Çizge verilerinde frekans analizi ve spektral gösterimlere dayalı yapay öğrenme

Frequency analysis and spectral machine learning methods on graphs

Elif Vural

Orta Doğu Teknik Üniversitesi
Elektrik-Elektronik Mühendisliği Bölümü

Bilim Akademisi Yapay Öğrenme Yaz Okulu, 2021

Graph Models in Machine Learning

- In many modern applications, data is acquired on an irregular network topology.

aelitta/iStock/Getty Images Plus

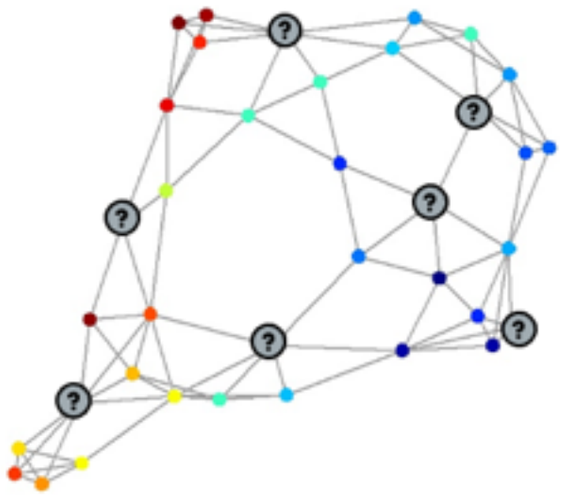
Social networks

https://www.semanticscholar.org/paper/Variational-Graph-Auto-Encoders-Kipf-Welling/

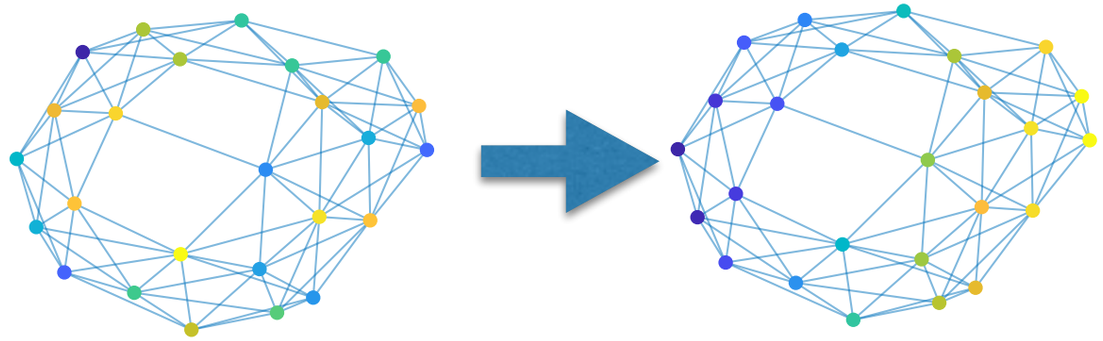
Sensor networks

- Graphs models offer the ability to analyze complex interactions over network structures.

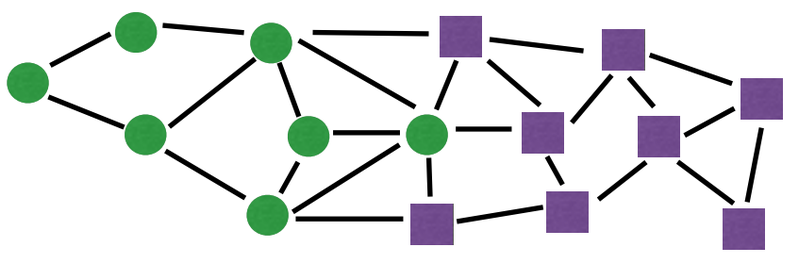
Inference Problems on Graphs



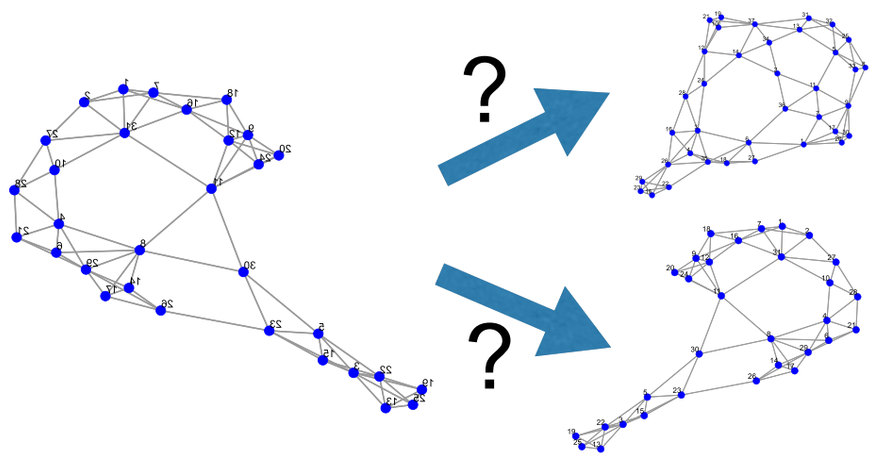
Estimation/Inpainting of graph signals



Denoising/Filtering



Node classification



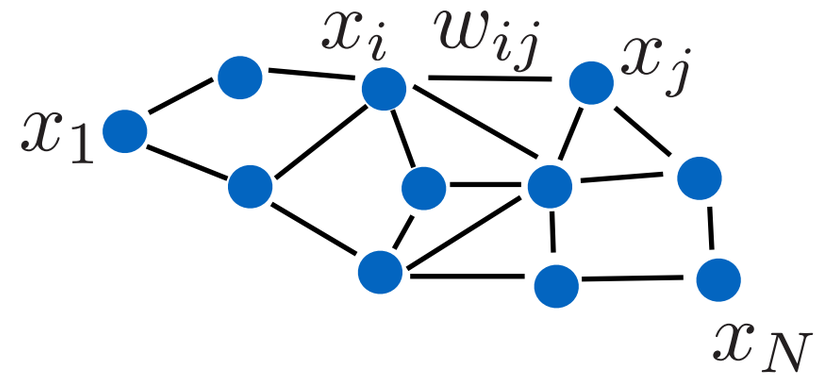
Graph classification

Outline

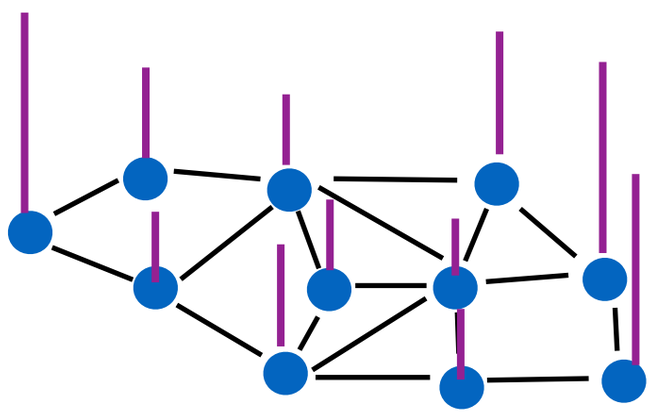
- Brief introduction to Spectral Graph Theory
- Extending common tools to graph domains:
  - Frequency analysis
  - Filtering
  - Frames, dictionaries, sparse representations
  - Neural networks
  - ….

Spectral Graph Theory: A Brief Overview

- $G = (\mathcal{V}, \mathcal{E}, W)$: Weighted graph



- $f : \mathcal{V} \to \mathbb{R}$: A graph signal, $f \in \mathbb{R}^N$



Graph Laplacian

Weight matrix

$$W = \begin{bmatrix} 0 & w_{12} & \dots & 0 & 0 \\ w_{21} & 0 & \dots & 0 & w_{2N} \\ \vdots & & & \vdots & \\ w_{i1} & 0 & \dots & w_{ij} & 0 \\ 0 & w_{N2} & 0 & \dots & 0 \end{bmatrix}$$

Degree matrix

$$D_{ii} = \sum_j W_{ij}$$

Graph Laplacian

$$L = D - W$$

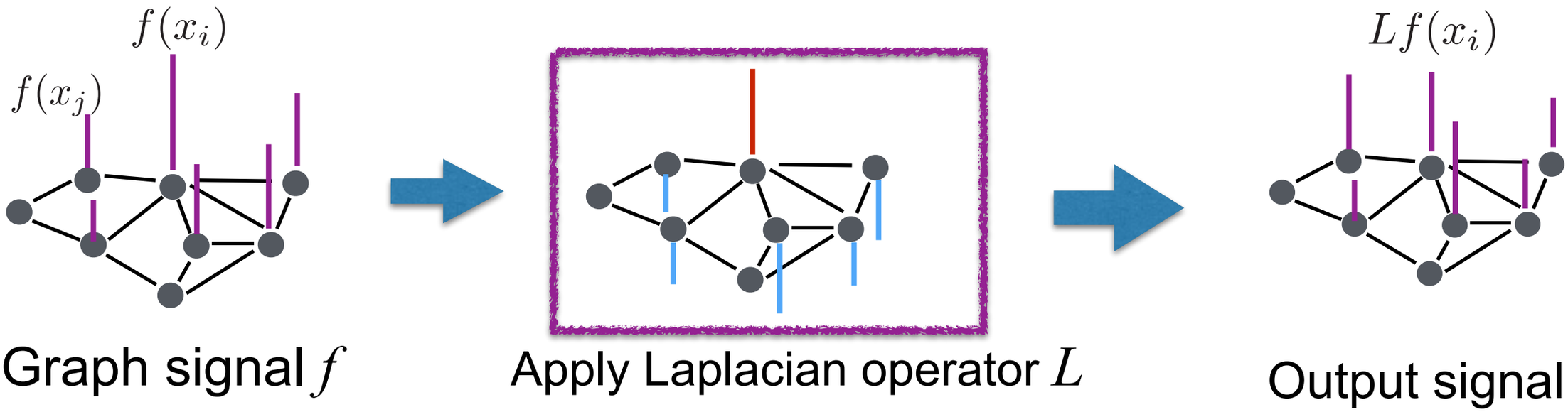
- L defines an operator on graph signals:

$$Lf = \begin{bmatrix} d_{11} & -w_{12} & \dots & 0 & 0 \\ -w_{21} & d_{22} & \dots & 0 & -w_{2N} \\ \vdots & & & \vdots & \\ -w_{i1} & 0 & \dots & & 0 \\ 0 & -w_{N2} & 0 & \dots & d_{NN} \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix}$$

$$(Lf)(x_i) = d_{ii} f(x_i) - \sum_{x_j \in \mathcal{N}(x_i)} W_{ij} f(x_j)$$

Graph Laplacian vs. Laplacian operator

$$(Lf)(x_i) = d_{ii} f(x_i) - \sum_{x_j \in \mathcal{N}(x_i)} W_{ij} f(x_j)$$

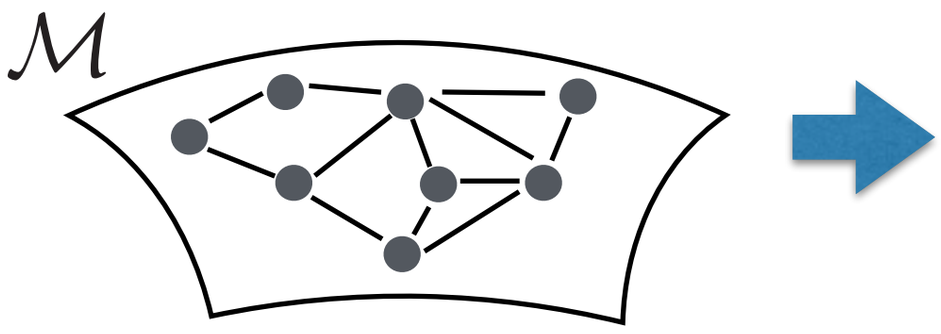


- Compare to the 2nd derivative kernel: [-1 2 -1]
- … or the general Laplacian operator $\Delta f = \mathrm{div}(\nabla f)$

$$\Delta f(x) = \frac{\partial^2 f}{\partial x^2} \qquad \Delta f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Frequency Analysis via Graph Laplacian

- In fact, Lf is indeed the graph equivalent of Δf



As $N \to \infty$, $Lf \to \Delta_{\mathcal{M}} f$

[Hein '05, Singer '06]

- Why is the Laplacian important for frequency analysis?
- Check the eigenfunctions of the Laplacian: $\Delta f = \lambda f$

$$f(t) = e^{j\Omega t} \qquad \Rightarrow \qquad -\Delta(e^{j\Omega t}) = \Omega^2 e^{j\Omega t}$$

- The eigenfunctions of Δf are complex exponentials

Fourier Bases on Graphs

- The "complex exponentials" in the graph domain are the eigenvectors of L: [Shuman, 2013]

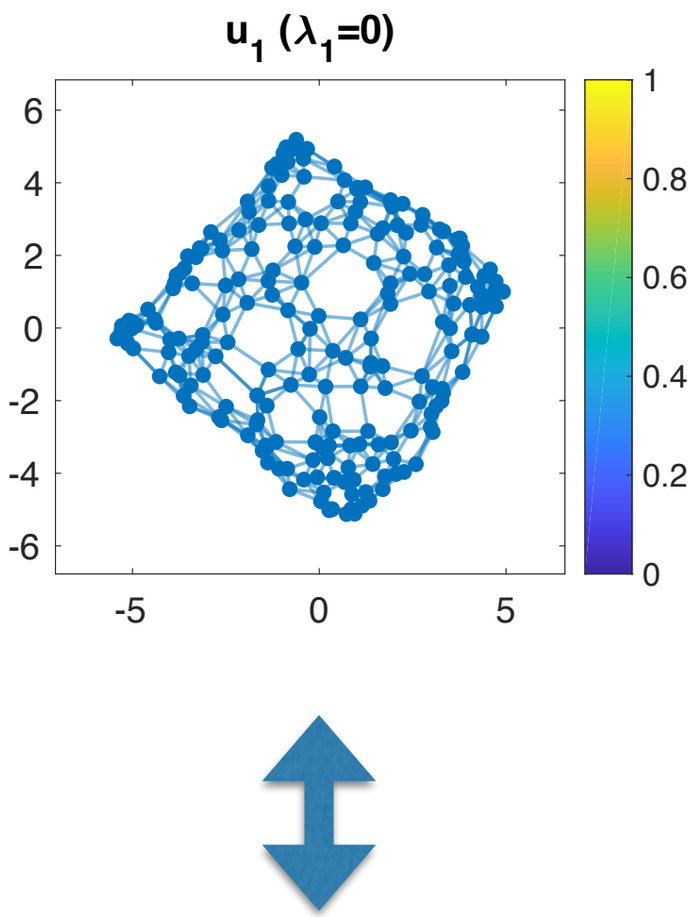
$$Lu_k = \lambda_k u_k \quad \text{for} \quad k = 1, \ldots, N$$

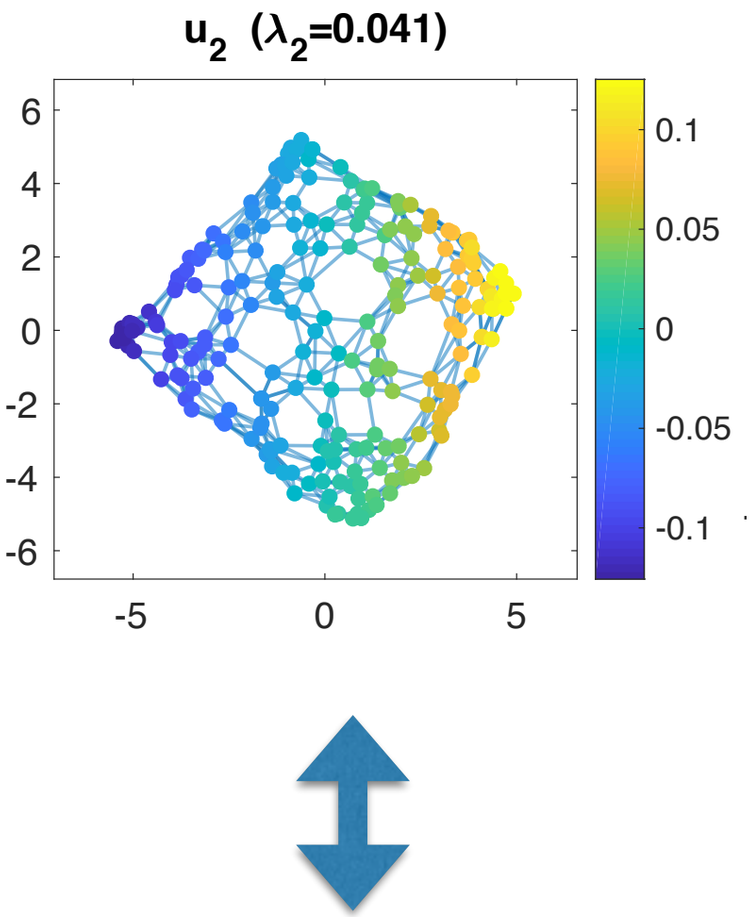
- Compare:

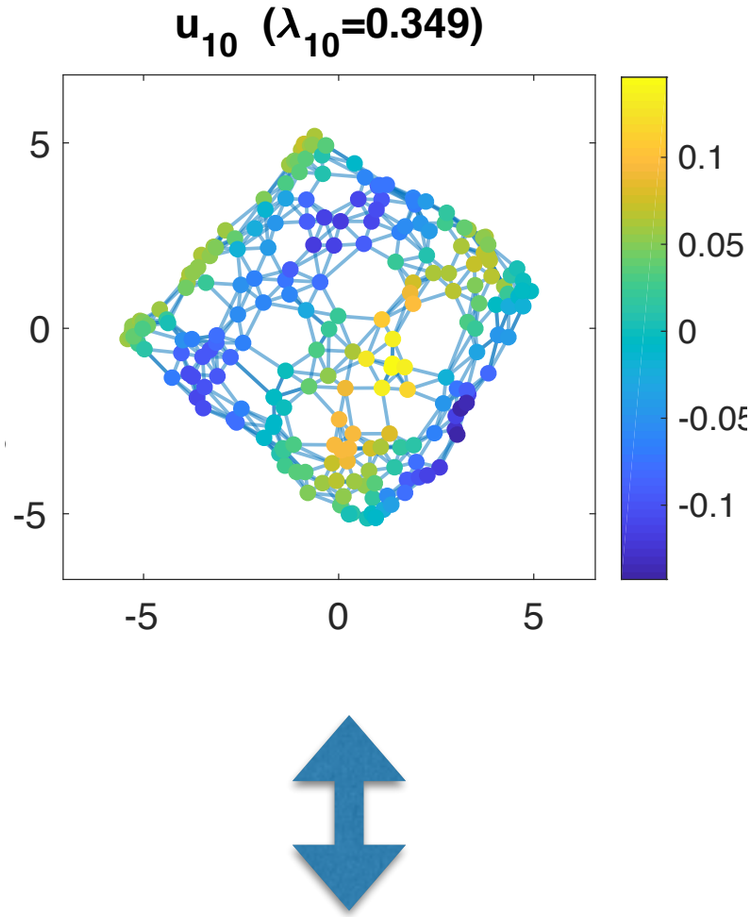
$$-\Delta(e^{j\Omega t}) = \boxed{\Omega^2} e^{j\Omega t} \quad \text{frequency} \qquad \Longleftrightarrow \qquad Lu_k = \boxed{\lambda_k} u_k \quad \text{frequency}$$

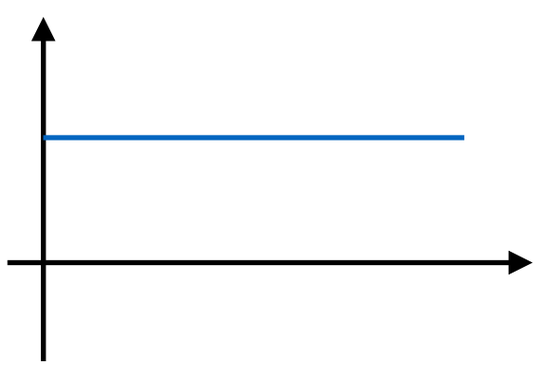
- As λ_k increases, u_k varies more rapidly on the graph.

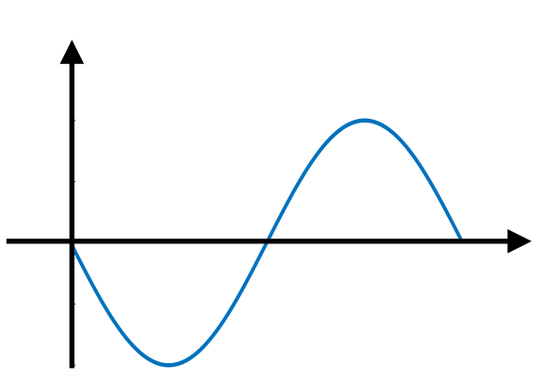
Fourier Bases on Graphs

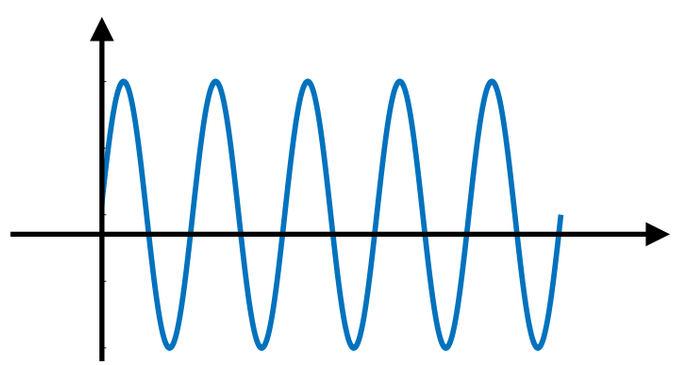












Graph Fourier Transform (GFT)

Classical Signal Processing

- Complex exponentials

$$e^{j\Omega t}$$

- Fourier Transform

$$\hat{f}(\Omega) = \langle f, e^{j\Omega t} \rangle = \int_{-\infty}^{\infty} f(t) e^{-j\Omega t} dt$$

- Inverse Fourier Transform

$$f = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\Omega) e^{j\Omega t} d\Omega$$

Graph Signal Processing

- Lap. eigenvectors $Lu_k = \lambda_k u_k$

$$u_k$$

- Graph Fourier Transform

$$\hat{f}(\lambda_k) = \langle f, u_k \rangle = \sum_{i=1}^{N} f(x_i) u_k(x_i)$$

$U = [u_1 \; u_2 \; \cdots \; u_N]$: Fourier basis

$\hat{f} = U^T f$: GFT of f

- Inverse Graph Fourier Transform

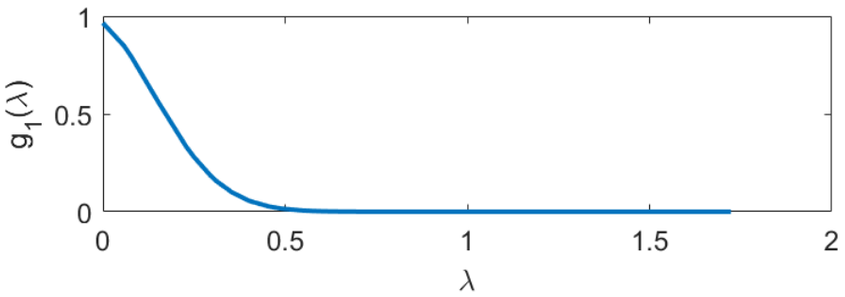
$$f = \sum_{k=1}^{N} \hat{f}(\lambda_k) u_k = U \hat{f}$$

Filtering on Graphs

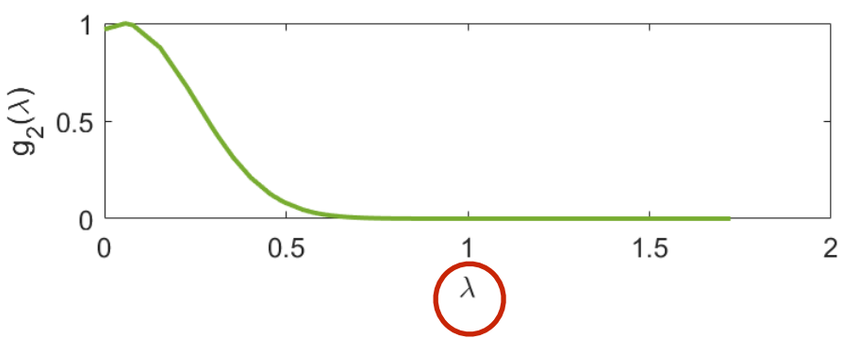
- Classical signal processing:
  - Filtering = Convolution
- Graph signal processing: Irregular topologies!
  - Hard to define convolution in vertex domain…
  - Define filtering operation in spectral domain
  - Filtering: Multiplication in spectral domain

Graph Filter Kernels

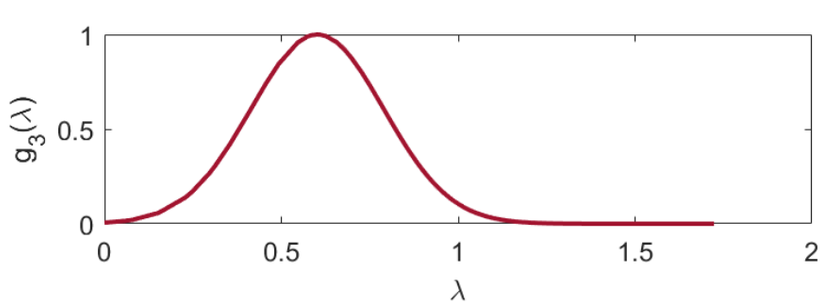
- Graph kernels in spectral domain



Lowpass kernel



λ : frequency variable



Bandpass kernel

- How to generate these kernels in the vertex domain?

$g(\lambda)$: Scalar function

$$L = U \Lambda U^T$$

$$\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix}$$

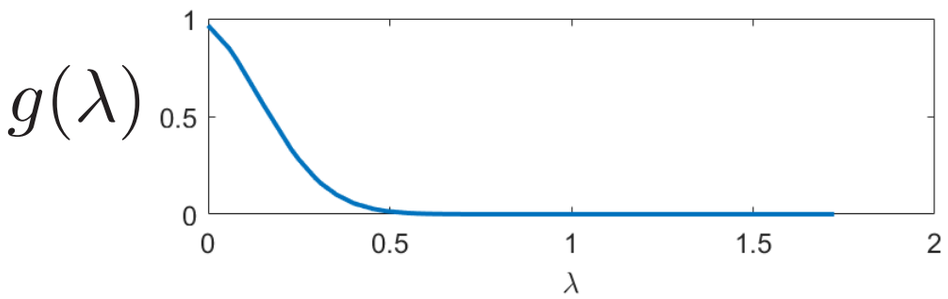
$g(L)$: Matrix function

$$g(L) = U\, g(\Lambda)\, U^T$$

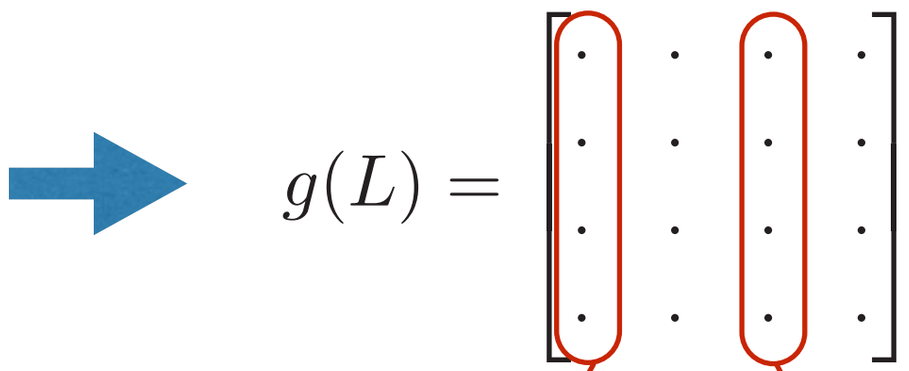
$$\begin{bmatrix} g(\lambda_1) & & & \\ & g(\lambda_2) & & \\ & & \ddots & \\ & & & g(\lambda_N) \end{bmatrix}$$

Localizing kernels on graph nodes

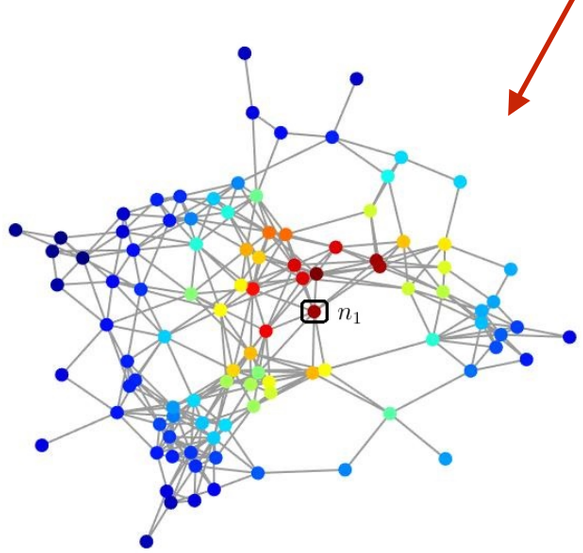
- The matrix $g(L) = U\ g(\Lambda)\ U^T$ localizes the kernel $g(\lambda)$ on the graph nodes:



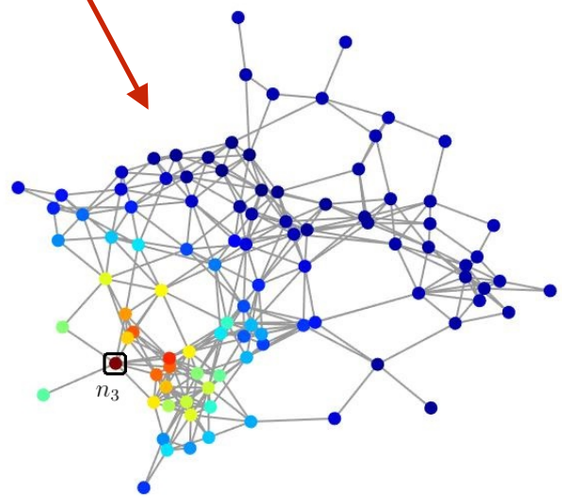
Lowpass kernel



$g(L) =$

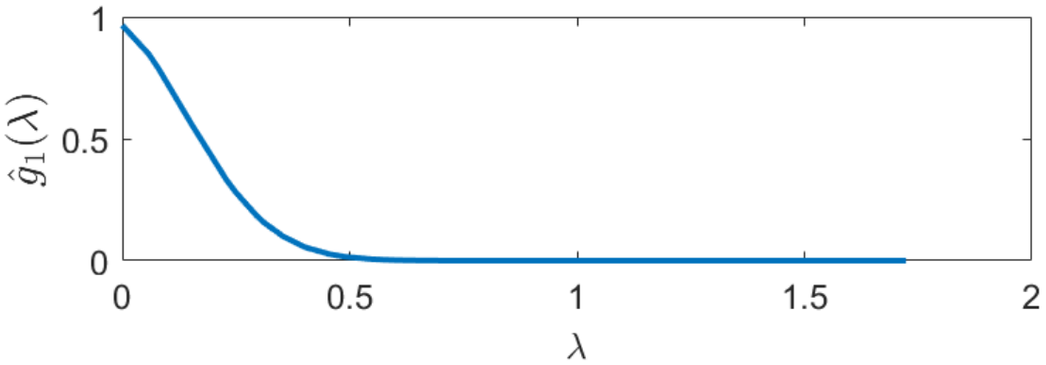


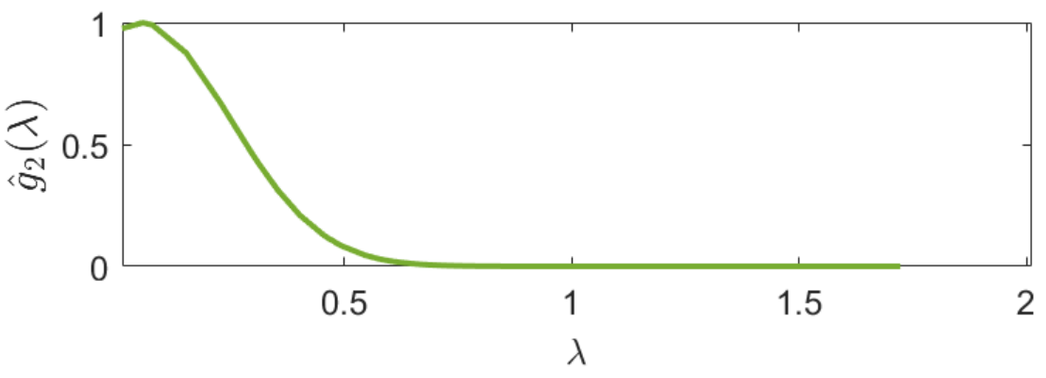
"Lowpass" graph signal localized at node 1

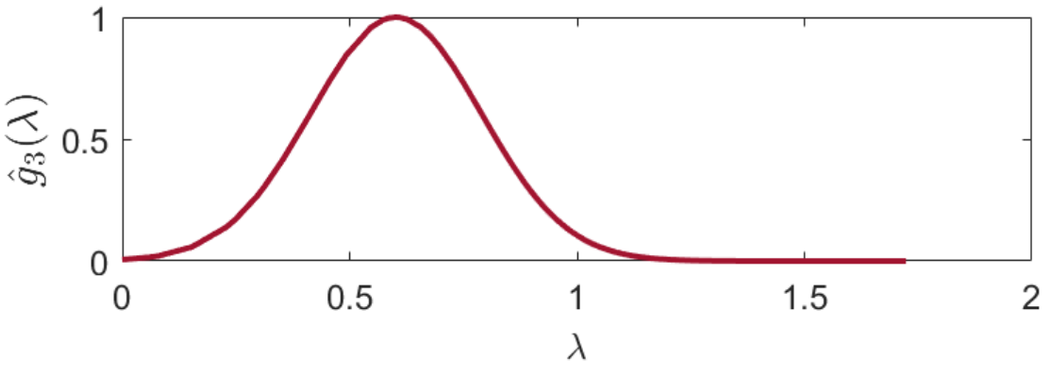


"Lowpass" graph signal localized at node 3

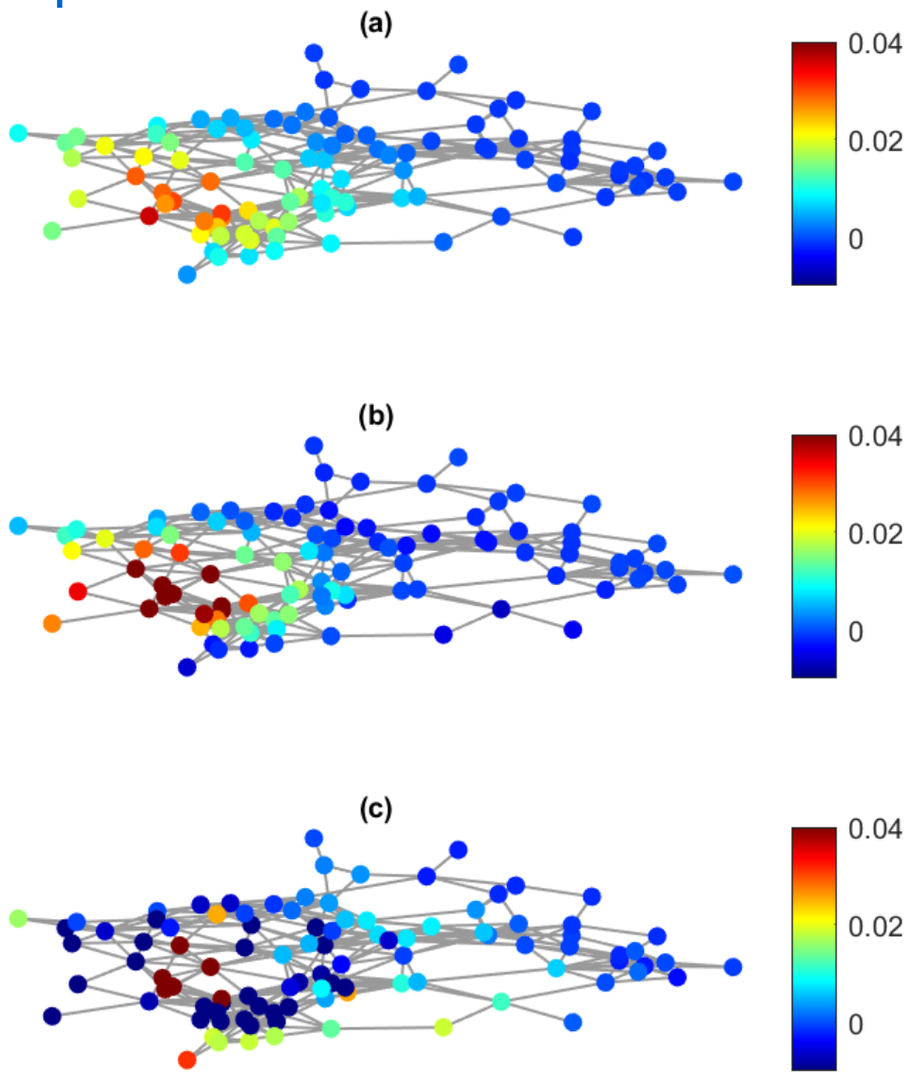
Effect of the choice of graph kernel







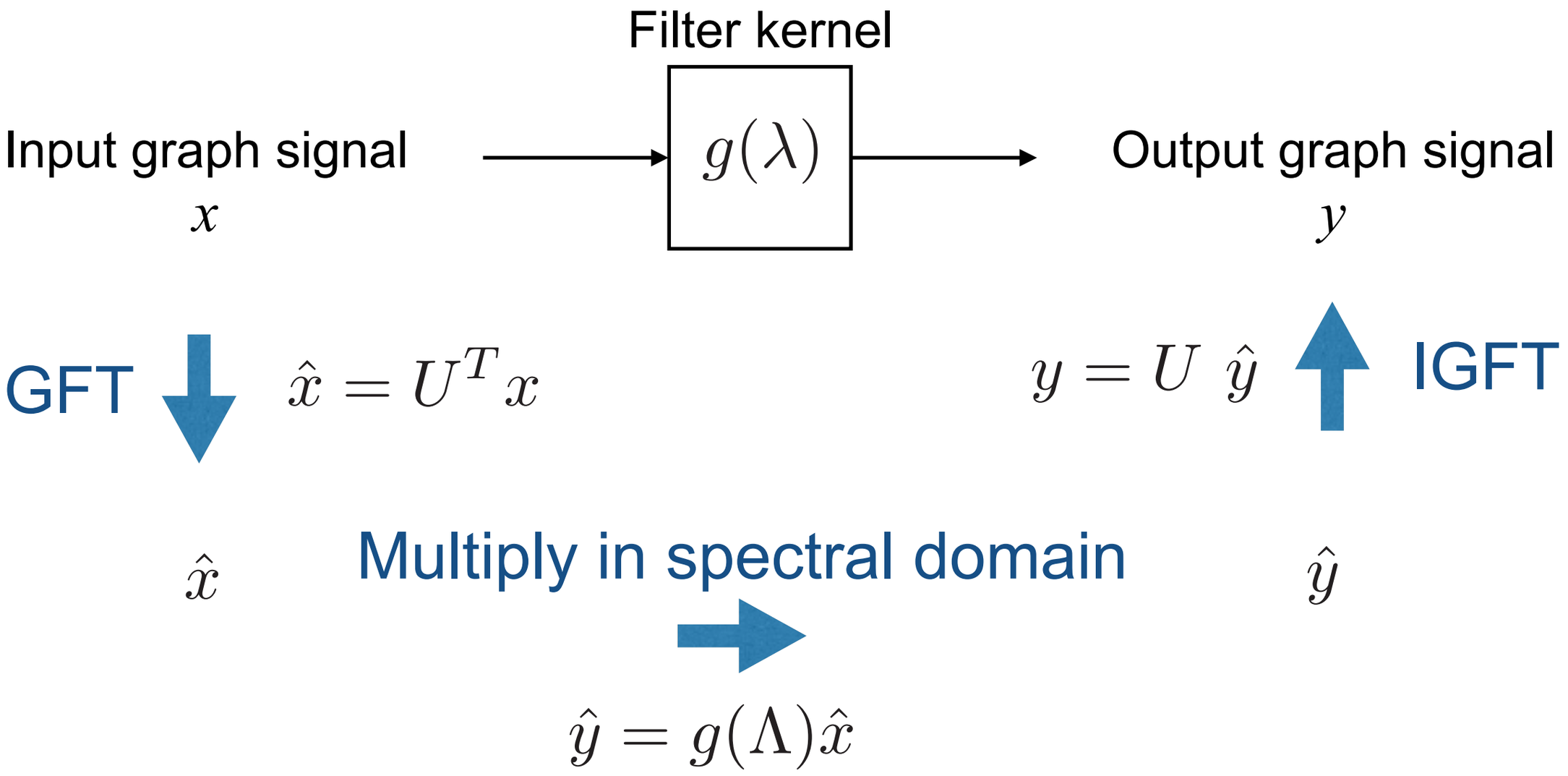
Low freq.



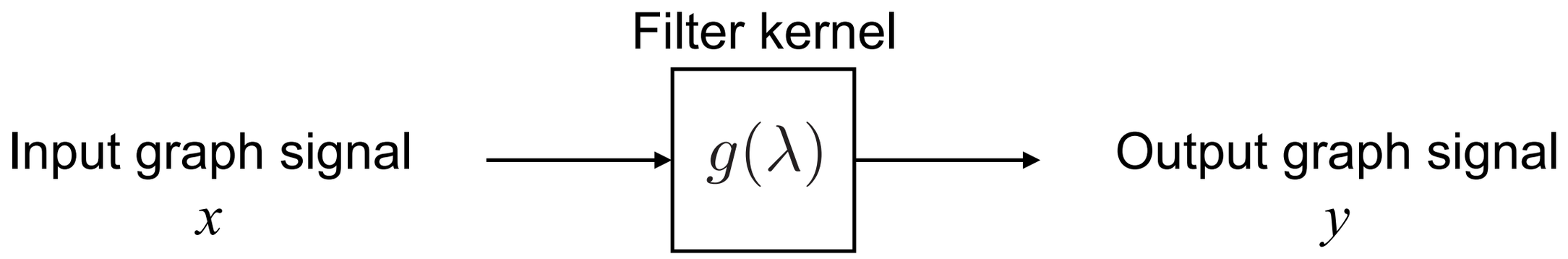
High freq.

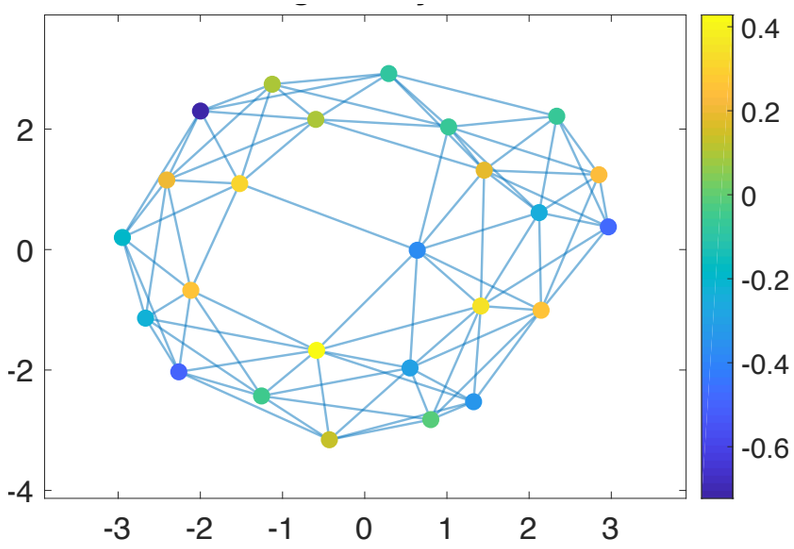
Back to the filtering problem …

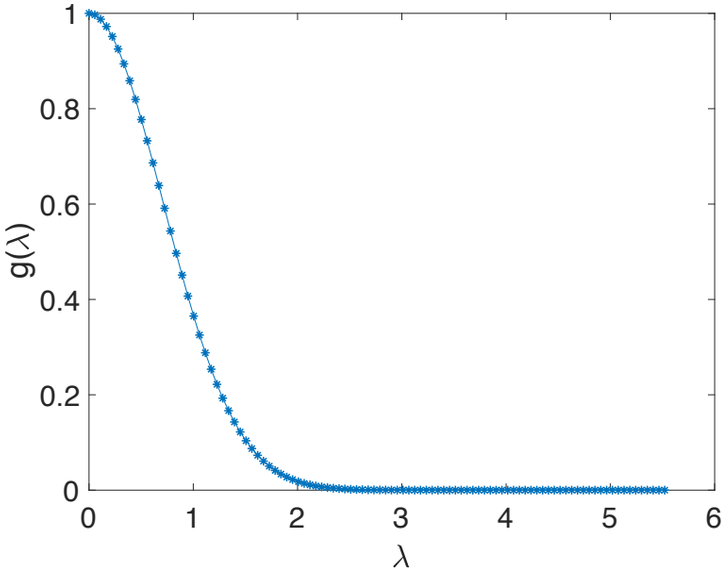
- Consider the filtering problem:

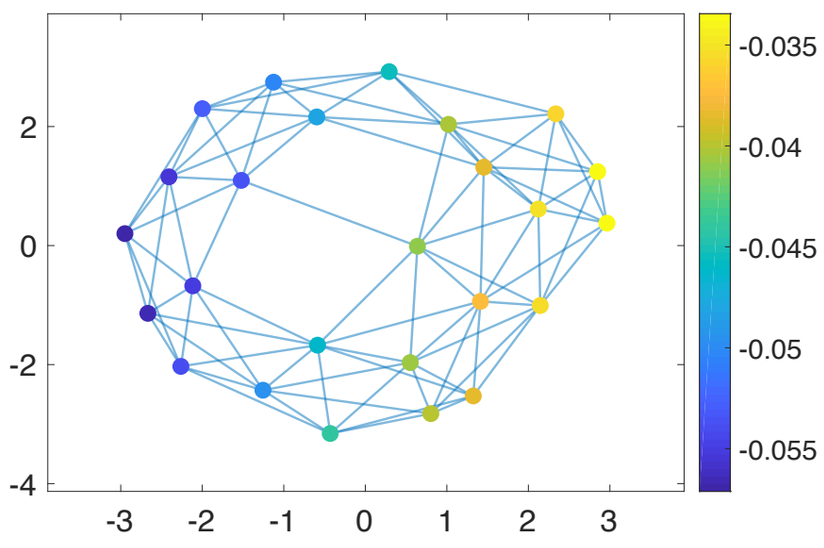


Example: Low-pass filtering a graph signal









Representations for Graph Signals

- Representations in transform domain: GFT

$$\begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} U \end{bmatrix} \begin{bmatrix} \hat{y} \end{bmatrix}$$

- Representations on overcomplete sets:

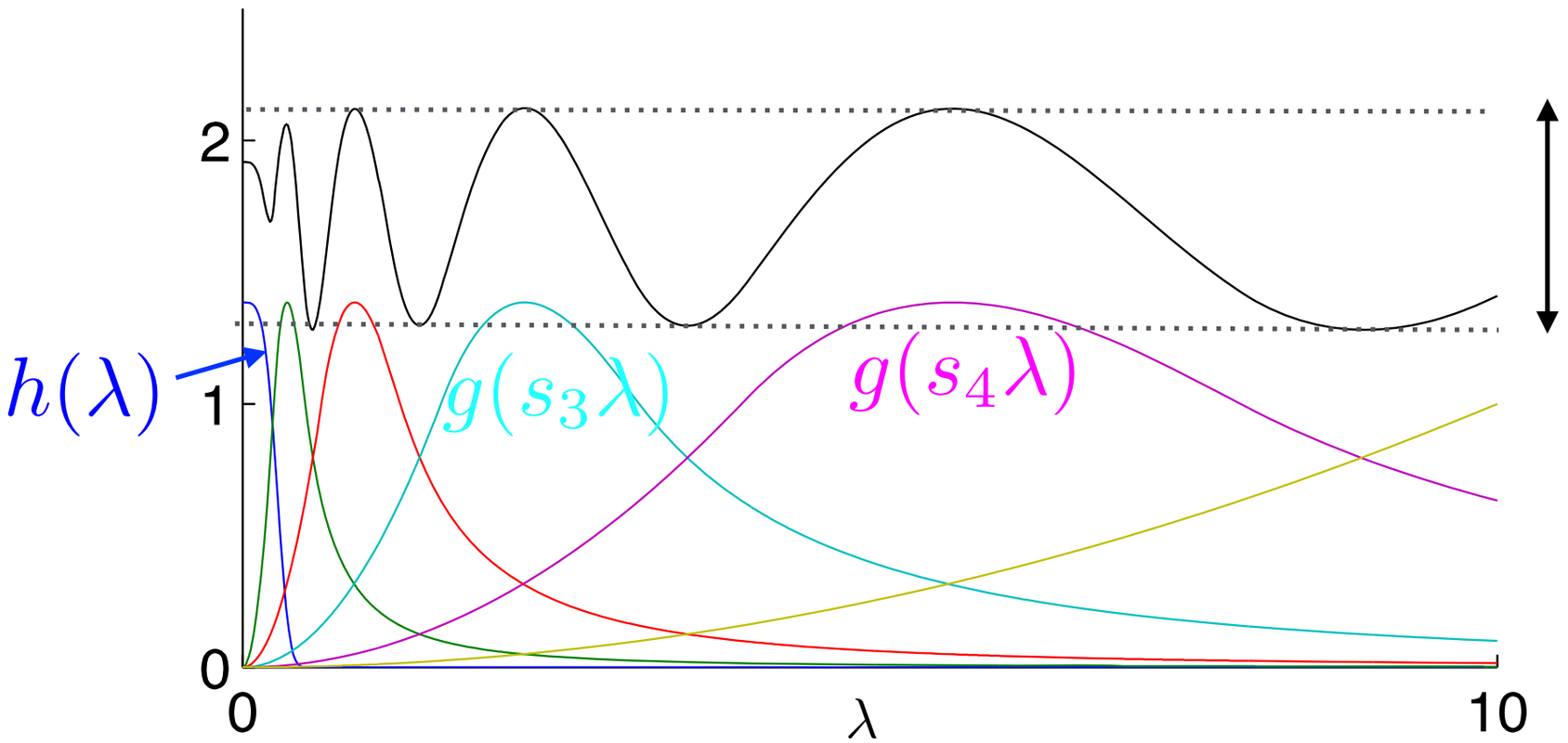
$$\begin{bmatrix} y \end{bmatrix} = \begin{bmatrix} \quad D \quad \end{bmatrix} \begin{bmatrix} x \\ \\ \end{bmatrix}$$

  - Predefined frames: Spectral Graph Wavelets [Hammond, 2011]

  - Adaptive representations: Graph Dictionaries [Zhang, 2012], [Thanou, 2014], …

Spectral Graph Wavelets

- Wavelet design problem is formulated in the spectral domain [Hammond, 2011]:
  - Low-pass scaling function: $h(\lambda)$
  - Wavelet-generating kernel: $g(\lambda)$
  - Band-pass wavelet functions: $g(s_j\lambda)$



Choose kernels to make total energy almost flat

Spectral Graph Wavelet Dictionaries

- Recall

$g(\lambda)$: Scalar function → $g(L) = U\ g(\Lambda)\ U^T$: Matrix function

Scaling function $h(\lambda)$

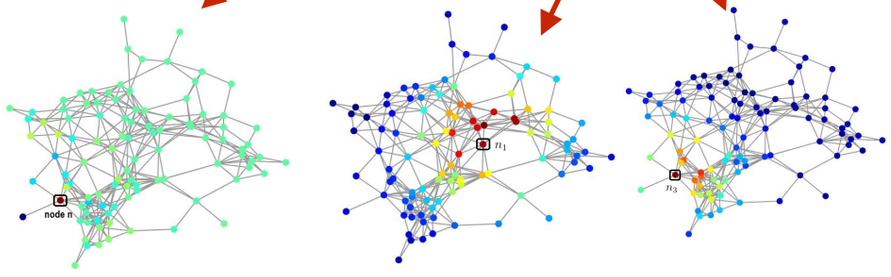
Wavelets $g(s_j\lambda)$

→ Wavelet dictionary:

$$D = [h(L)\ g(s_1 L)\ \ldots g(s_J L)]$$

- Graph signals can be sparsely represented over D

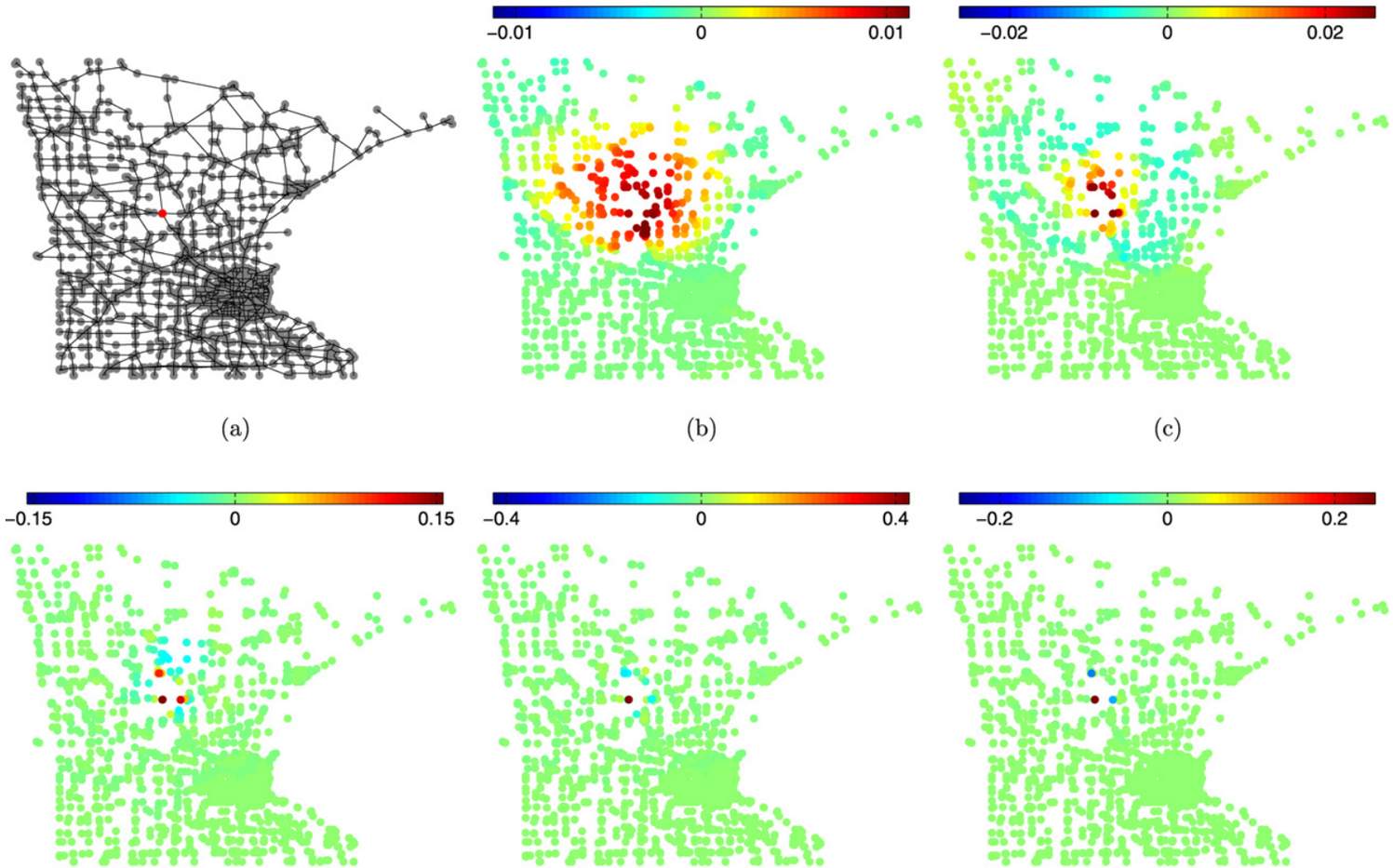
$$[y] = [\ D\ ][x]$$



Sparse coefficient vector

Spectral Graph Wavelets: Applications

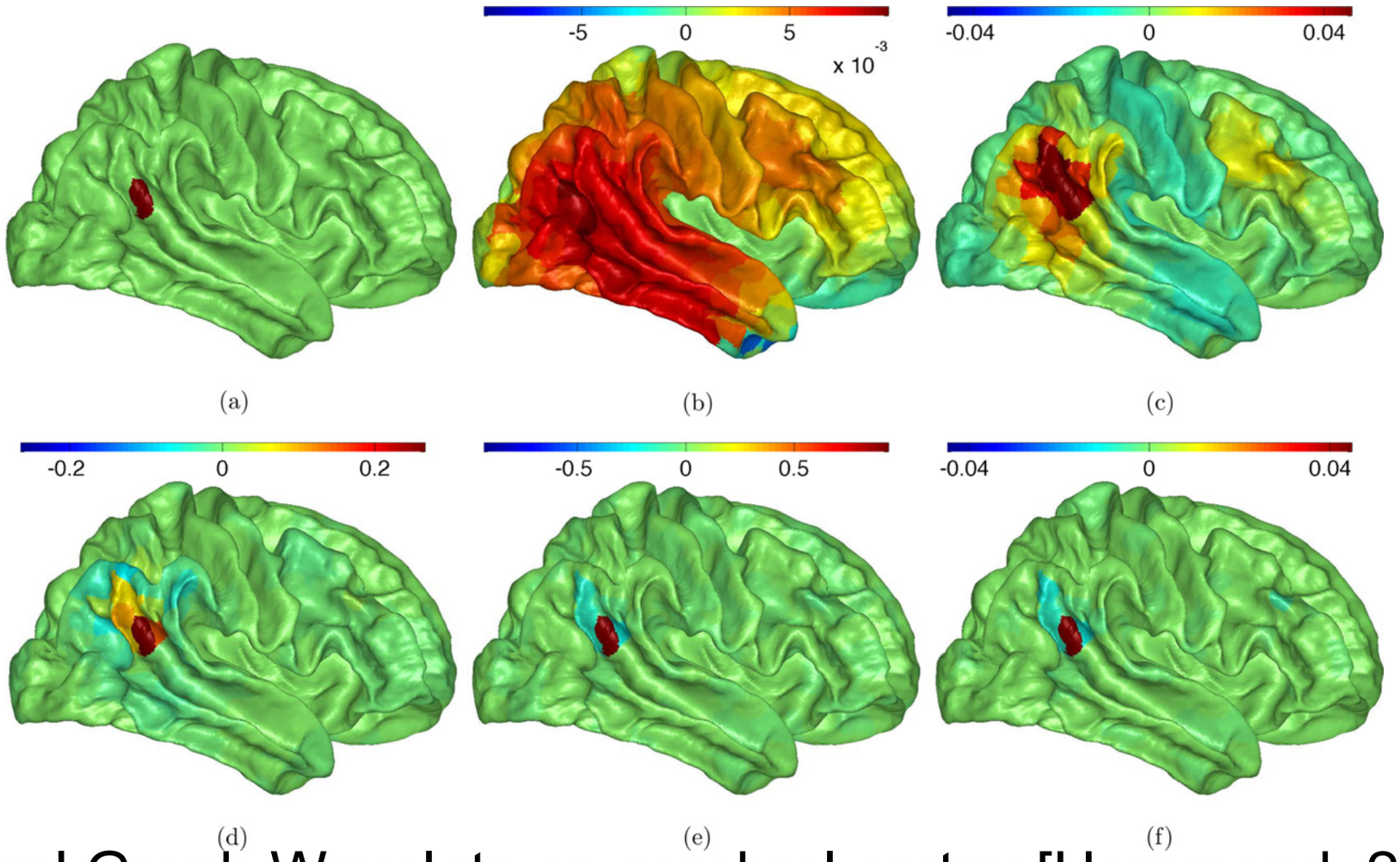
- Analysis of data on networks: (Movement along transportation networks, spread of epidemics, …)



Spectral Graph Wavelets on Minnesota road graph [Hammond, 2011]

Spectral Graph Wavelets: Applications

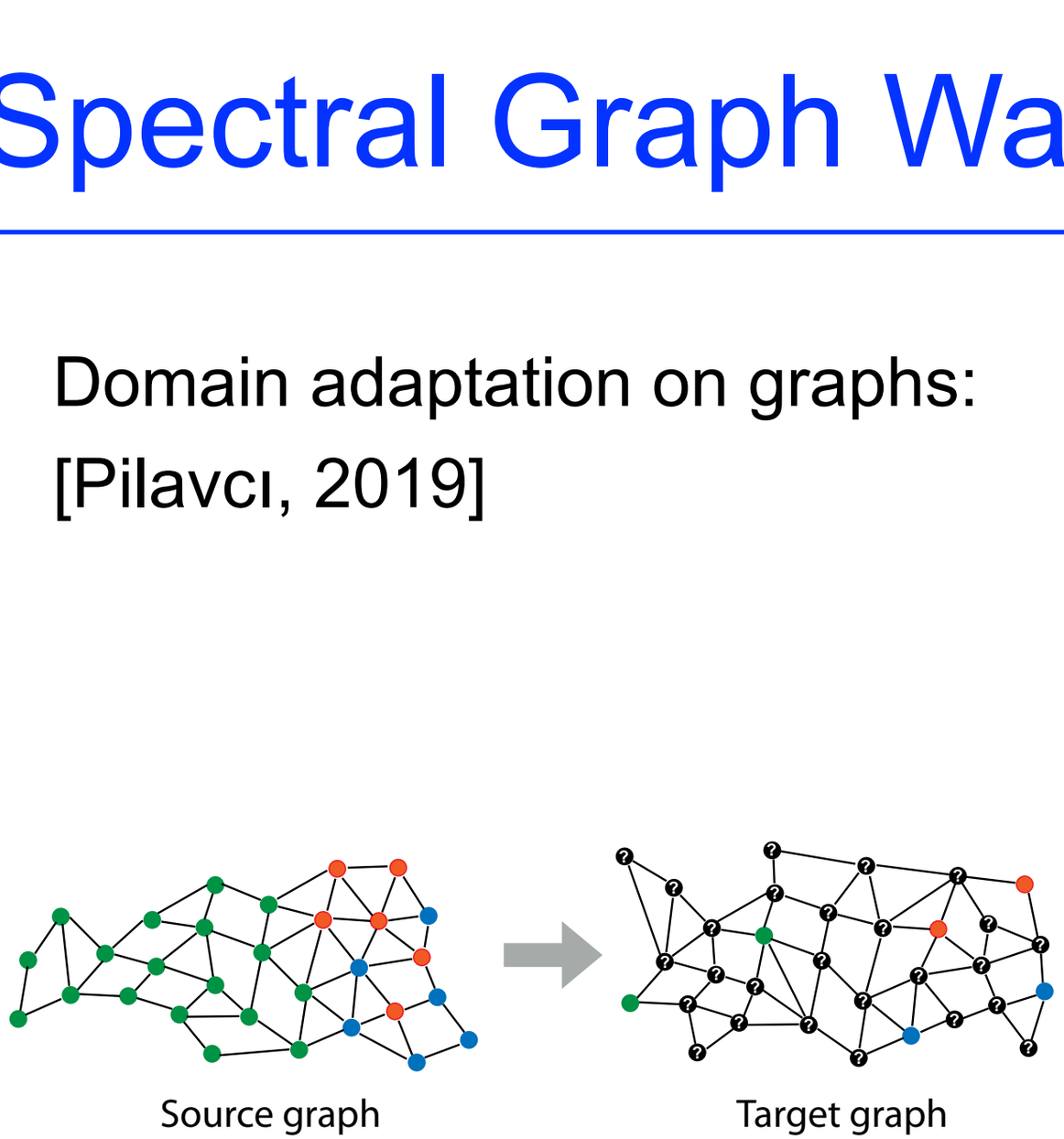
- Analysis/Denoising of functional MRI data, based on brain connectivity network models

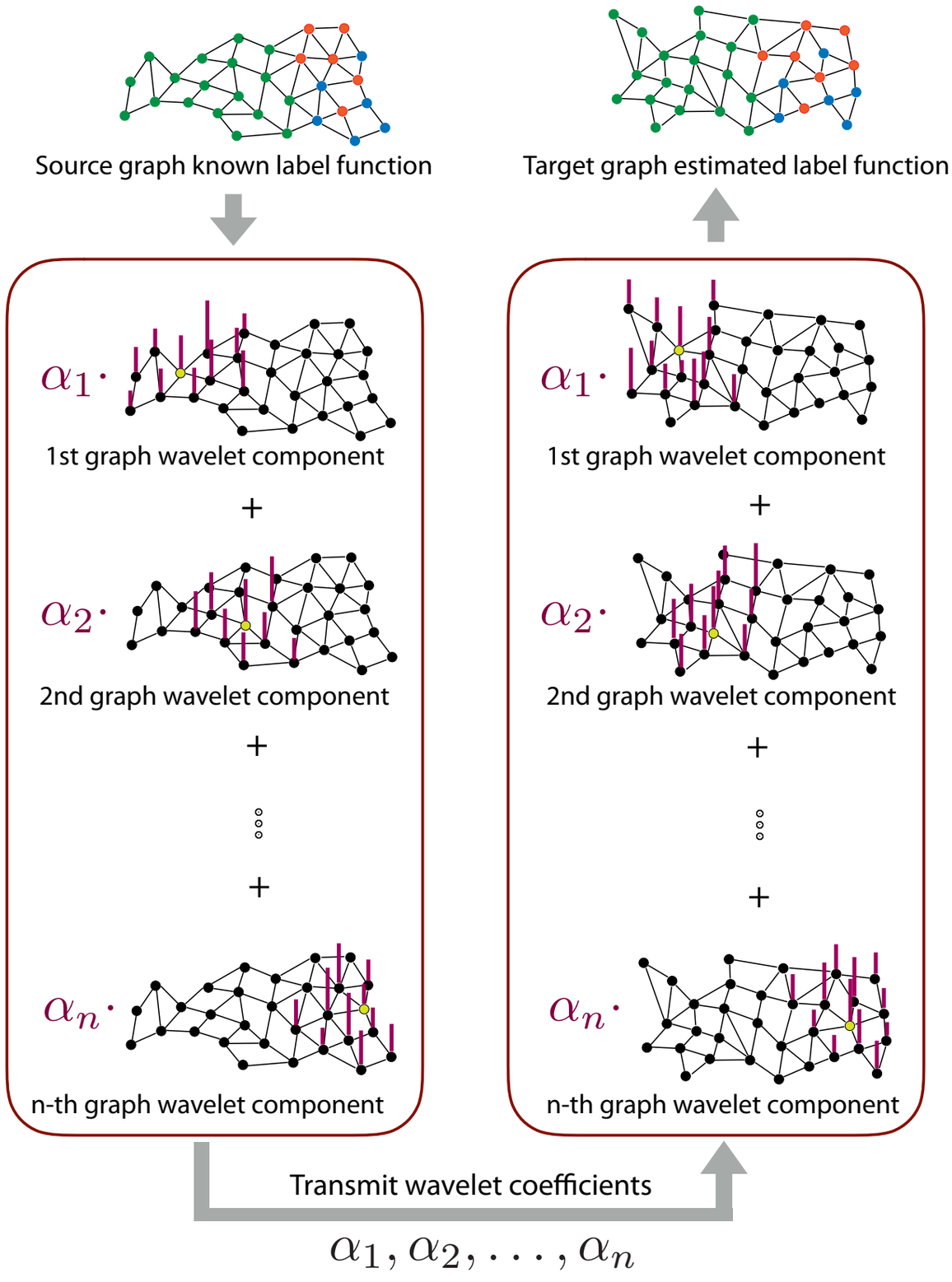


Spectral Graph Wavelets on cerebral cortex [Hammond, 2011]

Spectral Graph Wavelets: Applications

- Domain adaptation on graphs: [Pilavcı, 2019]





Adaptive Representations

- Classical sparse signal model: $\left[y\right] = \left[\quad D \quad\right]\left[x\right]$

  y: Signal, D: Dictionary, x: Sparse coefficients

- Classical dictionary learning problem:

$$\min_{D,X} \|Y - DX\|^2 + \gamma\|X\|_1$$

- Dictionary learning for graph signals:

  $y = D\ x$ y: Graph signal, D: Dictionary, x: Sparse coefficients

- Dictionaries must be adapted to graph topology:

$$D = [D_1\ D_2\ \ldots\ D_J]$$

$D_j = U\,\Sigma_j\,U^T$: j-th subdictionary

Learning D = Learning Σ_j 's

Graph Dictionary Learning

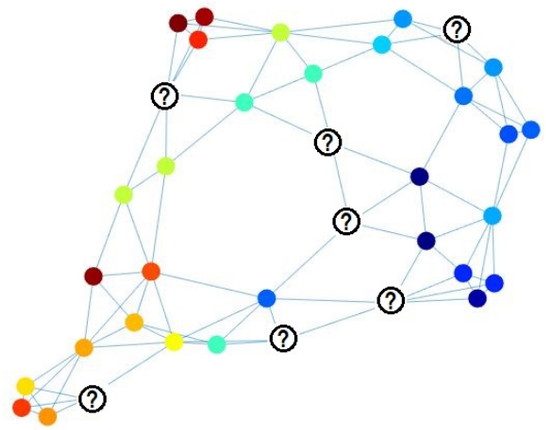
$D = [D_1 \; D_2 \; \ldots \; D_J]$ $\qquad$ $D_j = U \, \Sigma_j \, U^T$: j-th subdictionary

- Non-parametric dictionary learning [Zhang, 2012]:
  - Learn Σ_j's as nonparametric diagonal matrices
- Parametric dictionary learning [Thanou, 2014]:
  - Learn polynomial kernels: $\Sigma_j = g_j(\lambda) = \sum_k \alpha_{jk} \lambda^k$
  - Solve problem: $\min_{X,\alpha} \; \|Y - DX\|^2 + \gamma \|\alpha\|^2$

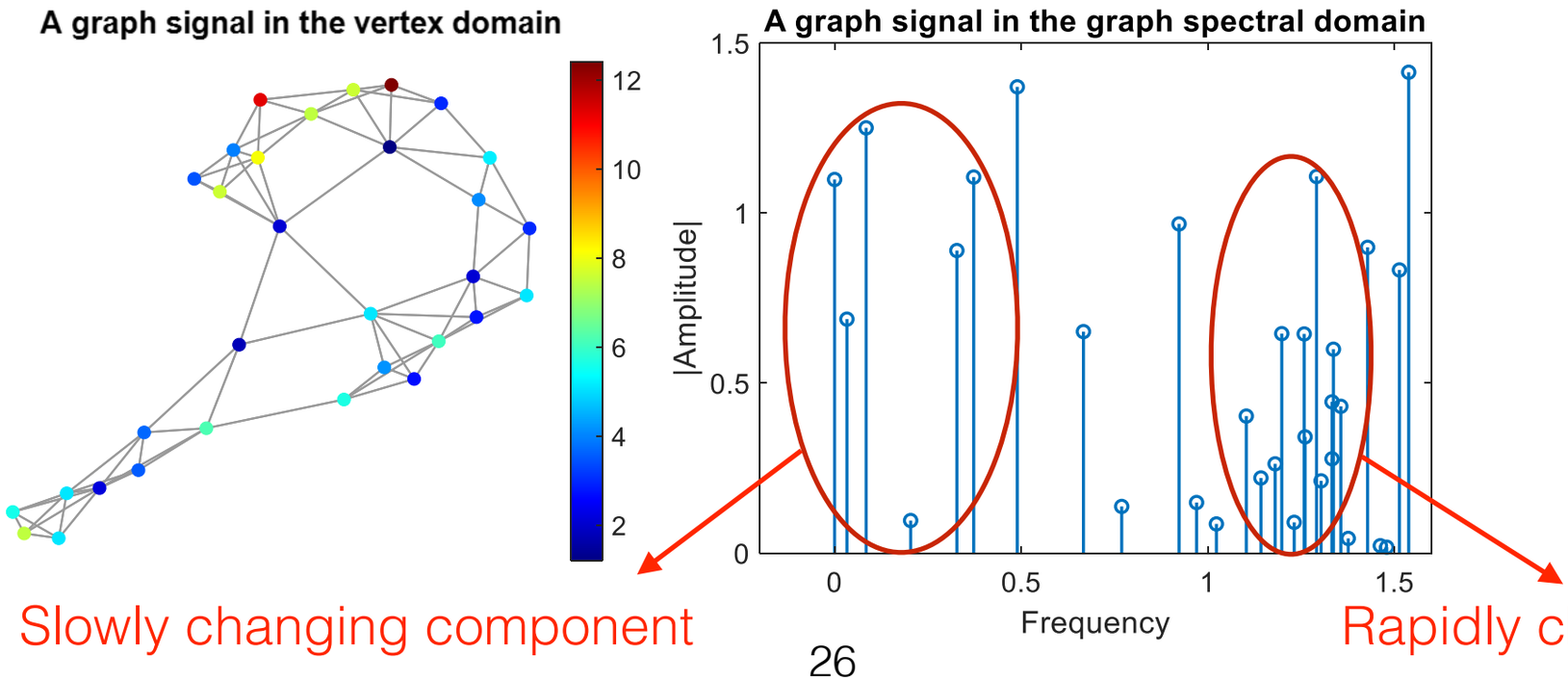
$$\text{s.t. } D_j = \sum_k \alpha_{jk} \, L^k, \quad \|X\|_0 \leq T$$

Dictionary Learning from Incomplete Graph Signals

- Aim: Estimate missing observations of partially observed graph signals



- Observation: Natural graph signals often contain components concentrated at different parts of the spectrum:
  - Example: Meteorological signals (wind speed measurements)



Slowly changing component

Rapidly changing component

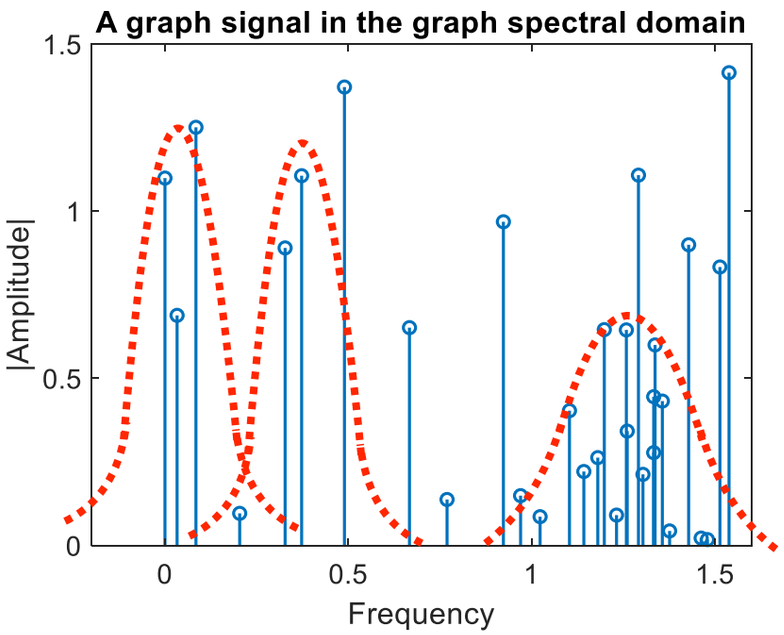
Dictionary Learning from Incomplete Graph Signals

- Idea: Learn narrowband kernels that fit to different components of the signal spectrum [Turhan, 2021]

$$D = [D_1 \; D_2 \; \ldots \; D_J]$$

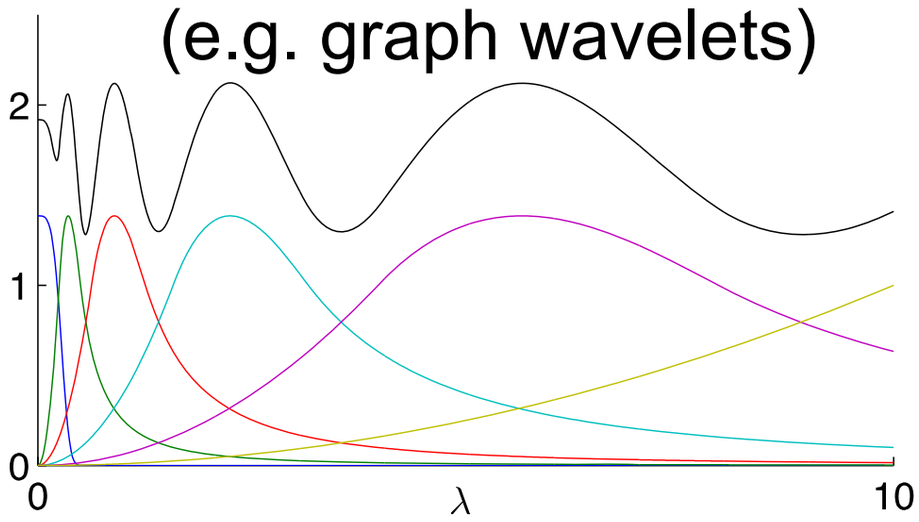
$$D_j = U \, \Sigma_j \, U^T \quad : \; j\text{-th subdictionary}$$

$$\Sigma_j = g_j(\lambda) \quad \Rightarrow \text{: Choose as narrowband Gaussian kernels. Learn kernel parameters from data}$$



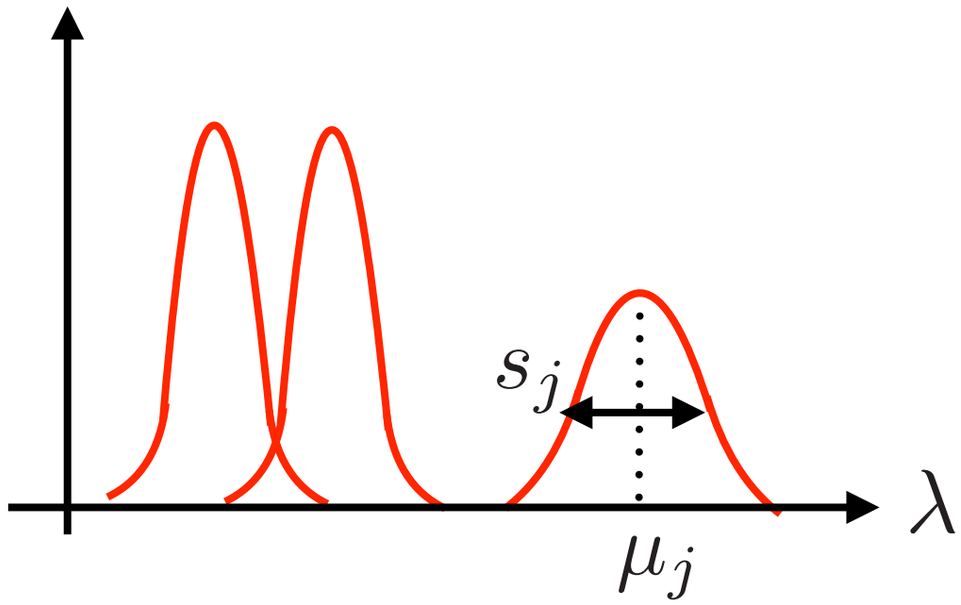
Compare to

Predefined dictionaries (e.g. graph wavelets)



Dictionary Learning from Incomplete Graph Signals

- Spectrally concentrated graph dictionary learning algorithm [Turhan, 2021]



Learn:

$$\mu = [\mu_1 \ \dots \ \mu_J]$$

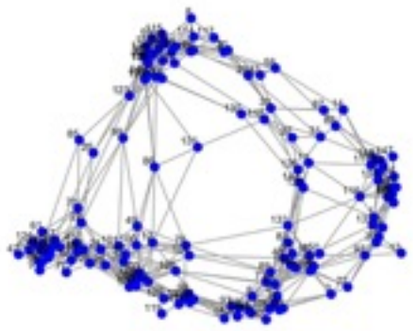
$$s = [s_1 \ \dots \ s_J]$$

- Optimization problem:

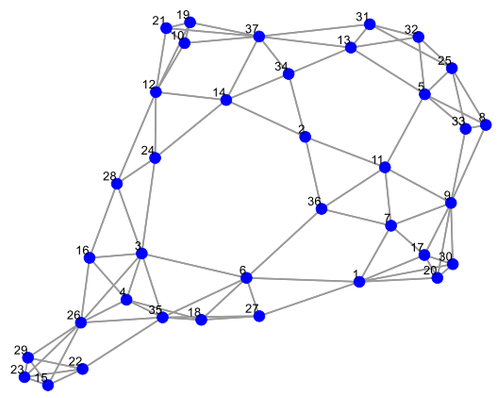
$$\min_{X,\mu,s} \|S\,Y - S\,D(\mu,s)\,X\|^2 + \gamma_1 \|X\|_1 + \gamma_2 R(\mu,s)$$

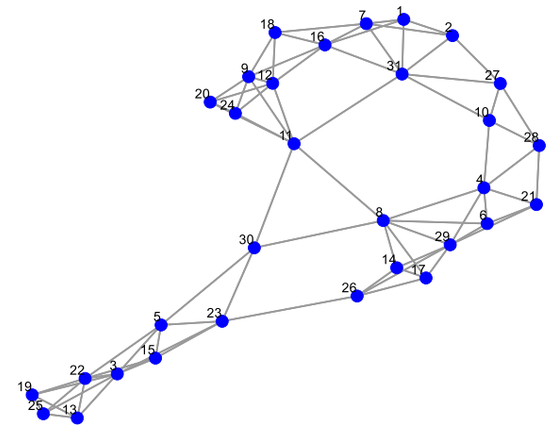
Signal Estimation Performance: Adaptive vs. Predefined Representations

Synthetic graphs

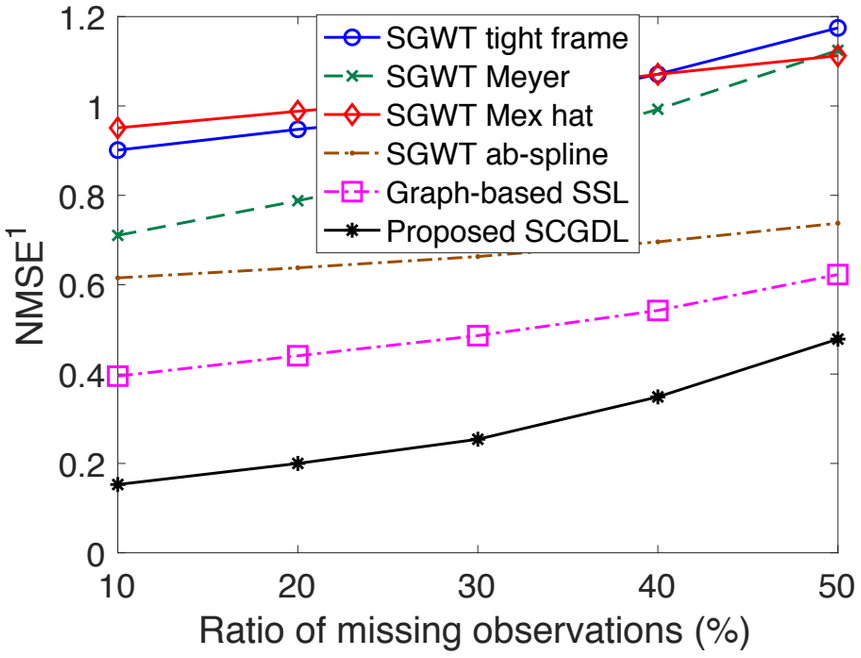


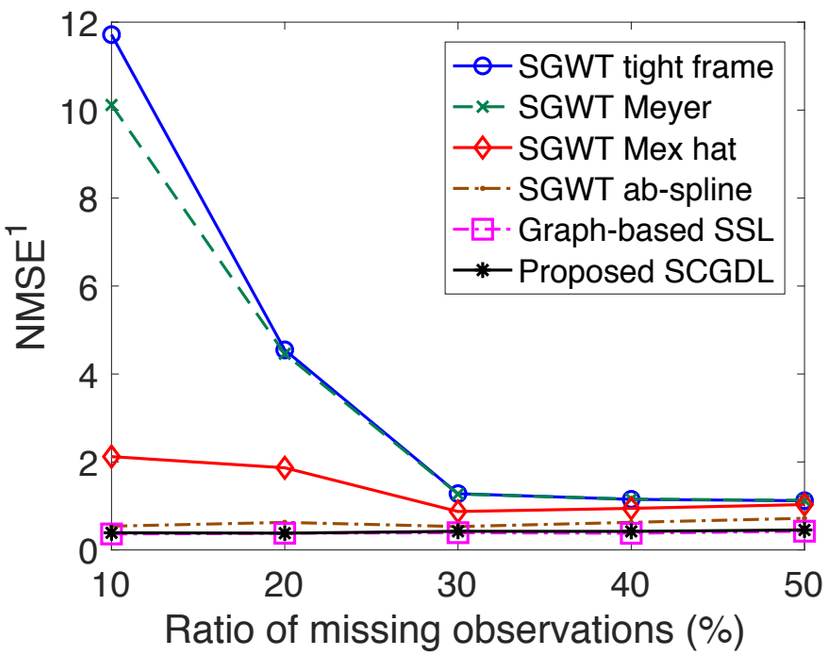
Molène weather network



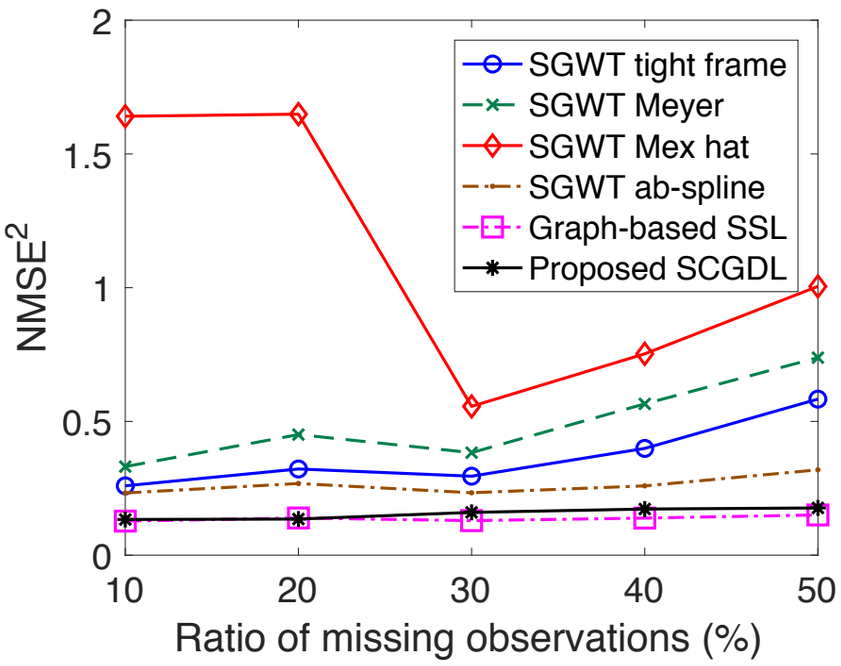


Temperature measurements





Wind speed measurements

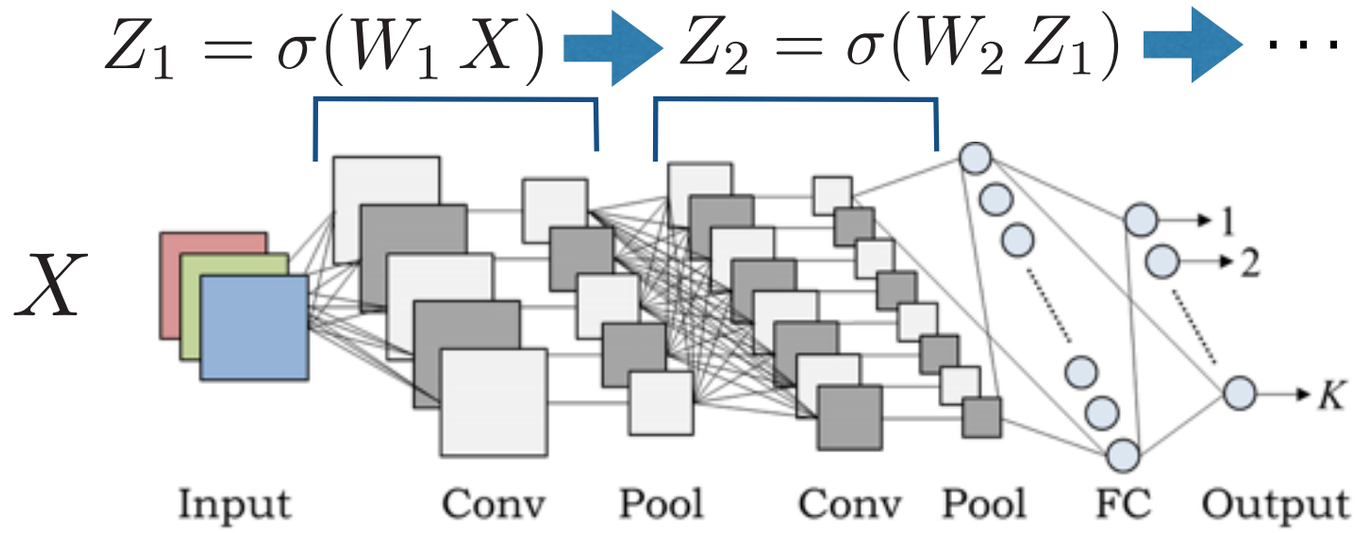


More elaborate techniques…?

- Transform-domain representations: GFT
- Frames, off-the-shelf dictionaries: SGWT
- Adaptive linear representations: Dictionary learning for graph signals
- Richer nonlinear models?

Extending neural networks to graphs

- Classical CNN setting:



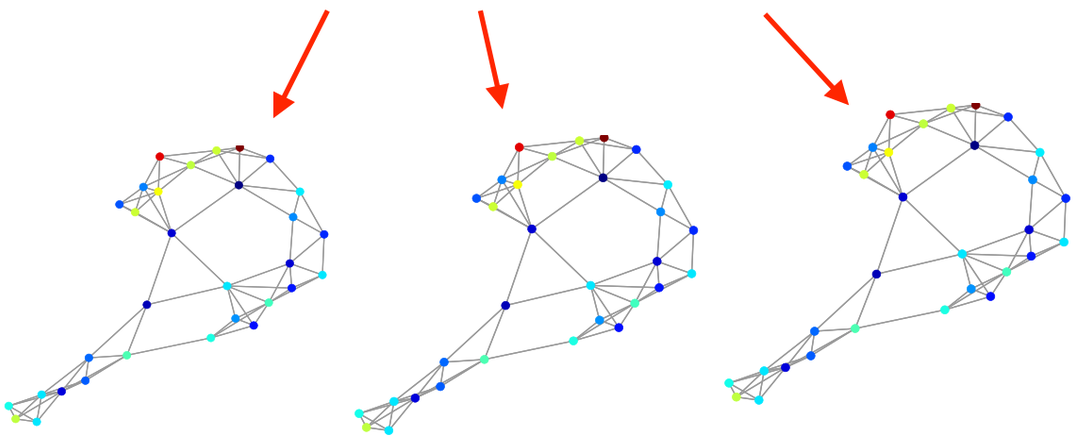
https://www.researchgate.net/

- Graph CNN setting: Input X has an underlying graph topology
  - Document classification in citation networks
  - Web page classification

Graph CNN

Feature vectors

$$X = [X_1 \; X_2 \; ... \; X_K] \in \mathbb{R}^{N \times K}$$



Graph CNN

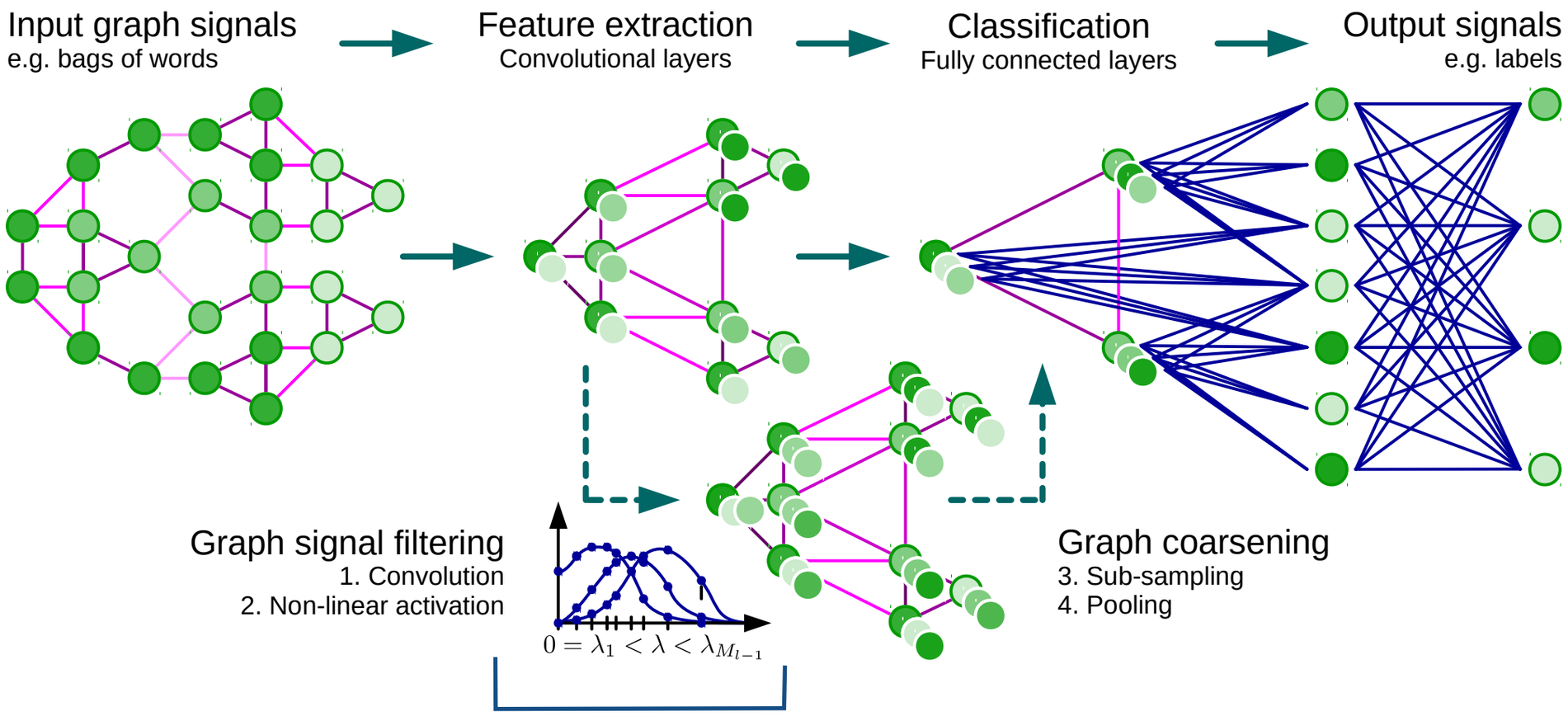
Possible outputs:

- Node class labels
- Graph class labels
- …

- Question: How to incorporate the information of the graph topology in the learnt representation?

- Solution: Formulate the convolution operation as a graph filter

ChebNet [Defferrard, 2016]



Feature extraction step:

$$X = [X_1 \ \cdots \ X_K] \longrightarrow Y_j = \sum_i g^{\theta}_{i,j}(L) X_i$$

Y_j: j-th learnt feature map

$g^{\theta}_{i,j}(L)$: Graph filter

X_i: i-th input feature map

ChebNet: Feature Learning

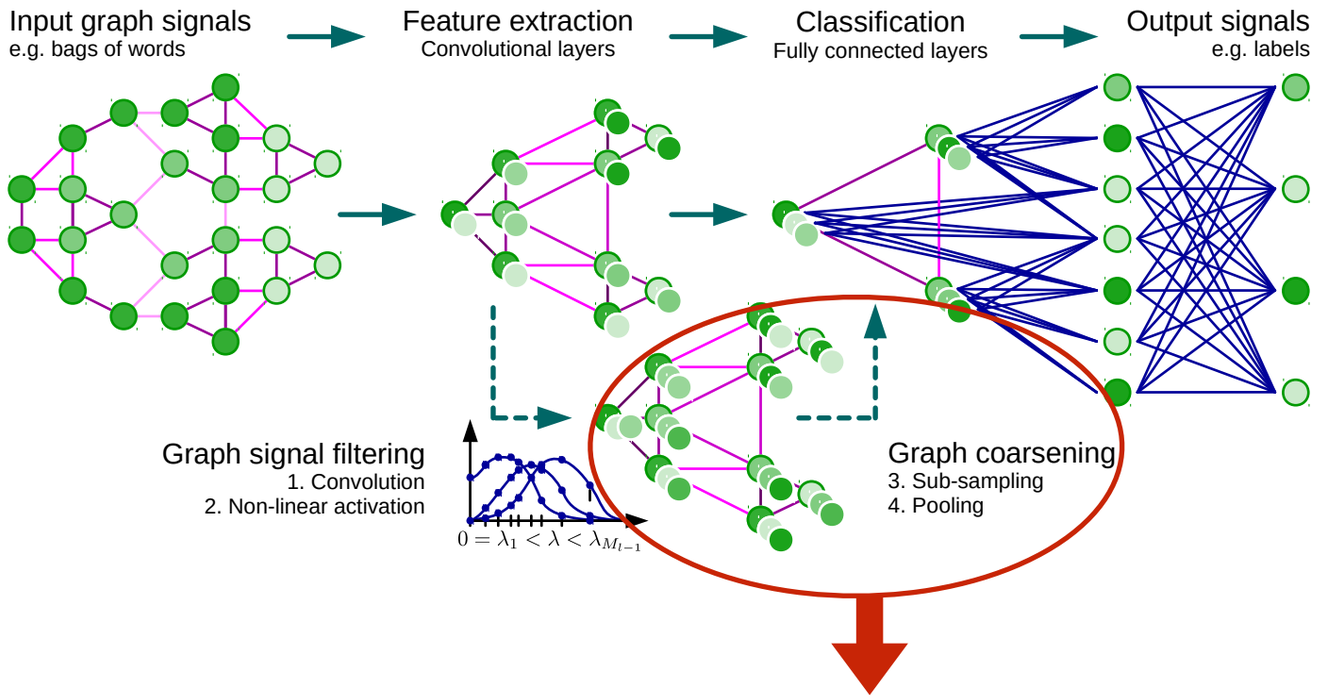
- Hidden layer relation: $Y_j = \sum_i g^{\theta}_{i,j}(L) X_i$
- Graph filter: $g^{\theta}_{i,j}(L) = U \; g^{\theta}_{i,j}(\Lambda) \; U^T$ where $L = U \; \Lambda \; U^T$
- Challenge: U is hard to compute in big graphs!
- ChebNet solution: Choose polynomial kernels

$$g^{\theta}(\Lambda) = \sum_k \theta_k \; \Lambda^k \quad \Rightarrow \quad g^{\theta}(L) = \sum_k \theta_k \; L^k$$

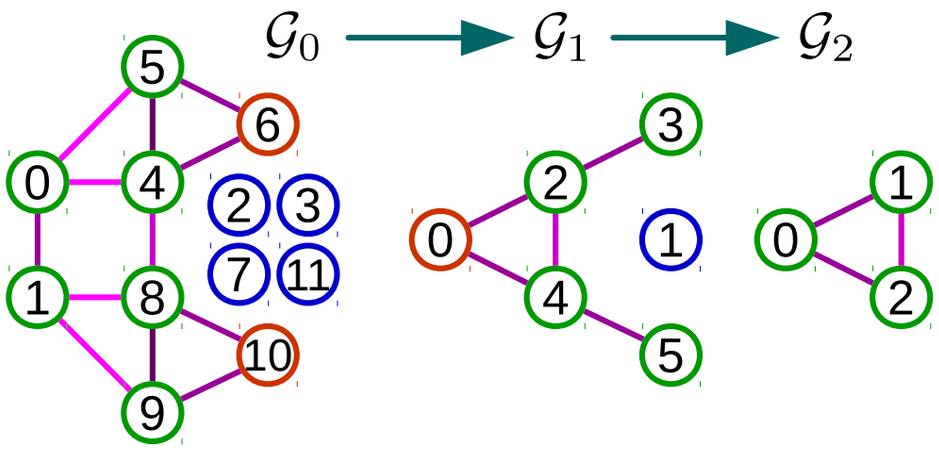
- For faster implementation: Use Chebyshev polynomials

$$g^{\theta}(L) = \sum_k \theta_k \; T_k(L) \quad \Rightarrow \quad \text{Learn} \;\; \theta_k \text{'s}$$

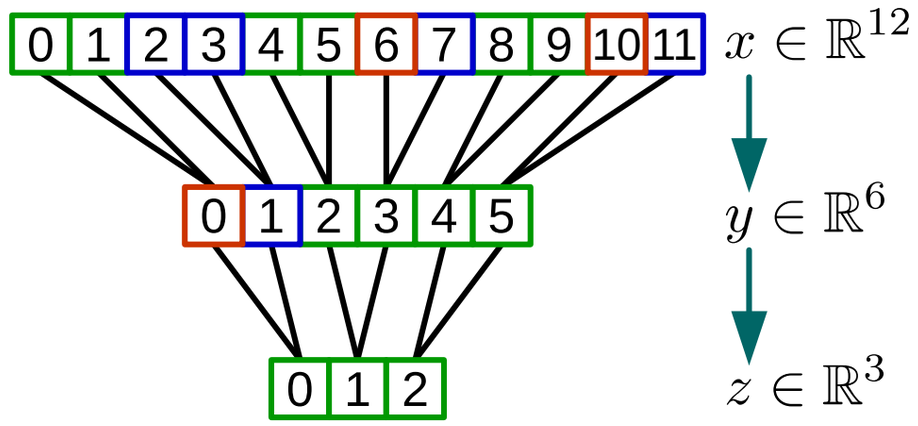
ChebNet: Graph Coarsening & Pooling



Graph coarsening



Graph signal pooling = 1-D pooling



Graph Convolutional Networks (GCN)

- Kipf and Welling (2016) simplified the ChebNet further.
- Choosing a degree-1 polynomial:

$$g^{\theta}(L) = \sum_{k} \theta_k \, T_k(L) \quad \Rightarrow \quad g^{\theta}(L) = \theta(I + D^{-1/2} W D^{-1/2})$$

- Let $\tilde{A} = W + I, \qquad \hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$
- A two-layer GCN becomes

$$Z = \underbrace{\mathrm{softmax}(\hat{A} \overbrace{\mathrm{ReLU}(\hat{A} X \Theta^0)}^{\text{First layer}} \Theta^1)}_{\text{Second layer}}$$

Graph Convolutional Networks (GCN)

- Two-layer GCN [Kipf, Welling 2016]

$$Z = \mathrm{softmax}(\hat{A}\ \mathrm{ReLU}(\hat{A}X\Theta^0)\ \Theta^1)$$

- Practical impact of architecture:
  - Smooth node features across neighbors

Graph Convolutional Networks (GCN)

- Two-layer GCN [Kipf, Welling 2016]

$$Z = \mathrm{softmax}(\hat{A}\ \mathrm{ReLU}(\hat{A}X\Theta^0)\ \Theta^1)$$

- Practical impact of architecture:
  - Smooth node features across neighbors
  - Extract features using learnt weights

Graph Convolutional Networks (GCN)

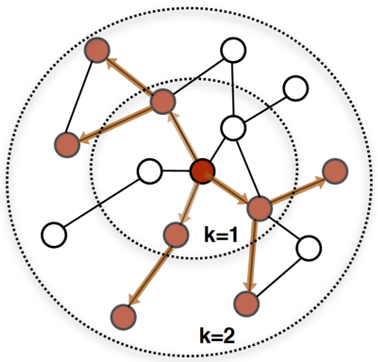
- Two-layer GCN [Kipf, Welling 2016]:

$$Z = \mathrm{softmax}(\hat{A}\,\mathrm{ReLU}(\hat{A}X\Theta^0)\;\Theta^1)$$

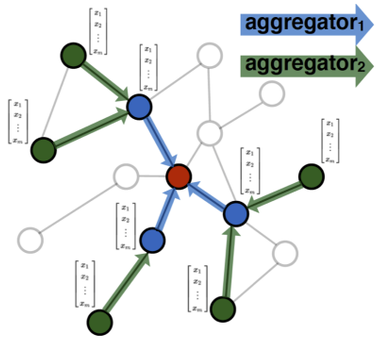
- Practical impact of architecture:
  - Smooth node features across neighbors
  - Extract features using learnt weights
  - Nonlinear activation function

GCN: Many recent extensions

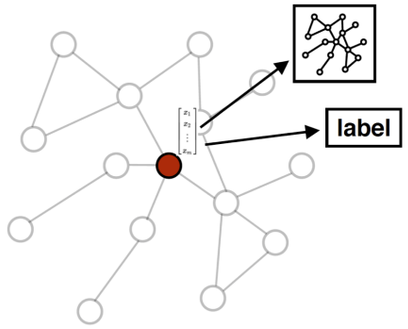
Inductive setting [Hamilton, 2017]



1. Sample neighborhood

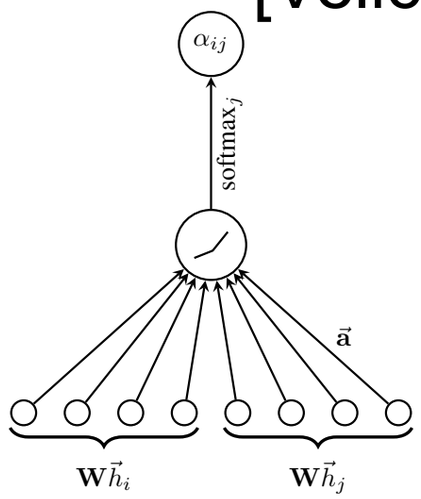


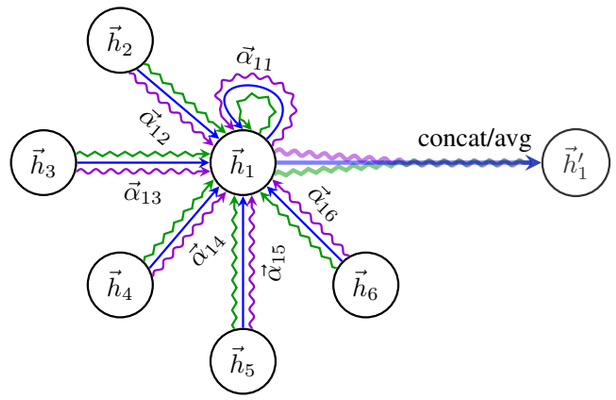
2. Aggregate feature information from neighbors



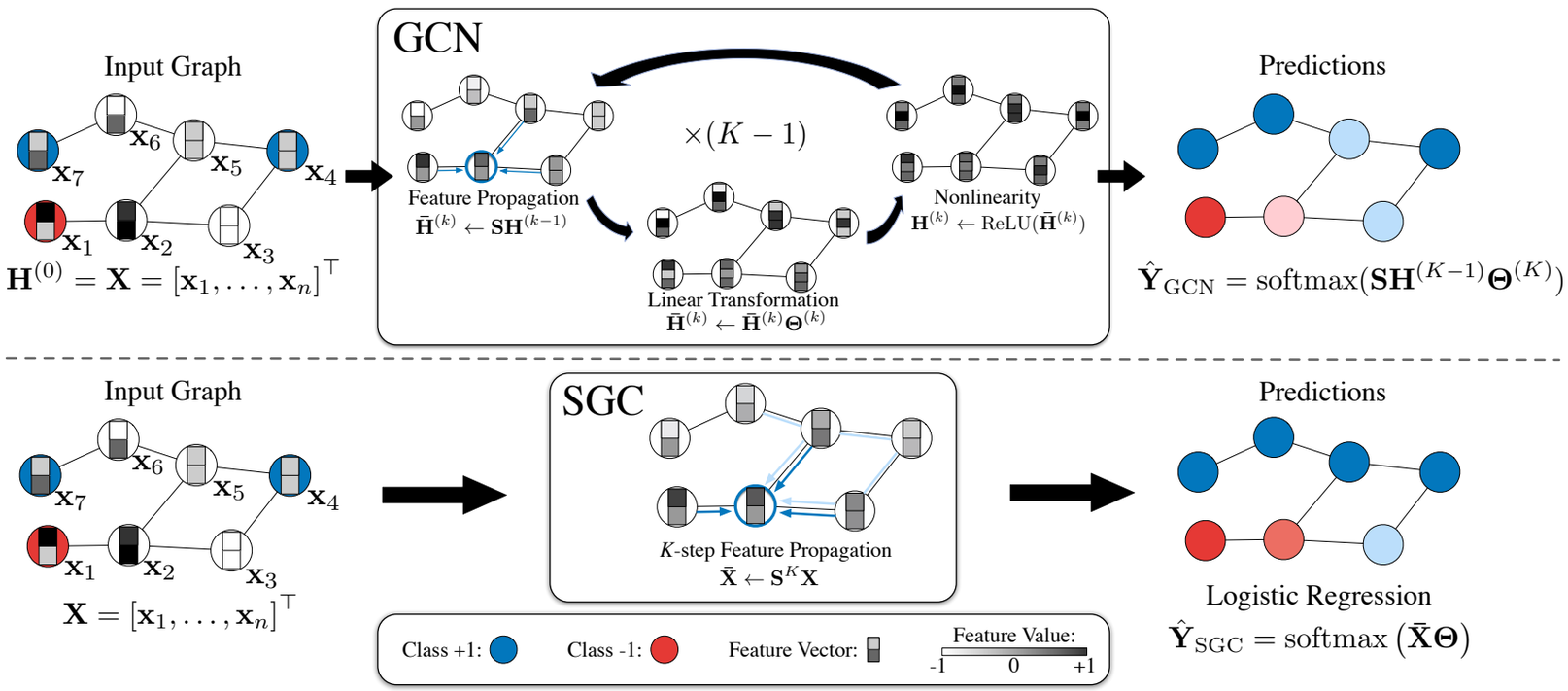
3. Predict graph context and label using aggregated information

Graph attention networks [Velickovic, 2018]





Simpler architectures [Wu, 2019]



Conclusion

- Analysis of data on graph domains
- Irregular topologies ➡ Nontrivial to extend classical techniques
- Perform operations in spectral domain
  - Filtering, Fourier transform
  - Linear signal models, sparse representations
  - Nonlinear models, graph neural networks
- Challenges, new directions
  - Big data, large graphs ➡ Approximate models are necessary
  - Generalizability: Unseen graphs, unseen nodes, dissimilar input topologies, …