain

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

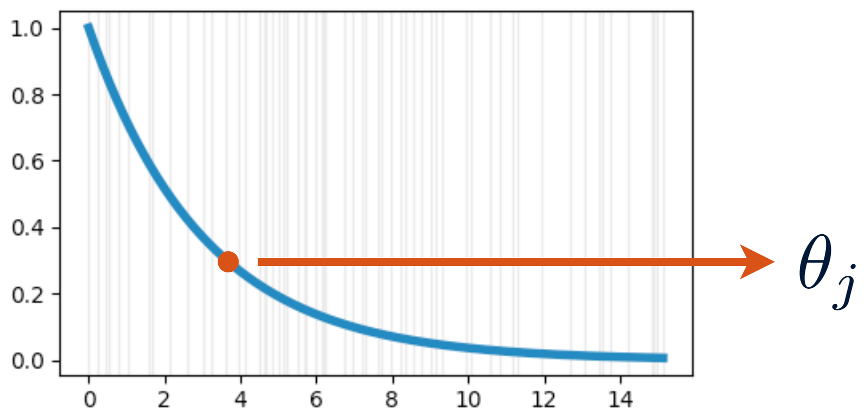
Dumoulin and Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

A non-parametric filter

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

learning a non-parametric filter:

$$\hat{g}_\theta(\Lambda) = \mathrm{diag}(\theta),\ \theta \in \mathbb{R}^N$$



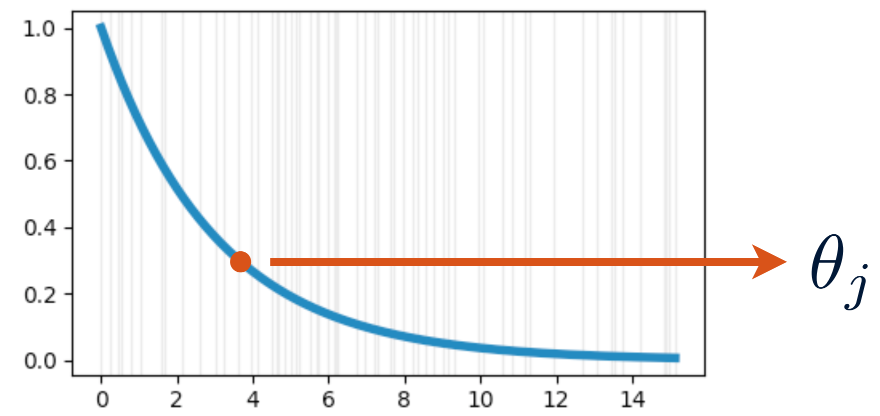
Bruna et al., "Spectral networks and deep locally connected networks on graphs," ICLR, 2014.

A non-parametric filter

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

learning a non-parametric filter:

$$\hat{g}_\theta(\Lambda) = \mathrm{diag}(\theta), \ \theta \in \mathbb{R}^N$$



- convolution expressed in the graph spectral domain
- no localisation in the spatial (node) domain
- computationally expensive

Bruna et al., "Spectral networks and deep locally connected networks on graphs," ICLR, 2014.

A parametric filter

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

parametric filter as polynomial of Laplacian

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

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

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

A parametric filter

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

parametric filter as polynomial of Laplacian

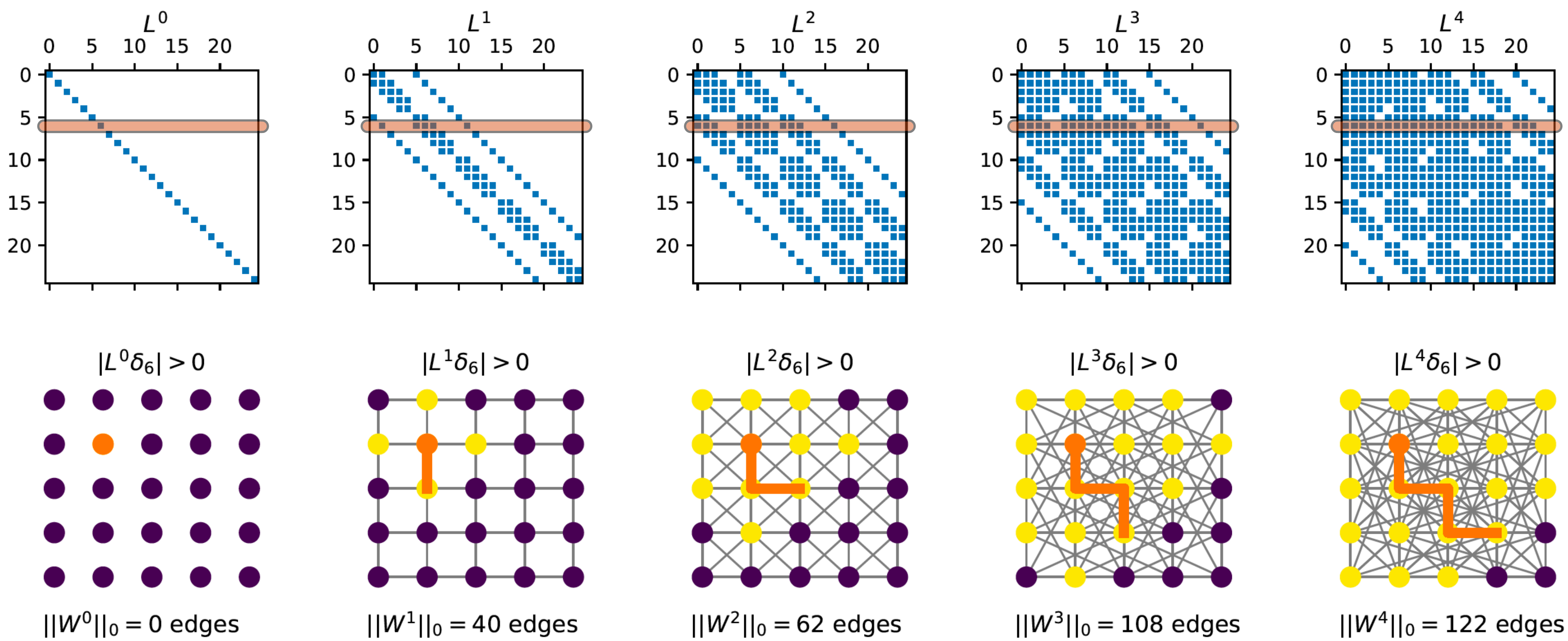
$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \; \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

what do powers of graph Laplacian capture?

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

Powers of graph Laplacian

L^k defines the k-neighborhood



(source: M. Deferrard)

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

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

A parametric filter

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

parametric filter as polynomial of Laplacian

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

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

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

A parametric filter

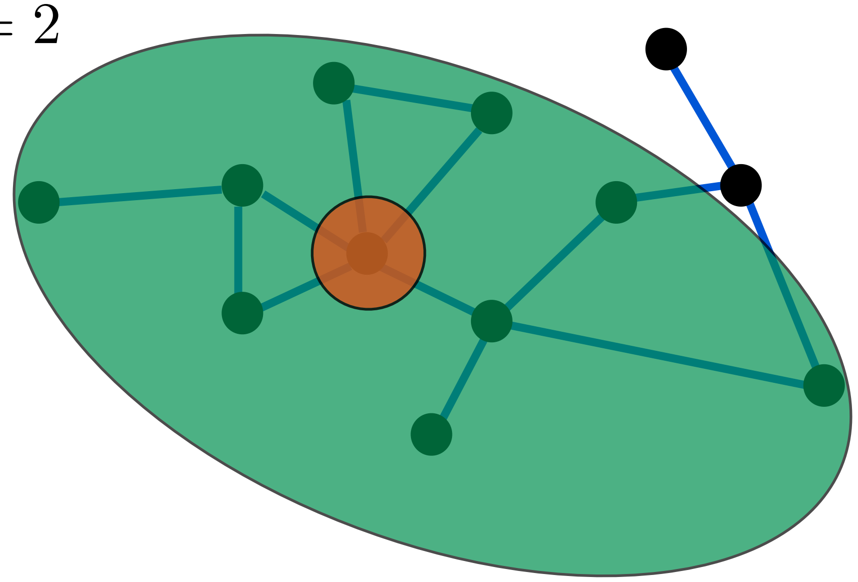
$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

parametric filter as polynomial of Laplacian

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

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

$K = 2$



- localisation within *K*-hop neighbourhood

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

A parametric filter

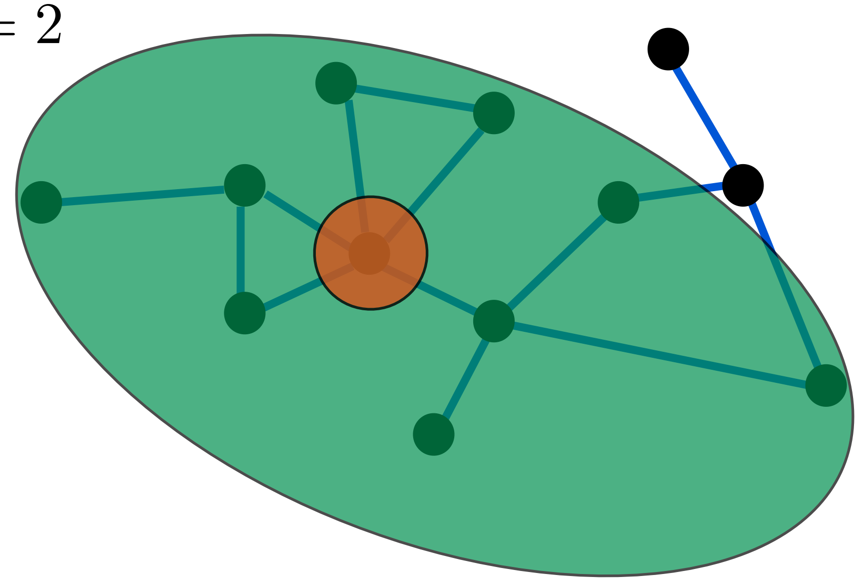
$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

parametric filter as polynomial of Laplacian

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

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

$K = 2$



- localisation within *K*-hop neighbourhood
- Chebyshev approximation enables efficient computation via recursive multiplication with 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.

A simplified parametric filter

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

simplified parametric filter

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

normalised Laplacian

$$\begin{aligned} L_{\text{norm}} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ &= D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} \\ &= I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = I - W_{\text{norm}} \end{aligned}$$

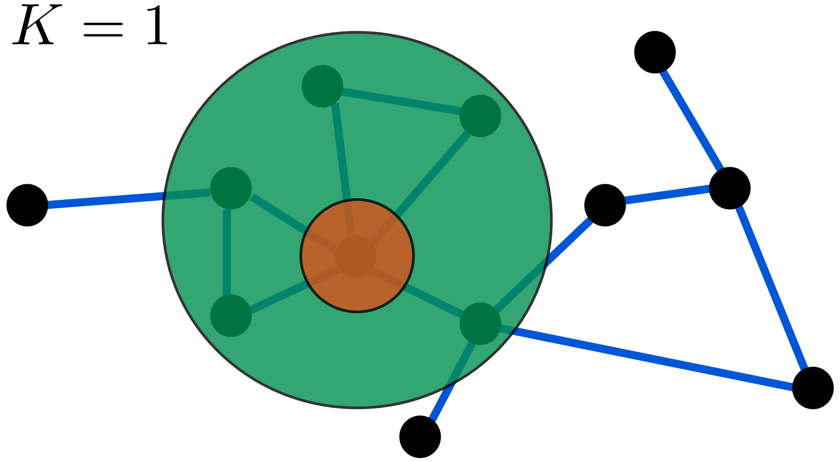
$K = 1$

normalised Laplacian

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

(localisation within 1-hop neighbourhood)

$K = 1$



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

A simplified parametric filter

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

simplified parametric filter

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

normalised Laplacian

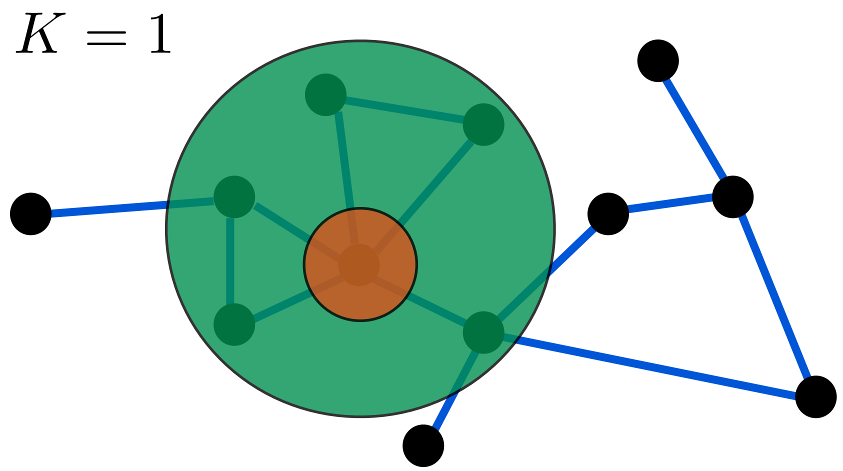
$$\begin{aligned} L_{\text{norm}} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ &= D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} \\ &= I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = I - W_{\text{norm}} \end{aligned}$$

$K = 1$

normalised Laplacian

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

(localisation within 1-hop neighbourhood)



$\alpha = \theta_0 = -\theta_1$

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

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

A simplified parametric filter

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

simplified parametric filter

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

normalised Laplacian

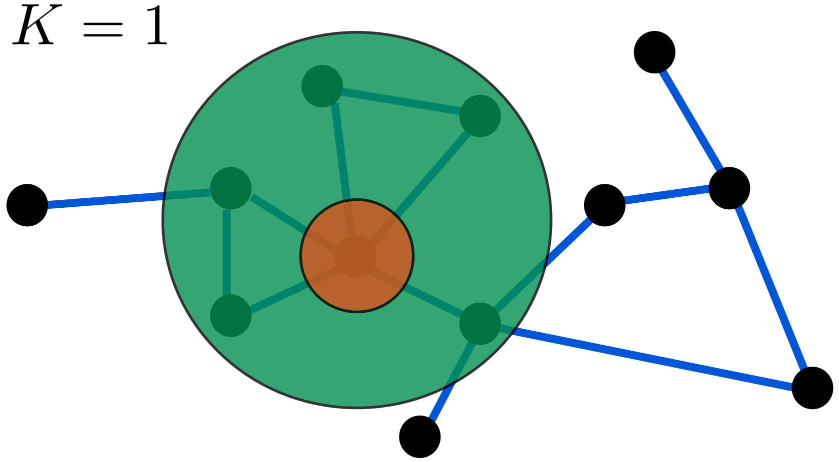
$$\begin{aligned} L_{\text{norm}} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ &= D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} \\ &= I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = I - W_{\text{norm}} \end{aligned}$$

$K = 1$
normalised Laplacian

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

(localisation within 1-hop neighbourhood)

$K = 1$



$\alpha = \theta_0 = -\theta_1$

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

renormalisation

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

renormalisation

$$\tilde{W} = W + I \qquad \tilde{D} = D + I$$

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

A simplified parametric filter

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

simplified parametric filter

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

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

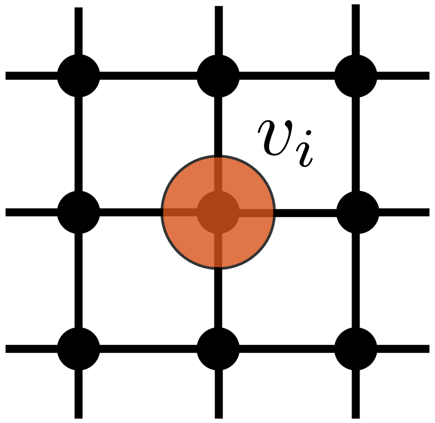
A simplified parametric filter

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

simplified parametric filter

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

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



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

A simplified parametric filter

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

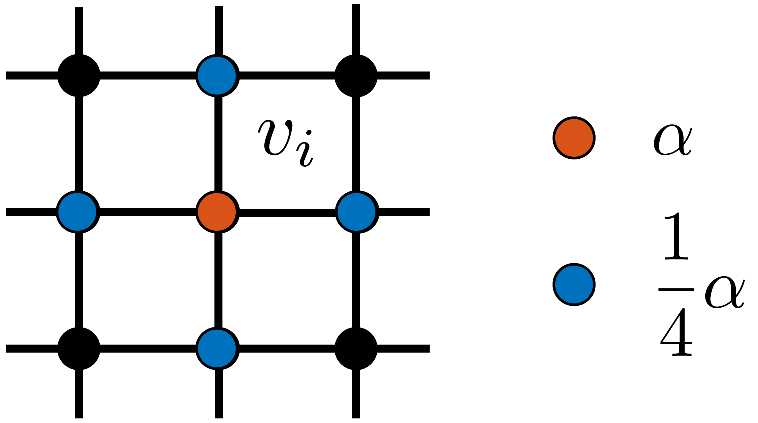
simplified parametric filter

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

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

unitary edge weights

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



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

A simplified parametric filter

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

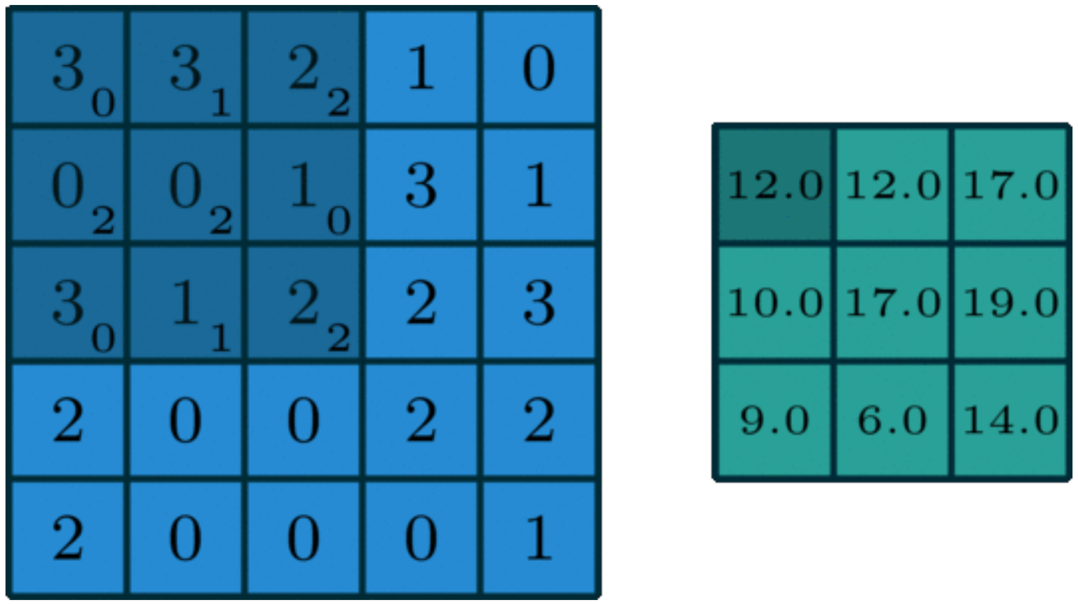
simplified parametric filter

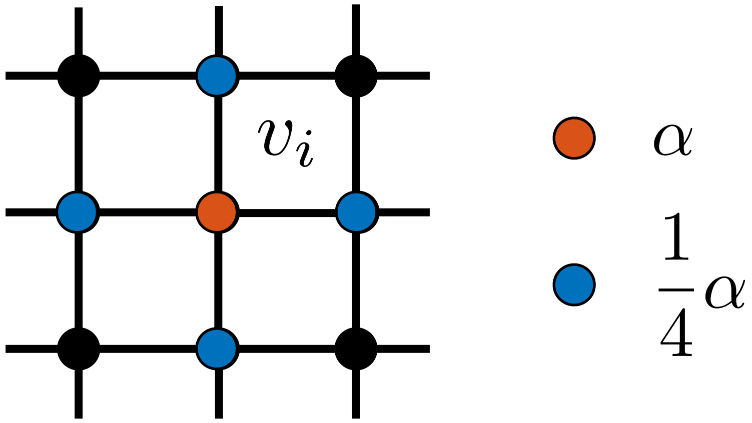
$$\hat{g}_\alpha(L) = \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

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

unitary edge weights

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





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

Convolution on graphs - Remarks

- 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 (node)** domain
  - simplified filter: $y = \hat{g}_\theta(L) f = \alpha(\tilde{D}^{-\frac{1}{2}} \tilde{W} \tilde{D}^{-\frac{1}{2}}) f$
  - interpretation: at each layer nodes exchange information in 1-hop neighbourhood
  - more generally: receptive field size determined by degree of polynomial

Convolution on graphs - Remarks

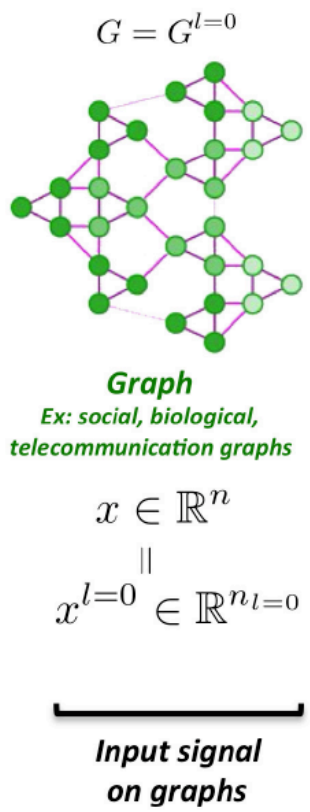
- Convolution is defined via the **graph spectral** domain..

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

- ..but can be implemented in the **spatial (node)** domain
  - simplified filter: $y = \hat{g}_\theta(L) f = \alpha(\tilde{D}^{-\frac{1}{2}} \tilde{W} \tilde{D}^{-\frac{1}{2}}) f$
  - interpretation: at each layer nodes exchange information in 1-hop neighbourhood
  - more generally: receptive field size determined by degree of polynomial

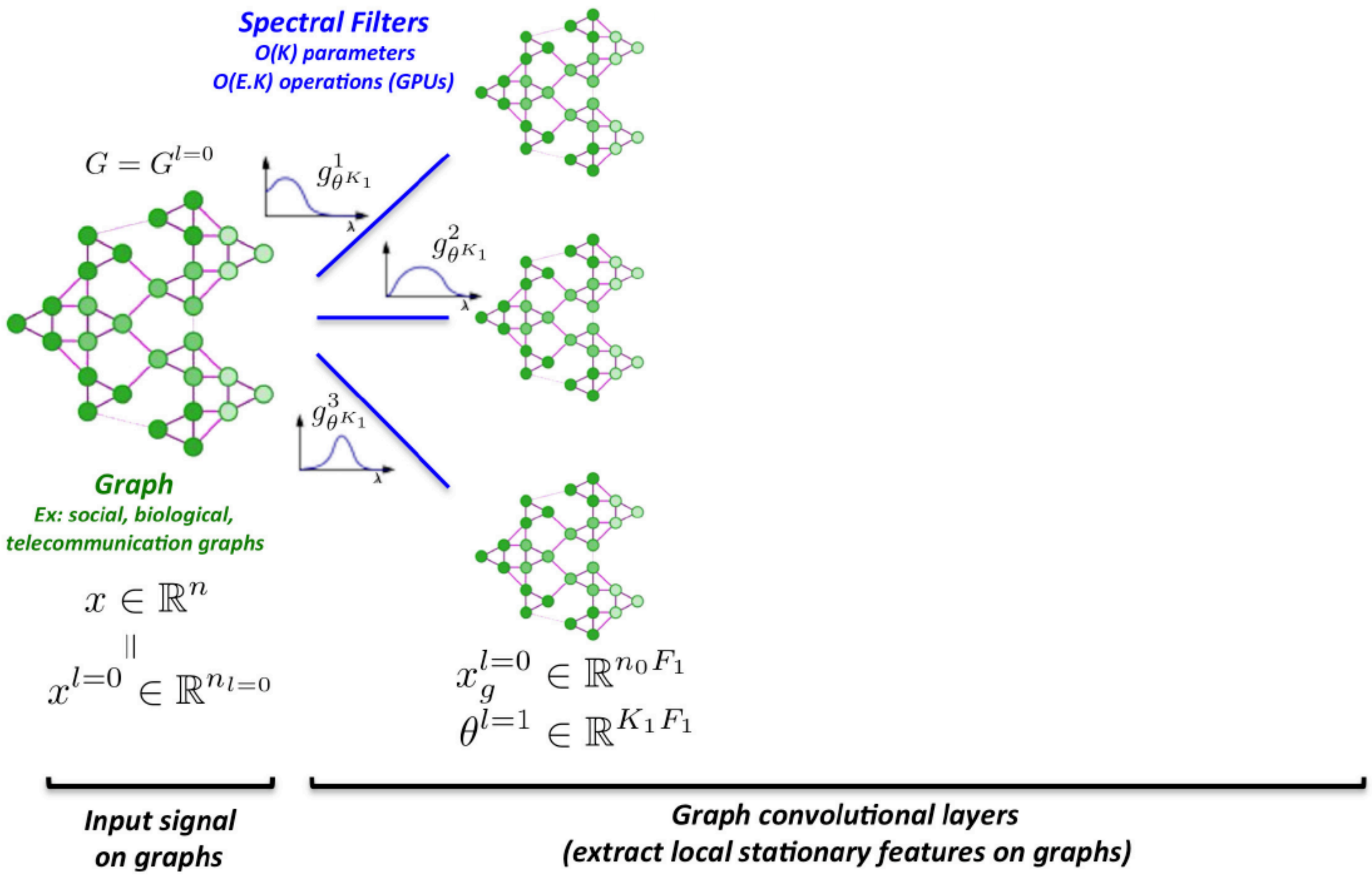
- Other possibilities exist (e.g., a direct spatial approach)

CNNs on graphs: ChebNet



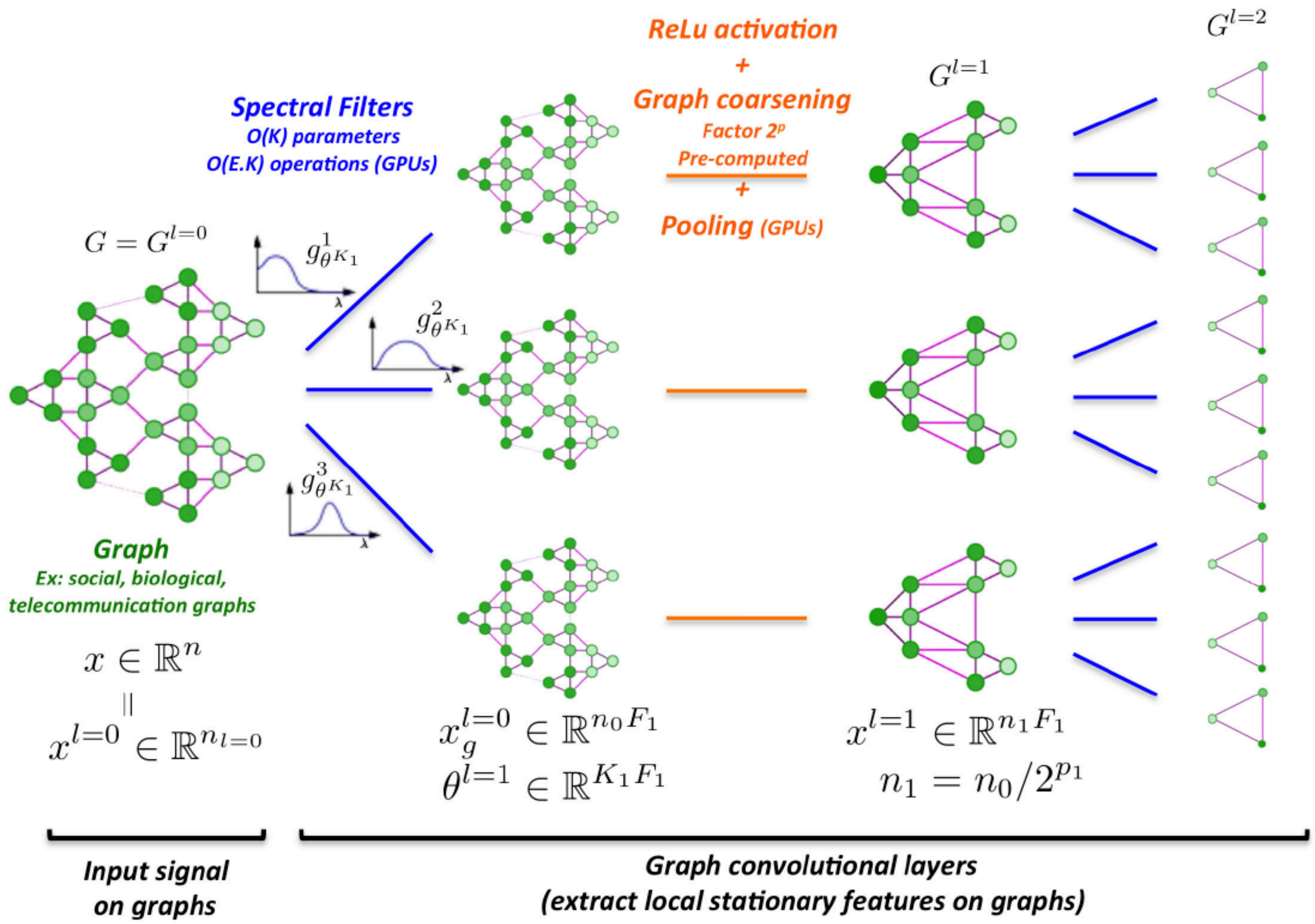
Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNNs on graphs: ChebNet



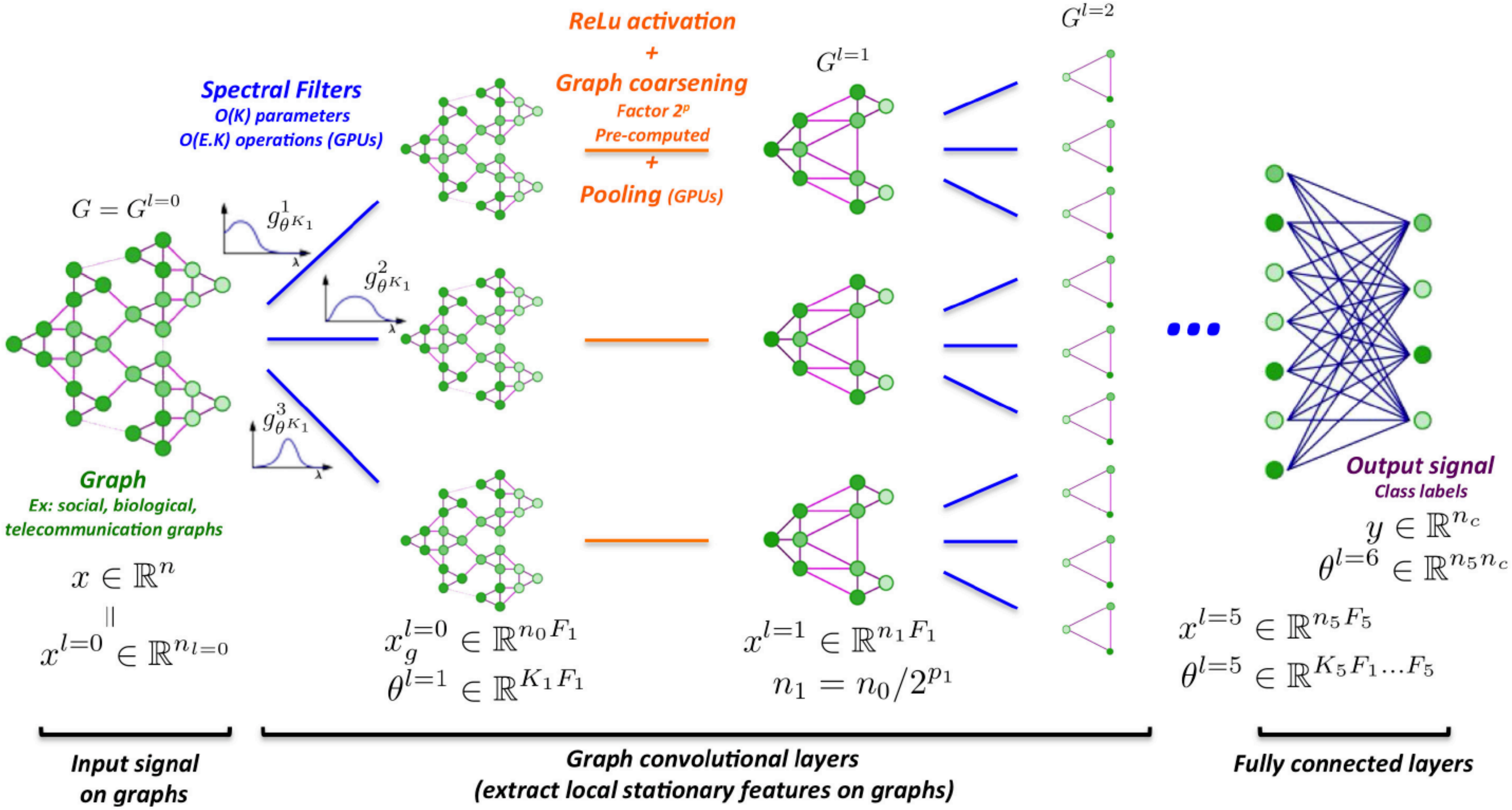
Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNNs on graphs: ChebNet



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

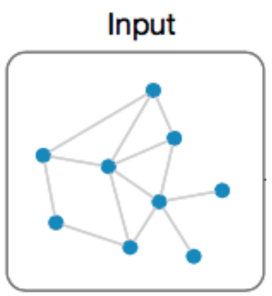
CNNs on graphs: ChebNet



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

CNNs on graphs: GCN

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

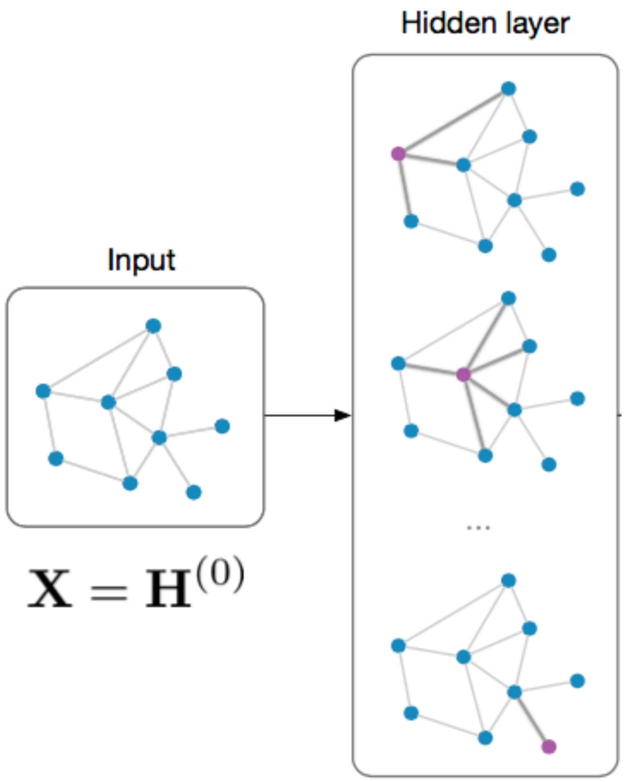


$\mathbf{X} = \mathbf{H}^{(0)}$

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

CNNs on graphs: GCN

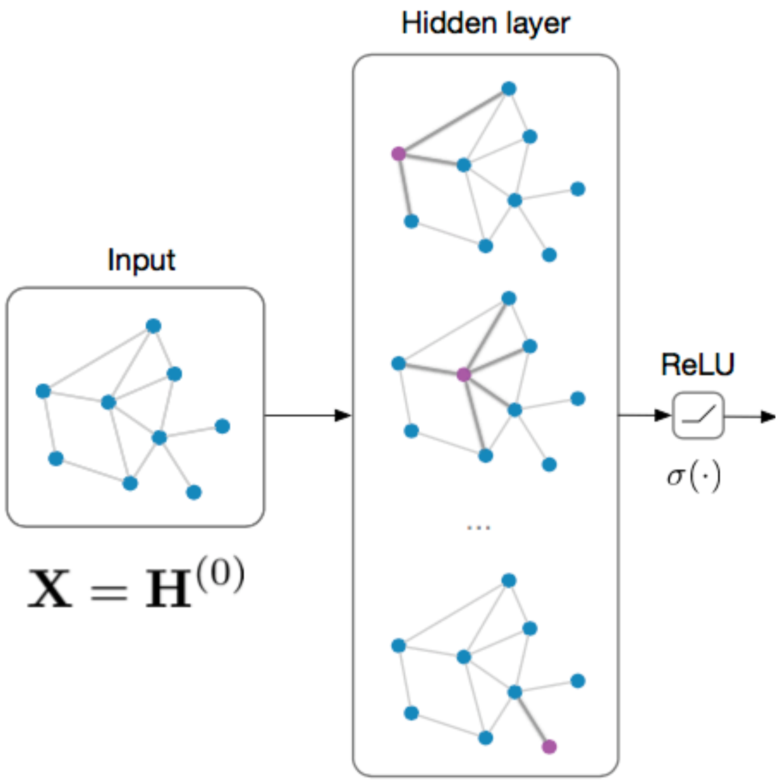
$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$



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

CNNs on graphs: GCN

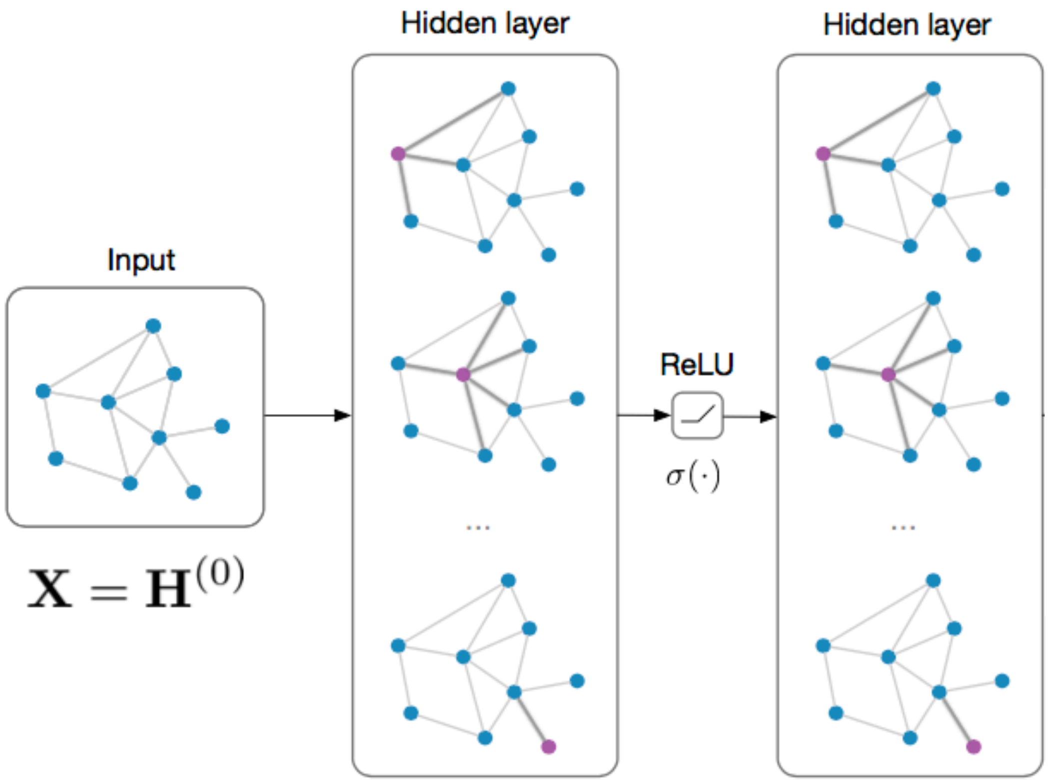
$$\hat{g}_{\theta^{(k+1)}}(L)\left(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\right)$$



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

CNNs on graphs: GCN

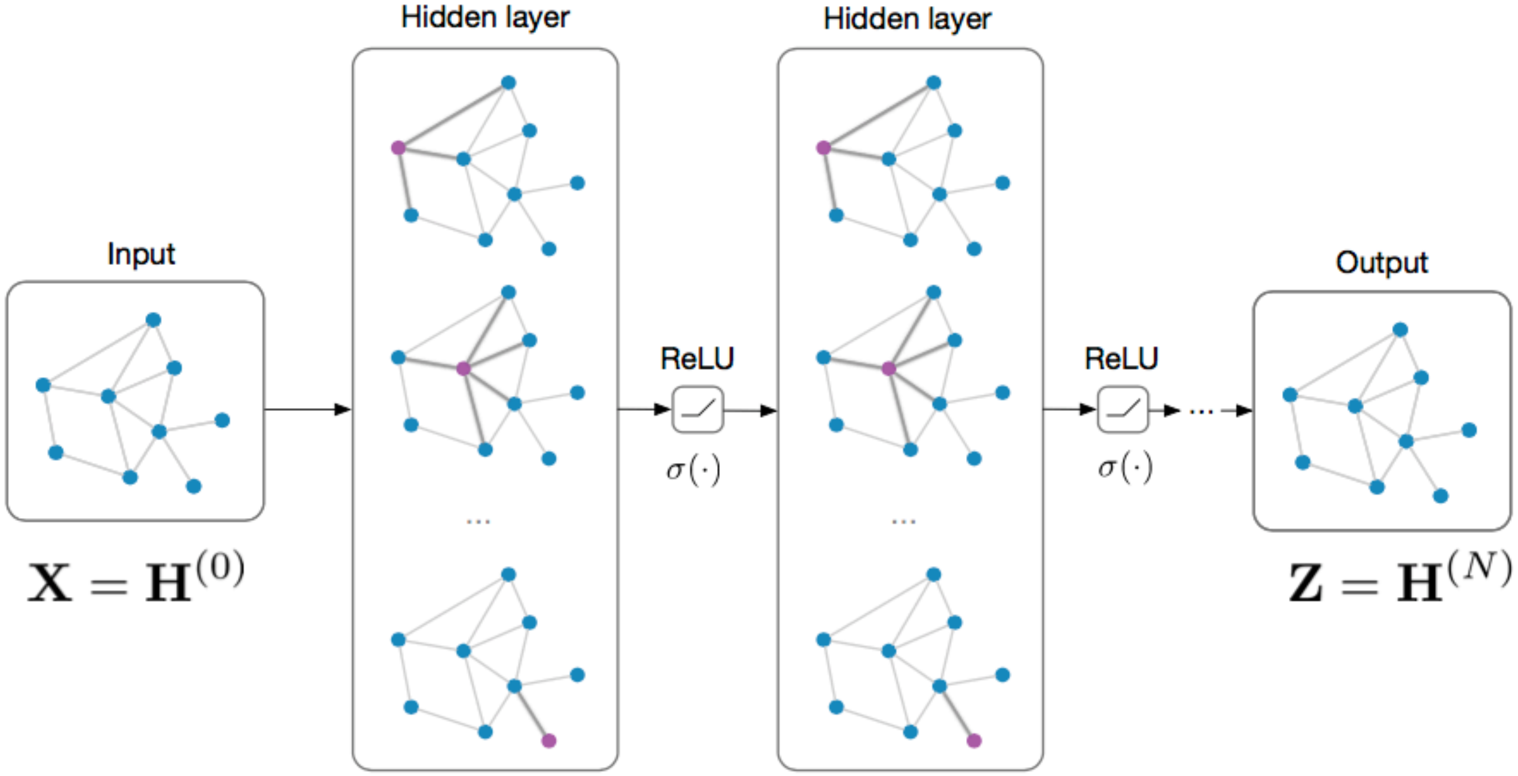
$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\text{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$



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

CNNs on graphs: GCN

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



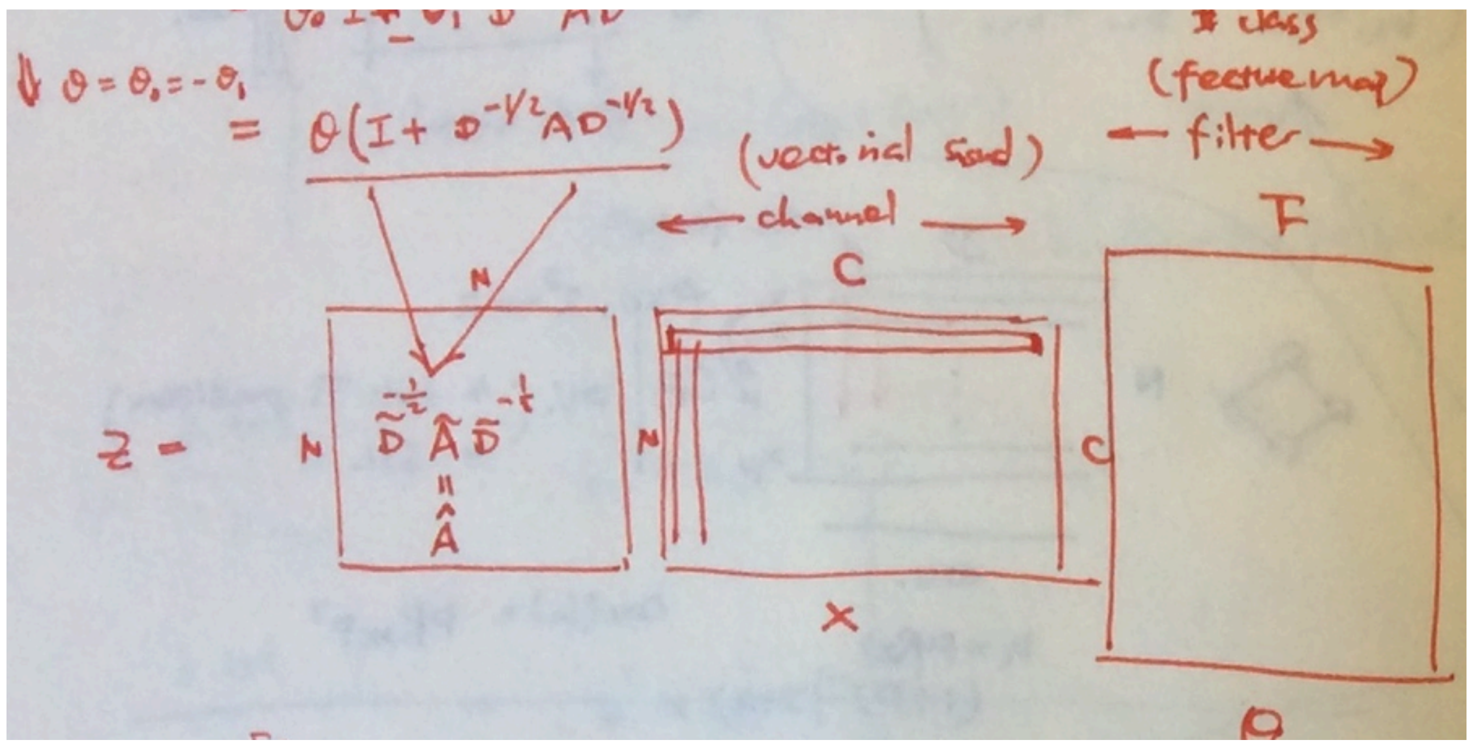
$$\mathbf{X} = \mathbf{H}^{(0)} \qquad \mathbf{Z} = \mathbf{H}^{(N)}$$

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

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

CNNs on graphs: GCN

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



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

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

Implementing CNNs on graphs

- Node-level task
  - cross-entropy loss function for (semi-supervised) node classification

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

set of labelled (training) nodes

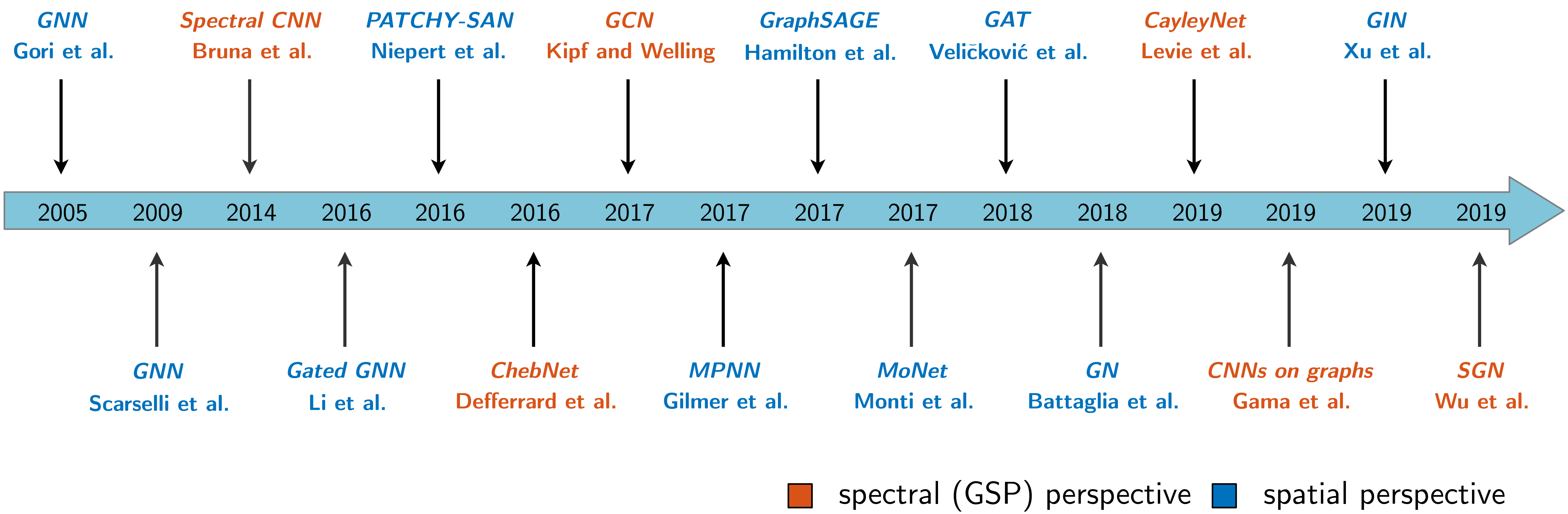
label groundtruth

label prediction (final layer node representation)

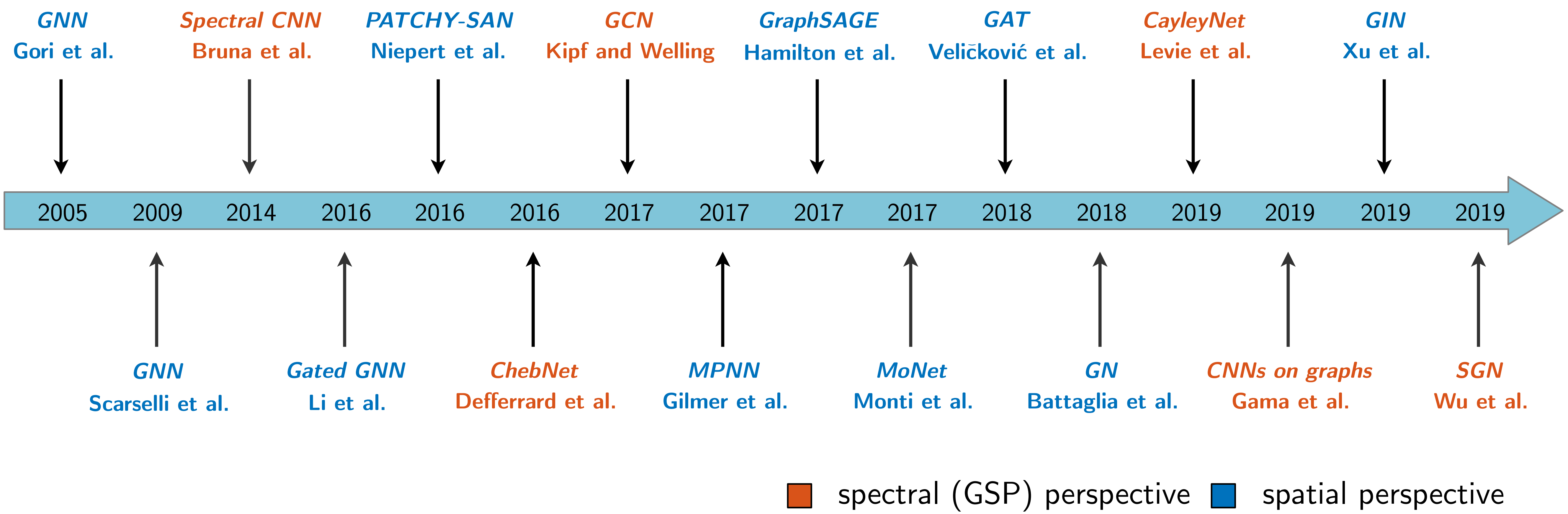
  - training by minimising loss function and making predictions on testing nodes

- Factors influencing model behaviour
  - what label distribution favours GCN in this task?
  - what about perturbation of input graph topology?

(More generally) Graph neural networks



(More generally) Graph neural networks



more recently: graph transformers and LLM-powered models

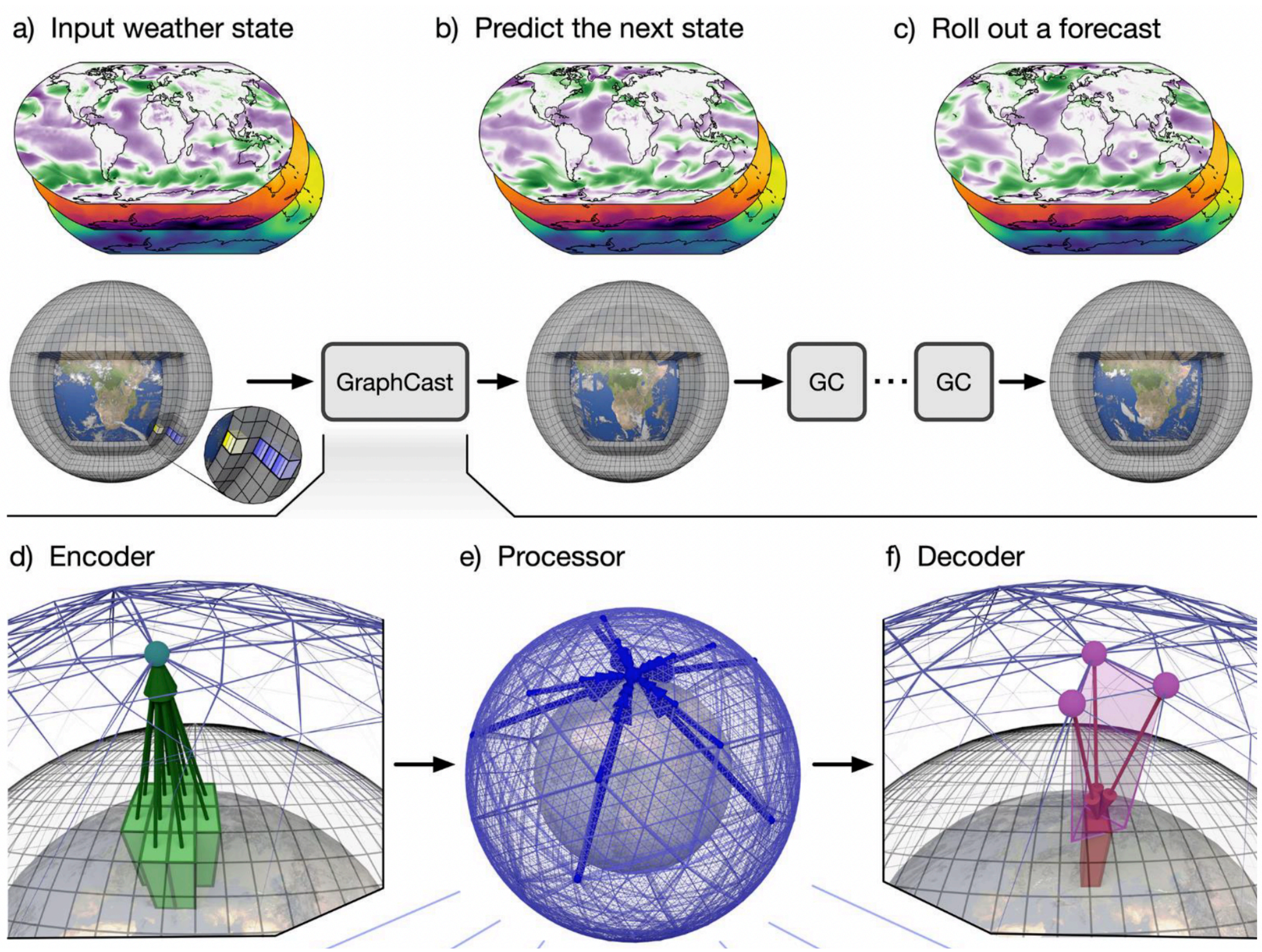
Application I: Traffic prediction

Derrow-Pinion et al., "ETA prediction with graph neural networks in Google Maps," CIKM, 2021.

Application I: Traffic prediction

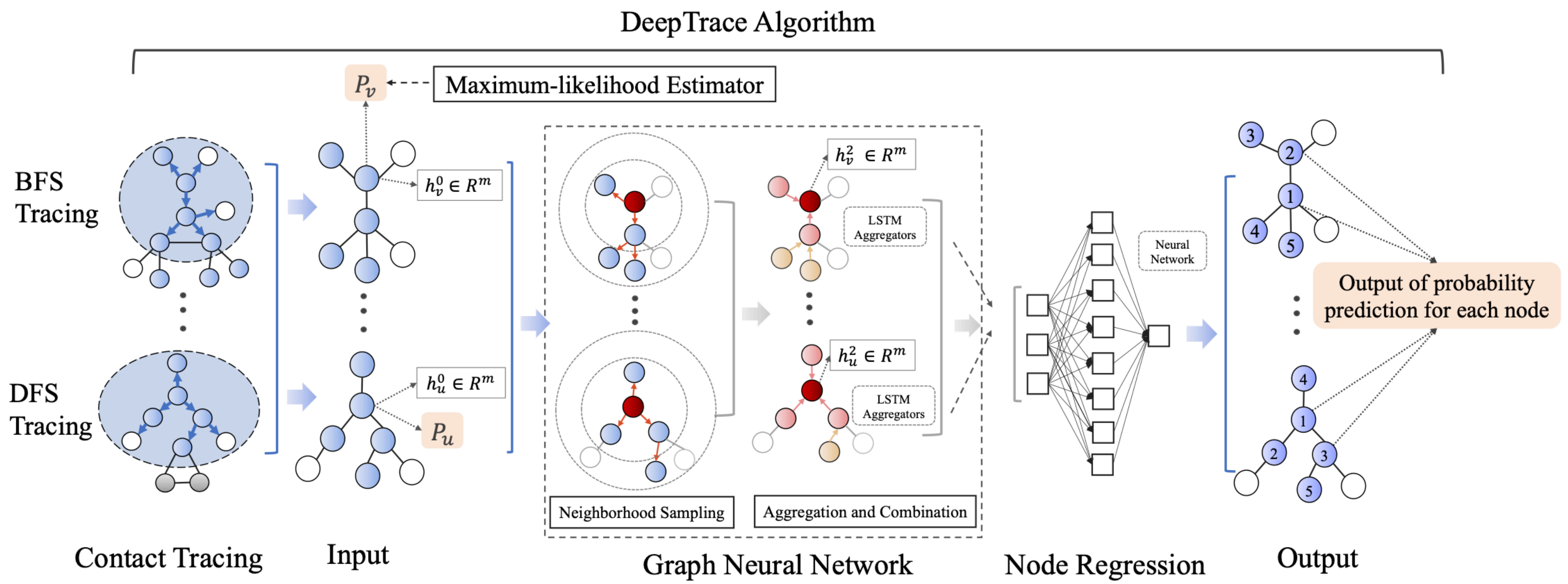
Derrow-Pinion et al., "ETA prediction with graph neural networks in Google Maps," CIKM, 2021.

Application II: Weather forecasting



Lam et al., "Learning skillful medium-range global weather forecasting," Science, 2023.

Application III: Contact tracing

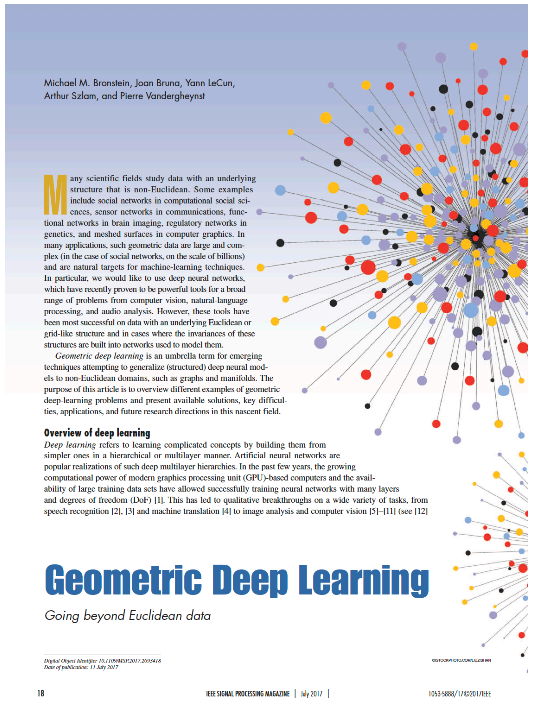


Tan et al., "DeepTrace: Learning to optimize contact tracing in epidemic networks with graph neural networks," IEEE TSIPN, 2025.

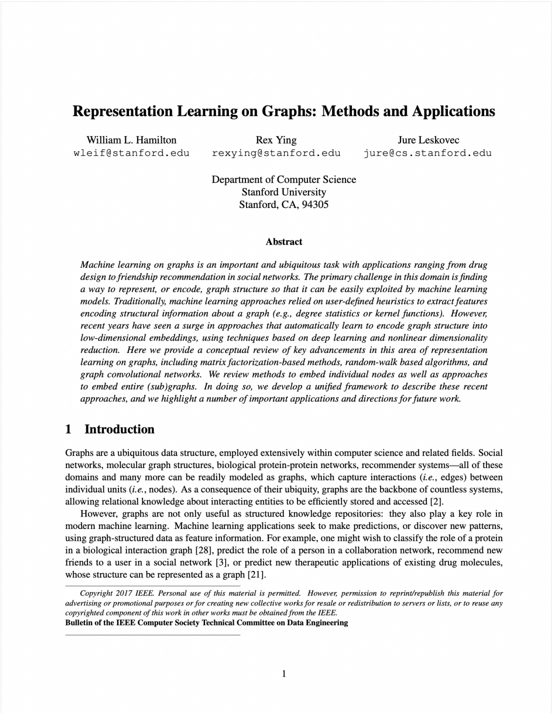
Graph machine learning - Summary

- Fast-growing field that extends data analysis to non-Euclidean domain
- Highly interdisciplinary: machine learning, signal processing, harmonic analysis, applies statistics, differential geometry
- Promising directions
  - going beyond convolutional models (e.g., graph transformers)
  - expressive power of graph ML models
  - robustness & generalisation & scalability
  - interpretability & causal inference
  - construction/refinement of initial graphs
  - applications (particularly in urban science)

References



Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst
Geometric Deep Learning
Going beyond Euclidean data



Representation Learning on Graphs: Methods and Applications
William L. Hamilton
Rex Ying
Jure Leskovec
Department of Computer Science
Stanford University
Stanford, CA, 94305

A Comprehensive Survey on Graph Neural Networks
Zonghan Wu, Shirui Pan, Member, IEEE, Fengwen Chen, Guodong Long, Chengqi Zhang, Senior Member, IEEE, and Philip S. Yu, Life Fellow, IEEE

Machine Learning on Graphs: A Model and Comprehensive Taxonomy
Ines Chami
Sami Abu-El-Haija
Bryan Perozzi
Christopher Ré
Kevin Murphy