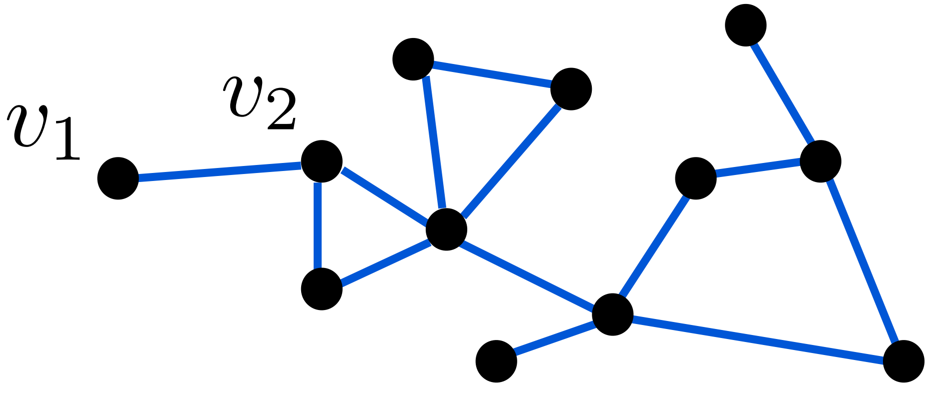
Applied Machine Learning

Graph machine learning - Part II

Xiaowen Dong
Department of Engineering Science

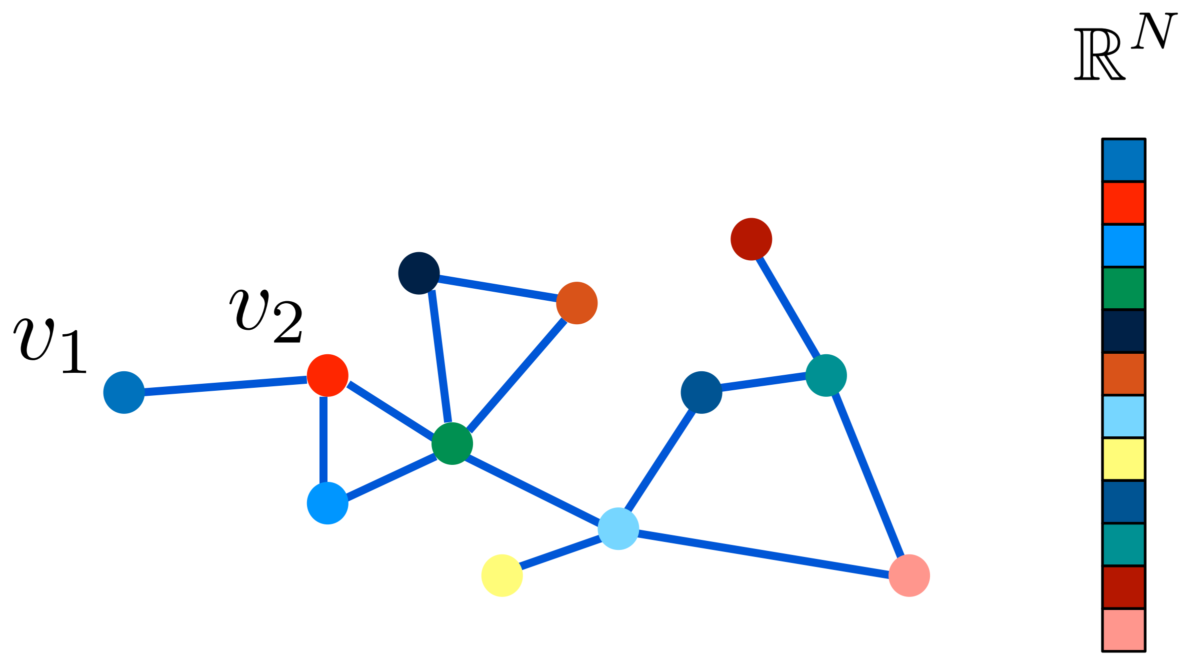
UNIVERSITY OF OXFORD

Recap: Graph signal processing



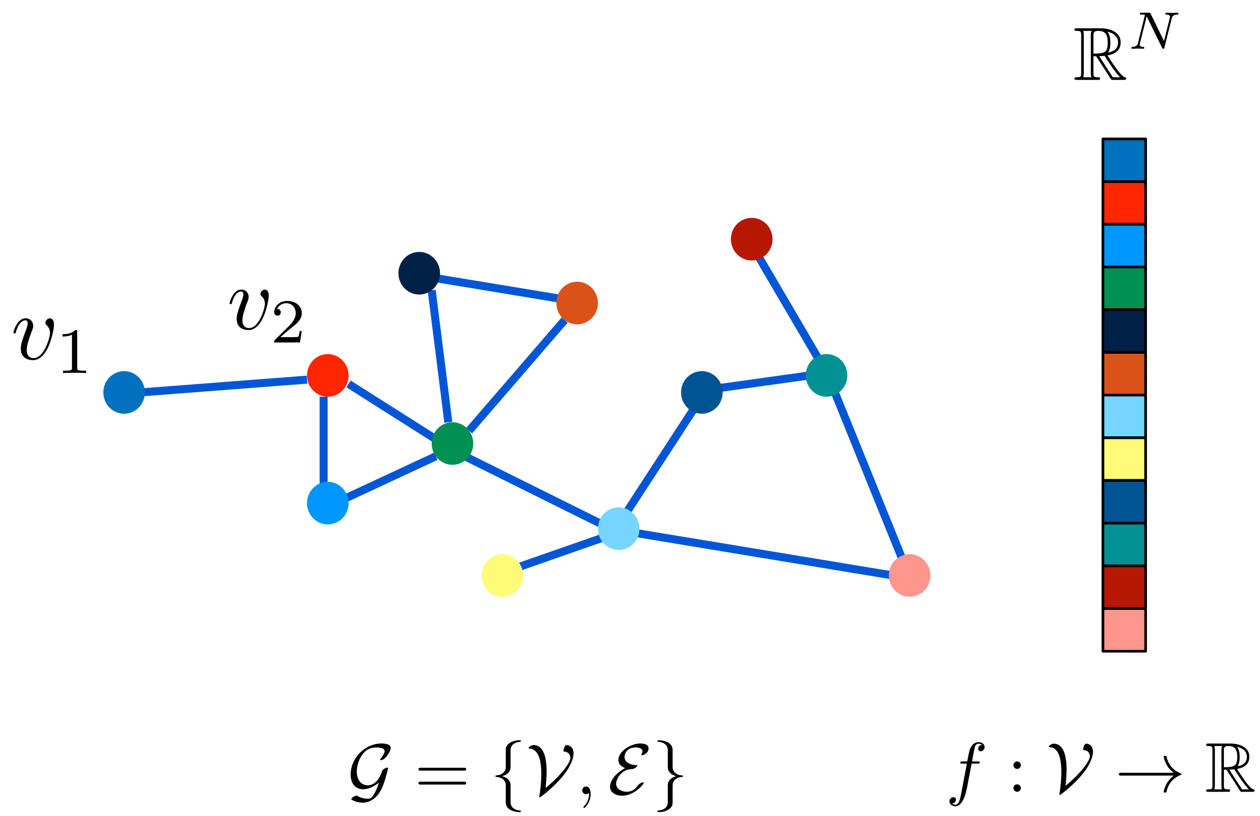
$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

Recap: Graph signal processing



$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ $f : \mathcal{V} \to \mathbb{R}$

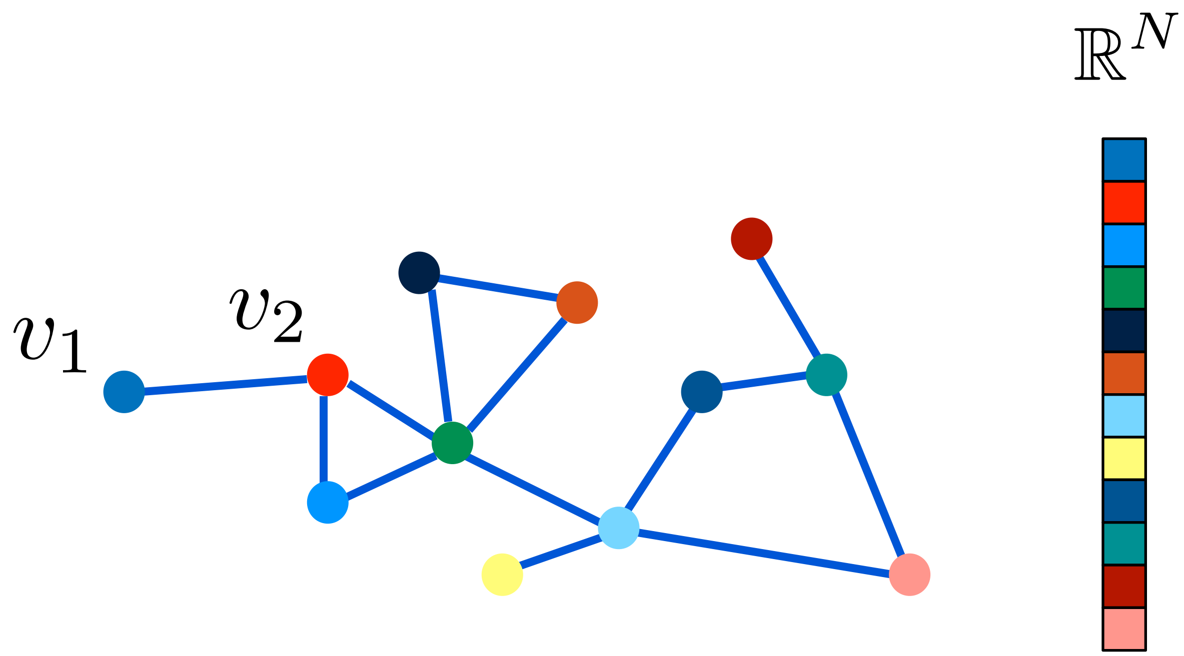
Recap: Graph signal processing



- key concepts
  - smoothness

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

Recap: Graph signal processing

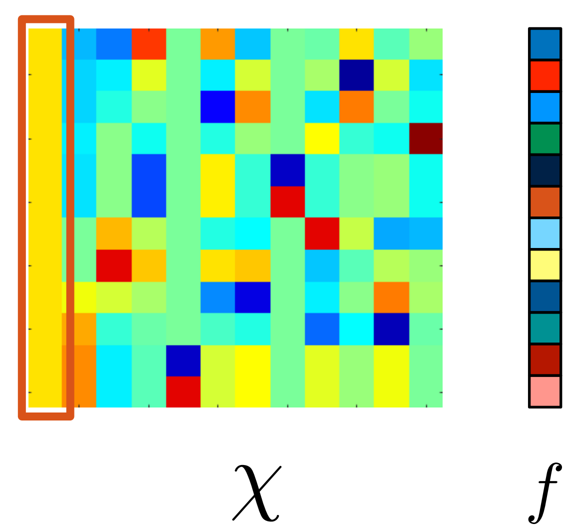


$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\} \qquad f : \mathcal{V} \to \mathbb{R}$$

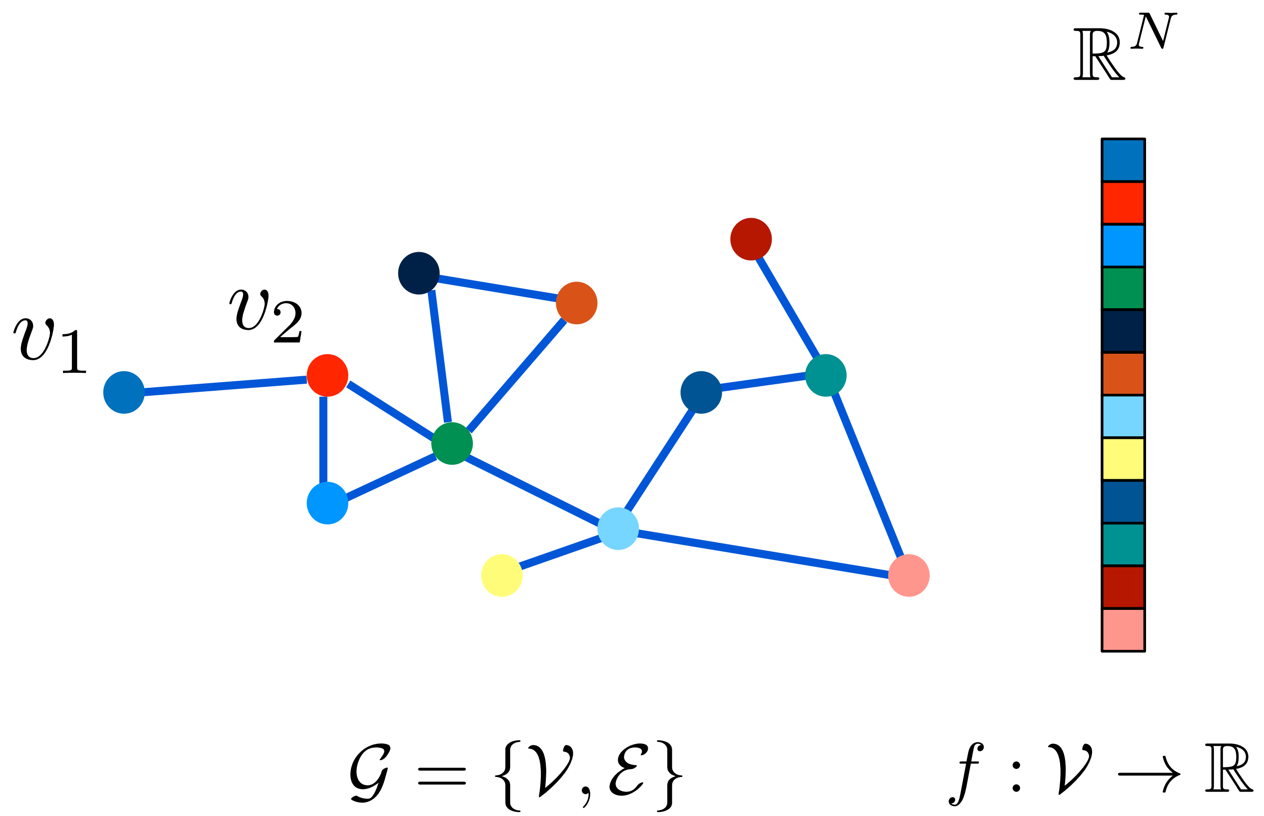
- key concepts
  - smoothness

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

  - Fourier-like analysis



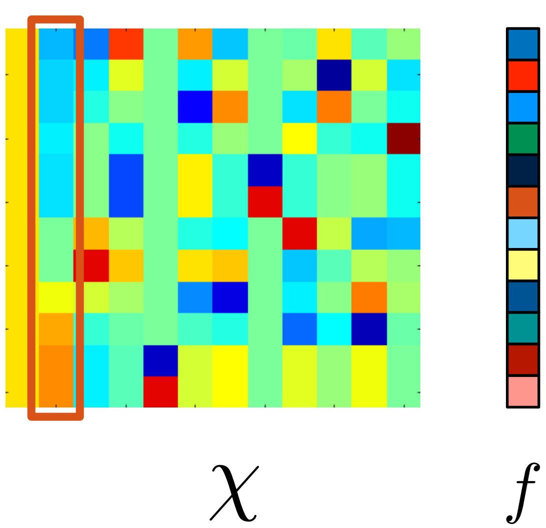
Recap: Graph signal processing



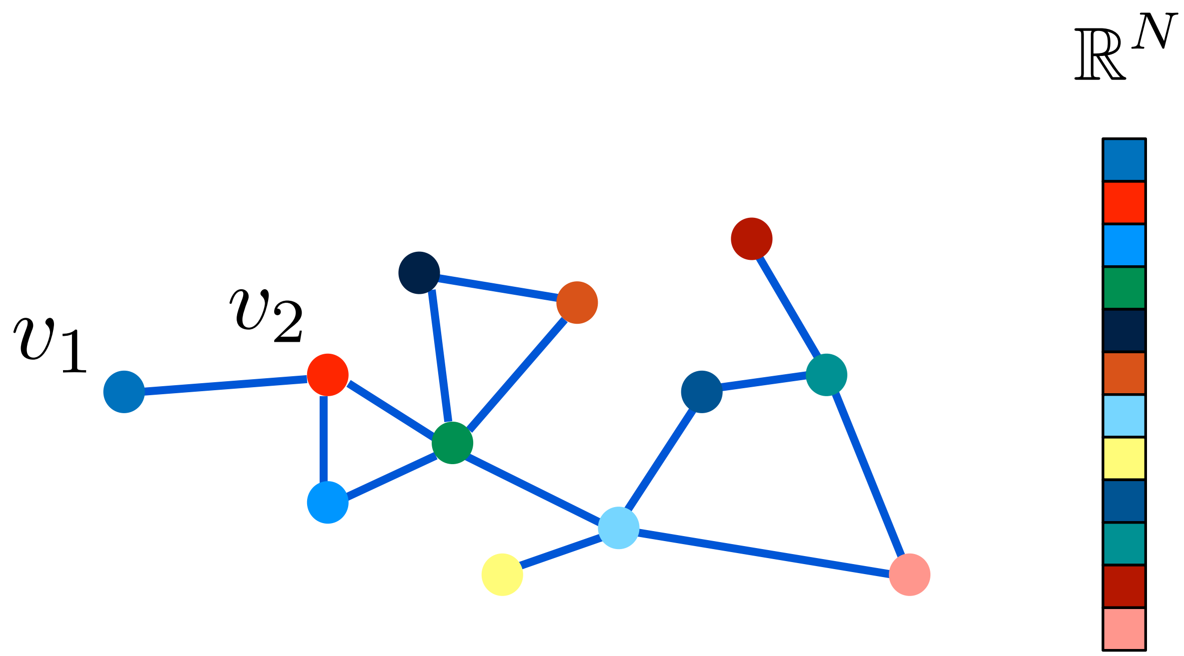
- key concepts
  - smoothness

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

  - Fourier-like analysis



Recap: Graph signal processing

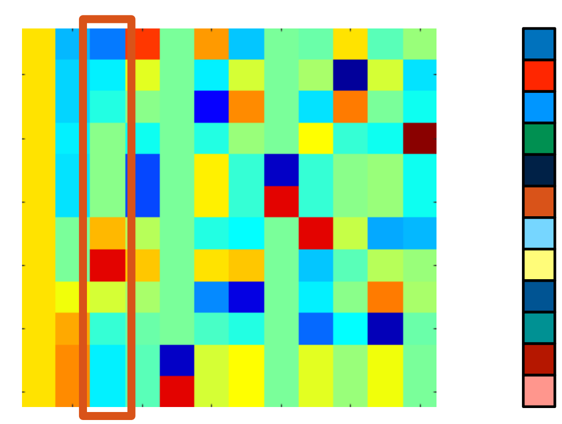


$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ $\quad f : \mathcal{V} \to \mathbb{R}$

- key concepts
  - smoothness

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

  - Fourier-like analysis

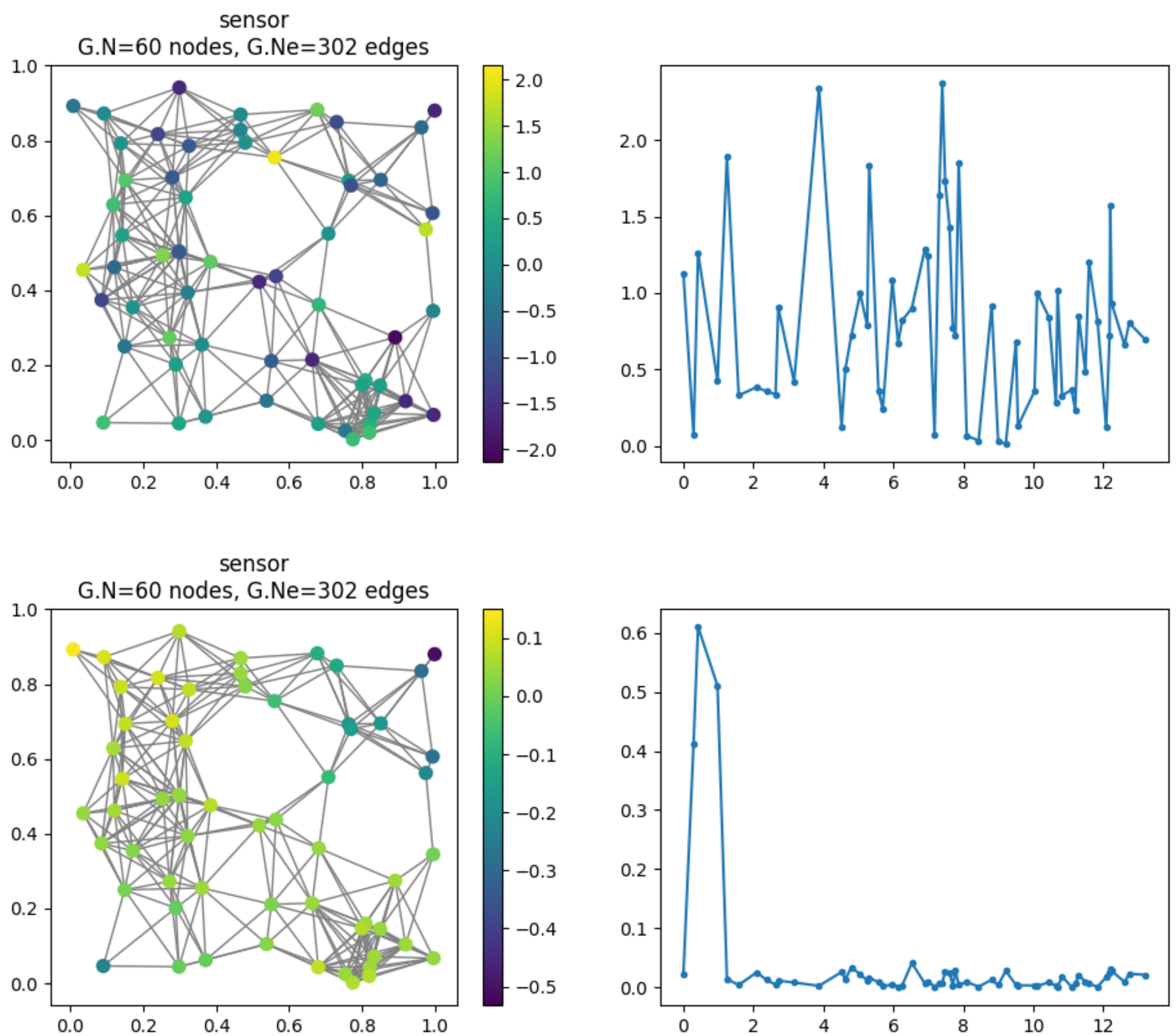


$\chi \qquad f$

Recap: Graph signal processing

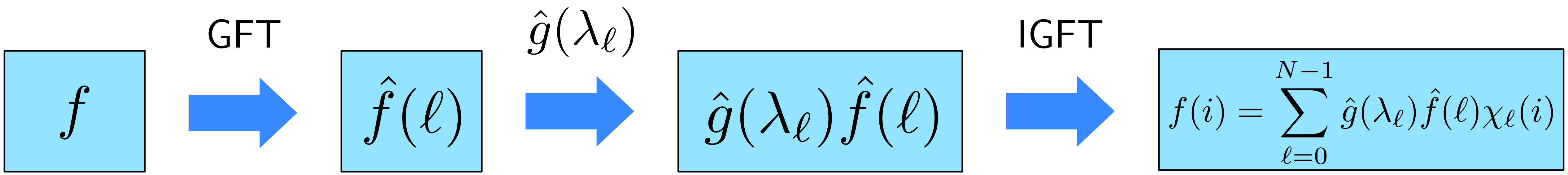
- Graph Fourier transform

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



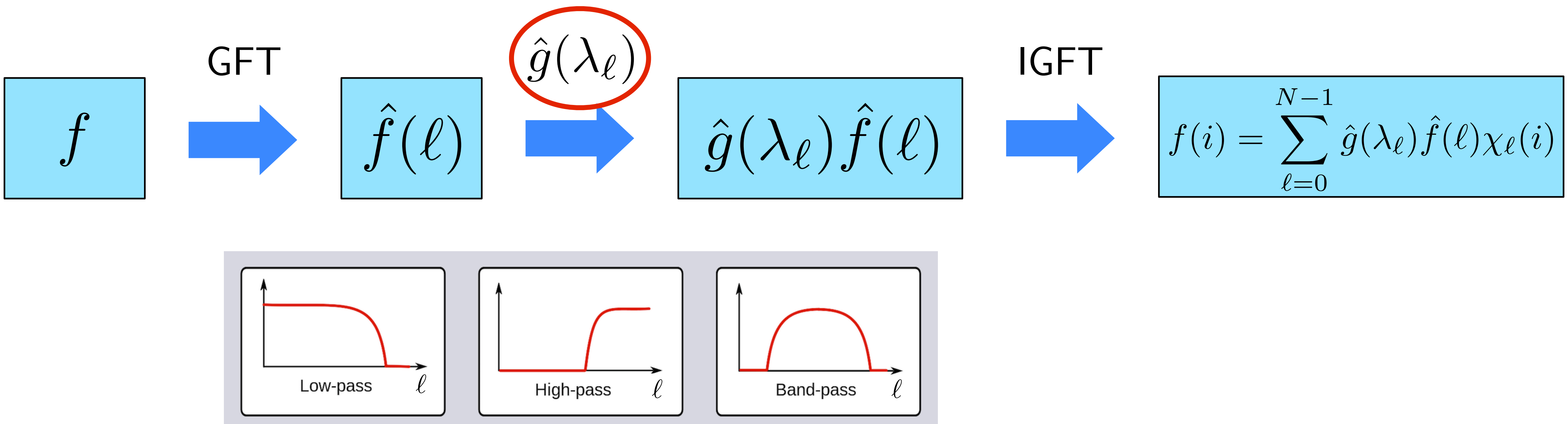
Recap: Graph signal processing

- Graph spectral filtering



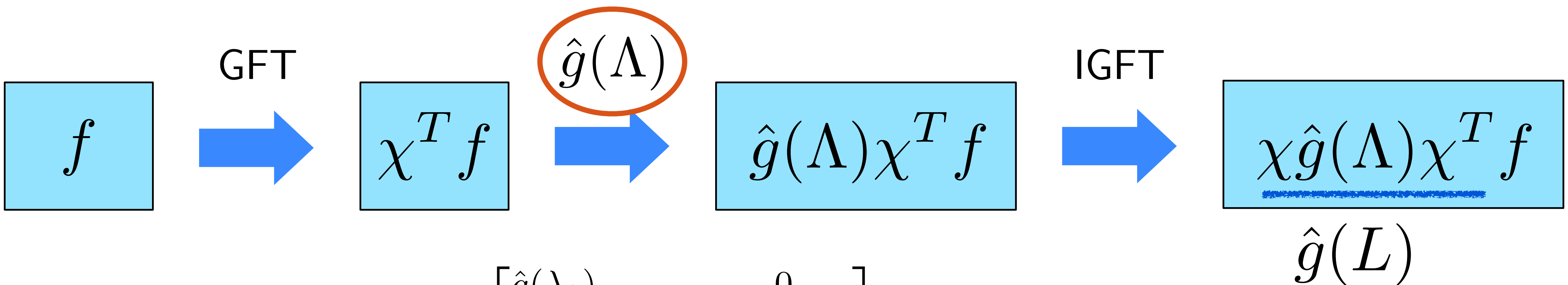
Recap: Graph signal processing

- Graph spectral filtering



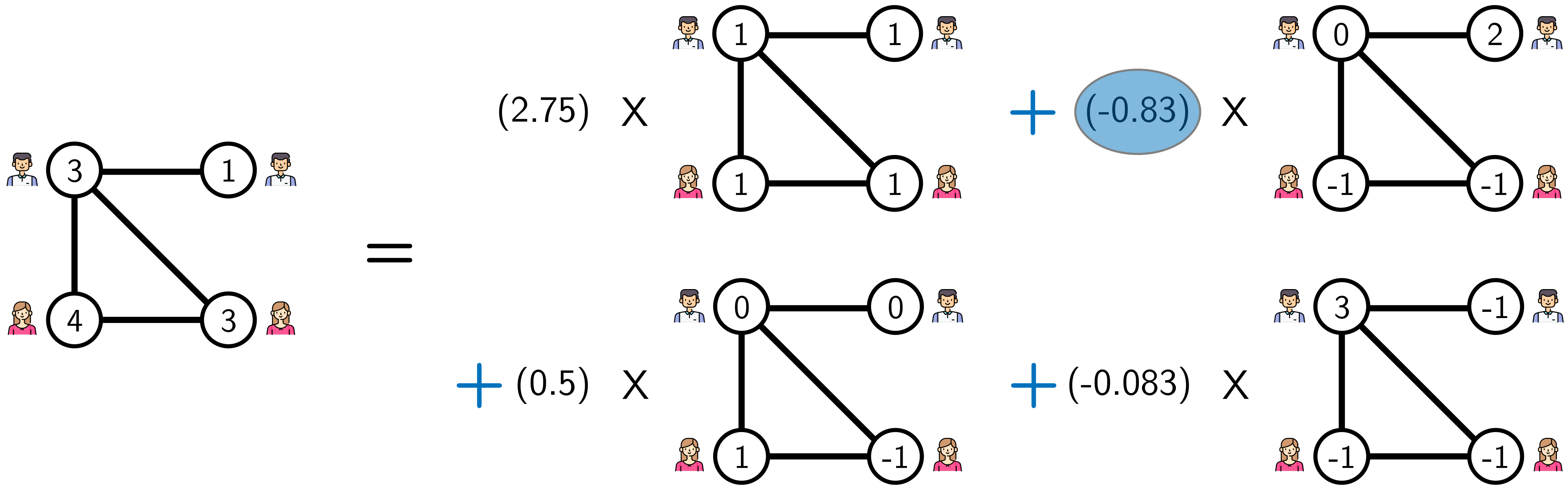
Recap: Graph signal processing

- Graph spectral filtering

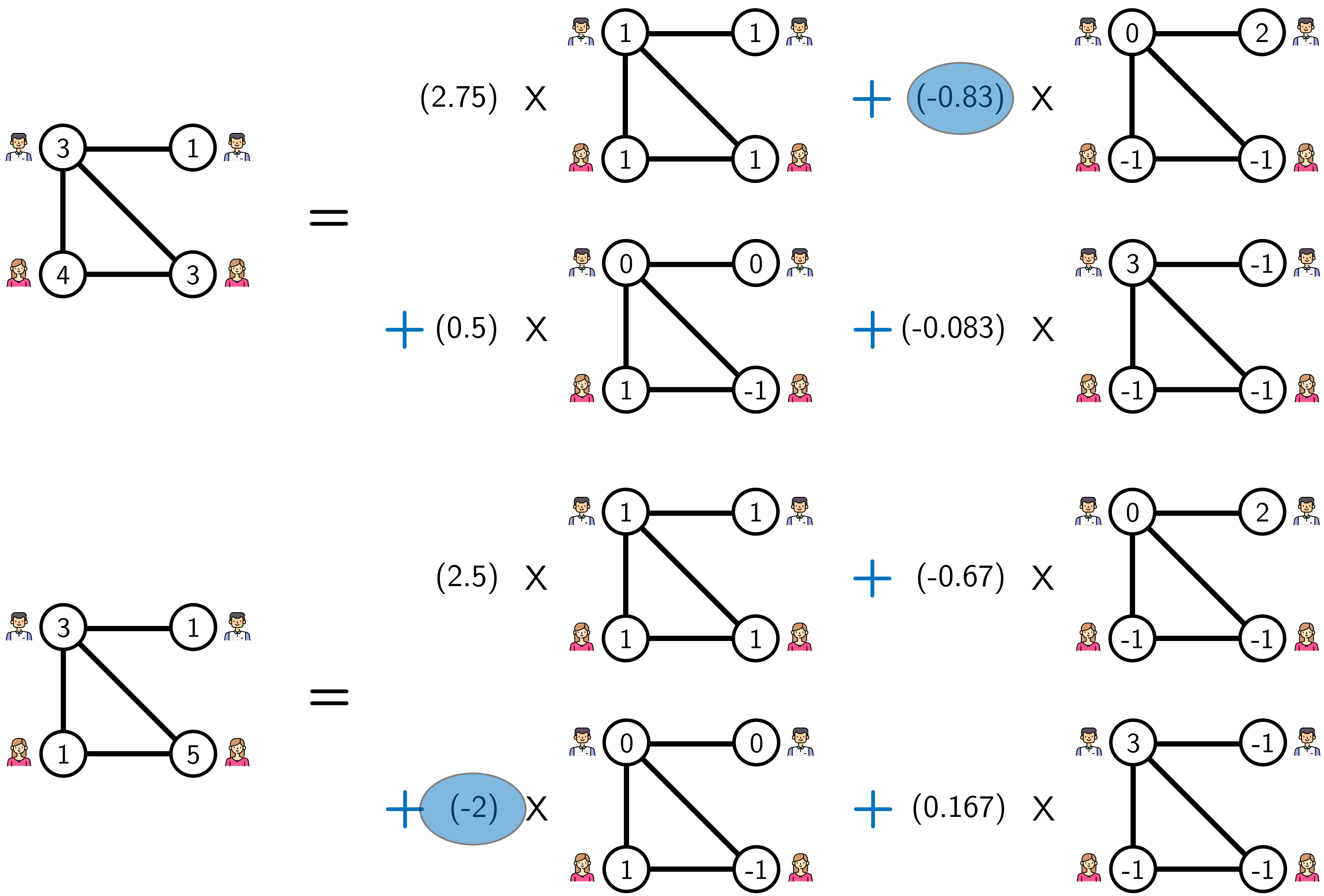


$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

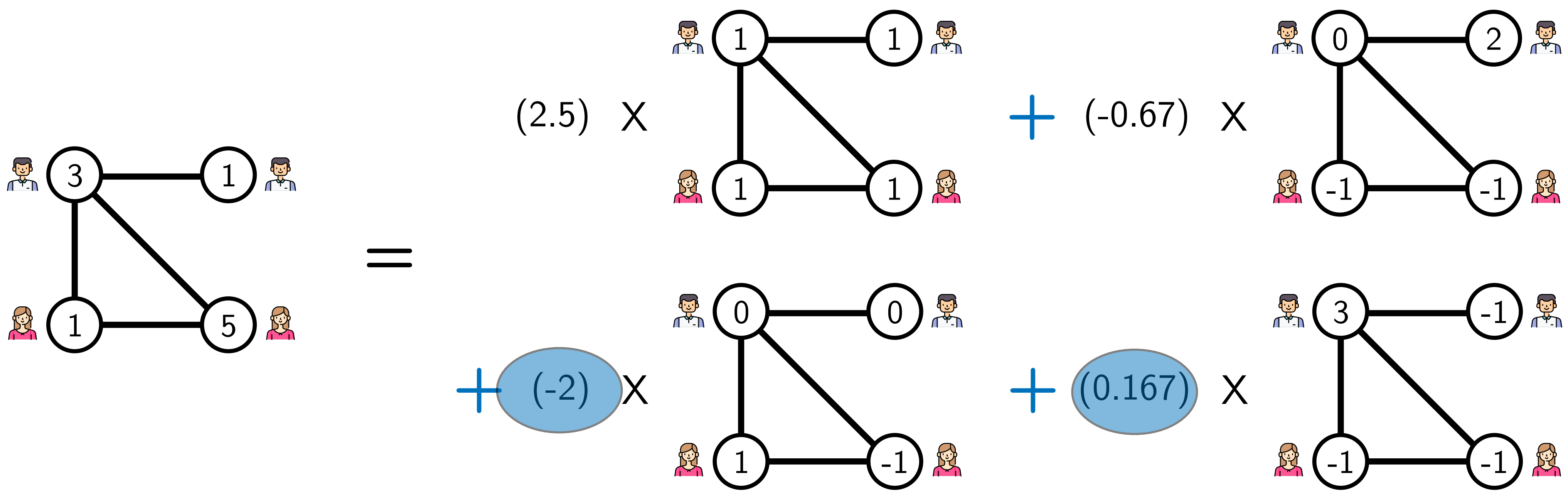
An example on movie rating



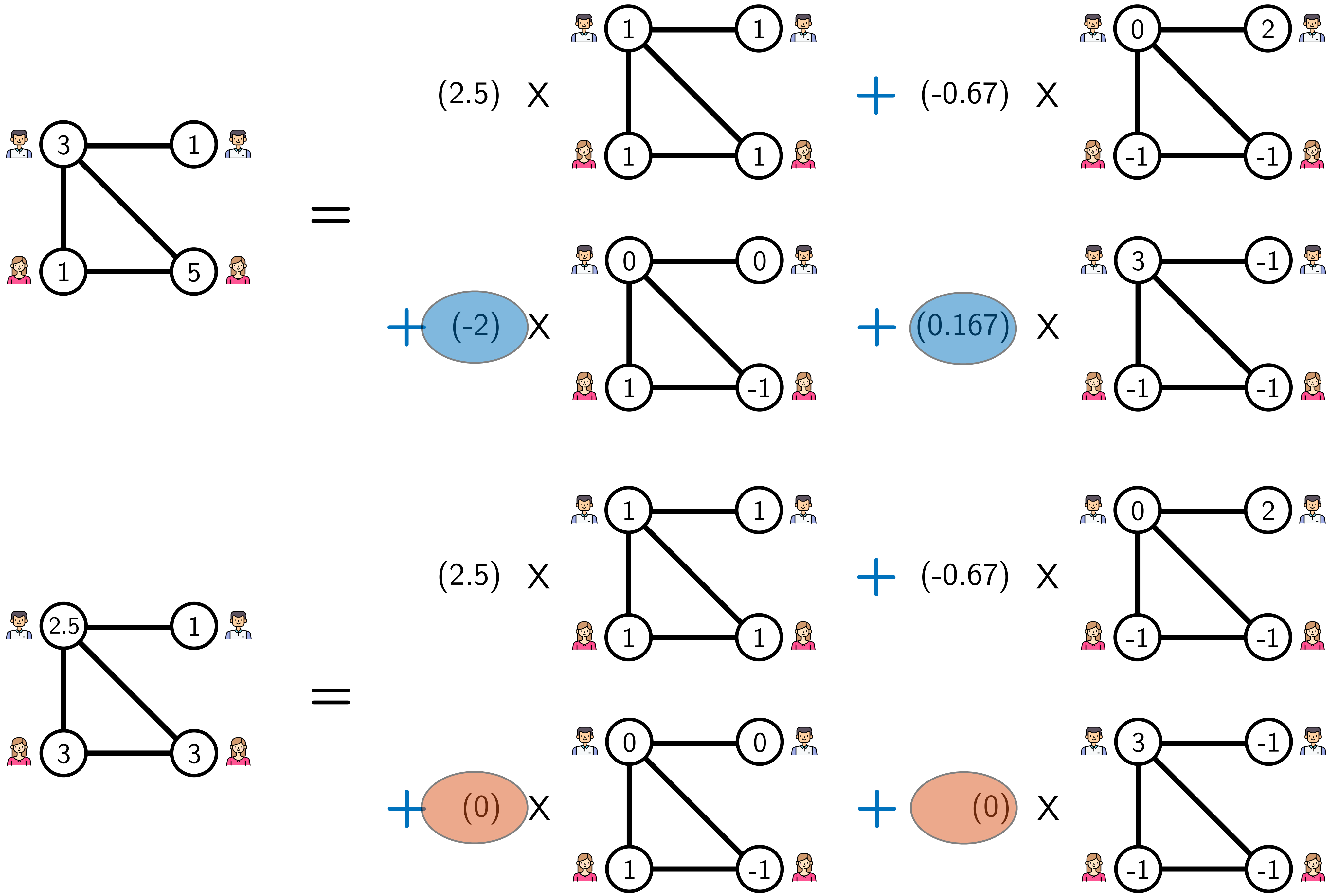
An example on movie rating



An example on movie rating

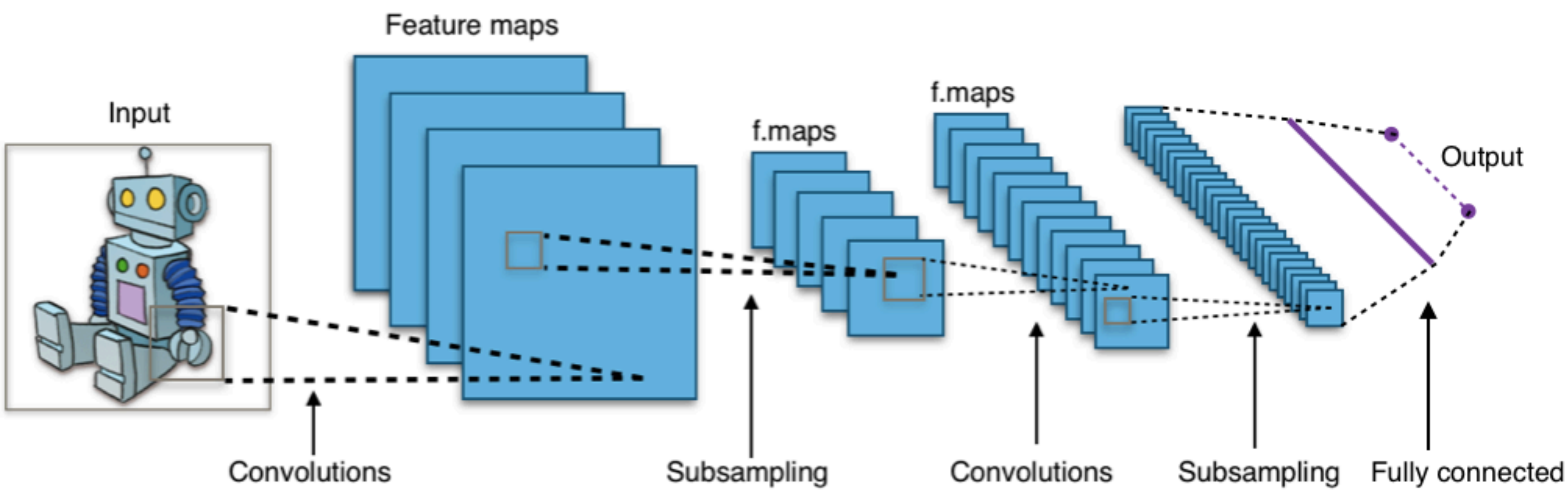


An example on movie rating



Graph Neural Networks

CNNs exploit structure within data

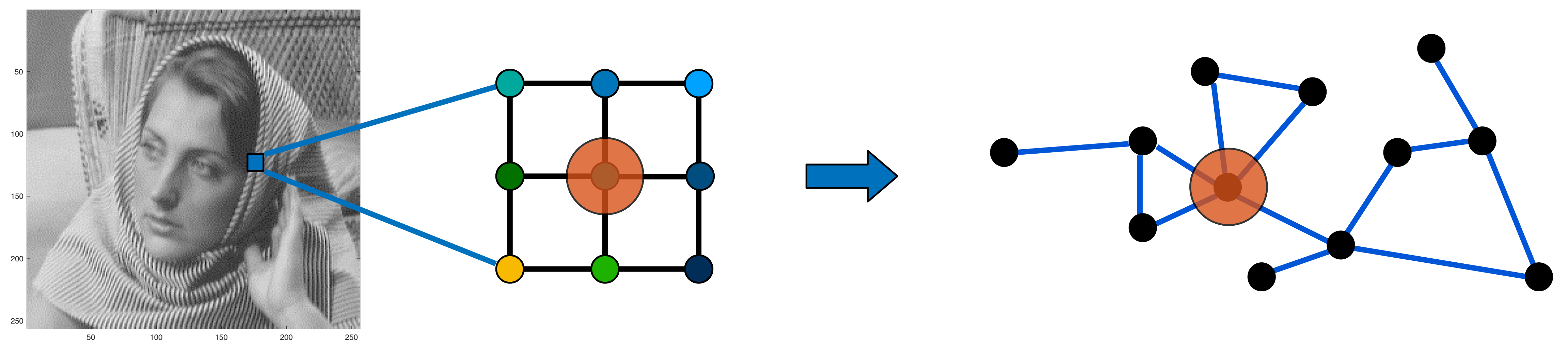


checklist

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

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

CNNs on graphs?



checklist

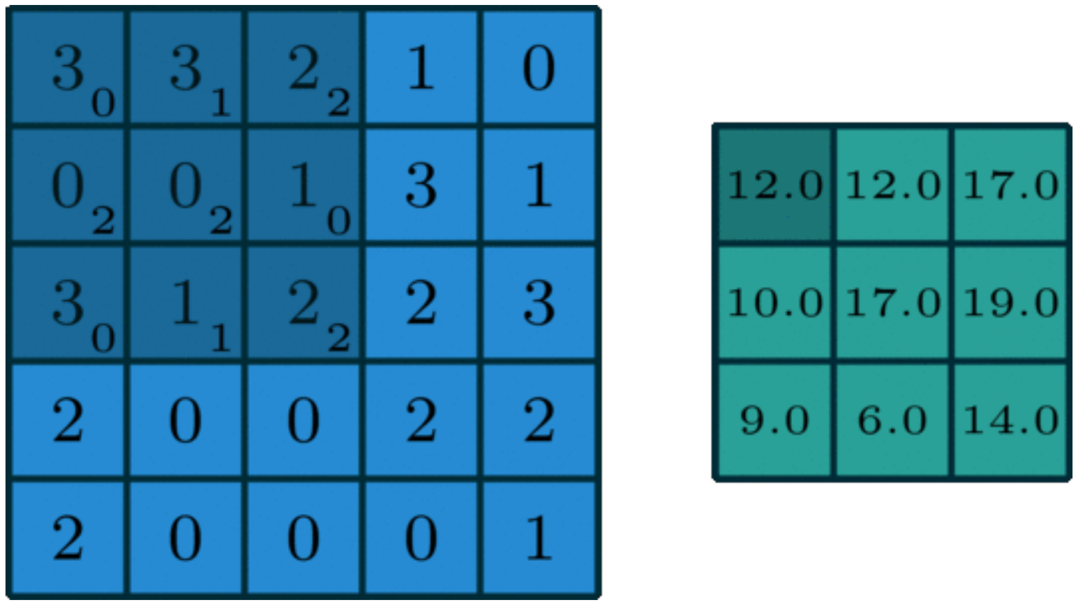
- **convolution:** how to do it on graphs?
- **localisation:** what's the notion of locality?
- **multi-scale:** how to down-sample on graphs?
- **efficiency:** how to keep the computational complexity low?

Convolution on graphs

classical convolution

time domain

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



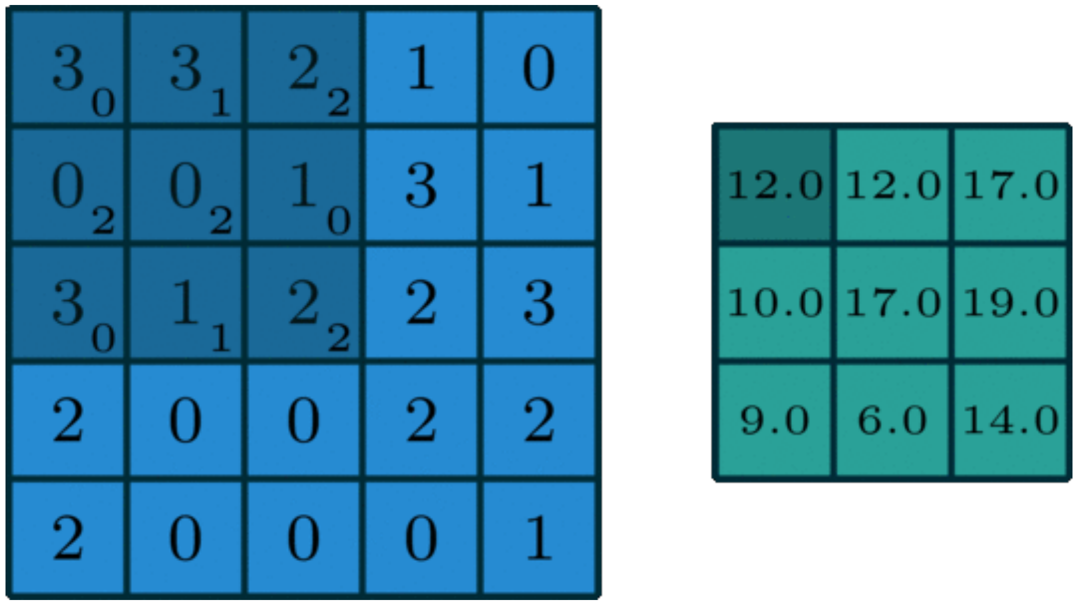
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution on graphs

classical convolution

time domain

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



Convolution on graphs

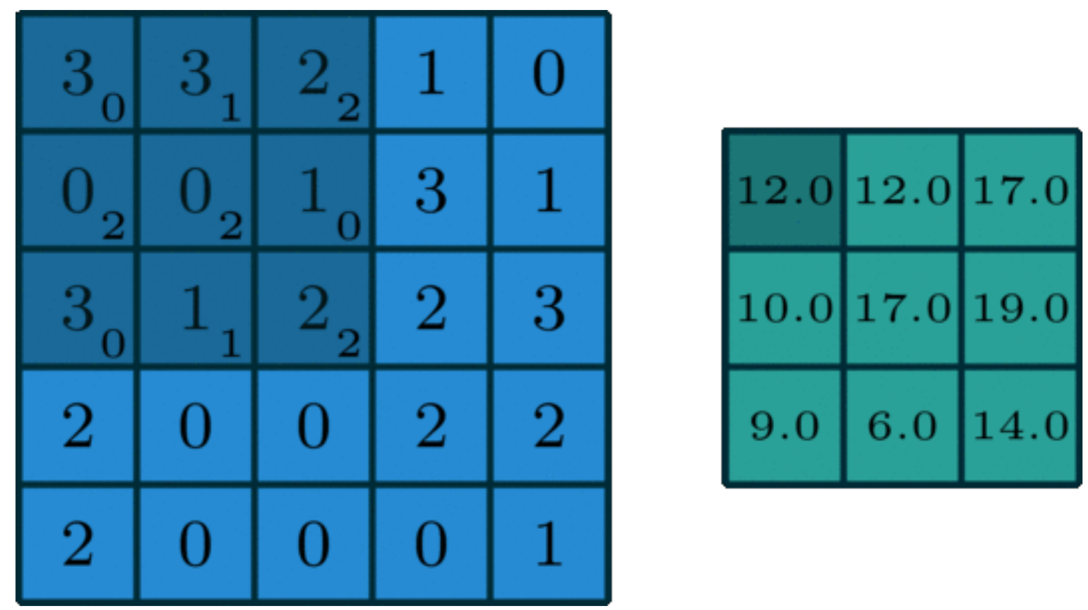
classical convolution

time domain

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

frequency domain

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



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

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

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

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

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

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

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

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

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

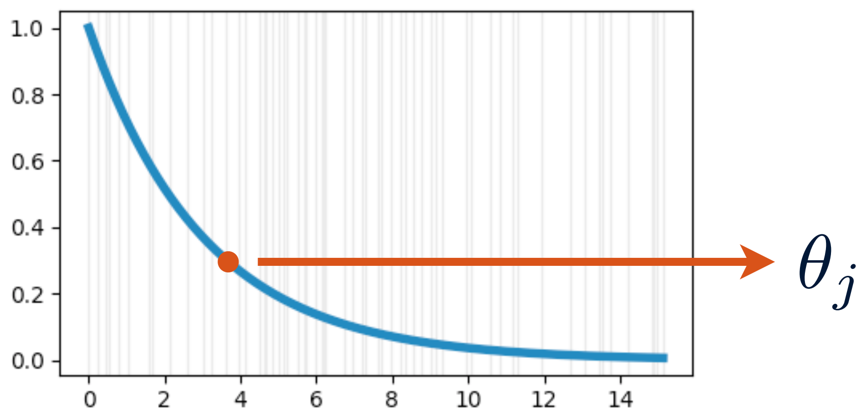
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

A non-parametric filter

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

learning a non-parametric filter:

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



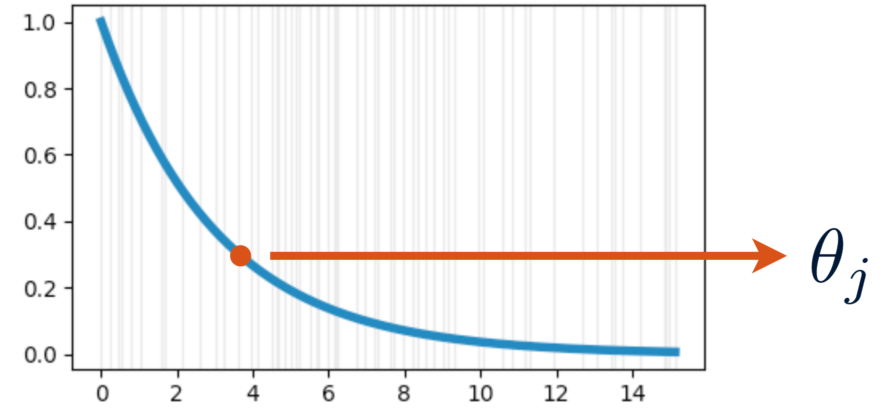
Bruna et al., "Spectral networks and deep locally connected networks on graphs," ICLR, 2014.

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

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

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

A parametric filter

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

parametric filter as polynomial of Laplacian

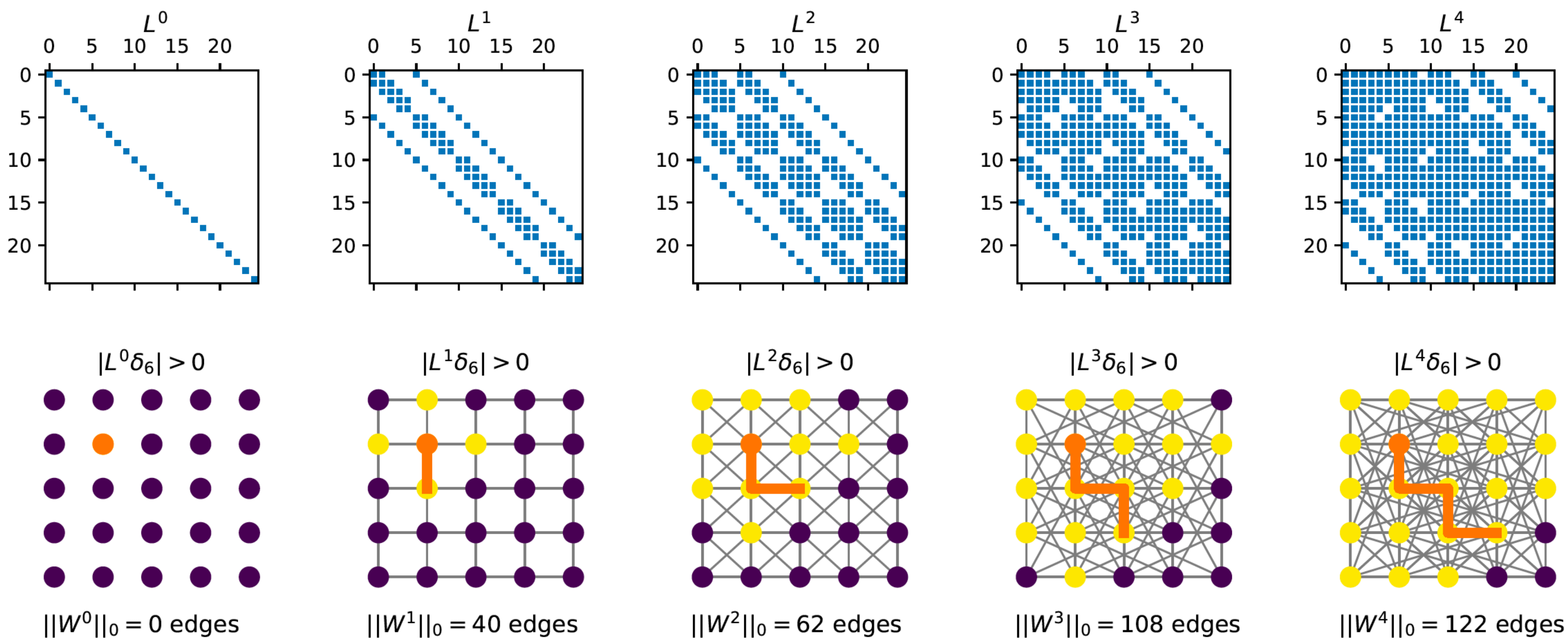
$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1} \quad \Rightarrow \quad \hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

what do powers of graph Laplacian capture?

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

Powers of graph Laplacian

L^k defines the k-neighborhood



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

(slide by Michaël Deferrard)

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

A parametric filter

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

⇓

parametric filter as polynomial of Laplacian

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

⇒

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

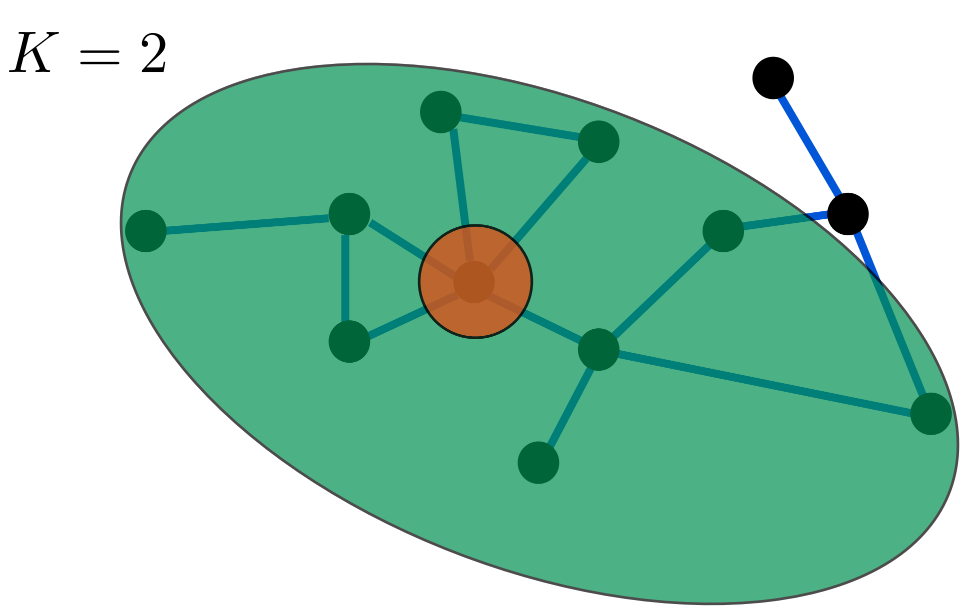
Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

A parametric filter

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

parametric filter as polynomial of Laplacian

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



- localisation within *K*-hop neighbourhood
- efficient computation via recursive multiplication with L

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

A simplified parametric filter

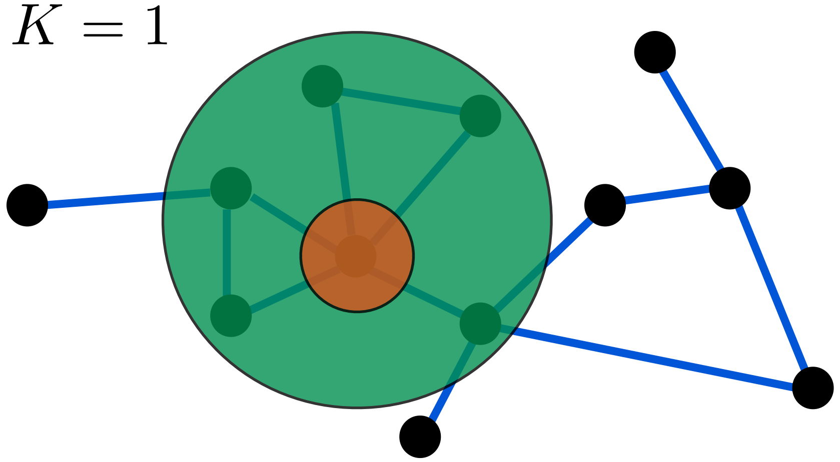
$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

simplified parametric filter

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

(localisation within 1-hop neighbourhood)

$K = 1$



A simplified parametric filter

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

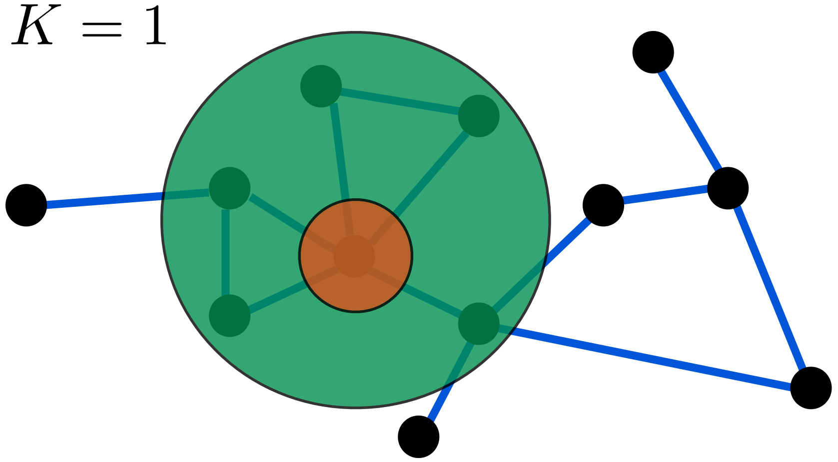
simplified parametric filter

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

$K = 1$

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

(localisation within 1-hop neighbourhood)



$\alpha = \theta_0 = -\theta_1$

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

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

A simplified parametric filter

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

simplified parametric filter

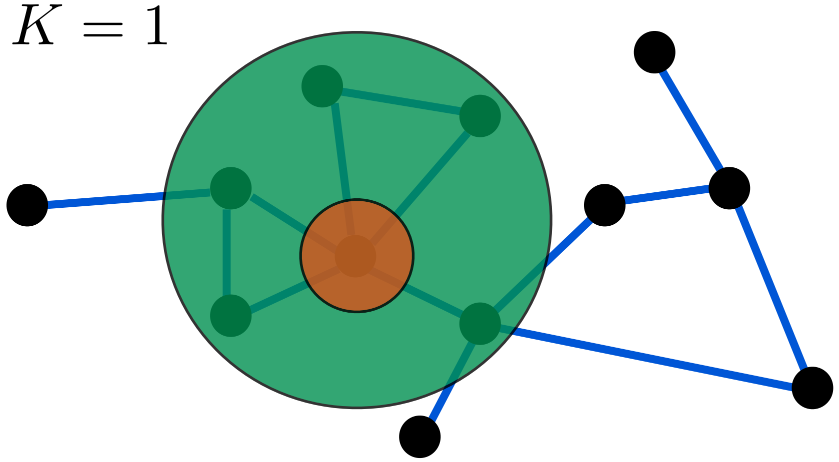
$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

$K = 1$

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

(localisation within 1-hop neighbourhood)

$K = 1$



$\alpha = \theta_0 = -\theta_1$

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

renormalisation

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

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.

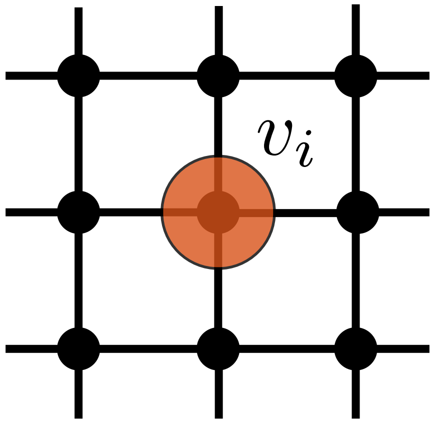
A simplified parametric filter

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

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

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

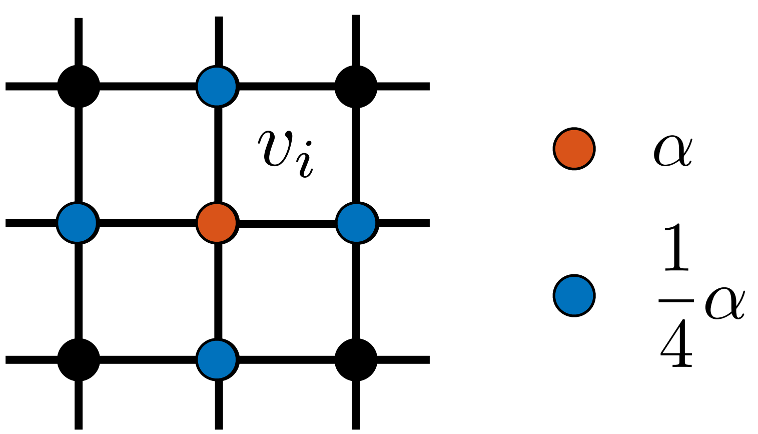
simplified parametric filter

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

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

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

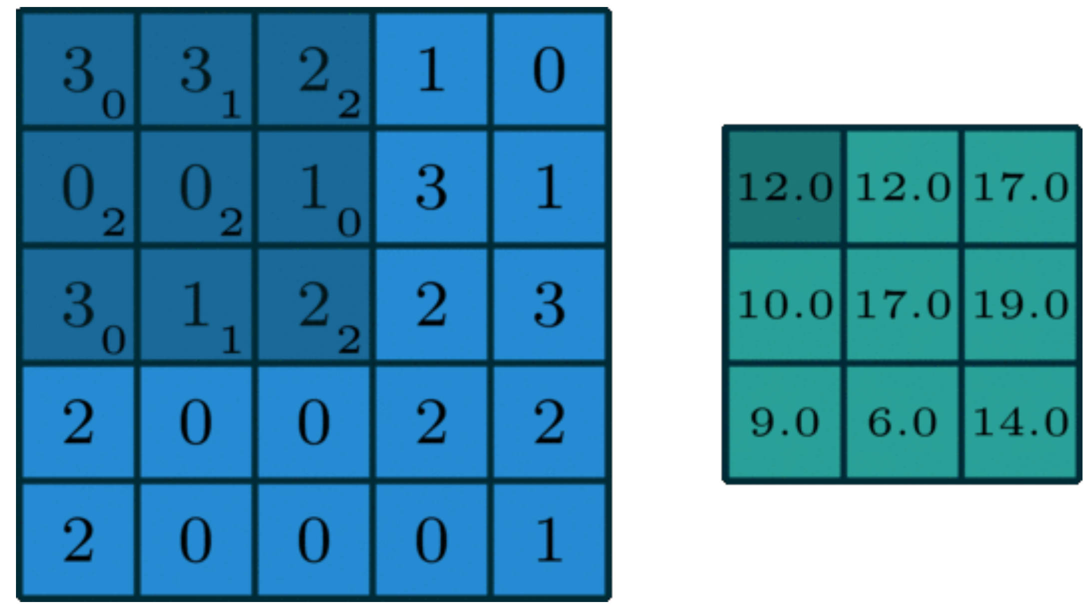
$$\hat{g}_\alpha(L) = \alpha(I + D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$$

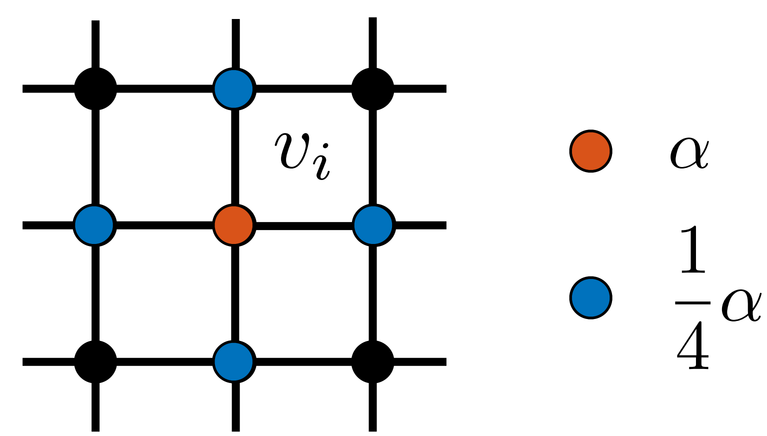
⬇

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

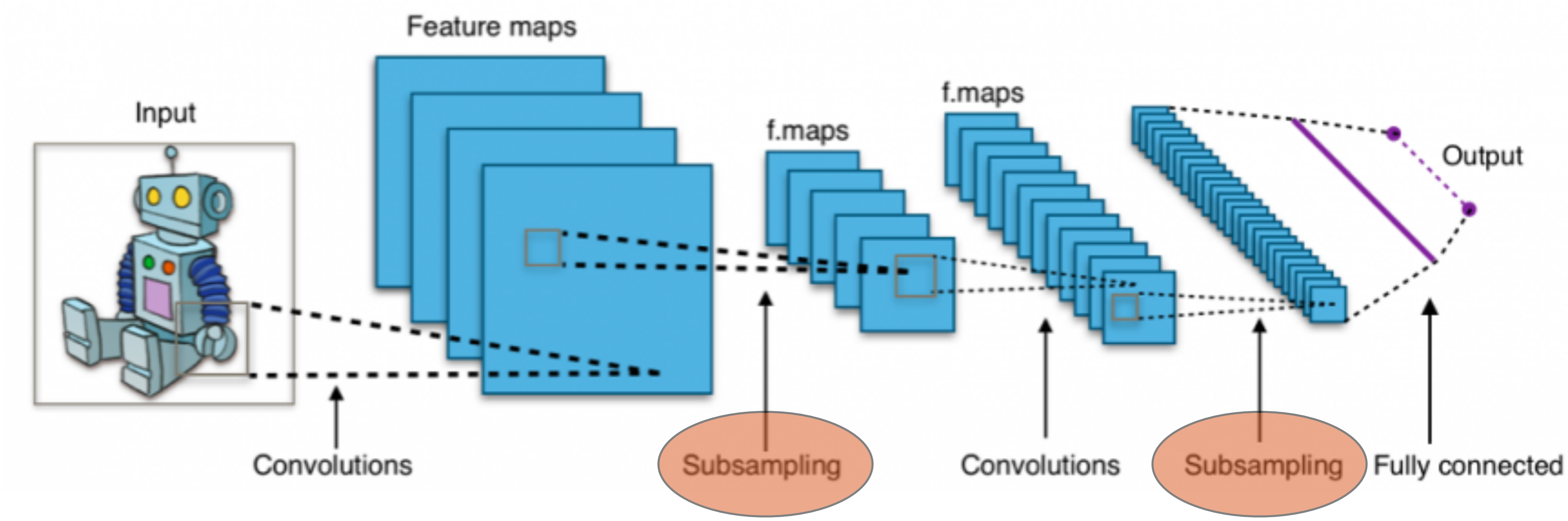
⬇ unitary edge weights

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

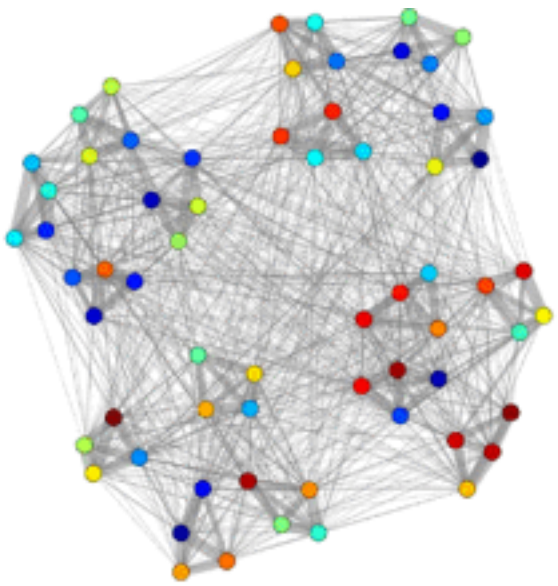




Pooling on graphs

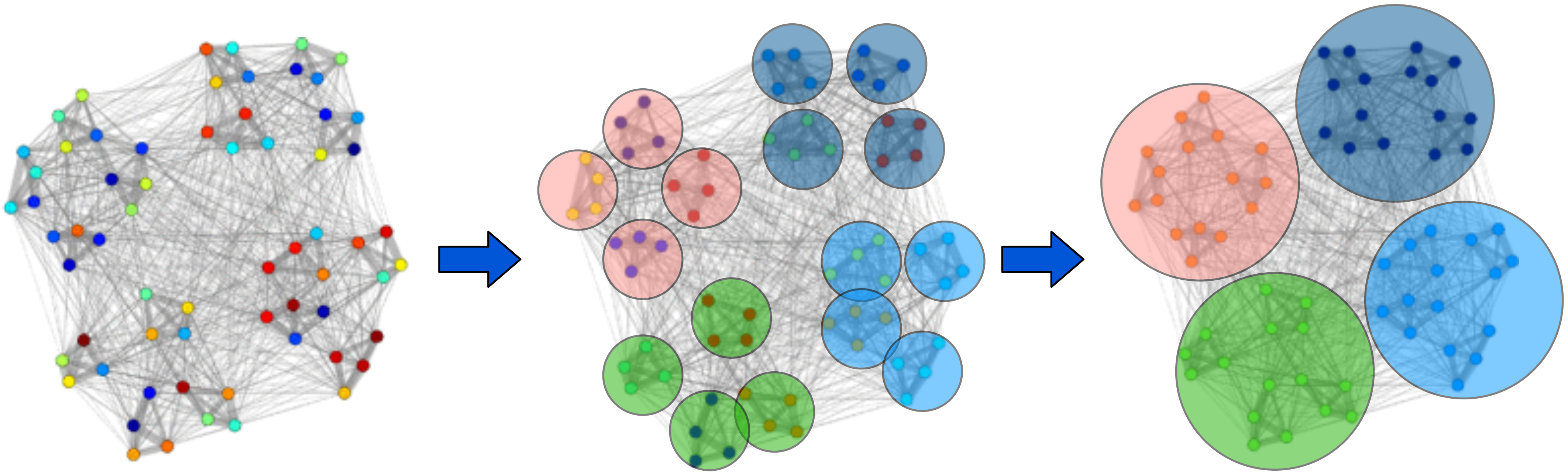


Pooling on graphs



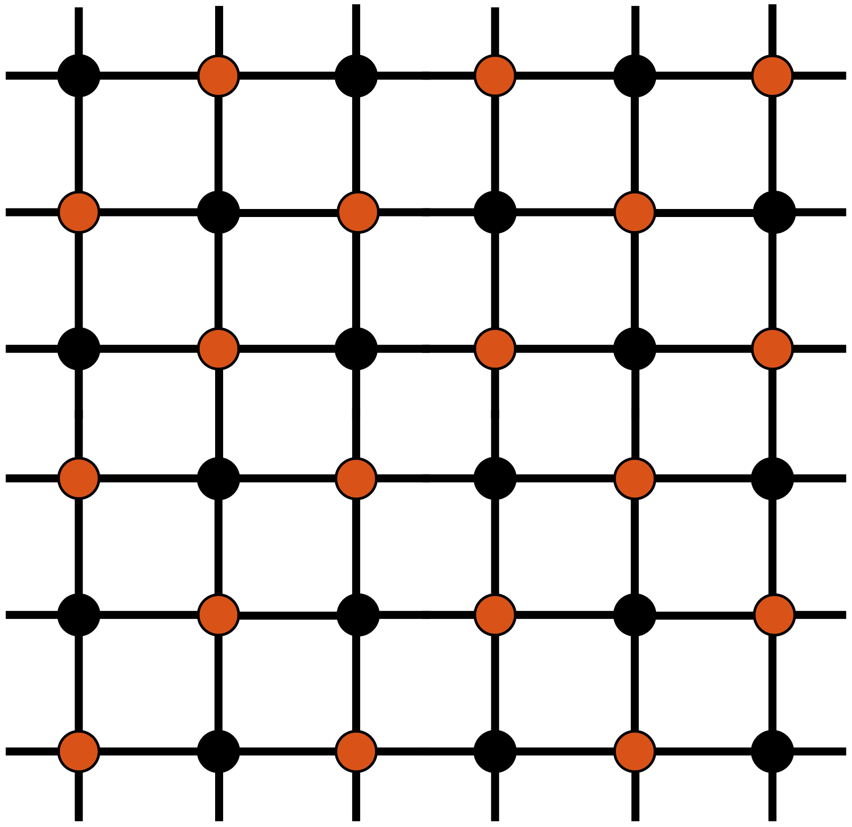
- pooling = downsampling on graphs, but how?

Pooling on graphs



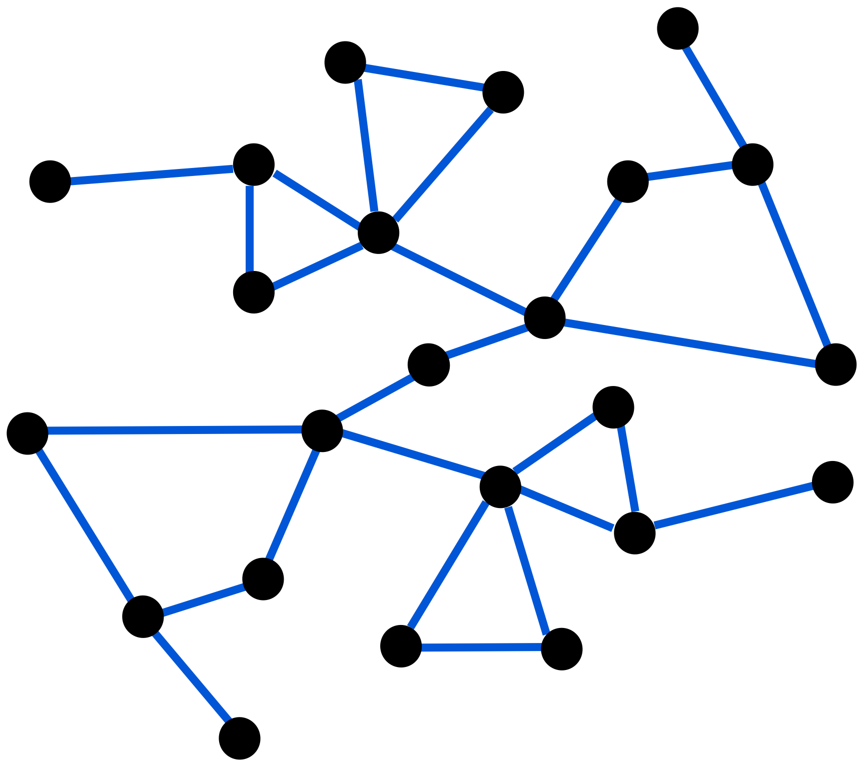
- pooling = downsampling on graphs, but how?
- natural idea: graph coarsening

Pooling on graphs



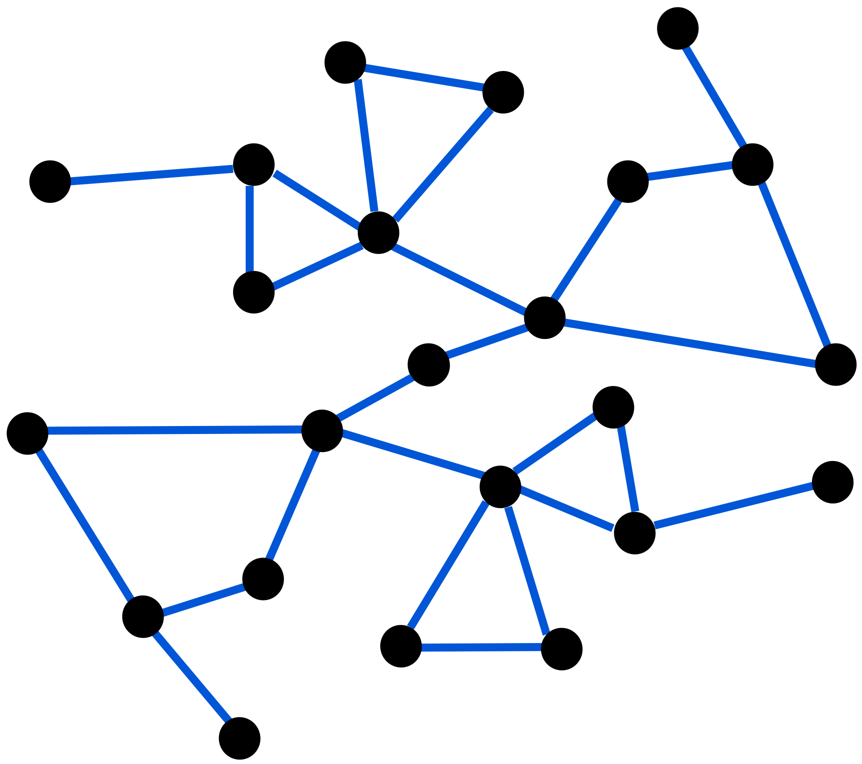
- coarsening is straightforward on regular grids

Pooling on graphs



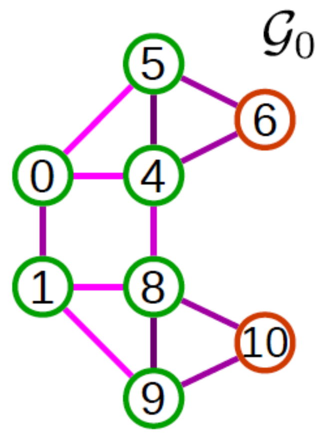
- coarsening is straightforward on regular grids
- not so much on irregular graphs

Pooling on graphs



- coarsening is straightforward on regular grids
- not so much on irregular graphs
- can be achieved via node clustering
  - multi-level partitioning
  - roughly fixed downsampling factor (e.g., 2)
  - need for efficiency

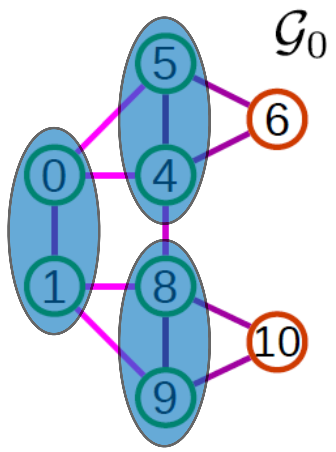
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

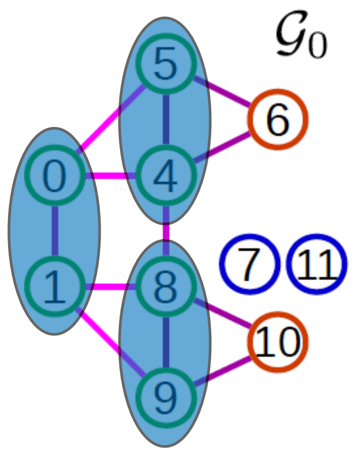
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

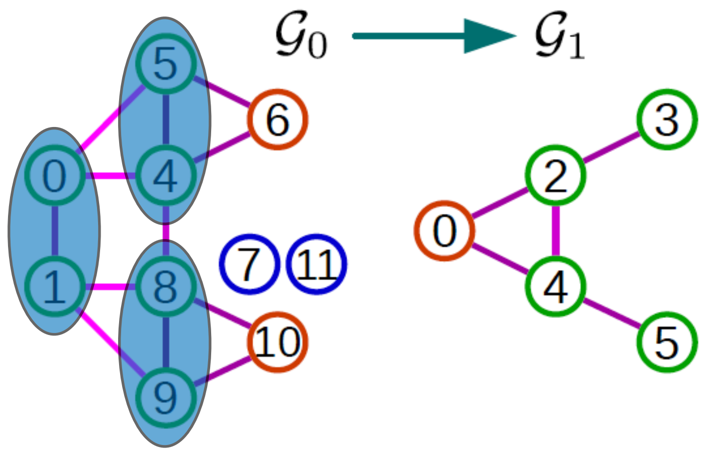
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

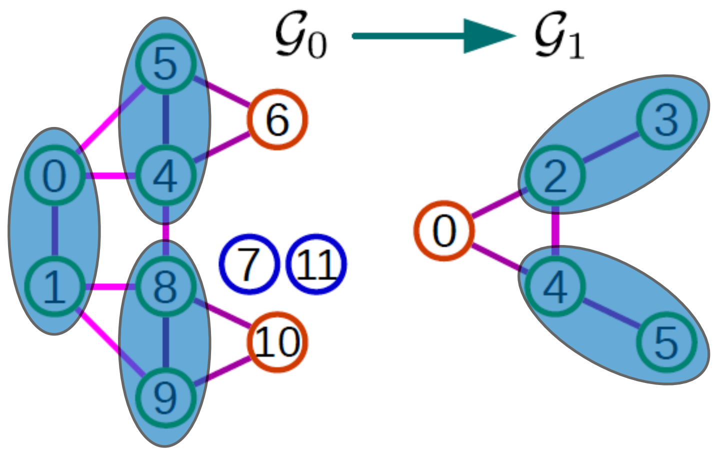
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

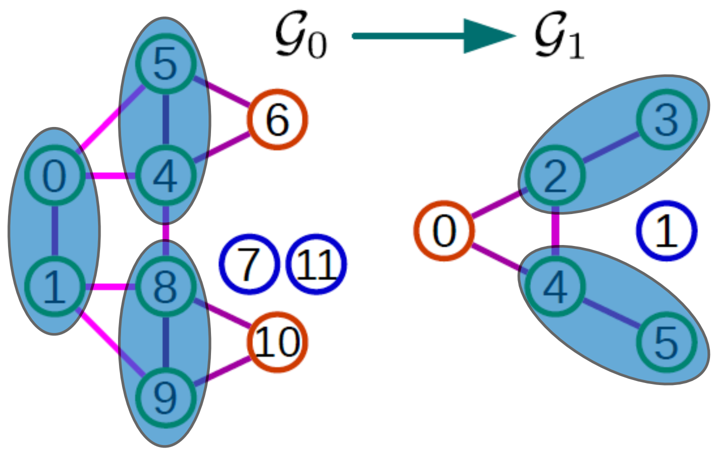
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

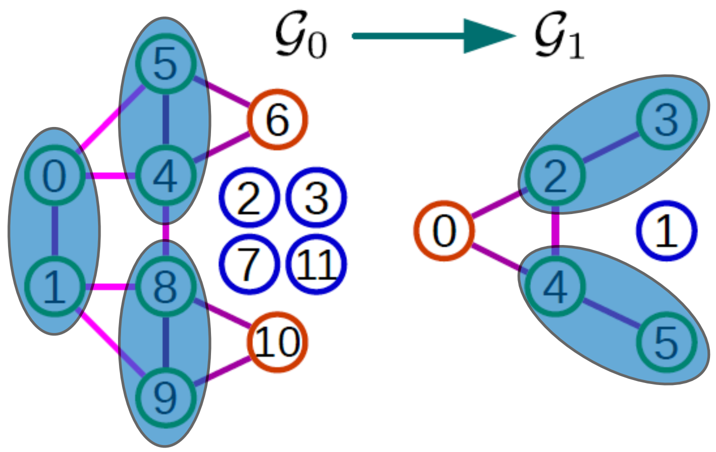
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

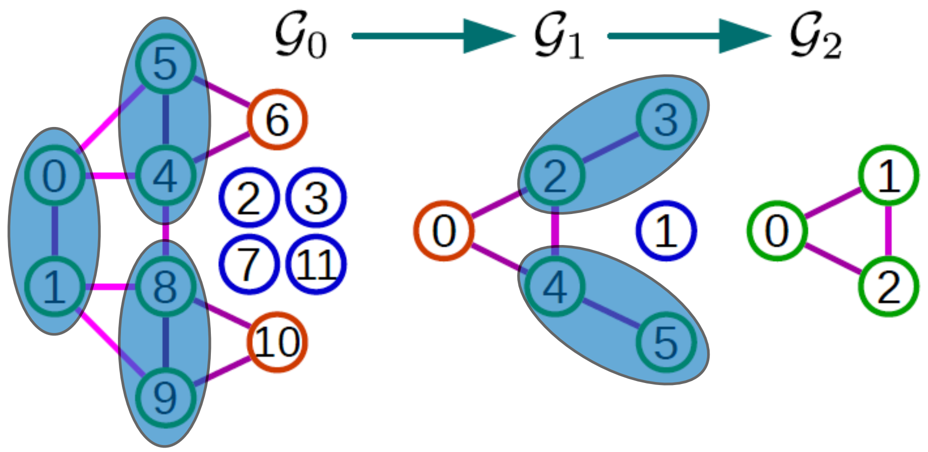
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

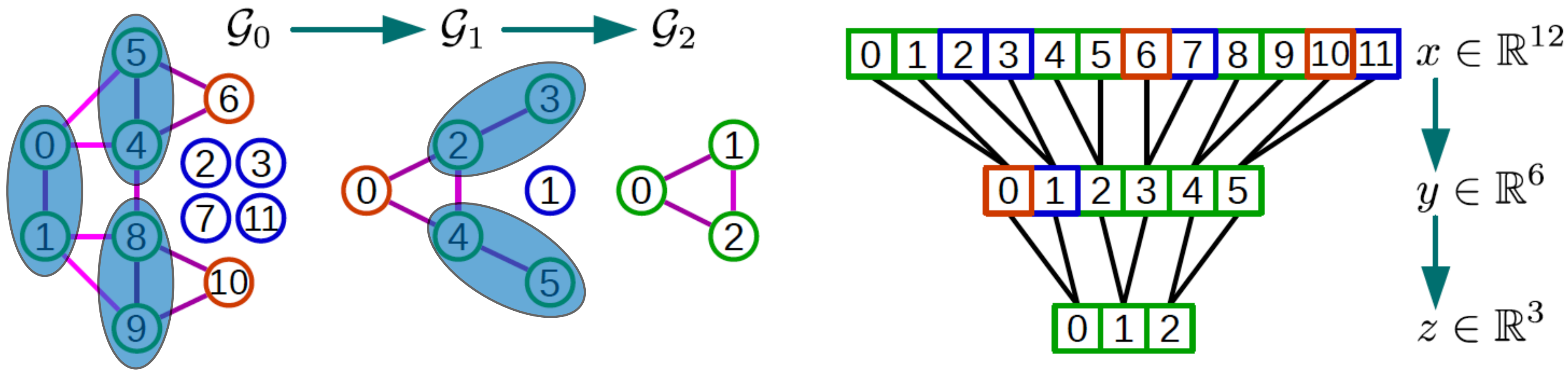
Pooling on graphs



- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

Pooling on graphs

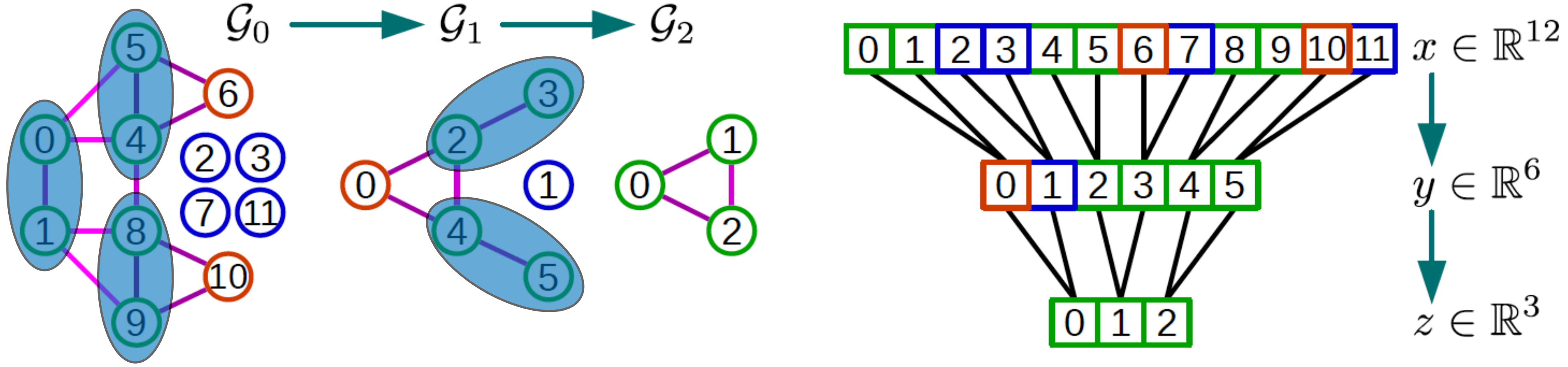


Defferrard et al. 2016

- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

Pooling on graphs

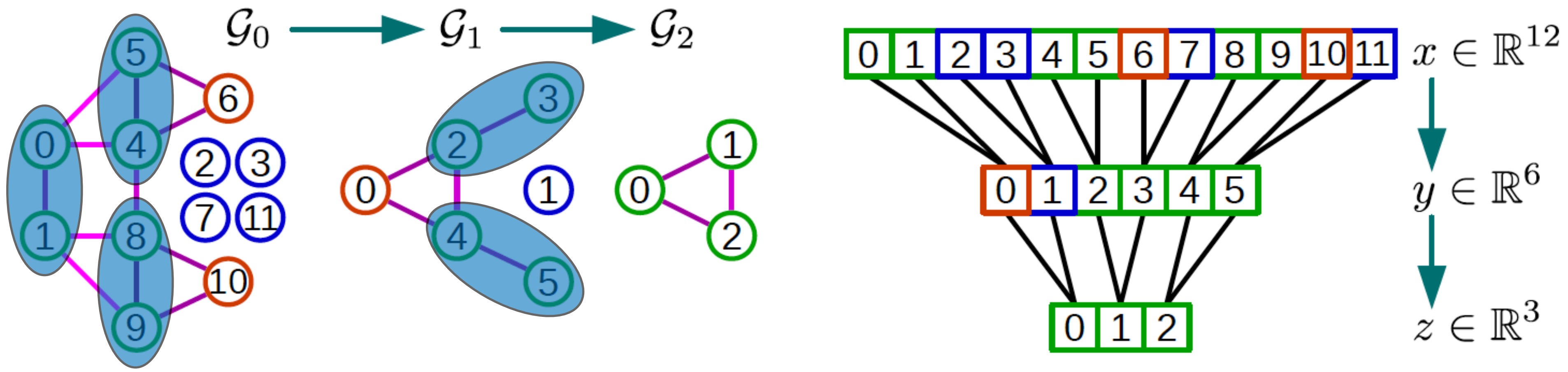


Defferrard et al. 2016

- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node
  - 1D grid pooling: [max(0,1) max(4,5,6) max(8,9,10)]

Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

Pooling on graphs

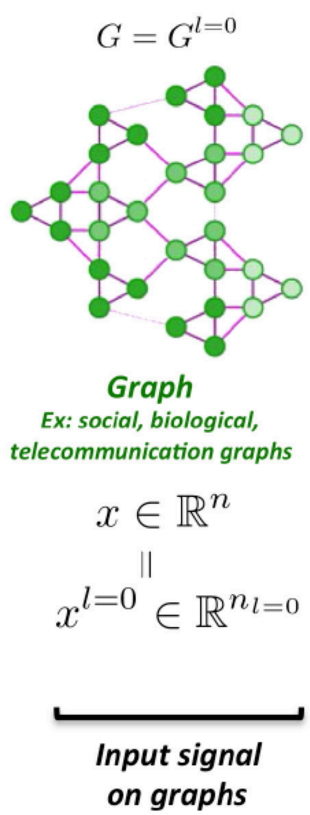


Defferrard et al. 2016

- pooling based on Graclus algorithm (Dhillon et al. 2007)
  - local greedy way of merging vertices: maximising $w_{ij}(1/d_i + 1/d_j)$
  - adding artificial vertices to ensure two children for each node
  - 1D grid pooling: [max(0,1) max(4,5,6) max(8,9,10)]
  - only based on graph (and no signal) information

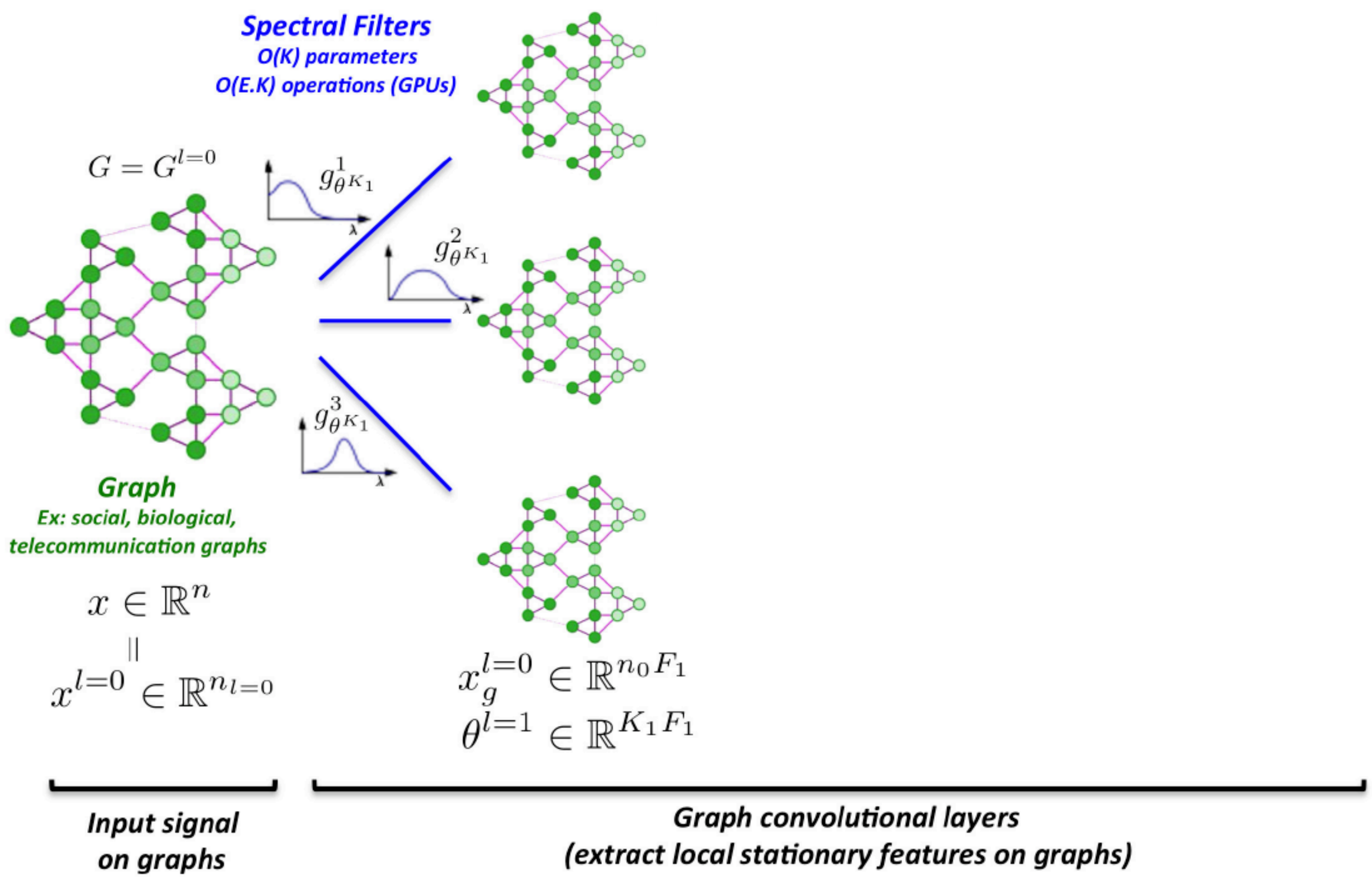
Dhillon et al., "Weighted graph cuts without eigenvectors: A multilevel approach," IEEE TPAMI, 2007.

CNN on graphs: Graph classification

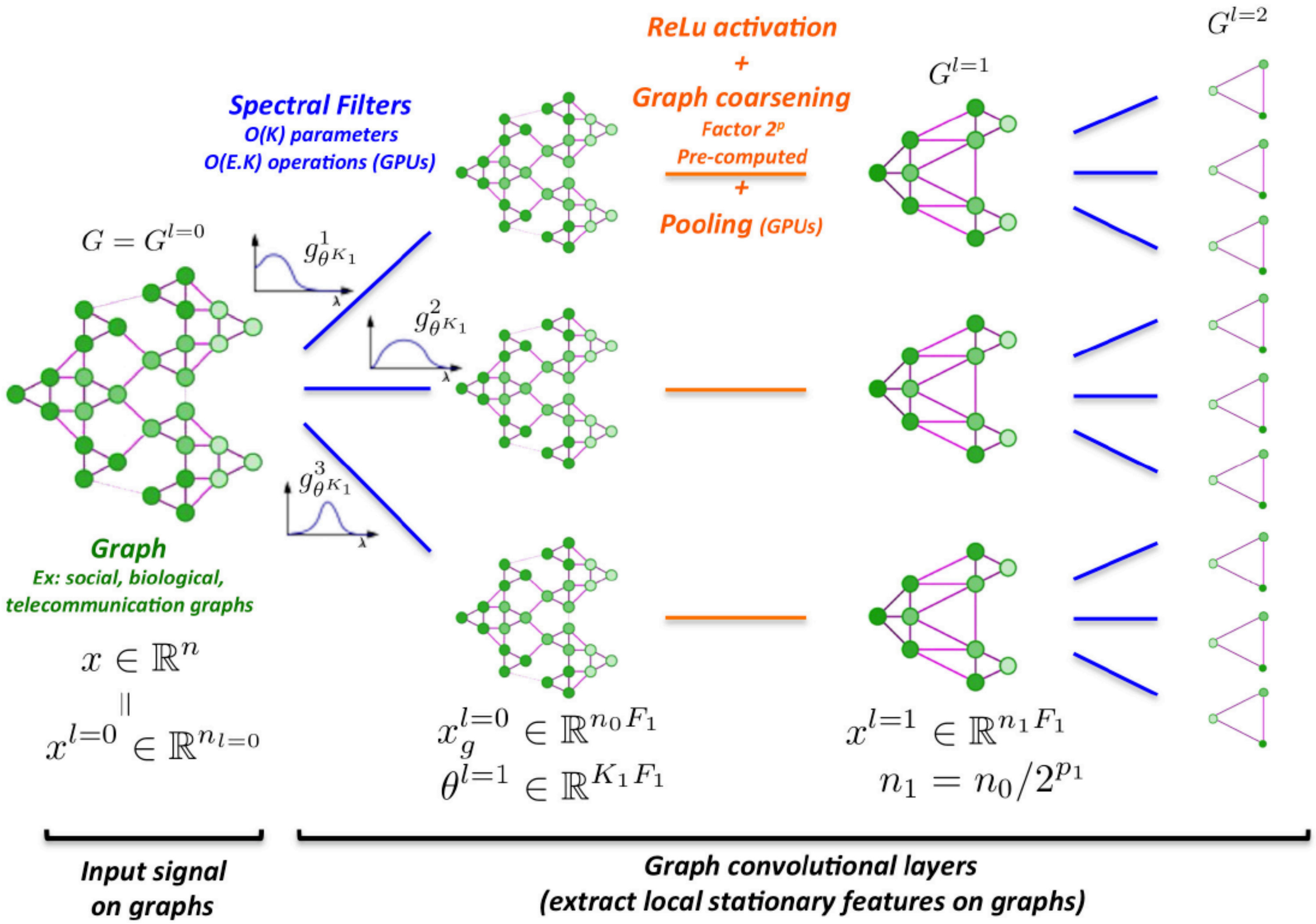


Defferrard et al., "Convolutional neural networks on graphs with fast localized spectral filtering," NIPS, 2016.

CNN on graphs: Graph classification

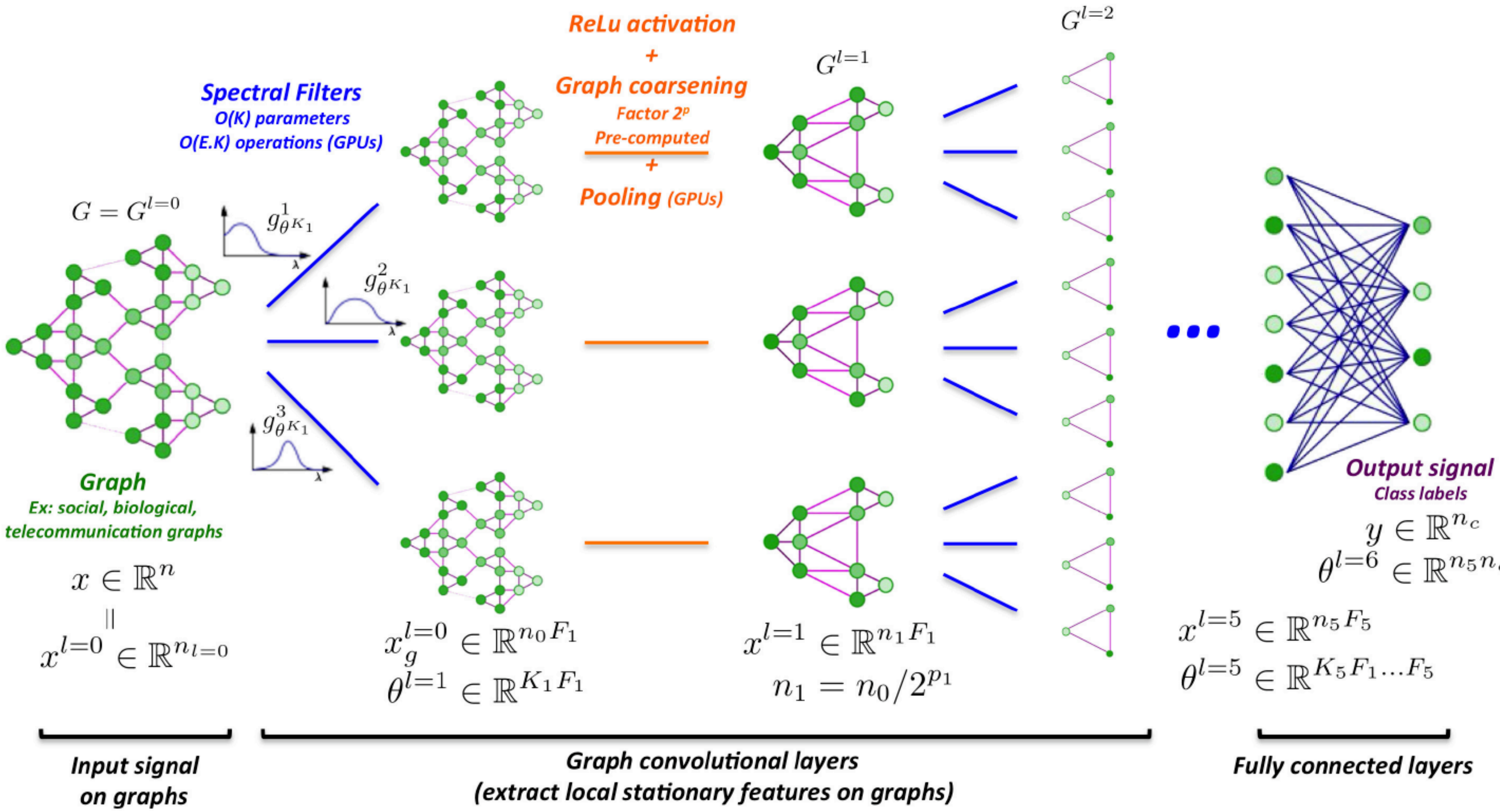


CNN on graphs: Graph classification



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

CNN on graphs: Graph classification



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

CNN on graphs: Node classification

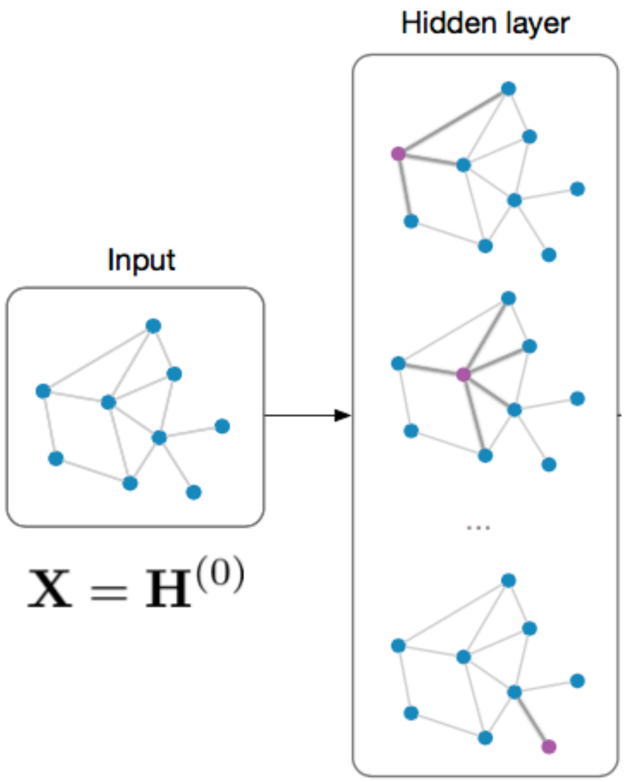
$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$

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

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

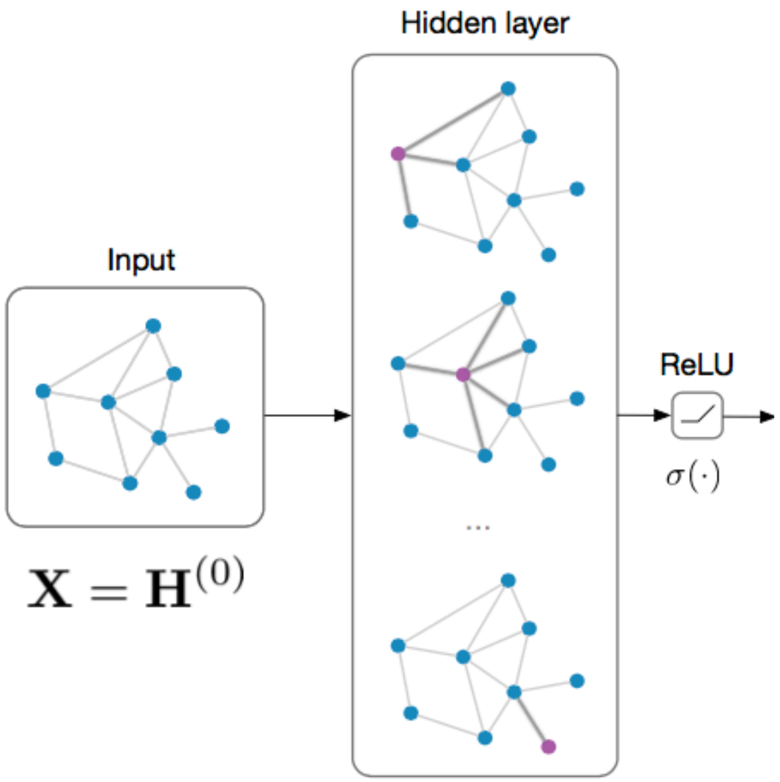
CNN on graphs: Node classification

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



CNN on graphs: Node classification

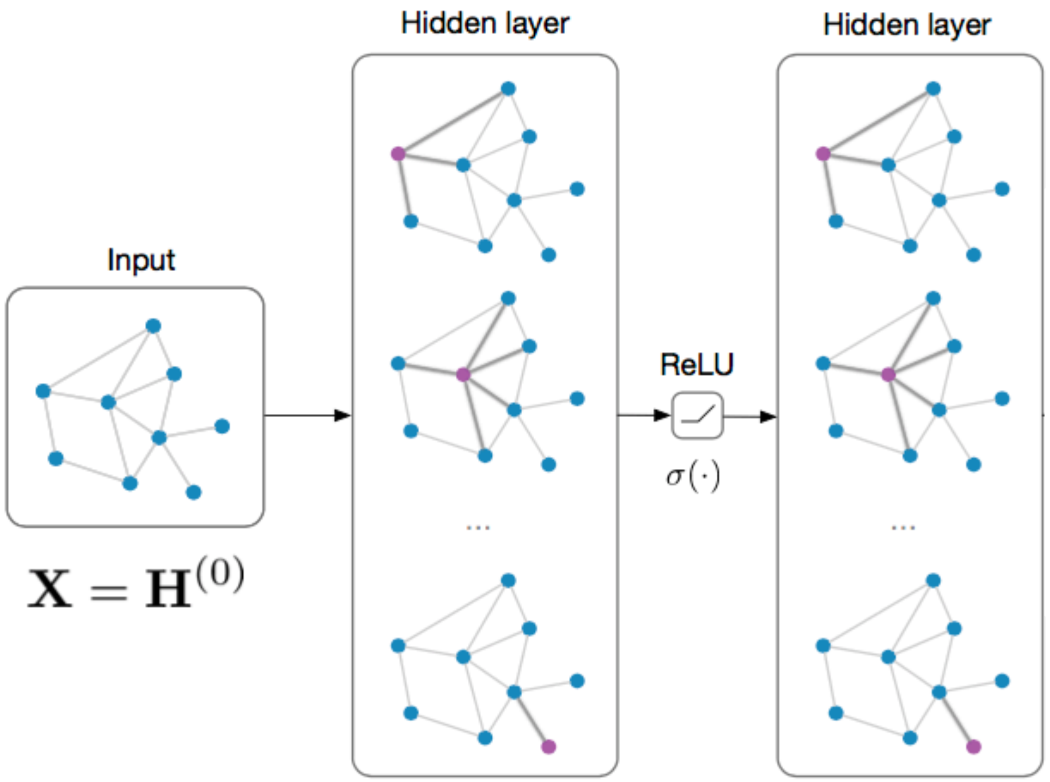
$$\hat{g}_{\theta^{(k+1)}}(L)\left(\text{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\right)$$



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

CNN on graphs: Node classification

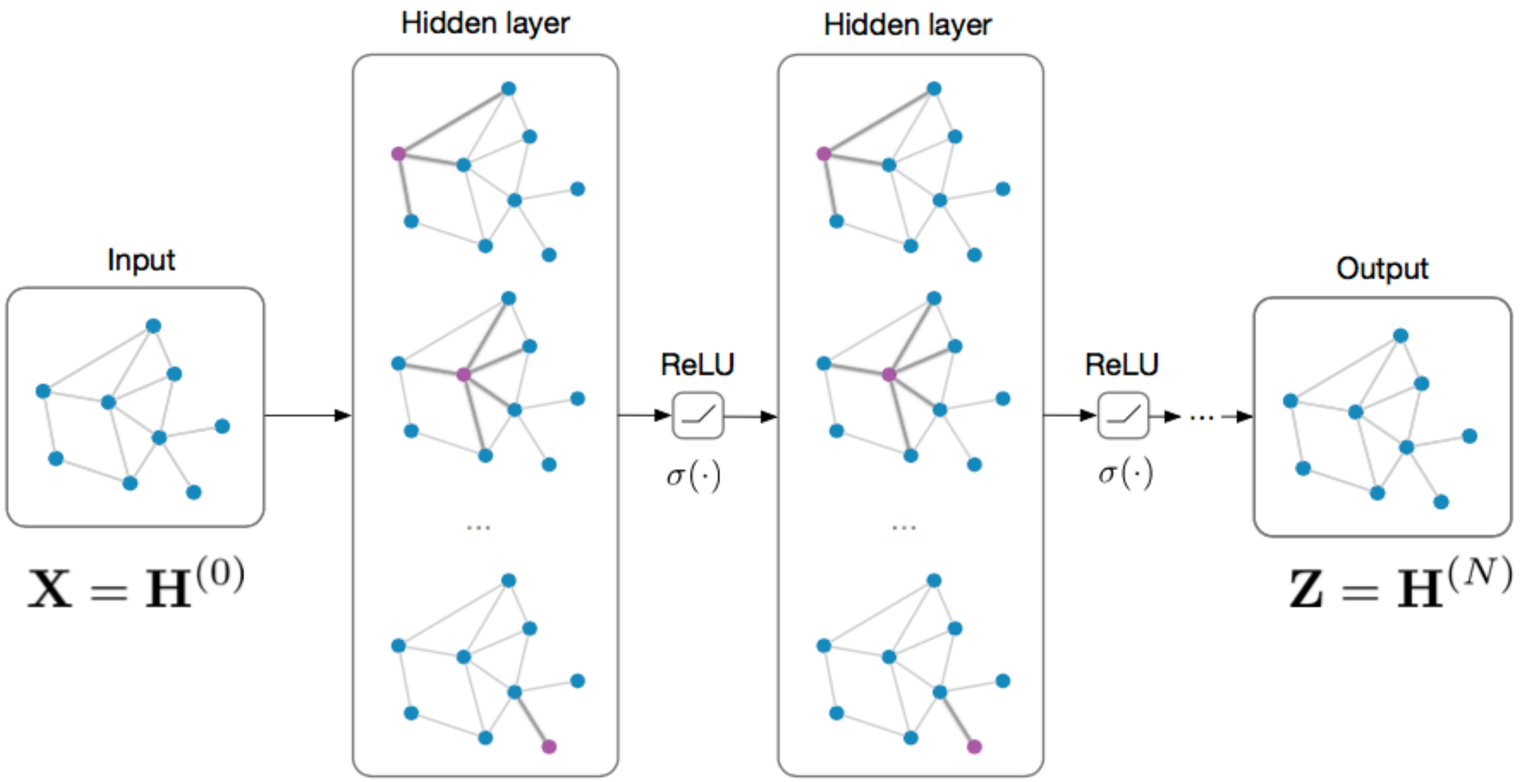
$$\hat{g}_{\theta^{(k+1)}}(L)\Big(\mathrm{ReLU}(\hat{g}_{\theta^{(k)}}(L)f)\Big)$$



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

CNN on graphs: Node classification

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

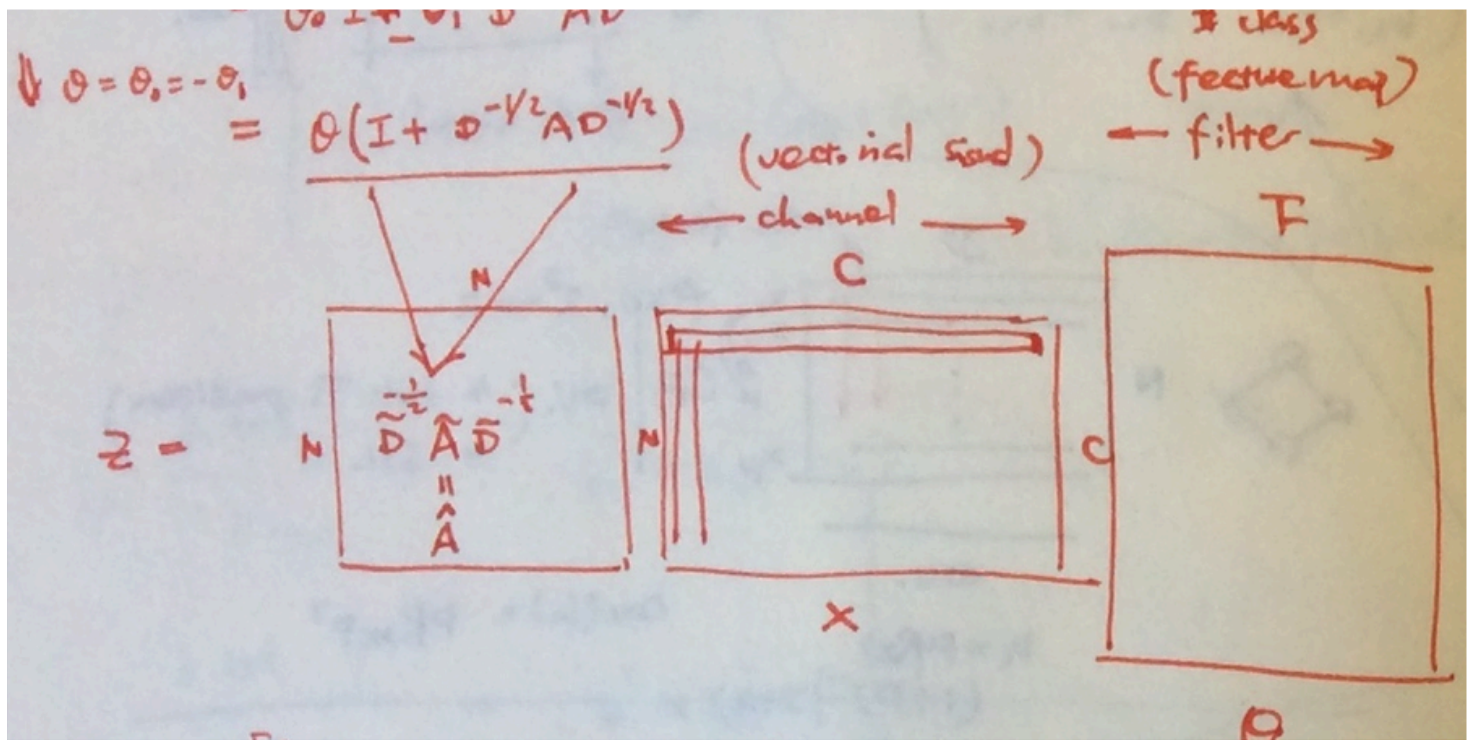


$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

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

CNN on graphs: Node classification

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



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

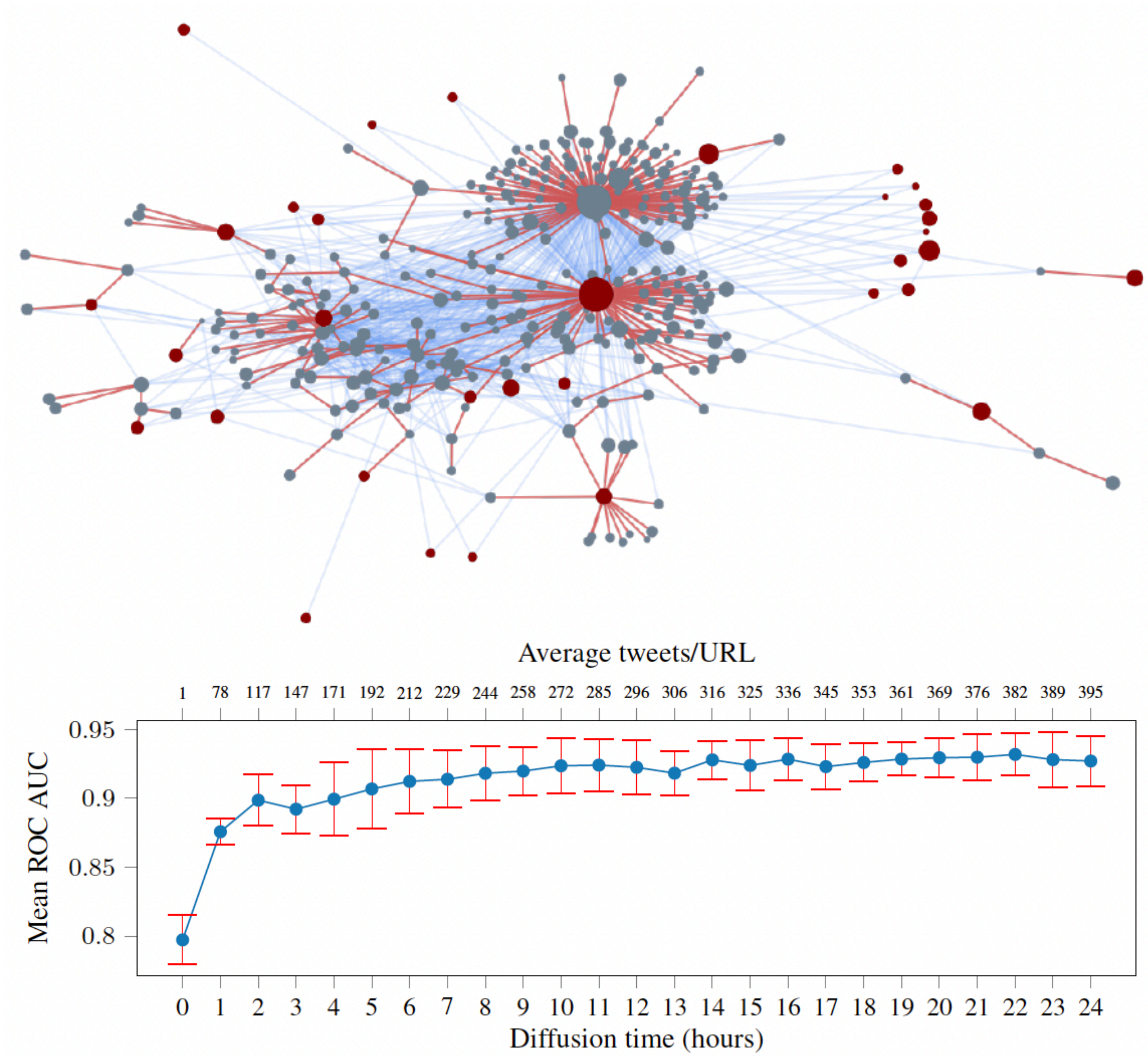
Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

Application I: Document classification

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

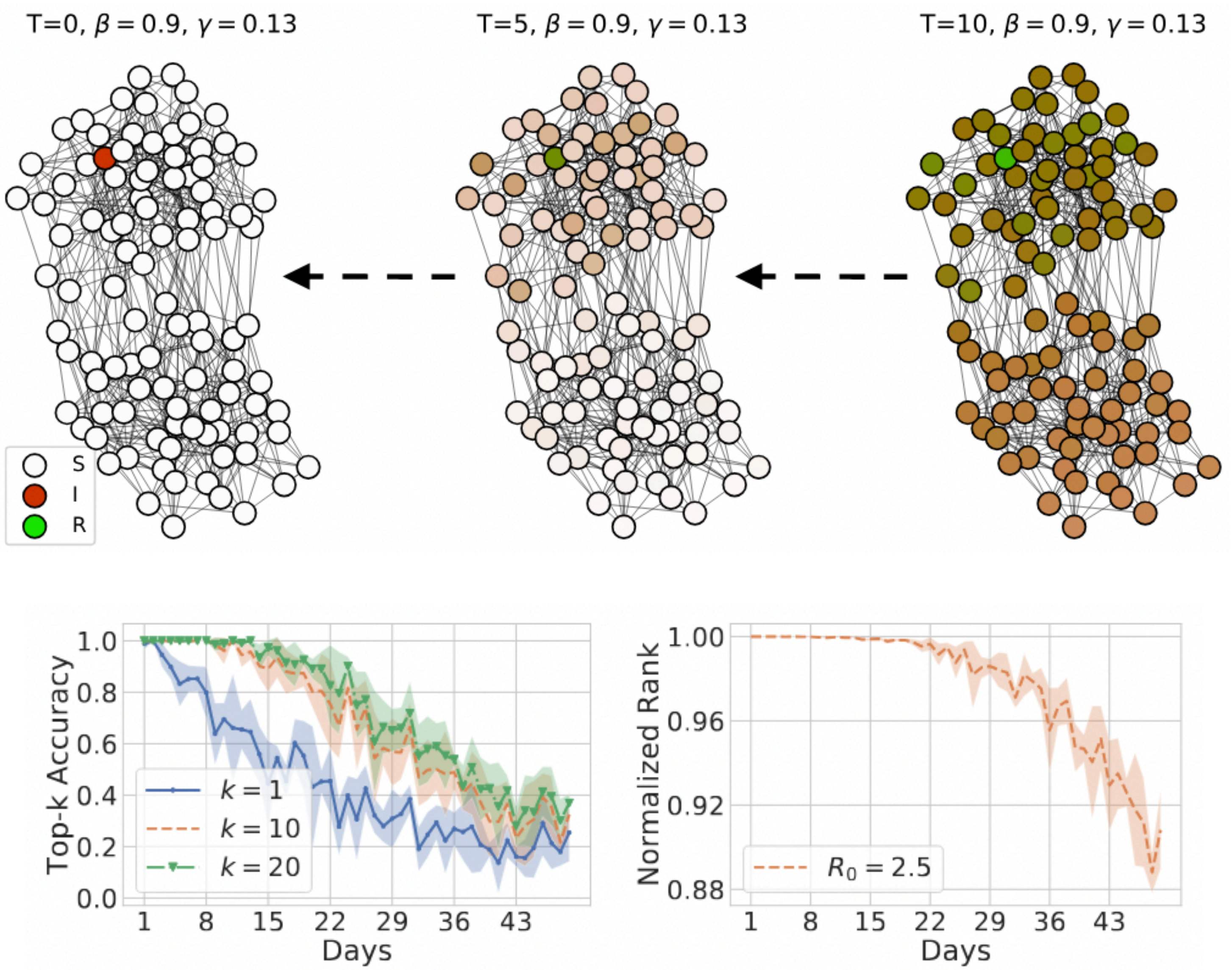
Kipf and Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017.

Application II: Fake news detection



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

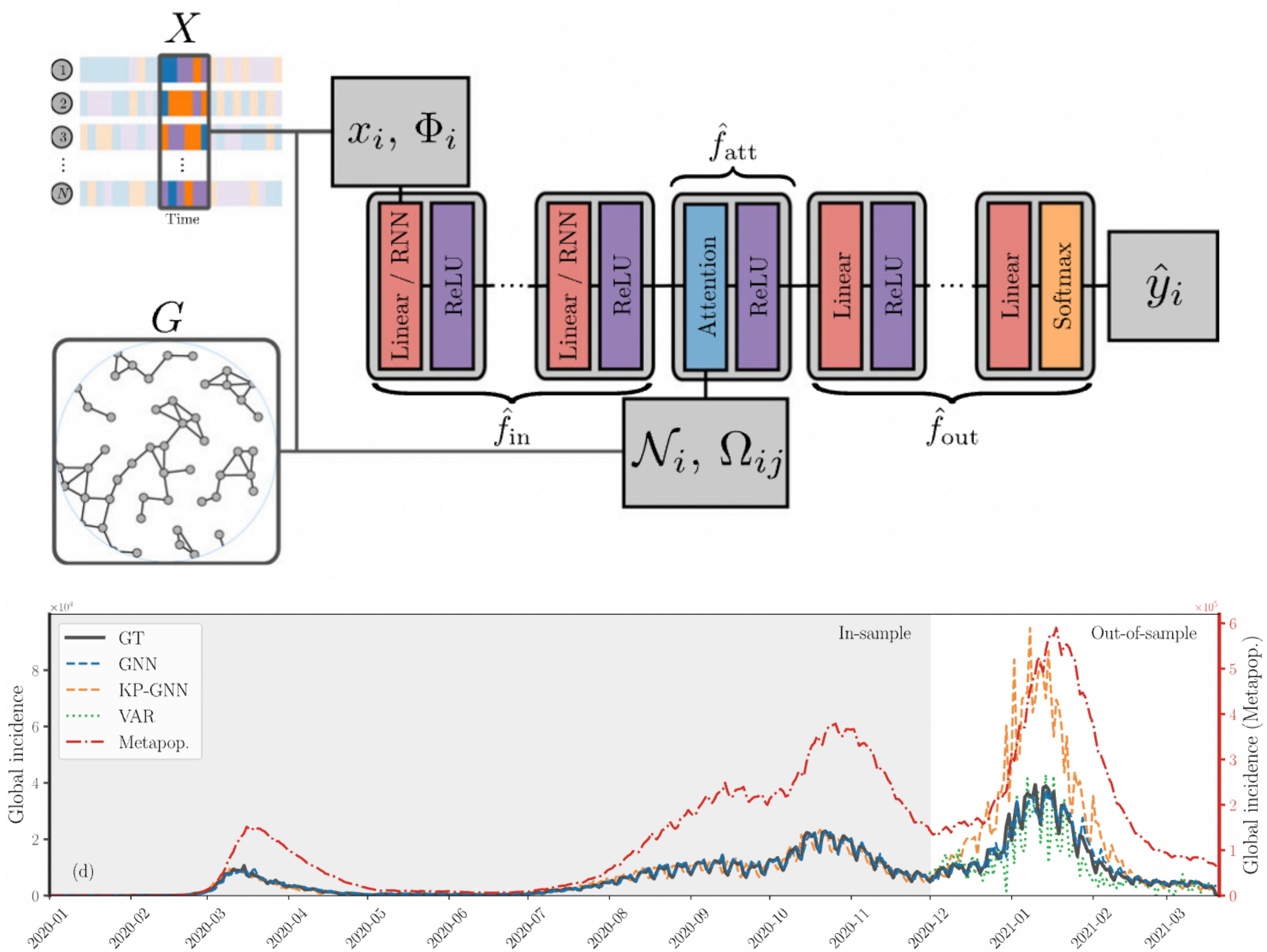
Application III: Finding patient zero



(a) Top-k accuracy (b) Normalized rank

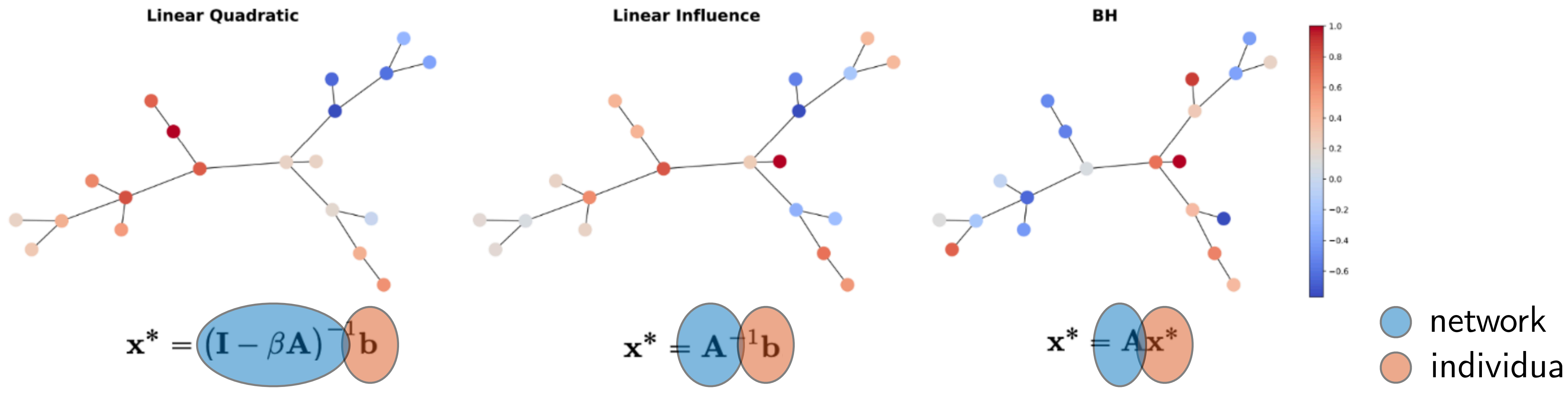
Shah et al., "Finding patient zero: Learning contagion source with graph neural networks," arXiv, 2020.

Application IV: Learning contagion dynamics



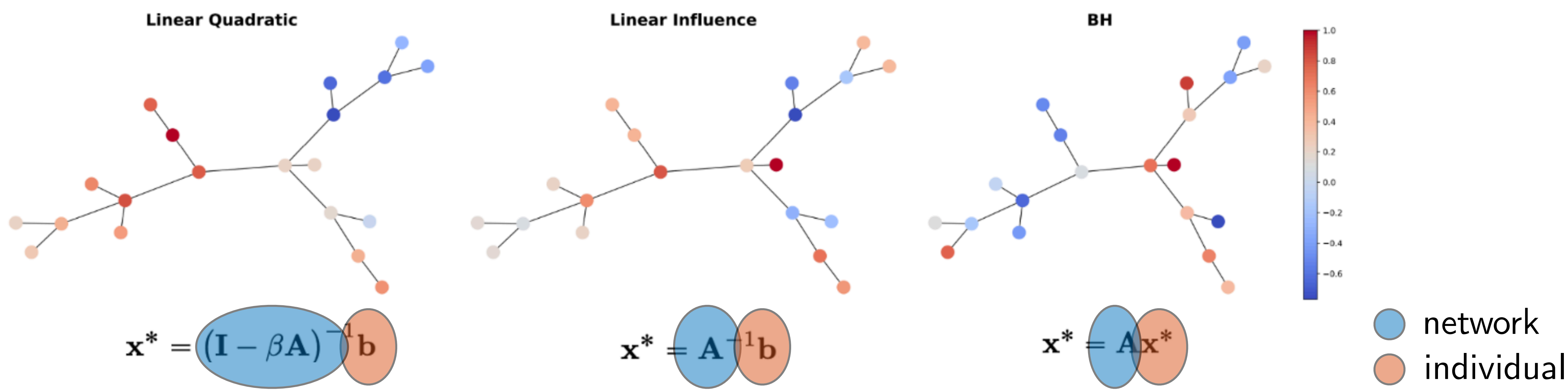
Murphy et al., "Deep learning of contagion dynamics oncomplex networks," Nature Communications, 2021.

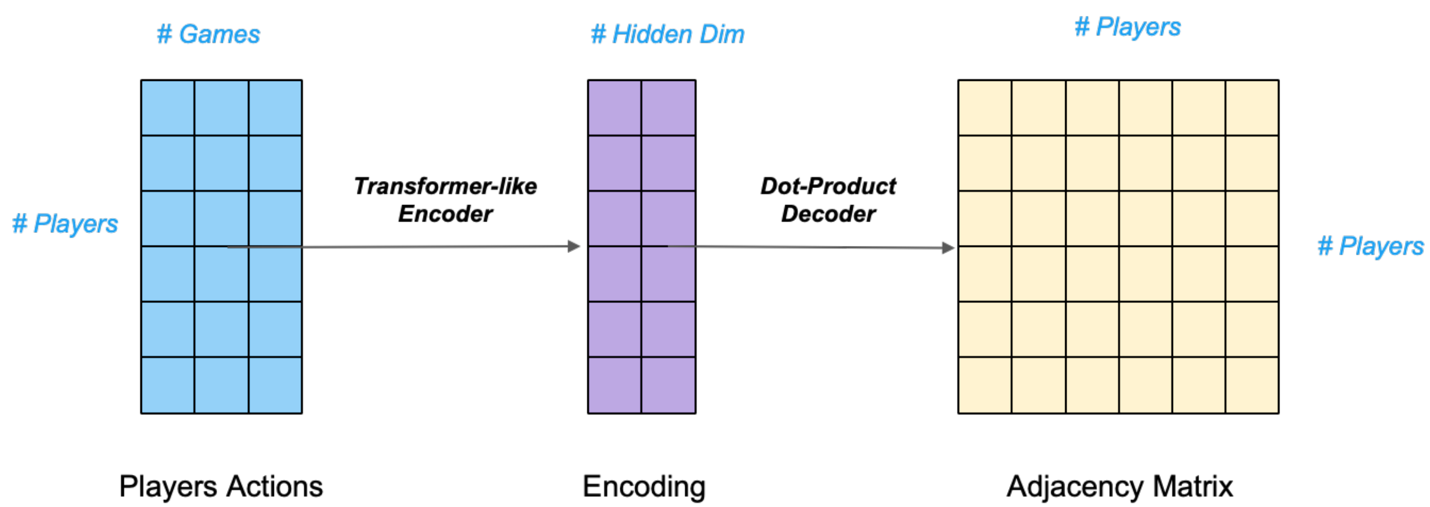
Application V: Learning social interactions



Rossi et al., "Learning to infer structures of network games," ICML, 2022.

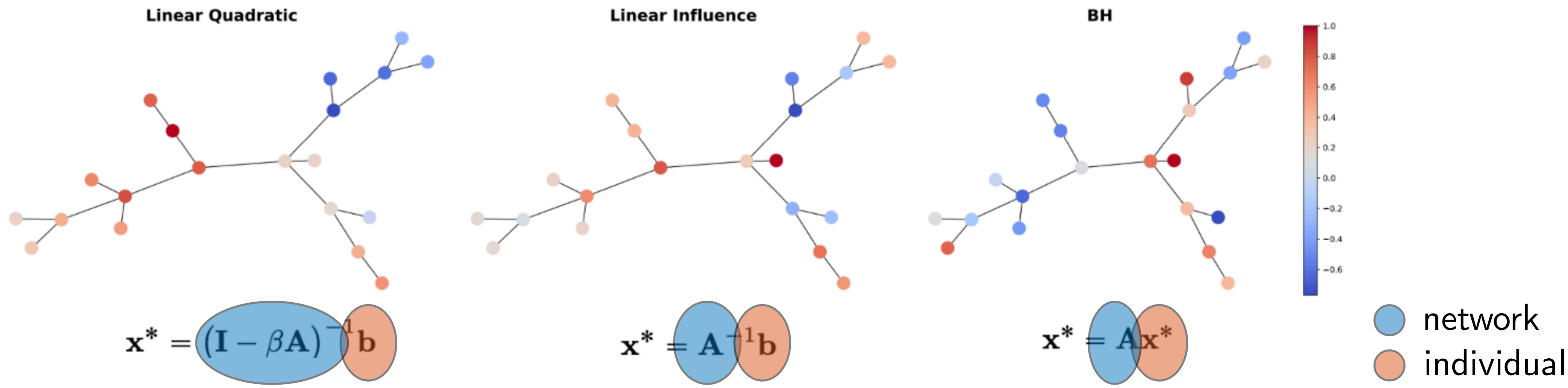
Application V: Learning social interactions

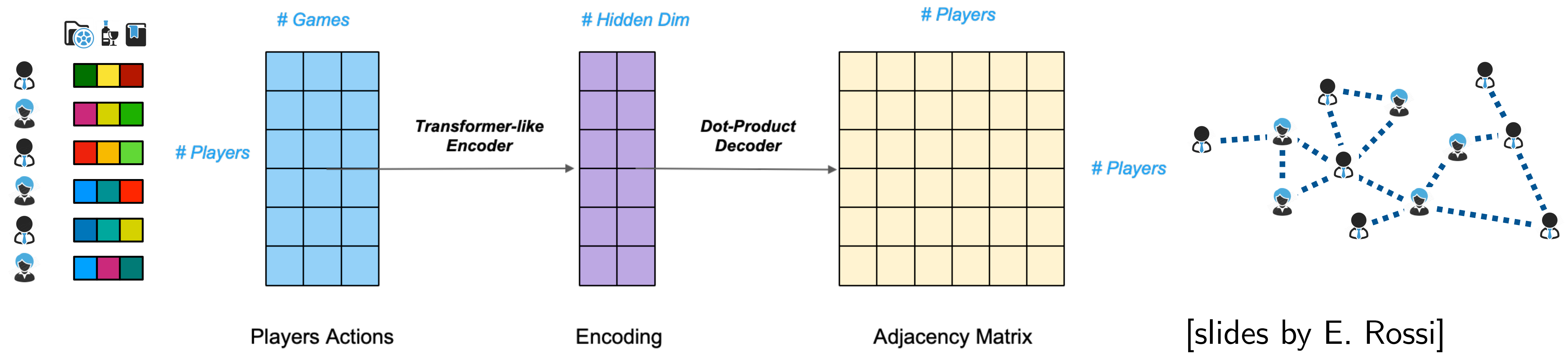




[slides by E. Rossi]

Application V: Learning social interactions

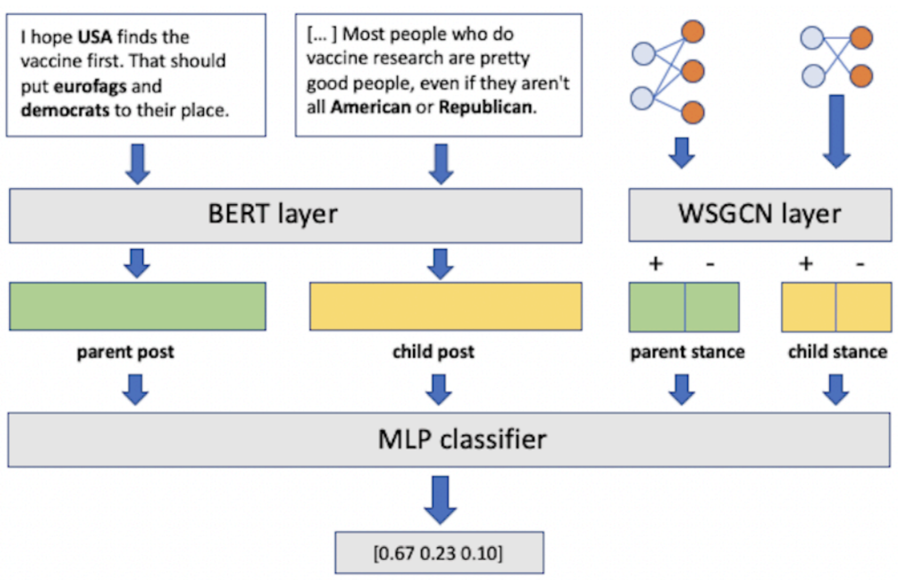




[slides by E. Rossi]

Rossi et al., "Learning to infer structures of network games," ICML, 2022.

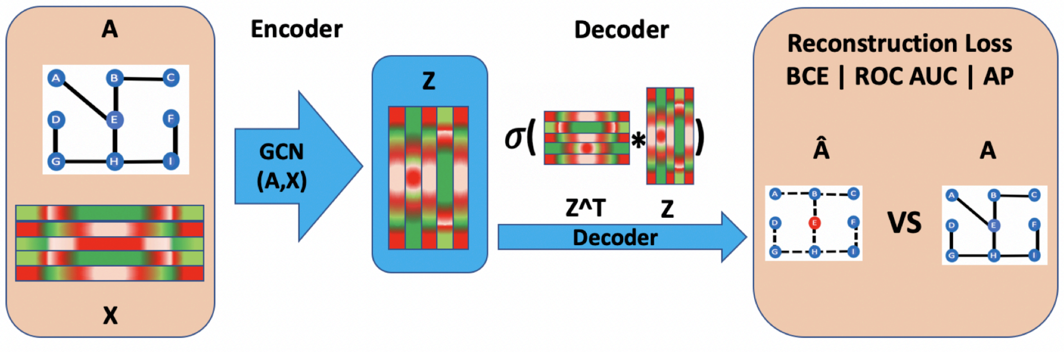
Application VI: Language and social media analysis



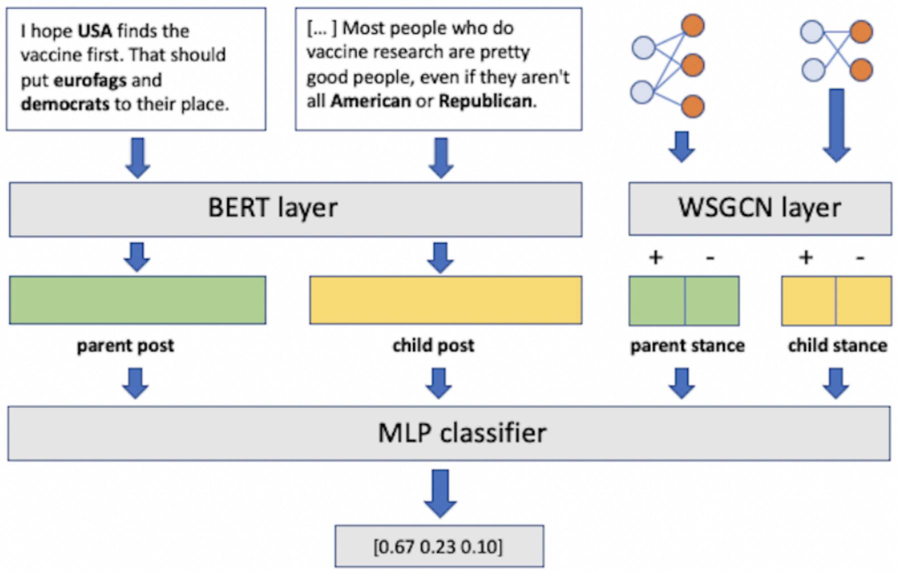
disagreement prediction

Hofmann et al., "Modeling ideological salience and framing in polarized online groups with graph neural networks and structured sparsity," NAACL, 2022.
Zhang et al., "Predicting polarisation of dynamic social networks via graph auto-encoders," IC2S2, 2023.
Lorge et al., "STEntConv: Predicting disagreement between reddit users with stance detection and a signed graph convolutional network," LREC-Coling 2024.

Application VI: Language and social media analysis



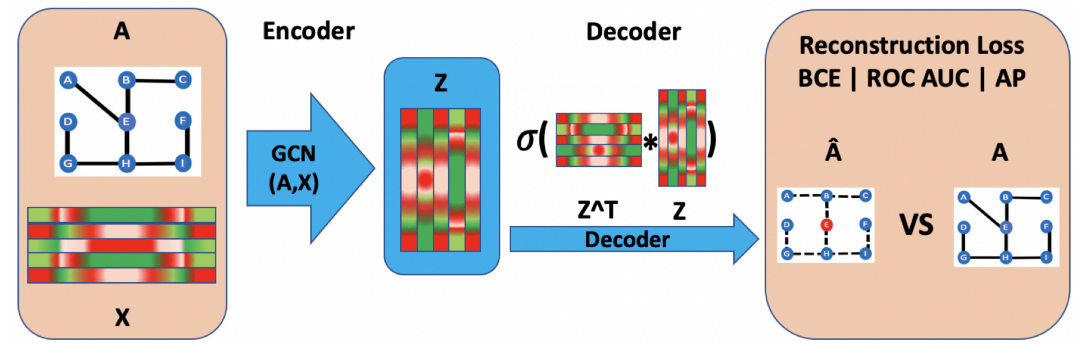
polarisation prediction



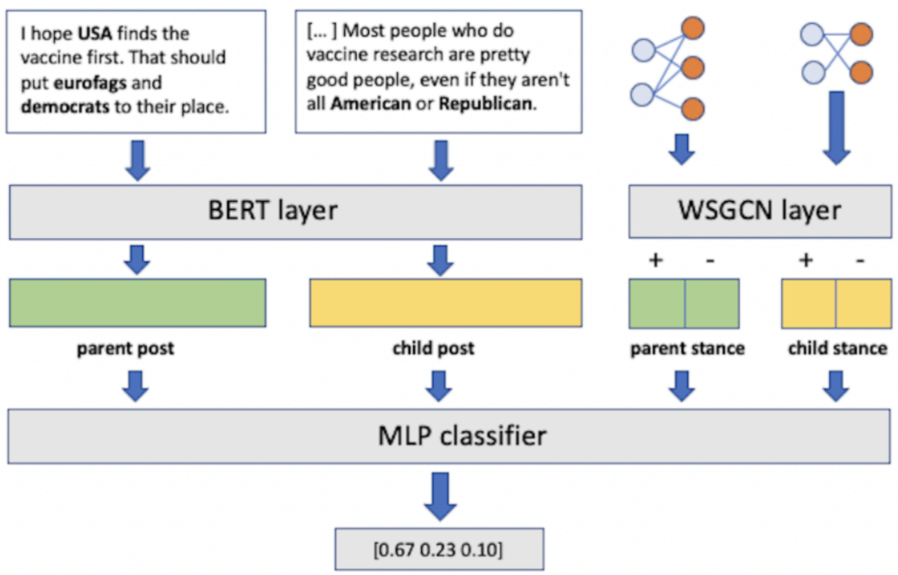
disagreement prediction

Hofmann et al., "Modeling ideological salience and framing in polarized online groups with graph neural networks and structured sparsity," NAACL, 2022.
Zhang et al., "Predicting polarisation of dynamic social networks via graph auto-encoders," IC2S2, 2023.
Lorge et al., "STEntConv: Predicting disagreement between reddit users with stance detection and a signed graph convolutional network," LREC-Coling 2024.

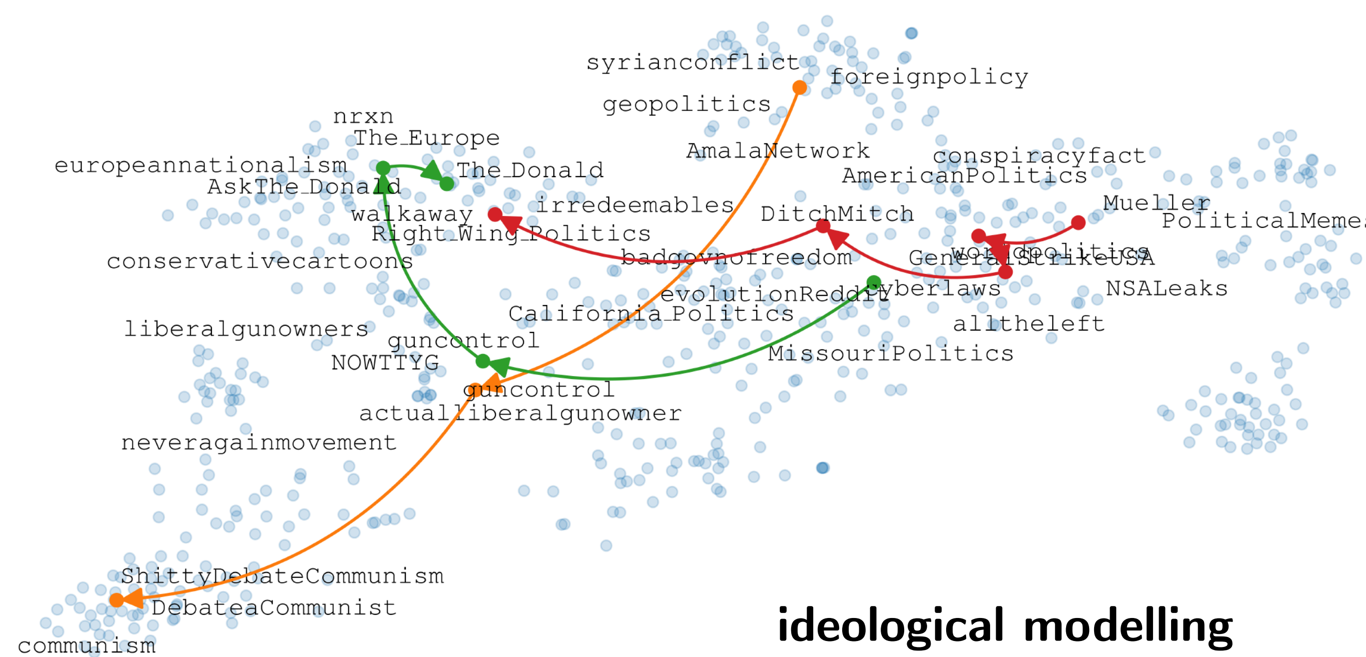
Application VI: Language and social media analysis



polarisation prediction



disagreement prediction



ideological modelling

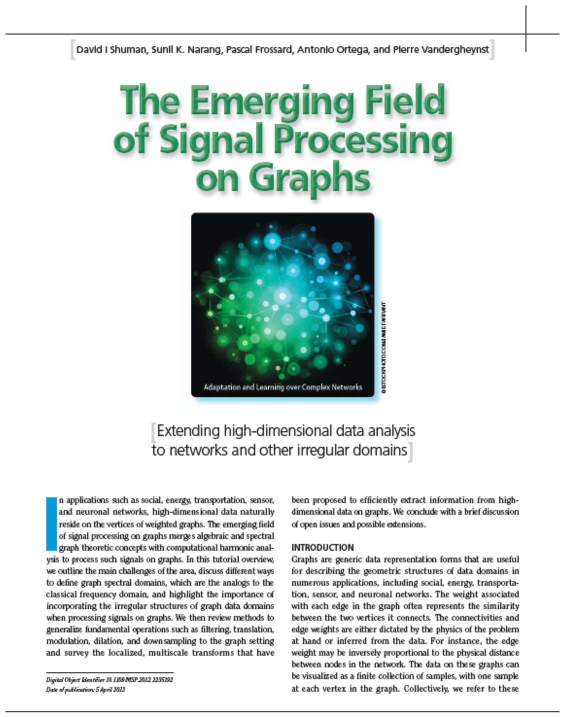
Hofmann et al., "Modeling ideological salience and framing in polarized online groups with graph neural networks and structured sparsity," NAACL, 2022.
Zhang et al., "Predicting polarisation of dynamic social networks via graph auto-encoders," IC2S2, 2023.
Lorge et al., "STEntConv: Predicting disagreement between reddit users with stance detection and a signed graph convolutional network," LREC-Coling 2024.

Some Final Thoughts

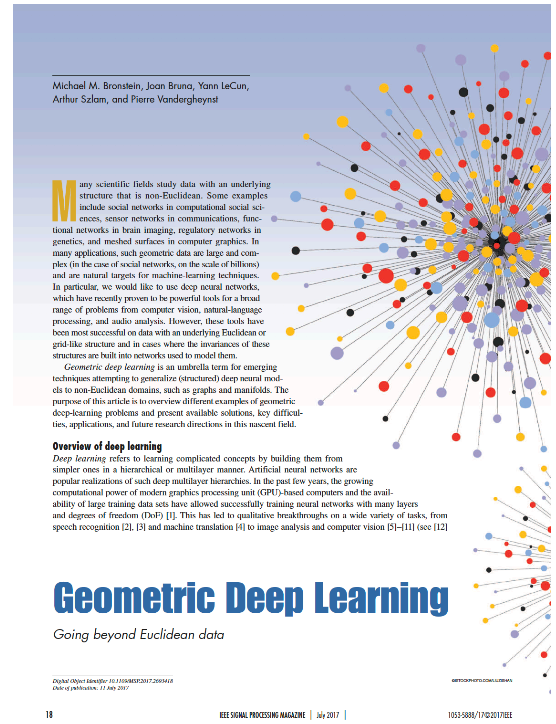
Summary

- Graph machine learning
  - fast-growing field that extends data analysis to non-Euclidean domain
  - highly interdisciplinary: machine learning, signal processing, harmonic analysis, applies statistics, differential geometry
- Limitations and open challenges
  - models on directed and signed graphs
  - models for temporal dynamics and online/adaptive settings
  - construction/refinement of initial graphs
  - robustness & generalisation & scalability
  - interpretability & causal inference
  - expect more applications in social sciences & economics!

References



David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst
The Emerging Field of Signal Processing on Graphs
Extending high-dimensional data analysis to networks and other irregular domains



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

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

Journal of Machine Learning Research 23 (2022) 1-64
Submitted 8/20; Revised 10/21; Published 05/22
Machine Learning on Graphs: A Model and Comprehensive Taxonomy
Ines Chami
Sami Abu-El-Haija
Bryan Perozzi
Christopher Ré
Kevin Murphy
Abstract

ANNUAL REVIEWS
Annual Review of Sociology
Machine Learning for Sociology
Mario Molina and Filiz Garip
Keywords
Abstract
ANNUAL REVIEWS CONNECT

ANNUAL REVIEWS
Annual Review of Political Science
Machine Learning for Social Science: An Agnostic Approach
Justin Grimmer, Margaret E. Roberts, and Brandon M. Stewart
ANNUAL REVIEWS CONNECT
Keywords
machine learning, text as data, research design
Abstract