COMP4222 Machine Learning with Structured Data

Graph Neural Networks 2

Instructor: Yangqiu Song

Slides credits: Yao Ma and Yiqi Wang, Tyler Derr, Lingfei Wu and Tengfei Ma

Overview

Foundations

Basic Graph Theory

Spectral Graph Theory

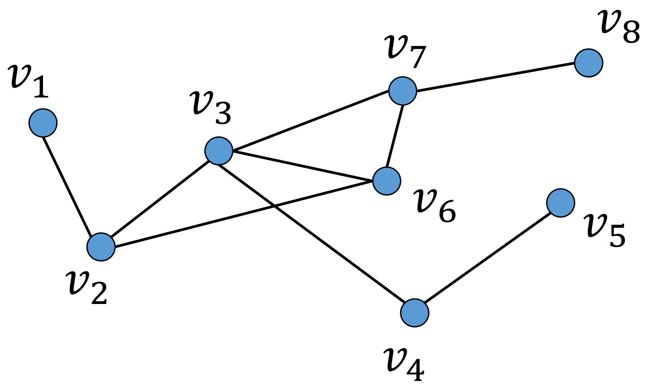
Graph Fourier Analysis

Models

Filtering Layers in GNN

Pooling Layers in GNN

Graphs and Graph Signals

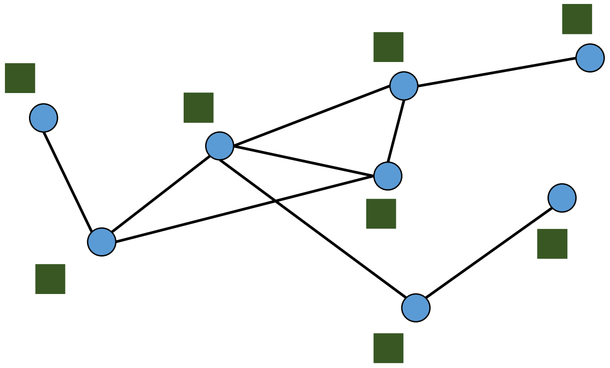


$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

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

Graphs and Graph Signals



Graph Signal: $f : \mathcal{V} \rightarrow \mathbb{R}^N$

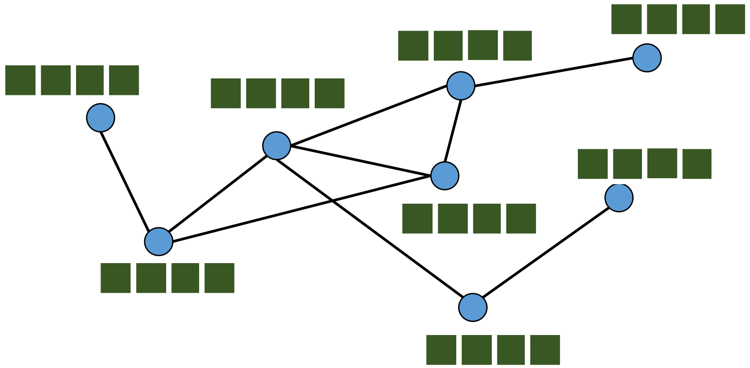
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

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

$$\mathcal{V} \longrightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

Graphs and Graph Signals



Graph Signal: $f : \mathcal{V} \rightarrow \mathbb{R}^{N \times d}$

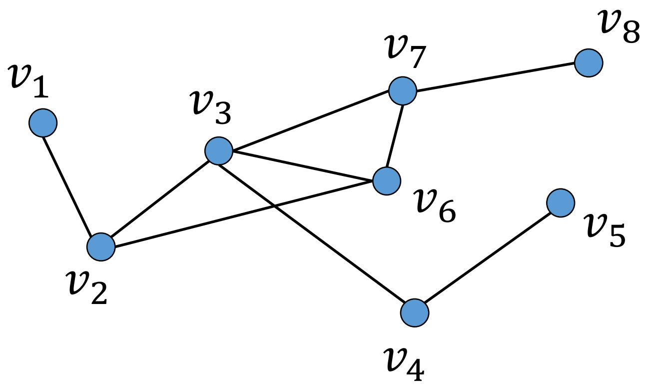
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

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

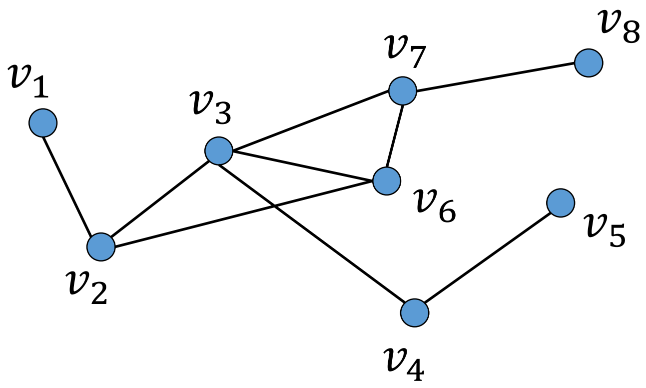
$$\mathcal{V} \longrightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

Matrix Representations of Graphs



Spectral graph theory. American Mathematical Soc.; 1997.

Matrix Representations of Graphs



Adjacency Matrix: $A[i,j] = 1$ if v_i is adjacent to v_j
$A[i,j] = 0$, otherwise

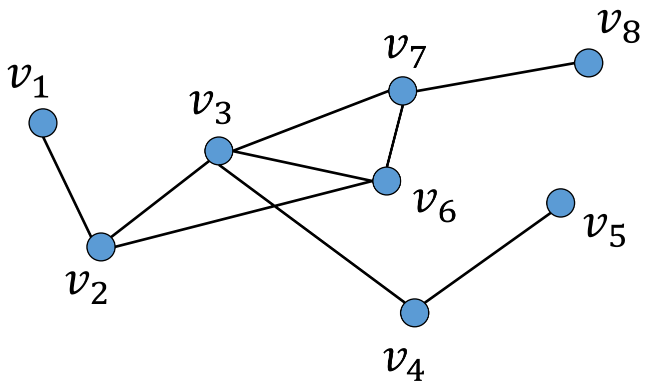
Adjacency Matrix

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

$\boldsymbol{A}$

Spectral graph theory. American Mathematical Soc.; 1997.

Matrix Representations of Graphs



Adjacency Matrix: $A[i, j] = 1$ if v_i is adjacent to v_j
$A[i, j] = 0$, otherwise

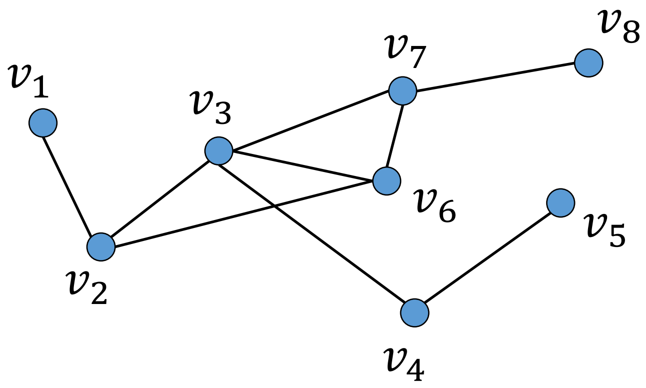
Degree Matrix: $\mathbf{D} = \mathrm{diag}(degree(v_1), \ldots, degree(v_N))$

Degree Matrix — Adjacency Matrix

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{D}} - \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}}_{\boldsymbol{A}}$$

Spectral graph theory. American Mathematical Soc.; 1997.

Matrix Representations of Graphs



Adjacency Matrix: $A[i, j] = 1$ if v_i is adjacent to v_j
$A[i, j] = 0$, otherwise

Degree Matrix: $\mathbf{D} = \mathrm{diag}(degree(v_1), \ldots, degree(v_N))$

Degree Matrix — Adjacency Matrix — Laplacian Matrix

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

Spectral graph theory. American Mathematical Soc.; 1997.

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$

Laplacian quadratic form:

$$\mathbf{f}^T \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{N} \mathbf{A}[i,j](\mathbf{f}(i) - \mathbf{f}(j))^2$$

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$

Laplacian quadratic form:

$$\mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{N} \mathbf{A}[i,j](\mathbf{f}(i) - \mathbf{f}(j))^2$$

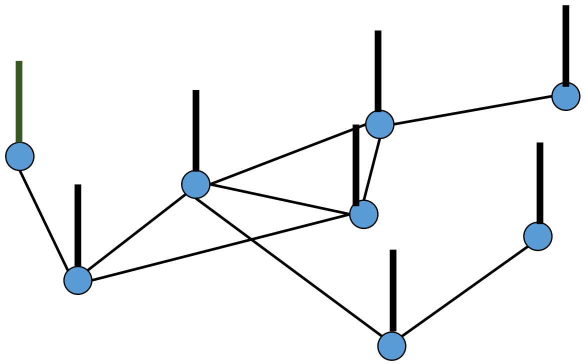
"Smoothness" or "Frequency" of the signal f

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$



Low frequency graph signal

Laplacian quadratic form:

$$\underline{\mathbf{f}^T \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{N} \mathbf{A}[i,j](\mathbf{f}(i) - \mathbf{f}(j))^2}$$

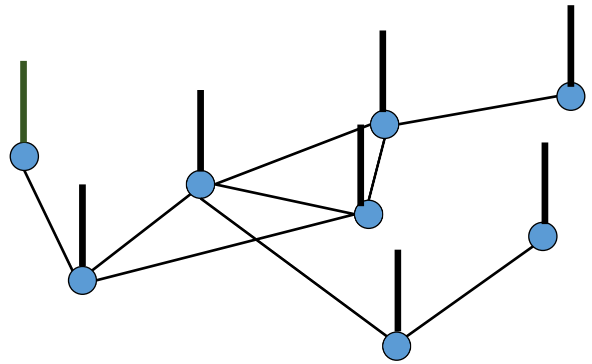
"Smoothness" or "Frequency" of the signal f

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$

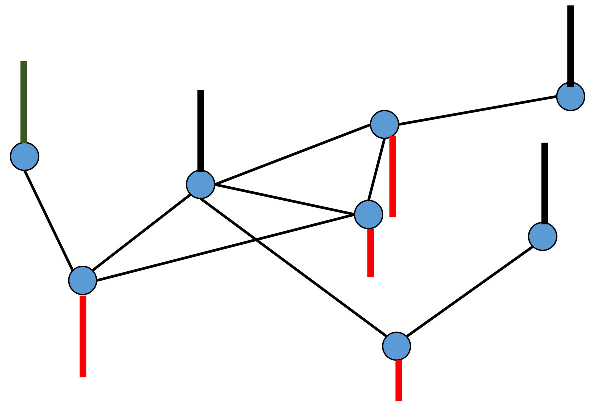


Low frequency graph signal

Laplacian quadratic form:

$$\underline{\mathbf{f}^T\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{N} \mathbf{A}[i,j](\mathbf{f}(i) - \mathbf{f}(j))^2}$$

"Smoothness" or "Frequency" of the signal f



High frequency graph signal

Eigen-decomposition of Laplacian Matrix

Laplacian matrix has a complete set of orthonormal eigenvectors:

$$\mathbf{L} = \begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}$$

Eigen-decomposition of Laplacian Matrix

Laplacian matrix has a complete set of orthonormal eigenvectors:

$$\mathbf{L} = \underbrace{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}}_{\boldsymbol{U}^{\boldsymbol{T}}}$$

$$\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U} = \boldsymbol{U}^{-\mathbf{1}}\boldsymbol{U} = \boldsymbol{I}$$

Here we assume that the graph is undirected, and L is symmetric

$$\boldsymbol{U}\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda} \leftrightarrow \boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-\mathbf{1}}$$

Eigen-decomposition of Laplacian Matrix

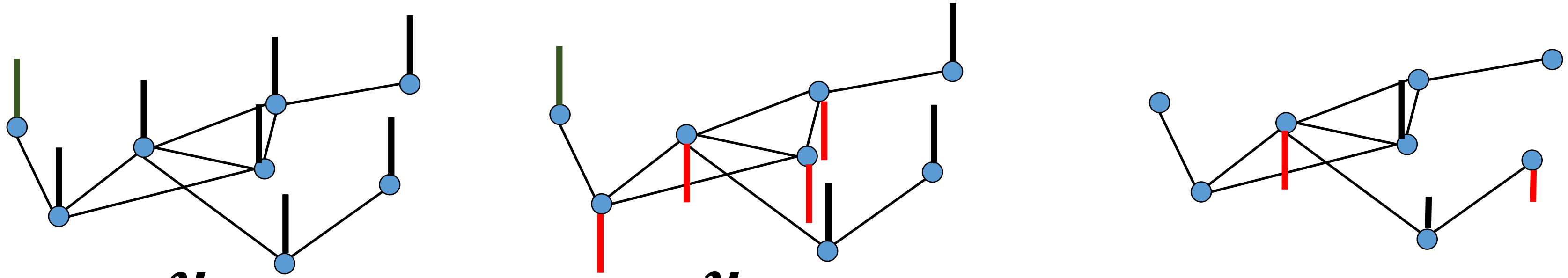
Laplacian matrix has a complete set of orthonormal eigenvectors:

$$\mathbf{L} = \underbrace{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}}_{\boldsymbol{U^T}}$$

Eigenvalues are sorted non-decreasingly:

$$0 = \lambda_0 < \lambda_1 \leq \cdots \lambda_{N-1}$$

Eigenvectors as Graph Signals



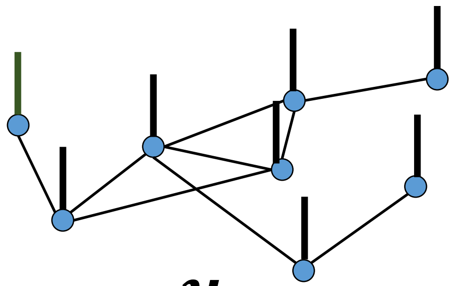
$\boldsymbol{u}_0$ $\boldsymbol{u}_1$ $\boldsymbol{u}_8$

Eigenvectors as Graph Signals

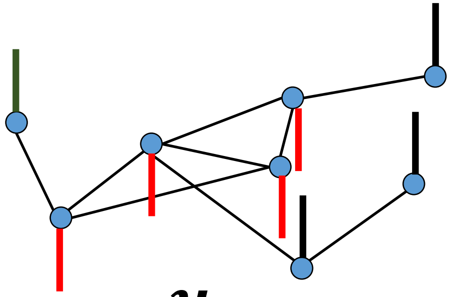
The frequency of an eigenvector of Laplacian matrix is its corresponding eigenvalue:

$$\underline{\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i} = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i$$

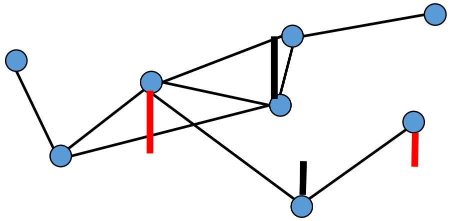
Frequency of the signal $\boldsymbol{u}_i$



$\boldsymbol{u}_0$



$\boldsymbol{u}_1$



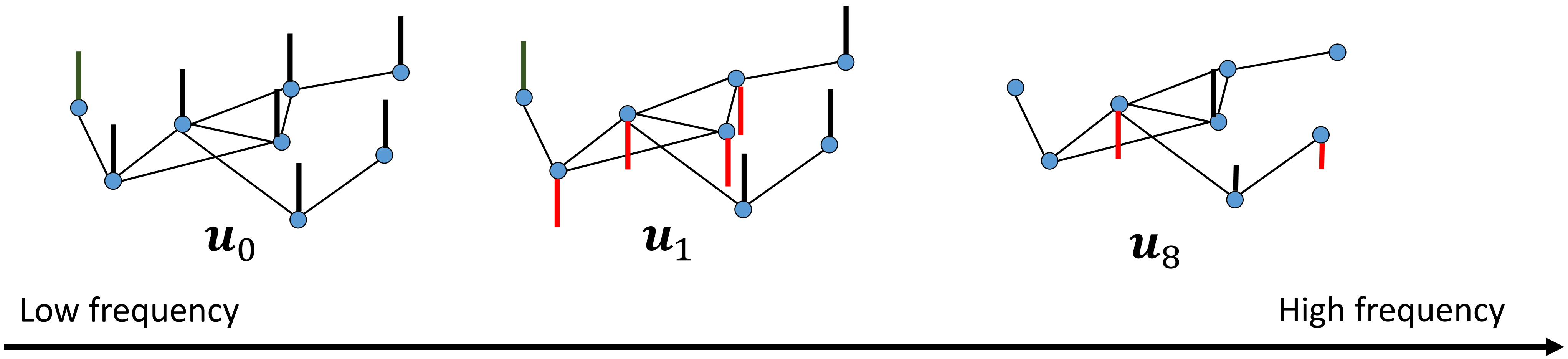
$\boldsymbol{u}_8$

$\boldsymbol{u}_0^T \boldsymbol{L} \boldsymbol{u}_0 = \boldsymbol{\lambda}_0 = \mathbf{0}$ $\boldsymbol{u}_1^T \boldsymbol{L} \boldsymbol{u}_1 = \boldsymbol{\lambda}_1$ $\boldsymbol{u}_8^T \boldsymbol{L} \boldsymbol{u}_8 = \boldsymbol{\lambda}_8$

Eigenvectors as Graph Signals

The frequency of an eigenvector of Laplacian matrix is its corresponding eigenvalue:

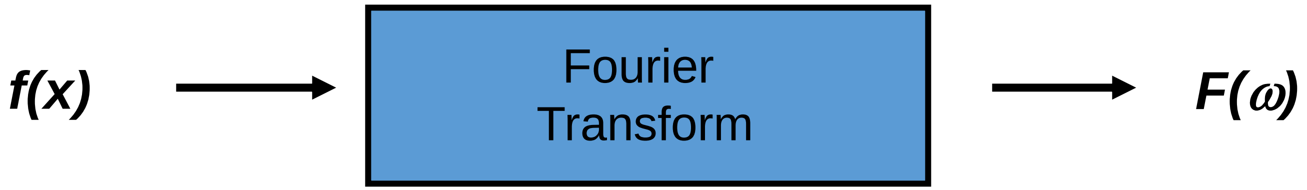
$$\underbrace{\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i}_{\text{Frequency of the signal } \boldsymbol{u}_i} = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i$$



$\boldsymbol{u}_0^T \boldsymbol{L} \boldsymbol{u}_0 = \boldsymbol{\lambda}_0 = \mathbf{0}$ $\quad$ $\boldsymbol{u}_1^T \boldsymbol{L} \boldsymbol{u}_1 = \boldsymbol{\lambda}_1$ $\quad$ $\boldsymbol{u}_8^T \boldsymbol{L} \boldsymbol{u}_8 = \boldsymbol{\lambda}_8$

Fourier Transform

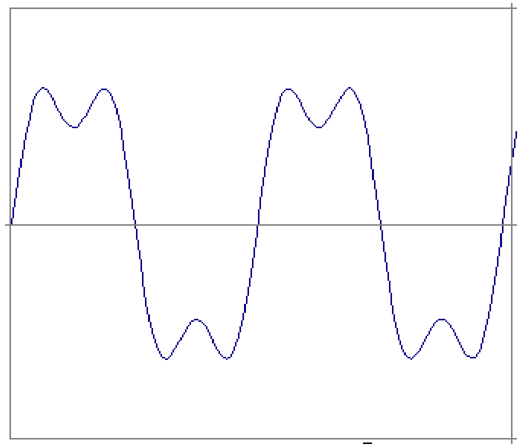
- We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:



- For every ω from 0 to inf, $\boldsymbol{F(\omega)}$ holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x+\phi)$

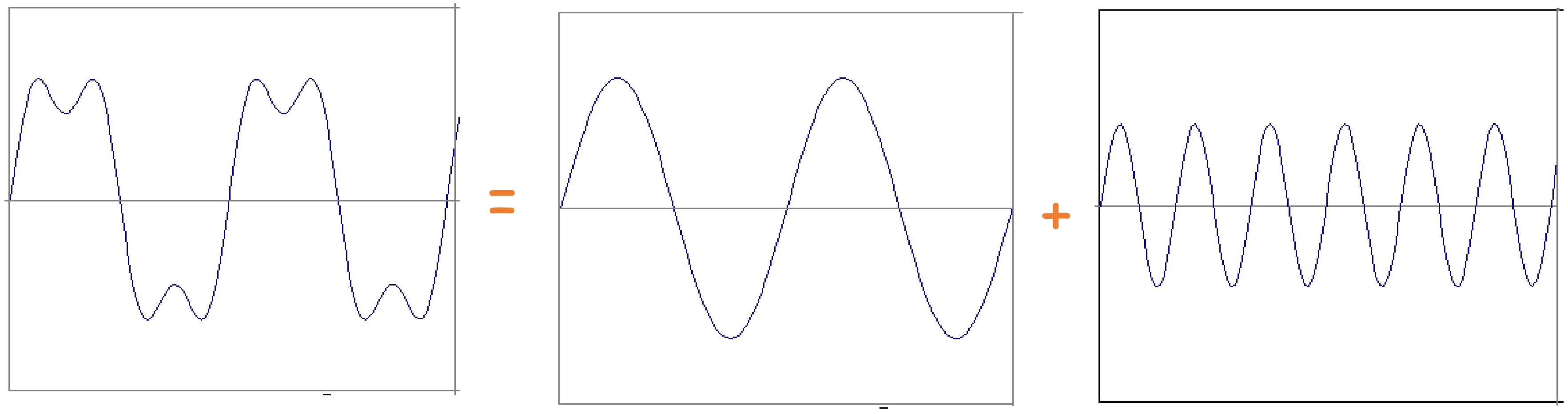
Time and Frequency

- example : $g(t) = \sin(2pi\, f\, t) + (1/3)\sin(2pi\, (3f)\, t)$



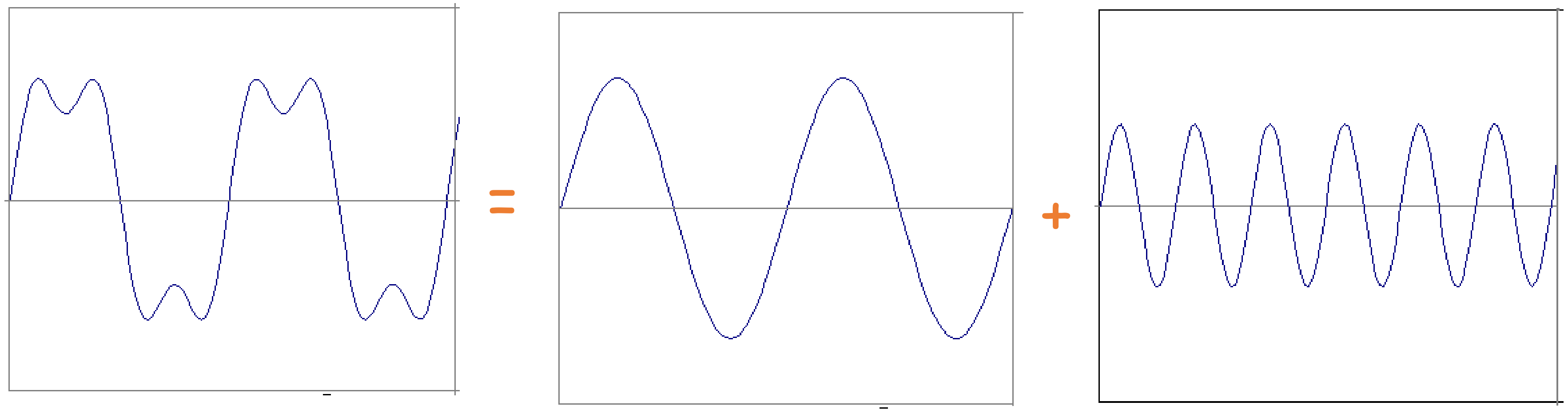
Time and Frequency

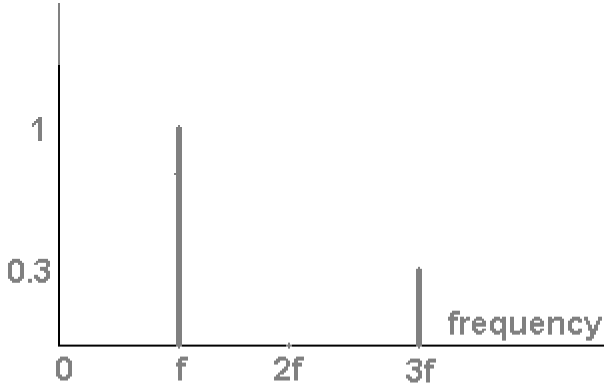
- example : $g(t) = \sin(2pi\, f\, t) + (1/3)\sin(2pi\, (3f)\, t)$



Frequency Spectra

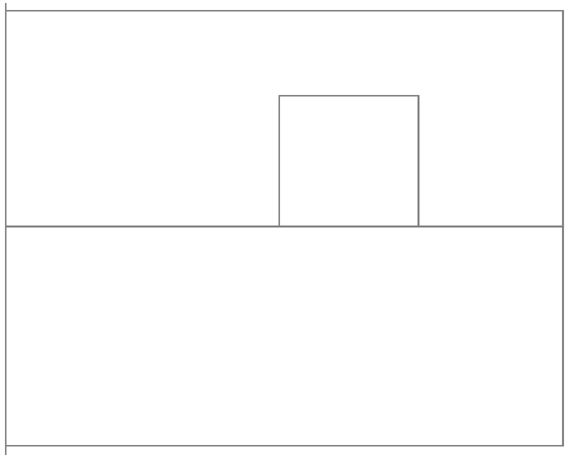
- example : $g(t) = \sin(2pi\ f\ t) + (1/3)\sin(2pi\ (3f)\ t)$



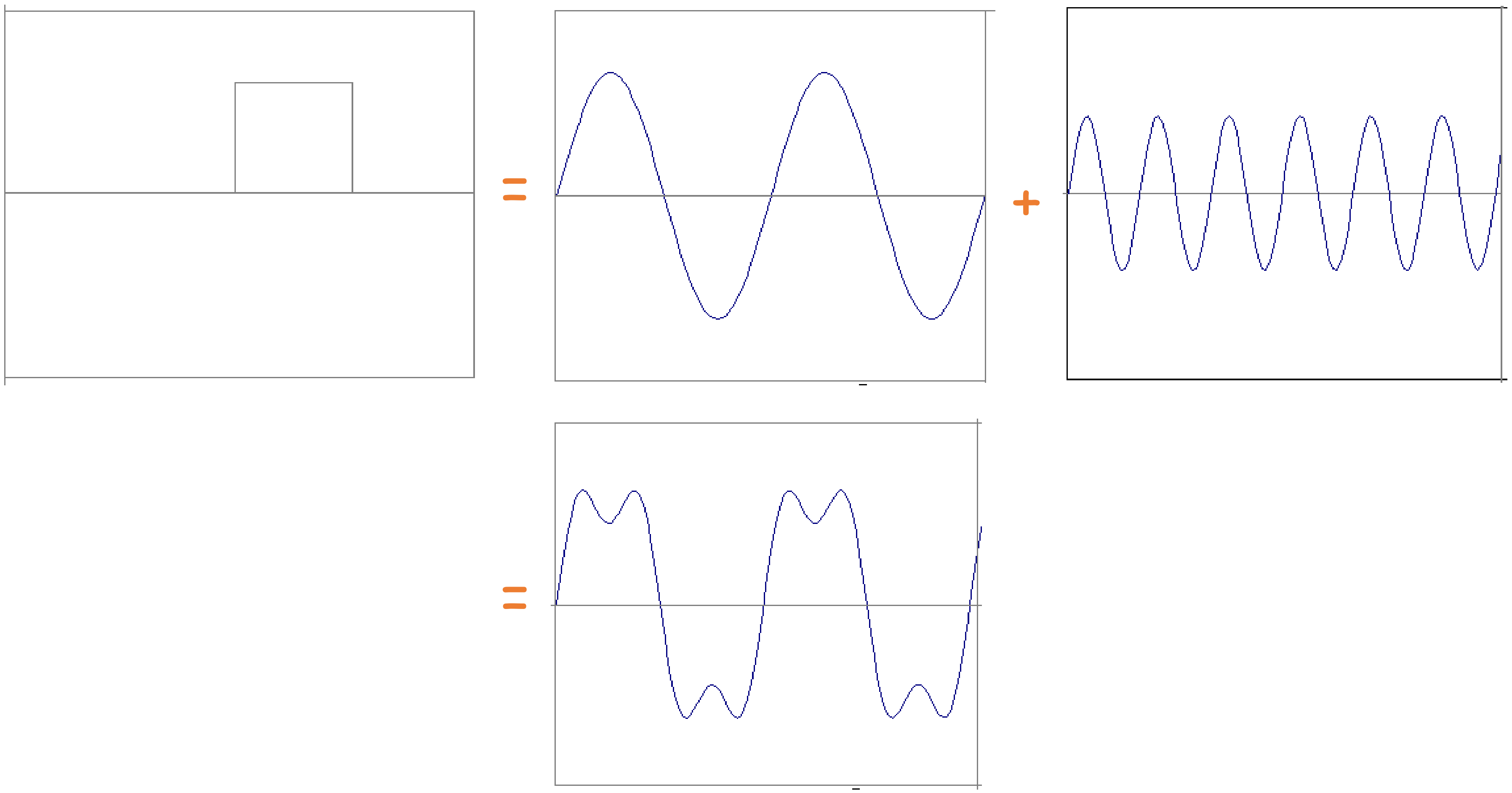


Frequency Spectra

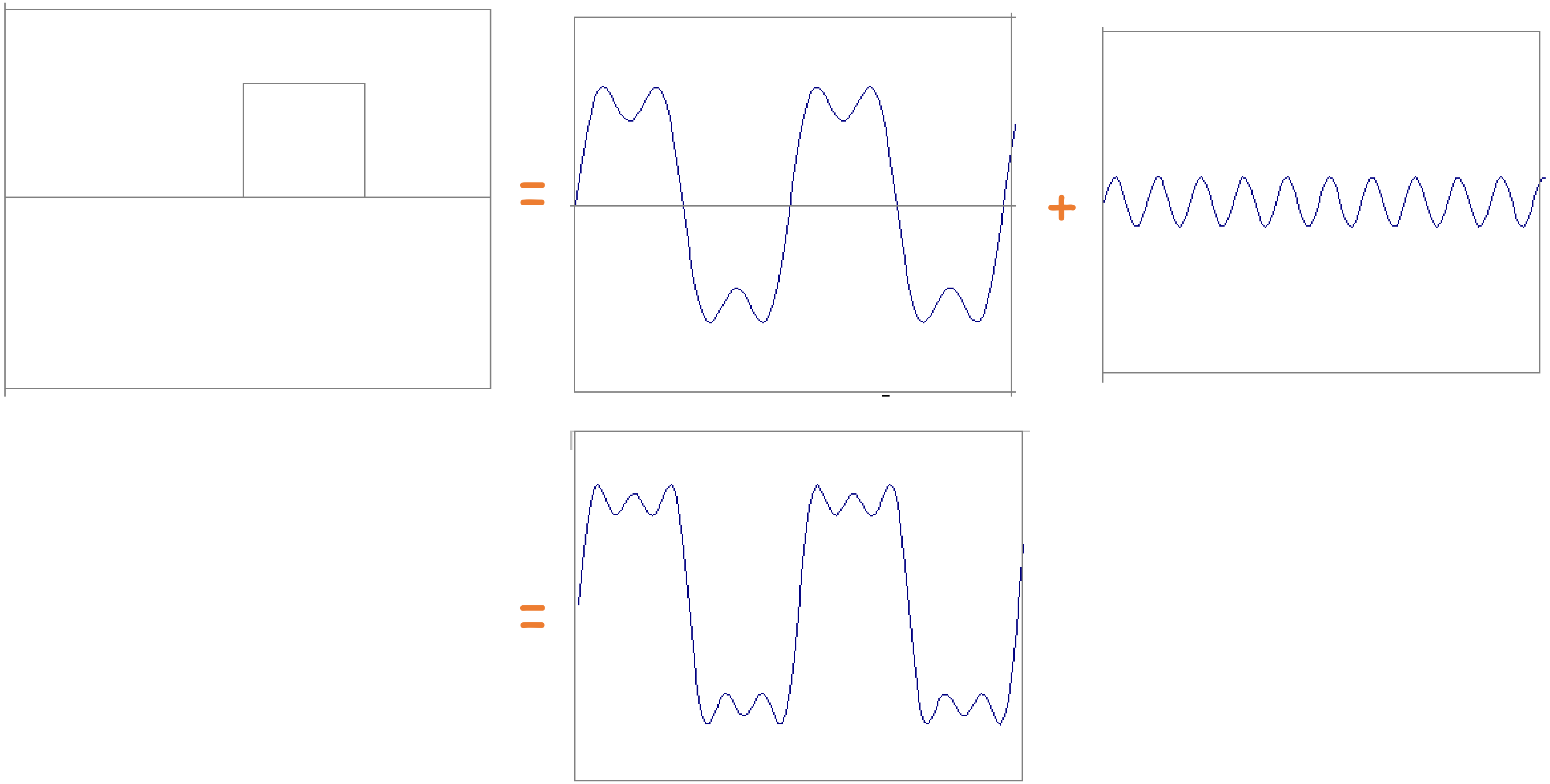
- Usually, frequency is more interesting than the phase



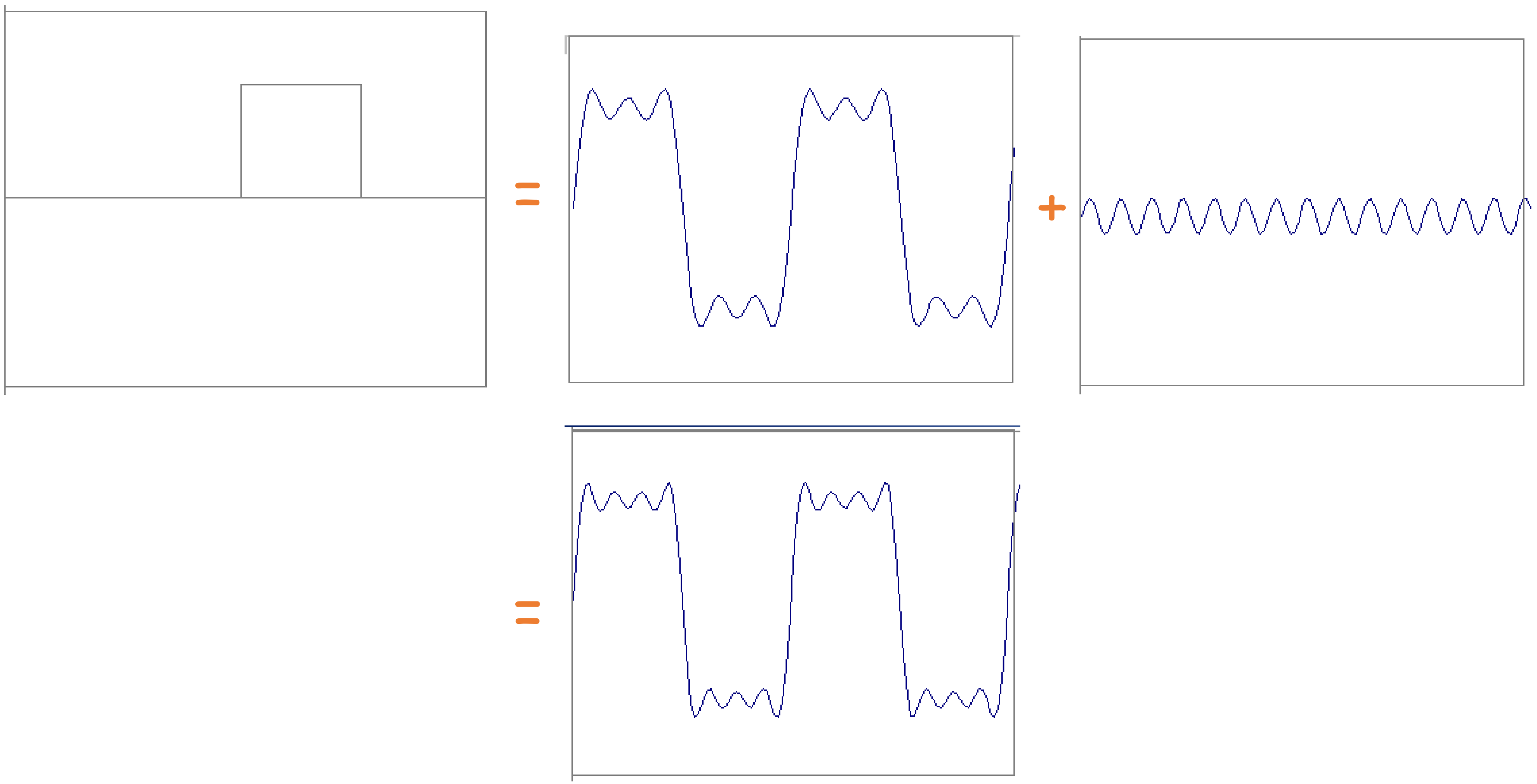
Frequency Spectra



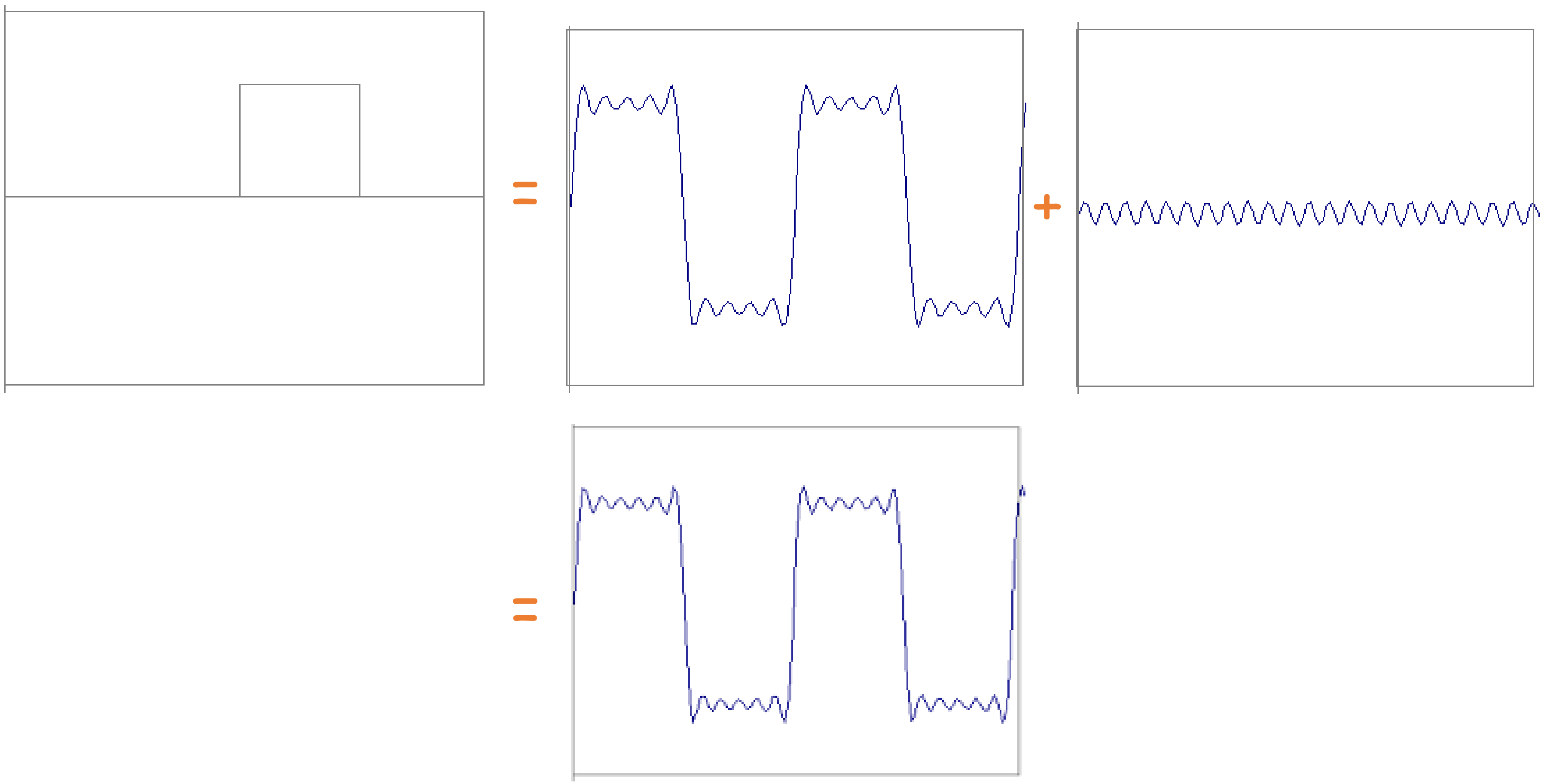
Frequency Spectra



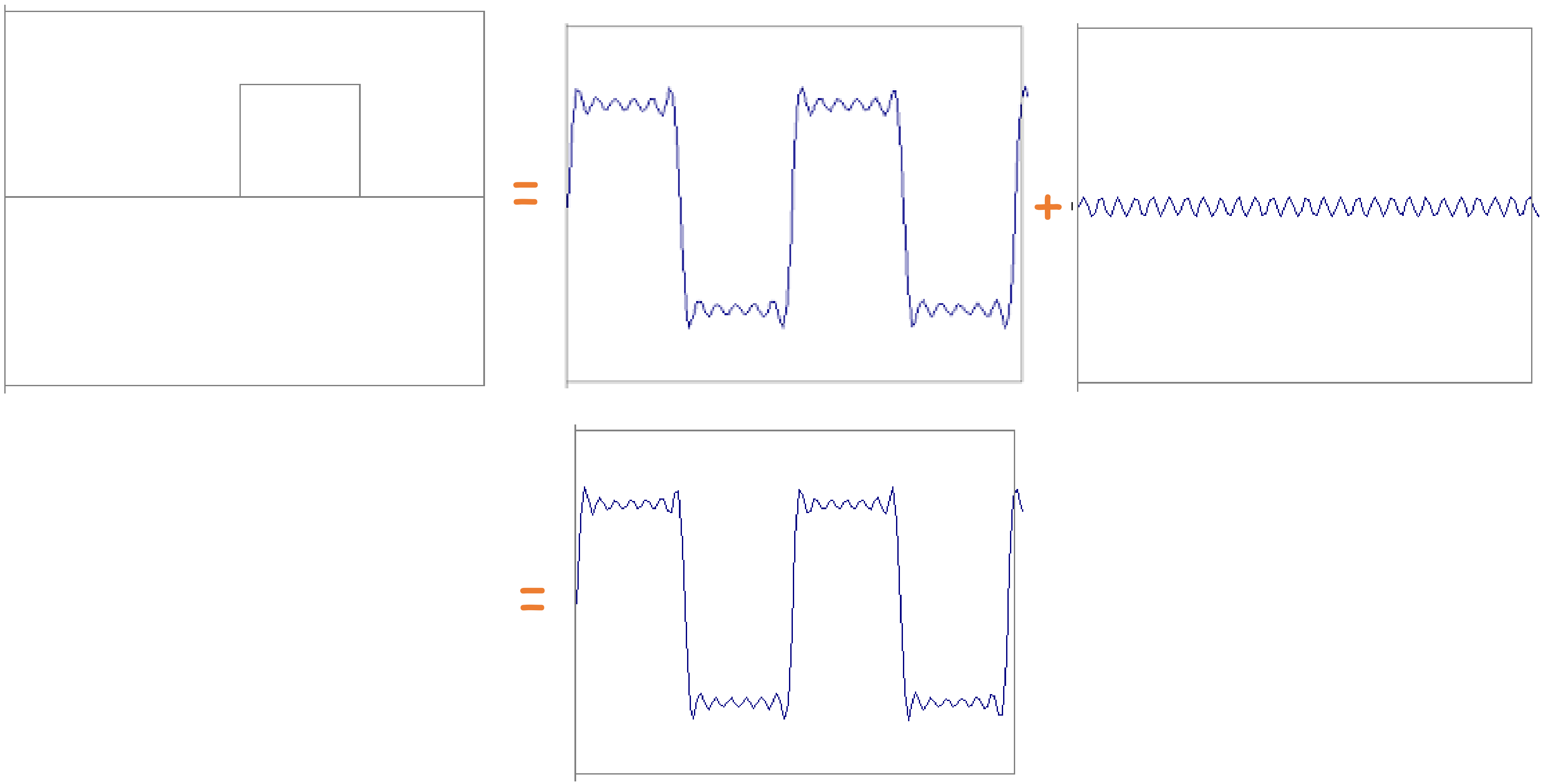
Frequency Spectra



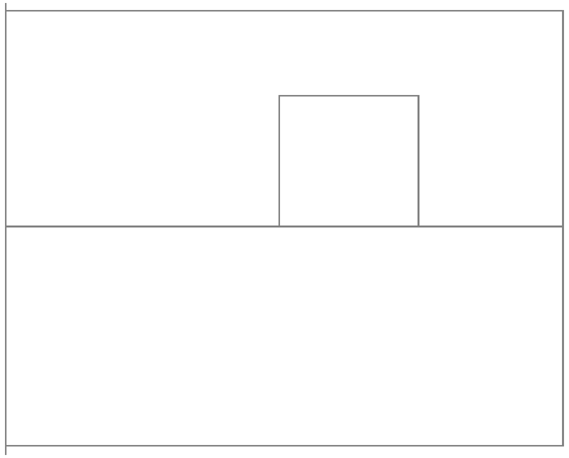
Frequency Spectra



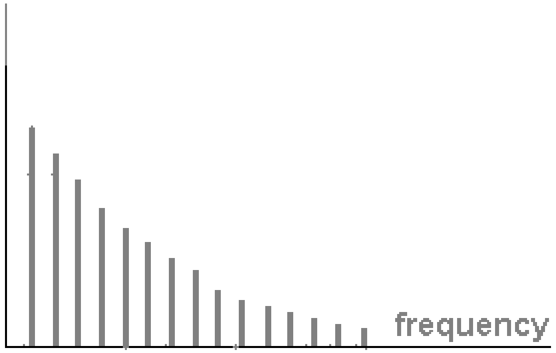
Frequency Spectra



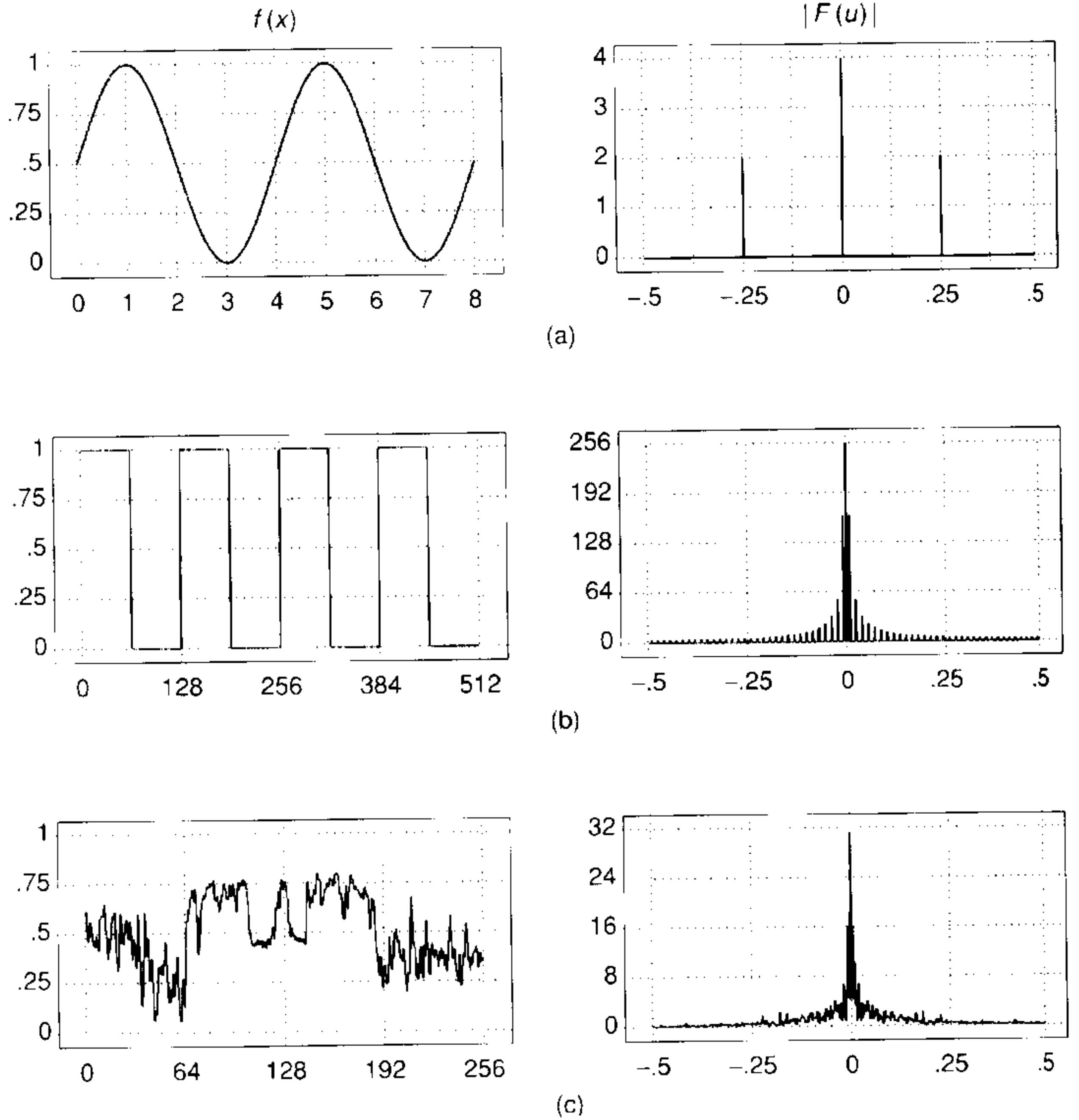
Frequency Spectra



$$= \boxed{A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi k t)}$$



Frequency Spectra



Graph Fourier Transform(GFT)

A signal f can be written as graph Fourier series:

$$f = \sum_{i=0}^{N-1} \hat{f}_i \cdot u_i$$

u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients

Graph Fourier Transform(GFT)

A signal f can be written as graph Fourier series:

$$f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients

$$\mathbf{L} = \underbrace{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}}_{\boldsymbol{U}^T}$$

Graph Fourier Transform(GFT)

A signal f can be written as graph Fourier series:

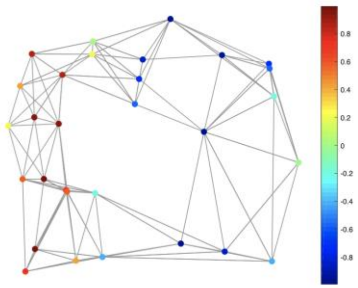
$$f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix}$$

u_i: graph Fourier mode

λ_i: frequency

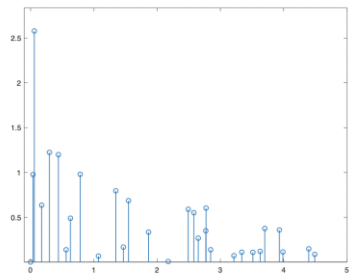
$\hat{f}_i$: graph Fourier coefficients



Spatial domain: $\boldsymbol{f}$

$$\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$$

Decompose signal f



Spectral domain: $\hat{\boldsymbol{f}}$

The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*

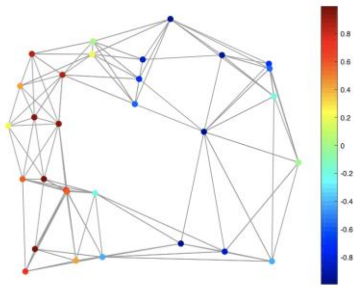
$$UL = U\Lambda \leftrightarrow L = U\Lambda U^{-1} \qquad U^T U = U^{-1} U = I$$

Inverse Graph Fourier Transform (IGFT)

A signal f can be written as graph Fourier series:

$$f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

u_i: graph Fourier mode $\qquad$ λ_i: frequency $\qquad$ $\hat{f}_i$: graph Fourier coefficients

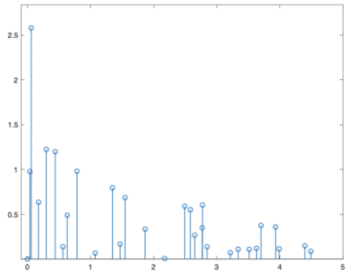


Spatial domain: $\boldsymbol{f}$

$$\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$$

Reconstruct signal $\boldsymbol{f}$

$$\boldsymbol{U}^T\boldsymbol{U} \leftrightarrow \boldsymbol{U} = {\boldsymbol{U}^T}^{-1}$$



Spectral domain: $\hat{\boldsymbol{f}}$

Overview

Foundations

Basic Graph Theory

Spectral Graph Theory

Graph Fourier Analysis

Models

Filtering Layers in GNN

Pooling Layers in GNN

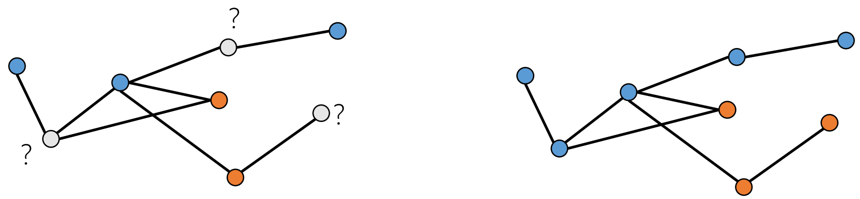
Tasks on Graph-Structured Data

Node-level

Link Prediction

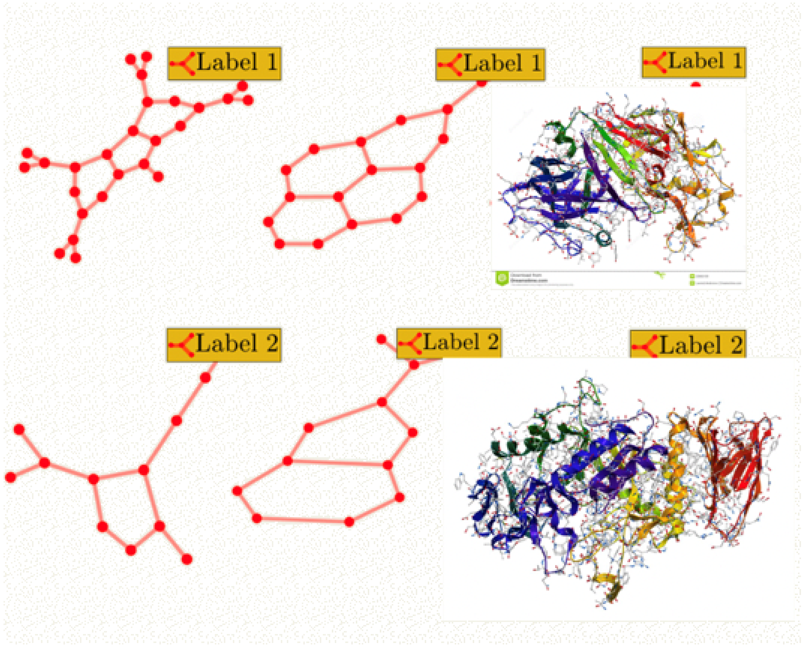
t t+a

Node Classification



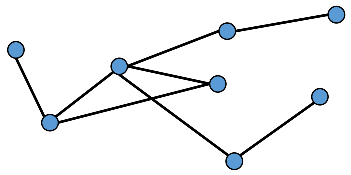
Graph-level

Graph Classification



Tasks on Graph-Structured Data

Node-level

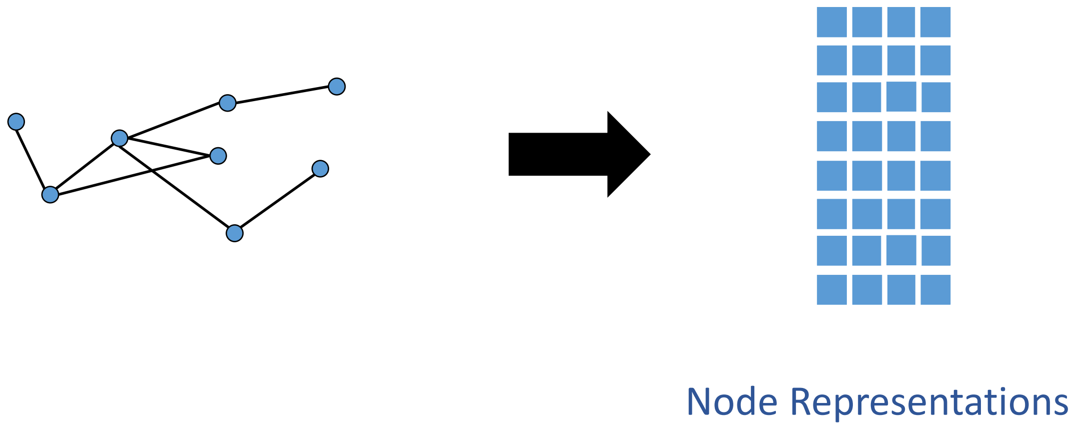


Graph-level

Tasks on Graph-Structured Data

Node-level

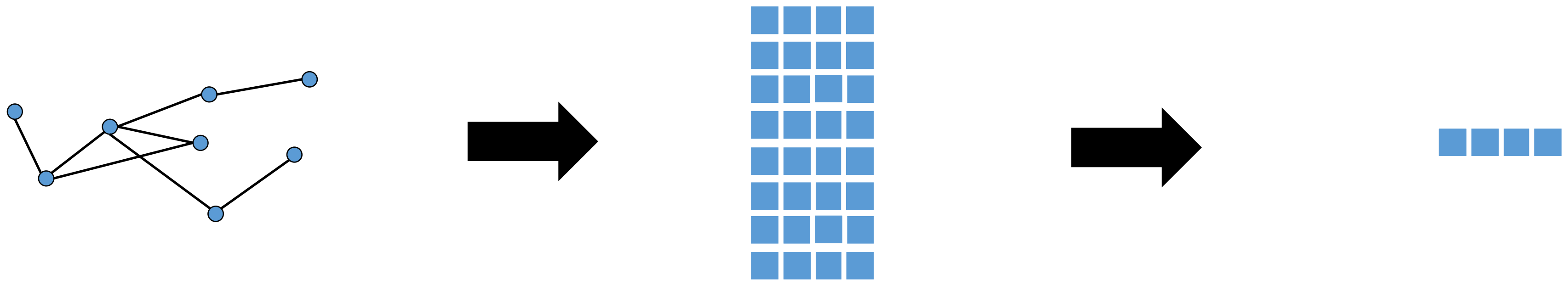
Graph-level



Tasks on Graph-Structured Data

Node-level

Graph-level



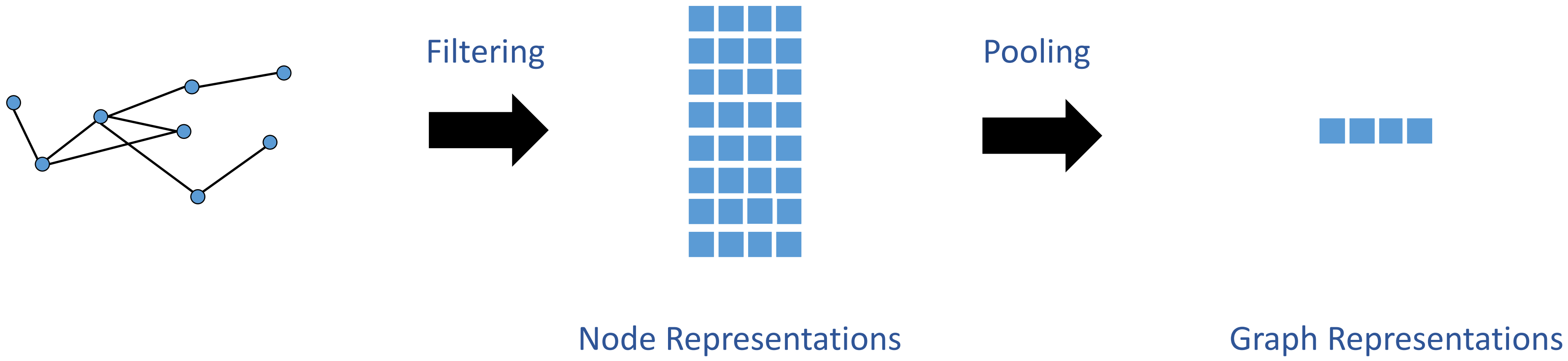
Node Representations

Graph Representation

Tasks on Graph-Structured Data

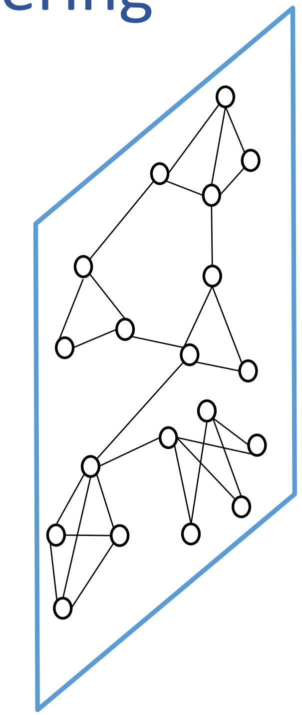
Node-level

Graph-level

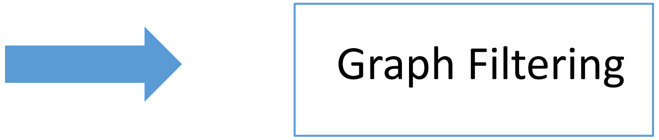


Two Main Operations in GNN

Graph Filtering

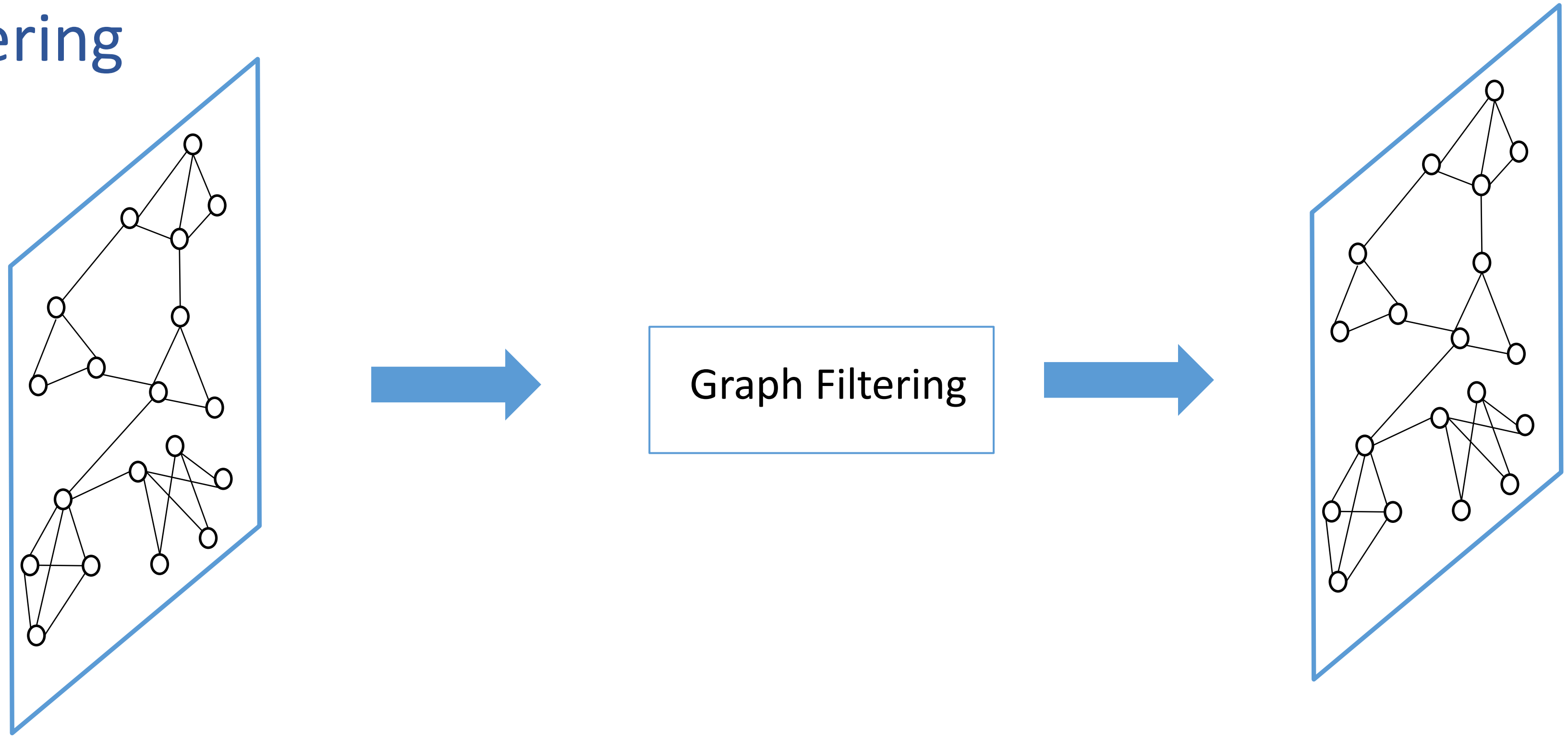


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$



Two Main Operations in GNN

Graph Filtering

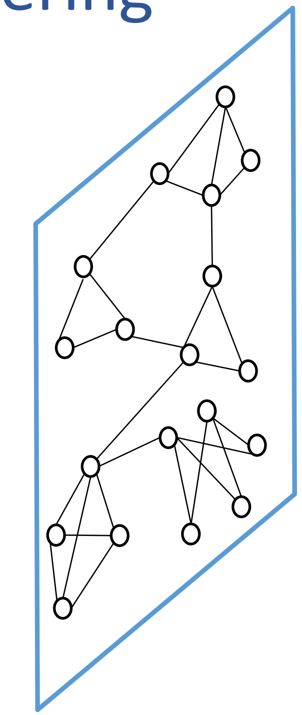


$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

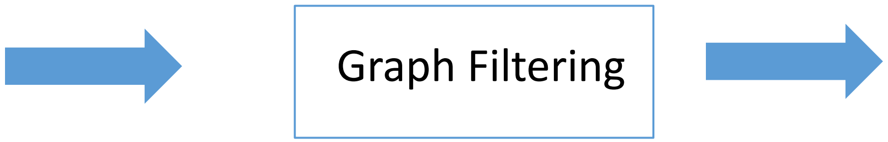
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X}_f \in \mathbb{R}^{n \times d_{new}}$

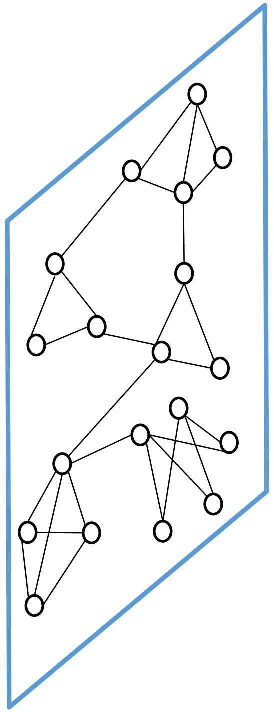
Two Main Operations in GNN

Graph Filtering



$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{X} \in \mathbb{R}^{n\times d}$



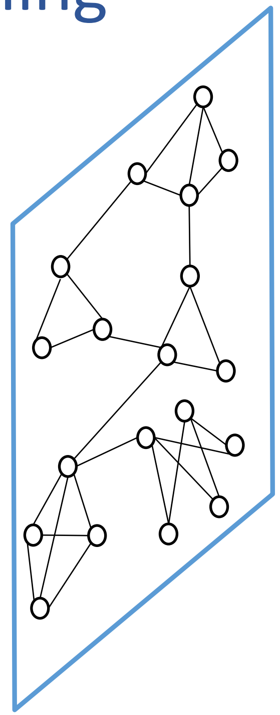


$\mathbf{A} \in \{0,1\}^{n\times n}, \underline{\mathbf{X}_f \in \mathbb{R}^{n\times d_{new}}}$

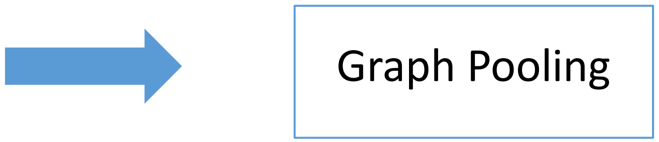
Graph filtering refines the node features

Two Main Operations in GNN

Graph Pooling

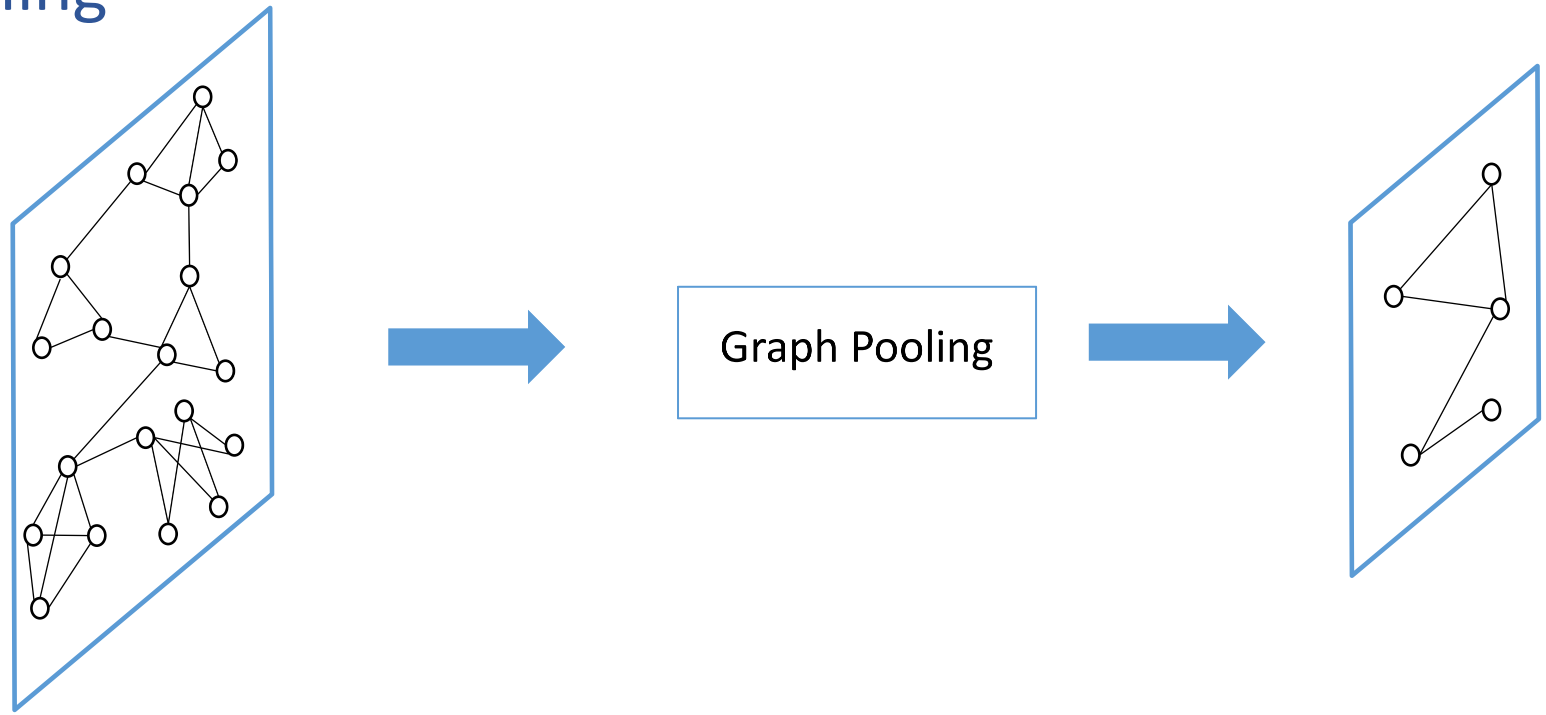


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$



Two Main Operations in GNN

Graph Pooling

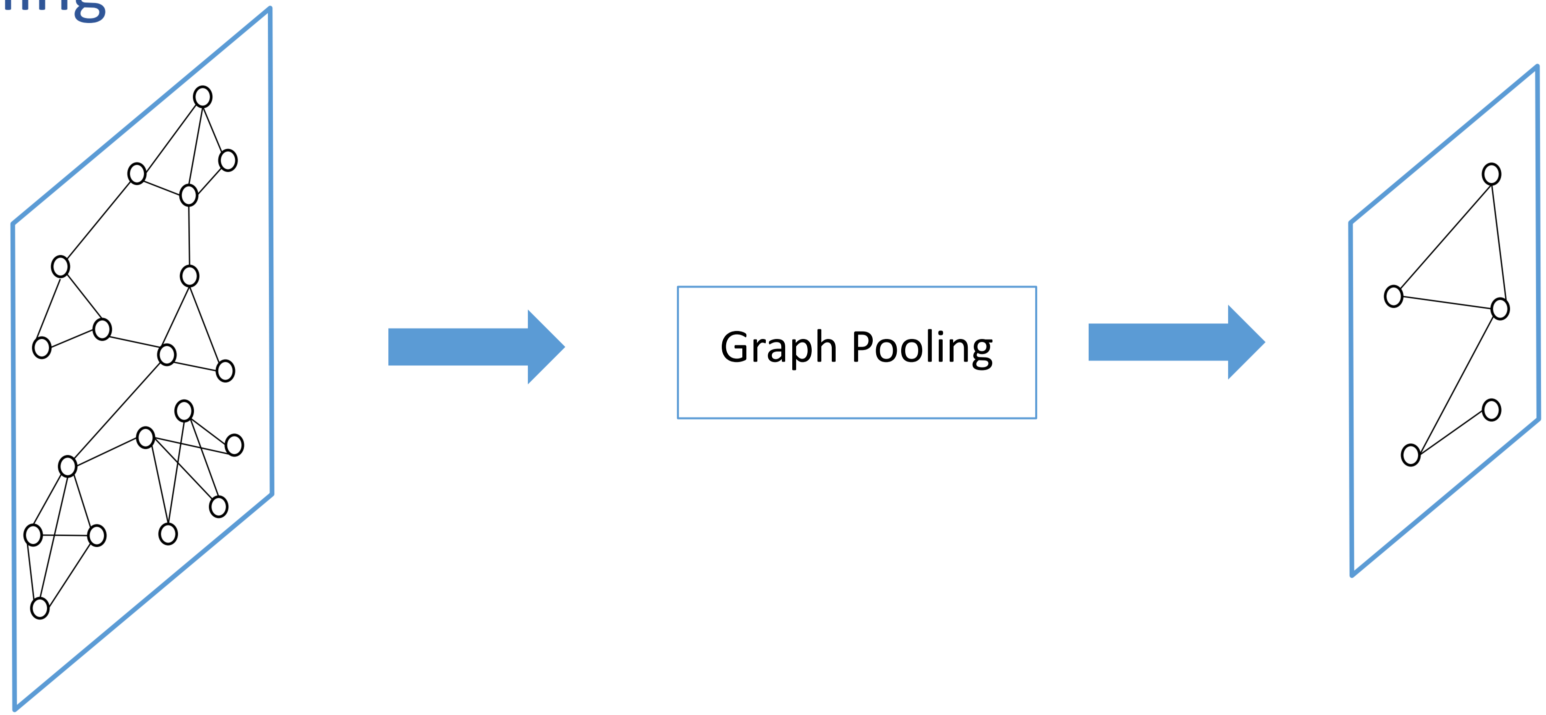


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Two Main Operations in GNN

Graph Pooling



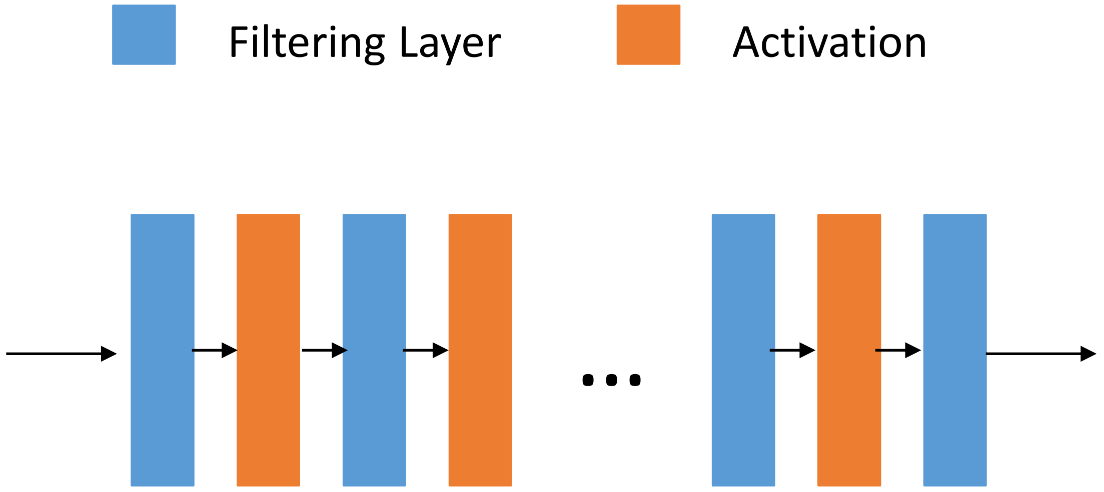
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$

Graph pooling generates a smaller graph

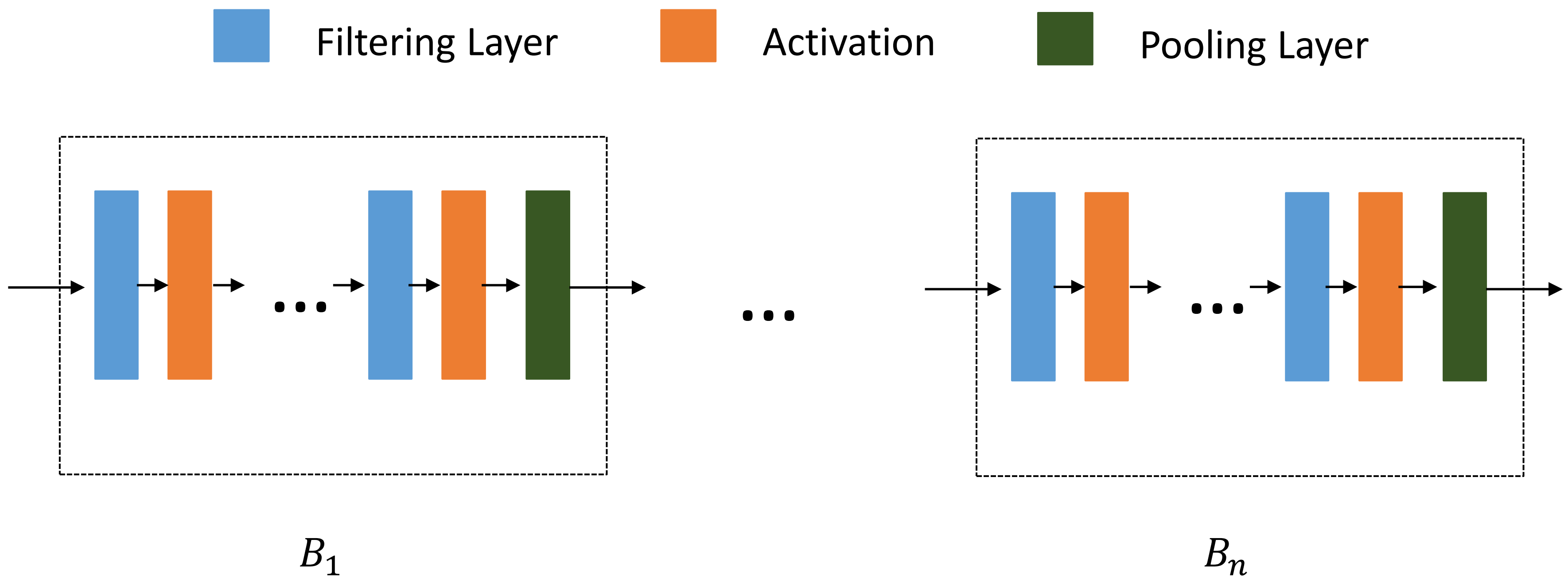
General GNN Framework

For node-level tasks

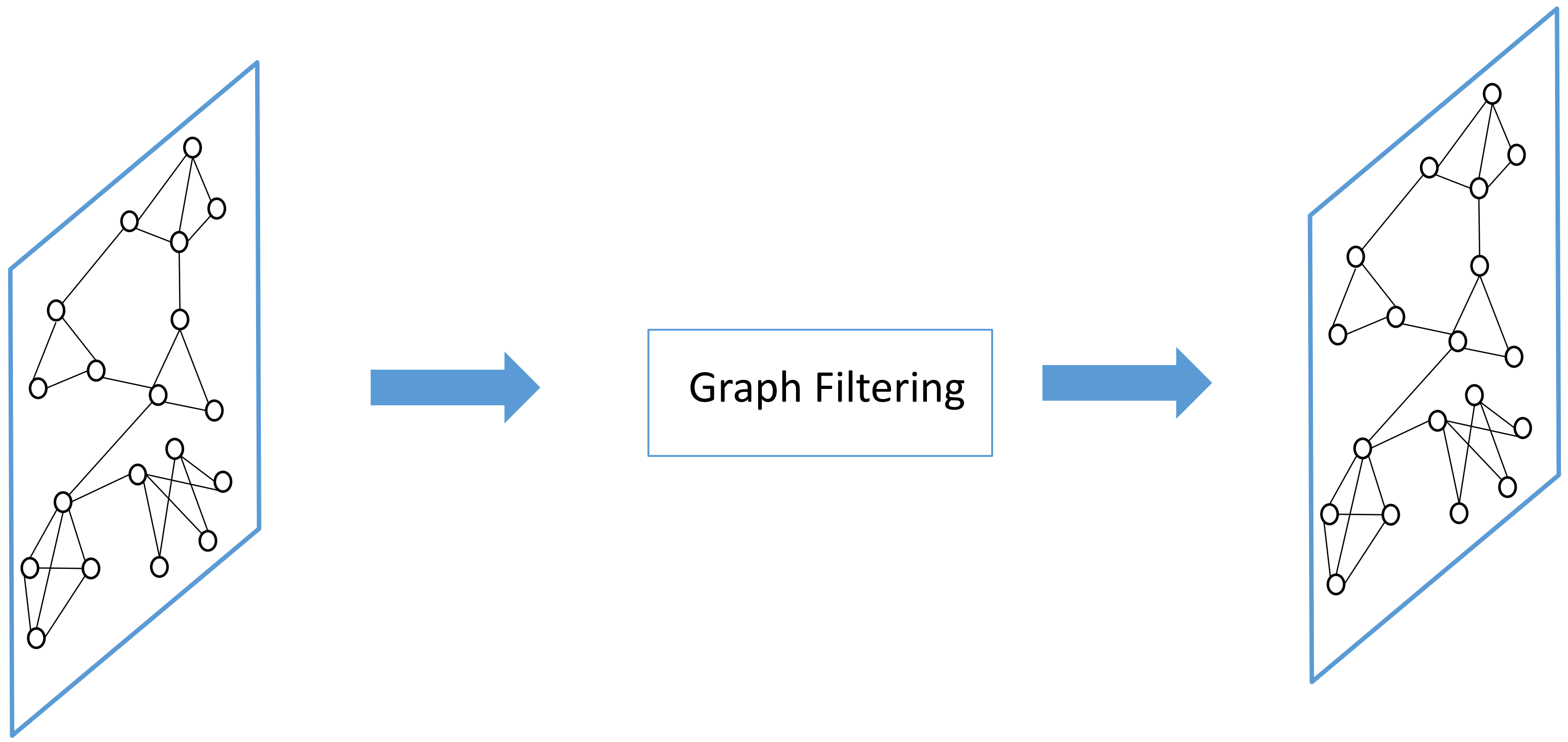


General GNN Framework

For graph-level tasks



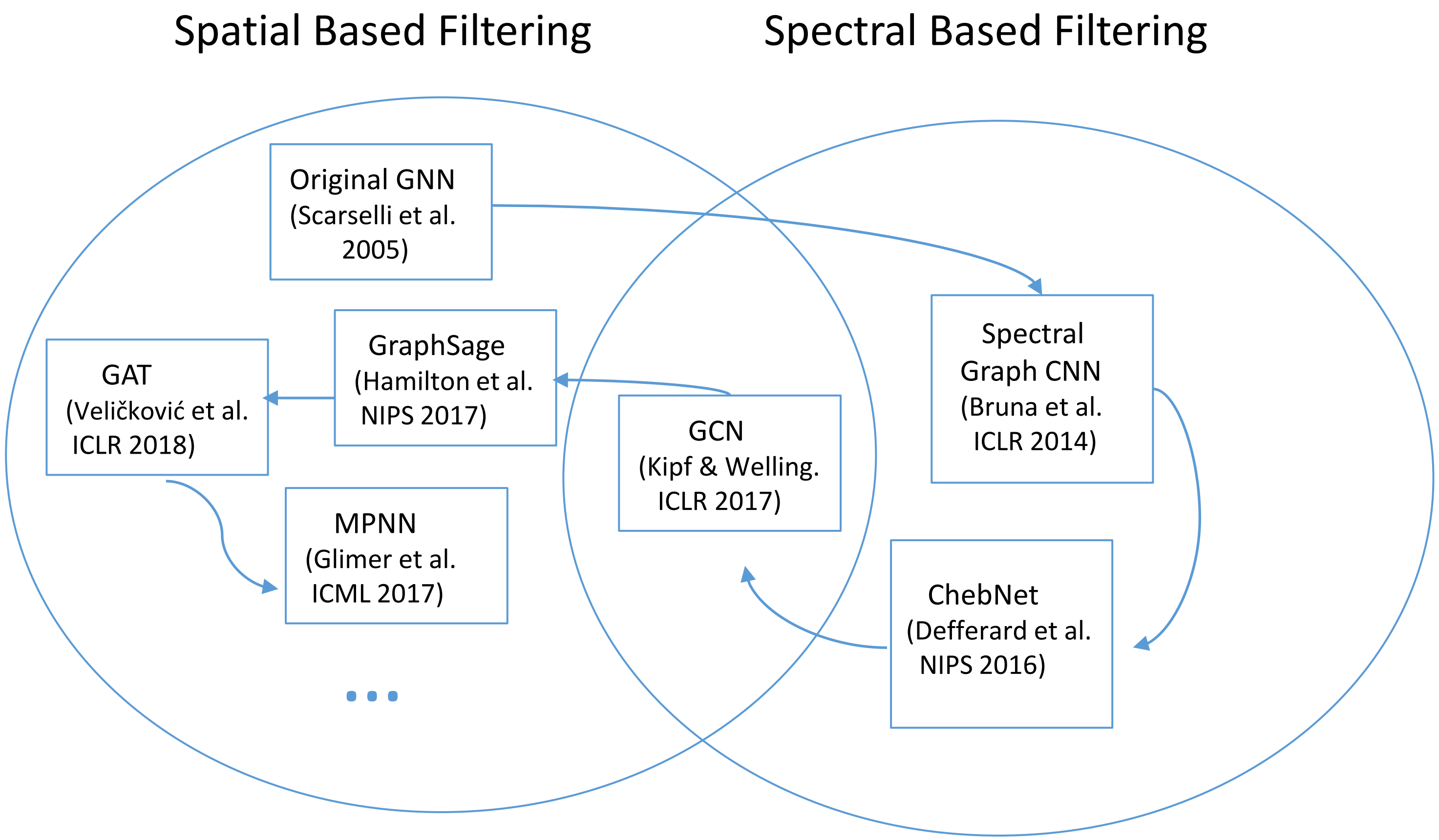
Graph Filtering Operation



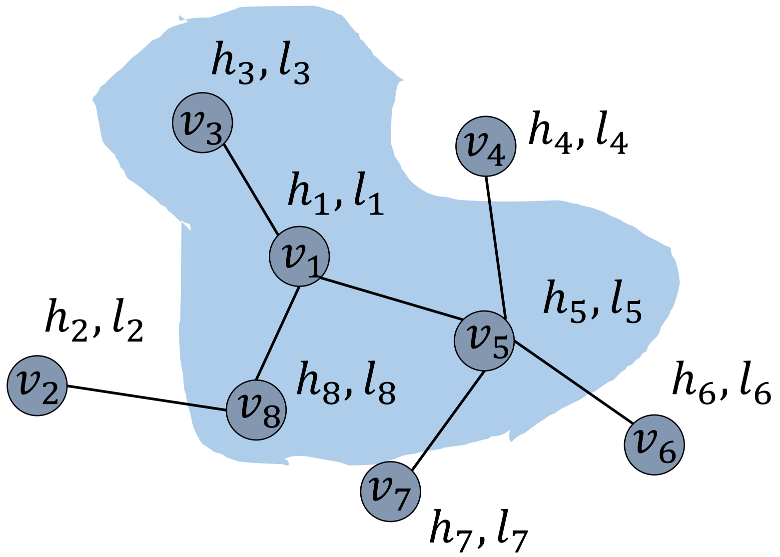
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X}_f \in \mathbb{R}^{n \times d_{new}}$

Two Types of Graph Filtering Operation



Graph Filtering in the First GNN Paper

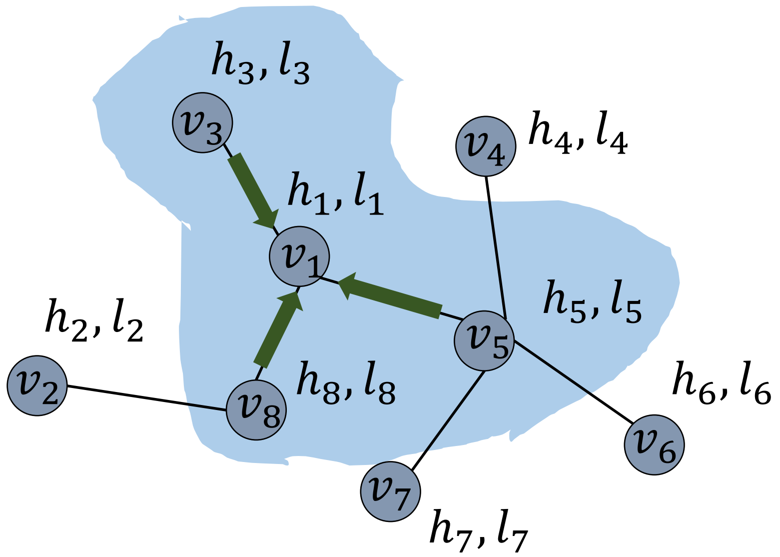


h_i: The hidden features

l_i: The input features

Graph neural networks for ranking web pages. *WI*. IEEE, 2005.

Graph Filtering in the First GNN Paper



h_i: The hidden features

l_i: The input features

$$h_i^{(k+1)} = \sum_{v_j \in N(v_i)} f\left(l_i, h_j^{(k)}, l_j\right), \qquad \forall\, v_i \in V.$$

$N(v_i)$: Neighbors of the node v_i.

$f(\cdot)$: Feedforward neural network.

Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

$\boxed{f}$

Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

$$\boxed{f} \xrightarrow[\text{Decompose}]{GFT} \boxed{u_i^T f}$$

Coefficients

Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

$$\boxed{f} \xrightarrow[\text{Decompose}]{GFT} \boxed{u_i^T f} \xrightarrow[\text{Filter}]{\hat{g}(\lambda_i)} \boxed{\hat{g}(\lambda_i) \cdot u_i^T f}$$

Coefficients

Filtered coefficients

Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

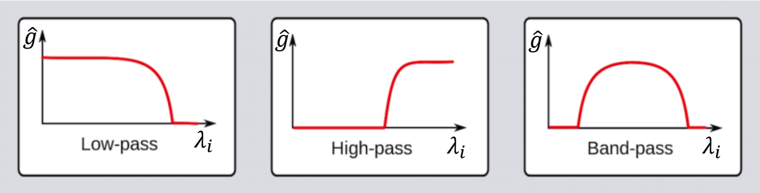
$$\boxed{f} \xrightarrow[\text{Decompose}]{GFT} \boxed{u_i^T f} \xrightarrow[\text{Filter}]{\hat{g}(\lambda_i)} \boxed{\hat{g}(\lambda_i) \cdot u_i^T f}$$

Coefficients

Filtered coefficients

Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Example:



Graph Spectral Filtering for Graph Signal

Recall:

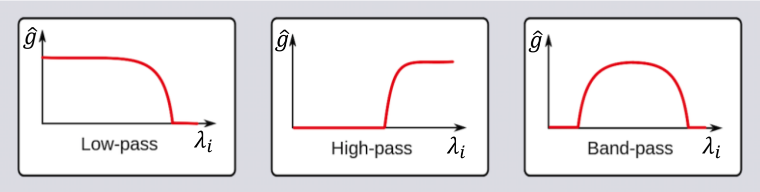
$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

$$f \xrightarrow[\text{Decompose}]{GFT} \underset{\text{Coefficients}}{u_i^T f} \xrightarrow[\text{Filter}]{\hat{g}(\lambda_i)} \underset{\text{Filtered coefficients}}{\hat{g}(\lambda_i) \cdot u_i^T f} \xrightarrow[\text{Reconstruct}]{IGFT} \sum_{i=0}^{N-1} \hat{g}(\lambda_i) \cdot u_i^T f \cdot u_i$$

Example:

Filter $\hat{g}(\lambda_i)$: Modulating the frequency



Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

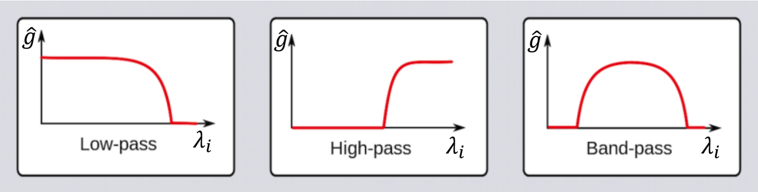
$$f \xrightarrow[\text{Decompose}]{GFT} U^T f \xrightarrow[\text{Filter}]{\hat{g}(\Lambda)} \hat{g}(\Lambda) U^T f \xrightarrow[\text{Reconstruct}]{IGFT} U \hat{g}(\Lambda) U^T f$$

Coefficients

Filtered coefficients

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

Example:



Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

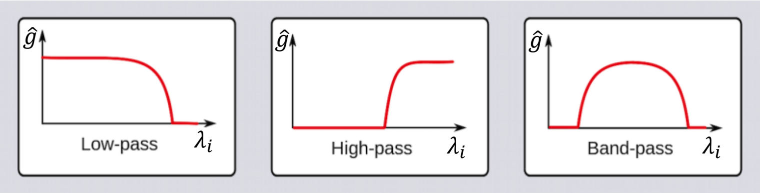
$$f \xrightarrow[\text{Decompose}]{GFT} U^T f \xrightarrow[\text{Filter}]{\hat{g}(\Lambda)} \hat{g}(\Lambda)U^T f \xrightarrow[\text{Reconstruct}]{IGFT} \underbrace{U\hat{g}(\Lambda)U^T}_{\hat{g}(L)} f$$

Coefficients

Filtered coefficients

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

Example:



Graph Spectral Filtering for Graph Signal

Recall:

$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f} \qquad f = \sum_{i=0}^{N-1} \underbrace{\hat{f}_i}_{f^T u_i} \cdot u_i$$

Filter a graph signal f:

$$f \xrightarrow[\text{Decompose}]{GFT} U^T f \xrightarrow[\text{Filter}]{\hat{g}(\Lambda)} \hat{g}(\Lambda)U^T f \xrightarrow[\text{Reconstruct}]{IGFT} \boxed{U\hat{g}(\Lambda)U^T} f$$

Coefficients

Filtered coefficients

$$\boxed{U\hat{g}(\Lambda)U^T} \Leftrightarrow \boxed{\hat{g}(L)}$$

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

$$f \xrightarrow[\text{Filtering}]{} \hat{g}(L)f$$

Graph Spectral Filtering for GNN

How to design the filter?

Graph Spectral Filtering for GNN

How to design the filter?

Data-driven! Learn $\hat{g}(\Lambda)$ from data!

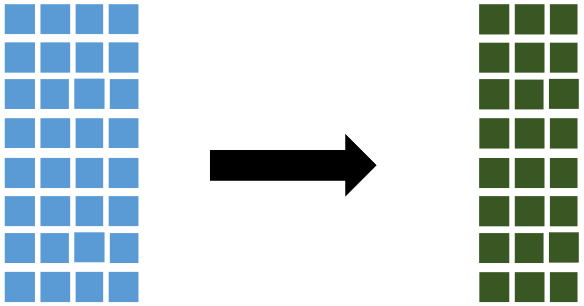
Graph Spectral Filtering for GNN

How to design the filter?

Data-driven! Learn $\hat{g}(\Lambda)$ from data!

How to deal with multi-channel signals?

$$\boldsymbol{F}_{in} \in \mathbb{R}^{N \times d_1} \rightarrow \boldsymbol{F}_{out} \in \mathbb{R}^{N \times d_2}.$$



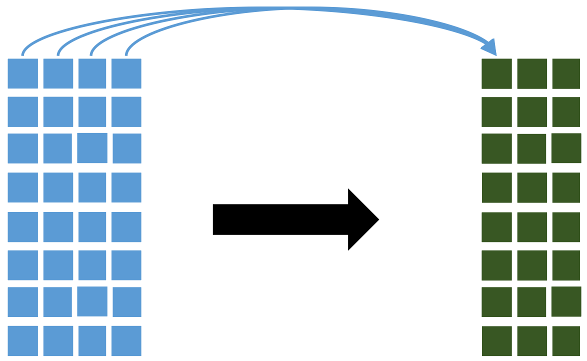
Graph Spectral Filtering for GNN

How to design the filter?

Data-driven! Learn $\hat{g}(\Lambda)$ from data!

How to deal with multi-channel signals?

$$\boldsymbol{F}_{in} \in \mathbb{R}^{N\times d_1} \to \boldsymbol{F}_{out} \in \mathbb{R}^{N\times d_2}.$$



Each input channel contributes to each output channel

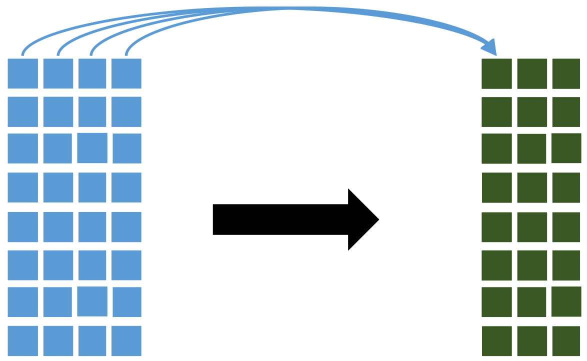
Graph Spectral Filtering for GNN

How to design the filter?

Data-driven! Learn $\hat{g}(\Lambda)$ from data!

How to deal with multi-channel signals?

$$\boldsymbol{F}_{in} \in \mathbb{R}^{N \times d_1} \rightarrow \boldsymbol{F}_{out} \in \mathbb{R}^{N \times d_2}.$$



Each input channel contributes to each output channel

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]}_{\text{Filter each input channel}} \quad i = 1, \dots d_2$$

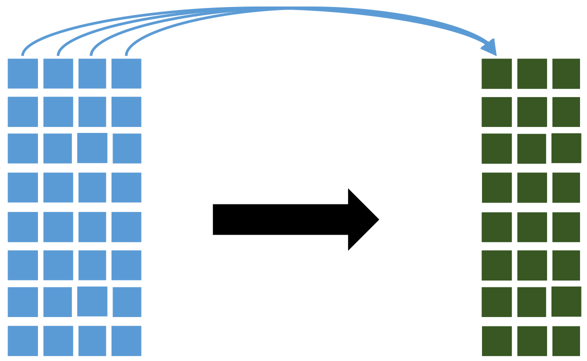
Graph Spectral Filtering for GNN

How to design the filter?

Data-driven! Learn $\hat{g}(\Lambda)$ from data!

How to deal with multi-channel signals?

$$\boldsymbol{F}_{in} \in \mathbb{R}^{N \times d_1} \rightarrow \boldsymbol{F}_{out} \in \mathbb{R}^{N \times d_2}.$$



Each input channel contributes to each output channel

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]}_{\text{Filter each input channel}} \quad i = 1, \ldots d_2$$

Learn $d_2 \times d_1$ filters

$\hat{g}(\Lambda)$: Non-parametric

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

$\hat{g}(\Lambda)$: Non-parametric

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \dots & \\ & & & \theta_N \end{bmatrix}$$

Spectral Networks and Locally Connected Networks on Graphs. ICLR 2014

$\hat{g}(\Lambda)$: Non-parametric

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \cdots & \\ & & & \theta_N \end{bmatrix}$$

$d_2 \times d_1 \times N$ parameters

$\hat{g}(\Lambda)$: Non-parametric

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \cdots & \\ & & & \theta_N \end{bmatrix}$$

$d_2 \times d_1 \times N$ parameters

$U\hat{g}(\Lambda)U^T f$

Expensive eigen-decomposition

$$\boldsymbol{L} = \boldsymbol{U\Lambda U}^T$$
$$\boldsymbol{L}^k = \boldsymbol{U\Lambda}^k \boldsymbol{U}^T$$
$$\boldsymbol{\Lambda}^k = \begin{bmatrix} \lambda_1^k & & \\ & \dots & \\ & & \lambda_N^k \end{bmatrix}$$

$\hat{g}(\Lambda)$: Polynomial Parametrized

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \dots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. NIPS 2016.

$\hat{g}(\Lambda)$: Polynomial Parametrized

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \cdots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

$d_2 \times d_1 \times K$ parameters

$\hat{g}(\Lambda)$: Polynomial Parametrized

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \dots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

$d_2 \times d_1 \times K$ parameters

$$\mathrm{U}\hat{g}(\Lambda)U^T f = \sum_{k=0}^{K} \theta_k L^k f$$

$\hat{g}(\Lambda)$: Polynomial Parametrized

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \cdots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

$d_2 \times d_1 \times K$ parameters

$$\mathrm{U}\hat{g}(\Lambda)U^T f = \sum_{k=0}^{K} \theta_k L^k f$$

No eigen-decomposition needed

Polynomial Parametrized Filter: a Spatial View

$$U\hat{g}(\Lambda)U^T f(i) = \sum_{j=0}^{N} \sum_{k=0}^{K} \theta_k L_{i,j}^k f(j)$$

Polynomial Parametrized Filter: a Spatial View

$$U\hat{g}(\Lambda)U^T f(i) = \sum_{j=0}^{N} \sum_{k=0}^{K} \theta_k L_{i,j}^k f(j)$$

If the node v_j is more than K-hops away from node v_i, then,

$$\sum_{k=0}^{K} \theta_k L_{i,j}^k = 0$$

Polynomial Parametrized Filter: a Spatial View

$$U\hat{g}(\Lambda)U^T f(i) = \sum_{j=0}^{N} \sum_{k=0}^{K} \theta_k L_{i,j}^k f(j)$$

If the node v_j is more than K-hops away from node v_i, then,

$$\sum_{k=0}^{K} \theta_k L_{i,j}^k = 0$$

The filter is localized within K-hops neighbors in the spatial domain

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

Unstable under perturbation of coefficients

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

Unstable under perturbation of coefficients

Chebyshev polynomials:

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

Unstable under perturbation of coefficients

Chebyshev polynomials:

Recursive definition:

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

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

Unstable under perturbation of coefficients

Chebyshev polynomials:

Recursive definition:

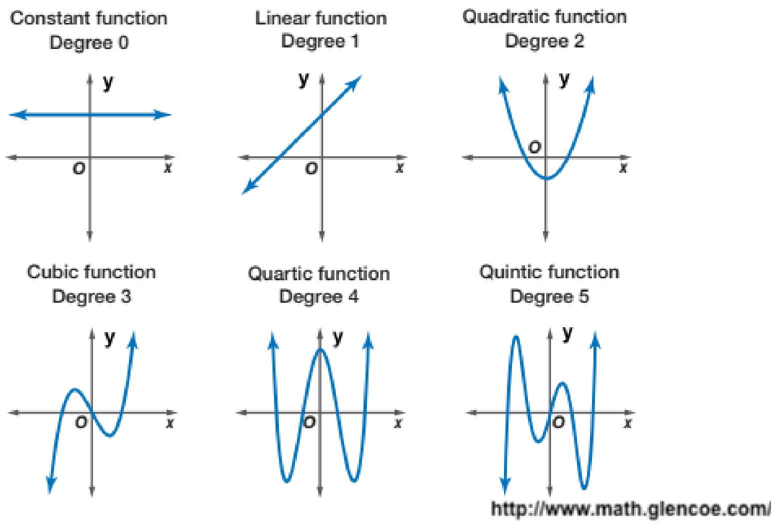
- $T_0(x) = 1; T_1(x) = x$
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

The Chebyshev polynomials $\{T_k\}$ form an orthogonal basis for the Hilbert space $L^2([-1,1], \frac{dy}{\sqrt{1-y^2}})$.

Chebyshev Polynomials

The polynomials adopted have non-orthogonal basis $1, x, x^2, x^3, \ldots$

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$



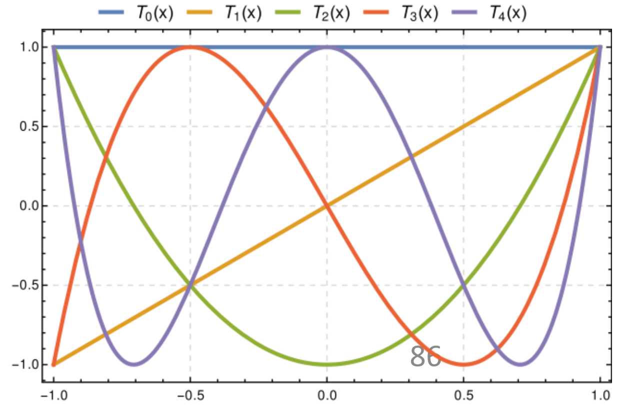
Unstable under perturbation of coefficients

Chebyshev polynomials:

Recursive definition:

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

$$g(x) = \theta_0 T_0(x) + \theta_1 T_1(x) + \theta_2 T_2(x) + \cdots$$



ChebNet

Parametrize $\hat{g}(\Lambda)$ with Chebyshev polynomials

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. NIPS 2016.

ChebNet

Parametrize $\hat{g}(\Lambda)$ with Chebyshev polynomials

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

$d_2 \times d_1 \times K$ parameters

ChebNet

Parametrize $\hat{g}(\Lambda)$ with Chebyshev polynomials

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

$d_2 \times d_1 \times K$ parameters

$\mathrm{U}\hat{g}(\Lambda)U^T f = \sum_{k=0}^{K} \theta_k T_k(\tilde{L})\, f$, with $\tilde{L} = \frac{2L}{\lambda_{max}} - I$

No eigen-decomposition needed

ChebNet

Parametrize $\hat{g}(\Lambda)$ with Chebyshev polynomials

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

$d_2 \times d_1 \times K$ parameters

$\mathrm{U}\hat{g}(\Lambda)U^T f = \sum_{k=0}^{K} \theta_k T_k(\tilde{L})\, f$, with $\tilde{L} = \frac{2L}{\lambda_{max}} - I$

No eigen-decomposition needed

Stable under perturbation of coefficients

GCN: Simplified ChebNet

Use Chebyshev polynomials with $K = 1$ and assume $\lambda_{max} = 2$

$$\hat{g}(\Lambda) = \theta_0 + \theta_1(\Lambda - I)$$

Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017.

GCN: Simplified ChebNet

Use Chebyshev polynomials with $K = 1$ and assume $\lambda_{max} = 2$

$$\hat{g}(\Lambda) = \theta_0 + \theta_1(\Lambda - I)$$

Further constrain $\theta = \theta_0 = -\theta_1$

$$\hat{g}(\Lambda) = \theta(2I - \Lambda)$$

$$U\hat{g}(\Lambda)U^T f = \theta(2I - L)f = \theta\left(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)f$$

GCN: Simplified ChebNet

Use Chebyshev polynomials with $K = 1$ and assume $\lambda_{max} = 2$

$$\hat{g}(\Lambda) = \theta_0 + \theta_1(\Lambda - I)$$

Further constrain $\theta = \theta_0 = -\theta_1$

$$\hat{g}(\Lambda) = \theta(2I - \Lambda)$$

$$U\hat{g}(\Lambda)U^T f = \theta(2I - L)f = \theta\left(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)f$$

Apply a renormalization trick

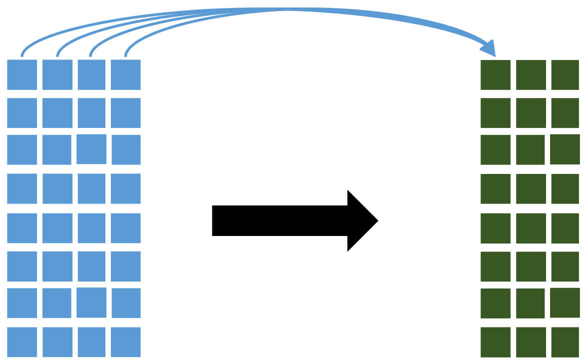
$$U\hat{g}(\Lambda)U^T f = \theta\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)f, with\ \hat{A} = A + I$$

GCN for Multi-channel Signal

Recall:

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underline{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]} \quad i = 1, \dots d_2$$

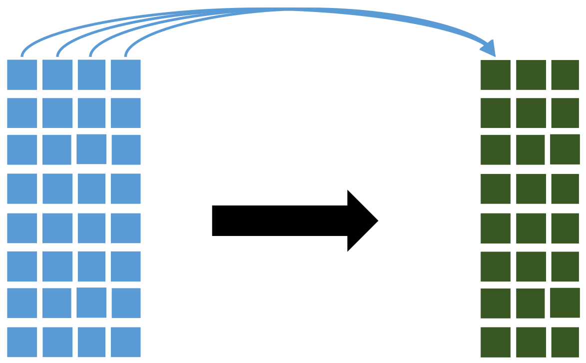
Filter each input channel



GCN for Multi-channel Signal

Recall:

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]}_{\text{Filter each input channel}} \quad i = 1, \dots d_2$$



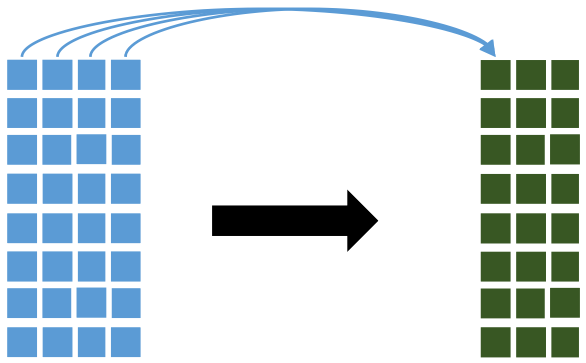
For GCN:

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\theta_{ji}(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}})}_{\text{GCN filter}}\boldsymbol{F}_{in}[:,j] \quad i = 1, \dots d_2$$

GCN for Multi-channel Signal

Recall:

$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]}_{\text{Filter each input channel}} \quad i = 1, \dots d_2$$



For GCN:

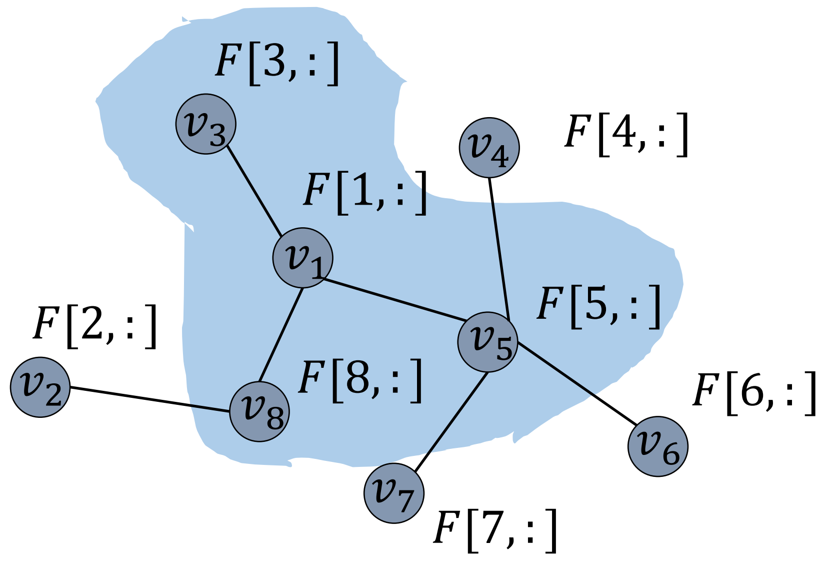
$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\theta_{ji}(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}})}_{\text{GCN filter}}\boldsymbol{F}_{in}[:,j] \quad i = 1, \dots d_2$$

In matrix form:

$$\boldsymbol{F}_{out} = (\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}})\boldsymbol{F}_{in}\Theta \text{ with } \Theta \in \mathbb{R}^{d_1 \times d_2} \text{ and } \Theta[j,i] = \theta_{ji}$$

A Spatial View of GCN Filter

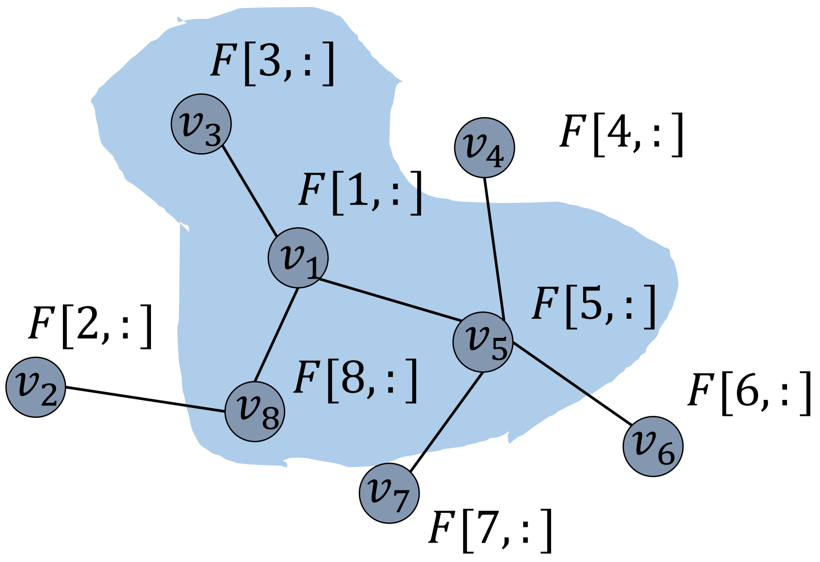
Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$



A Spatial View of GCN Filter

Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$

- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$



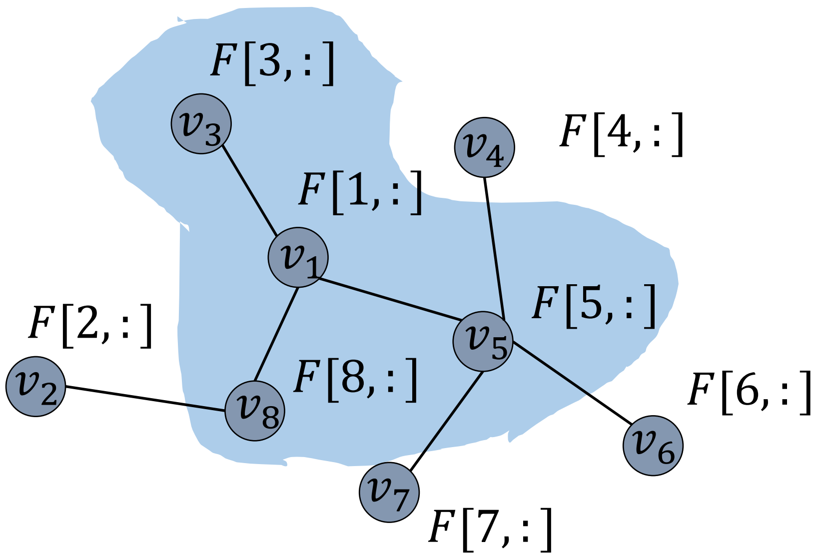
A Spatial View of GCN Filter

Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$

- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$

Observe that:

- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$



A Spatial View of GCN Filter

Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$

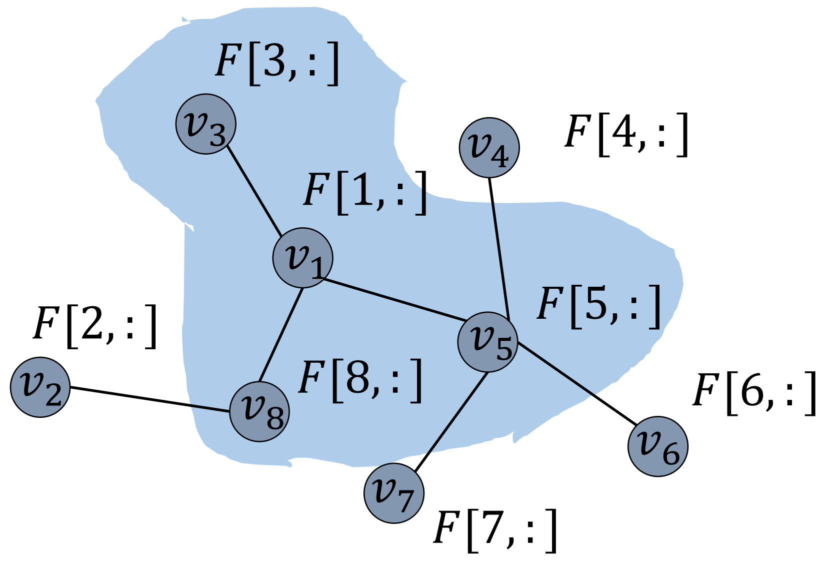
- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$

Observe that:

- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$

Hence,

$$\mathbf{F}_{out}[i,:] = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} C[i,j]\mathbf{F}_{in}[j,:]\Theta$$



A Spatial View of GCN Filter

Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$

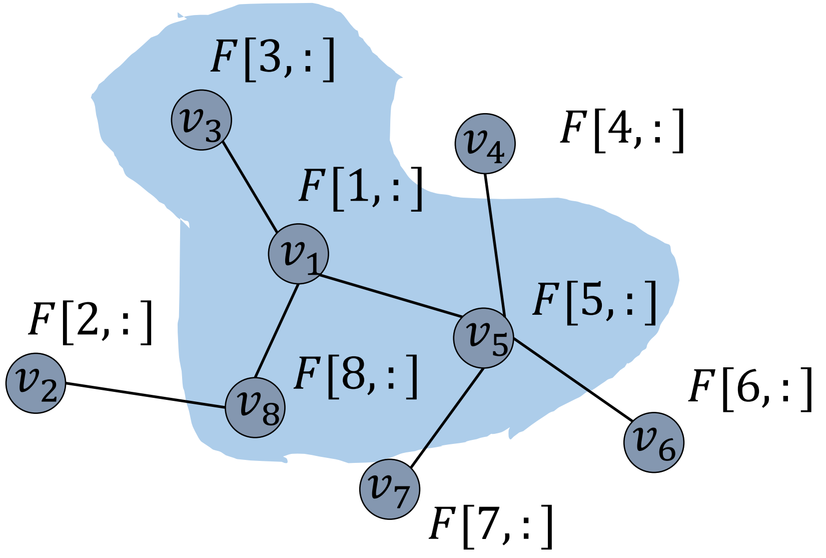
- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$

Observe that:

- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$

Hence,

$$\mathbf{F}_{out}[i,:] = \sum_{v_j \in \mathcal{N}(v_i)\cup\{v_i\}} C[i,j]\underbrace{\mathbf{F}_{in}[j,:]\Theta}_{\text{Feature transformation}}$$



A Spatial View of GCN Filter

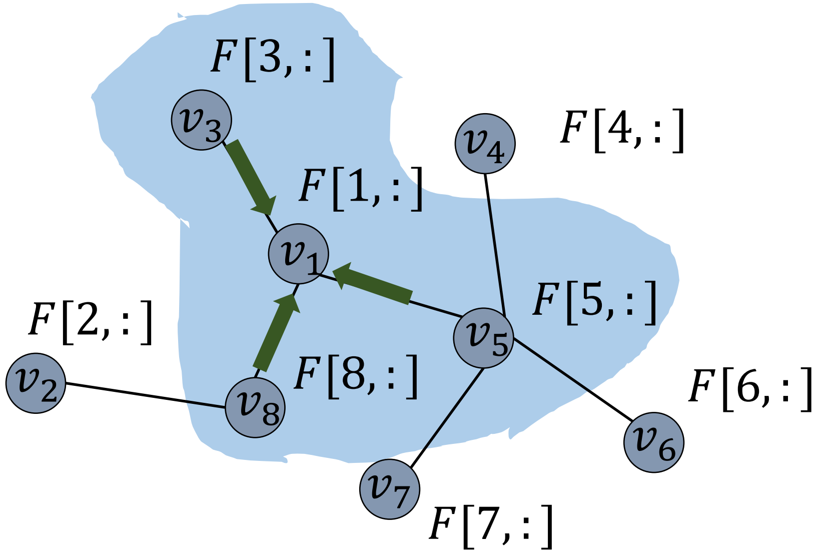
Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$

- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$

Observe that:

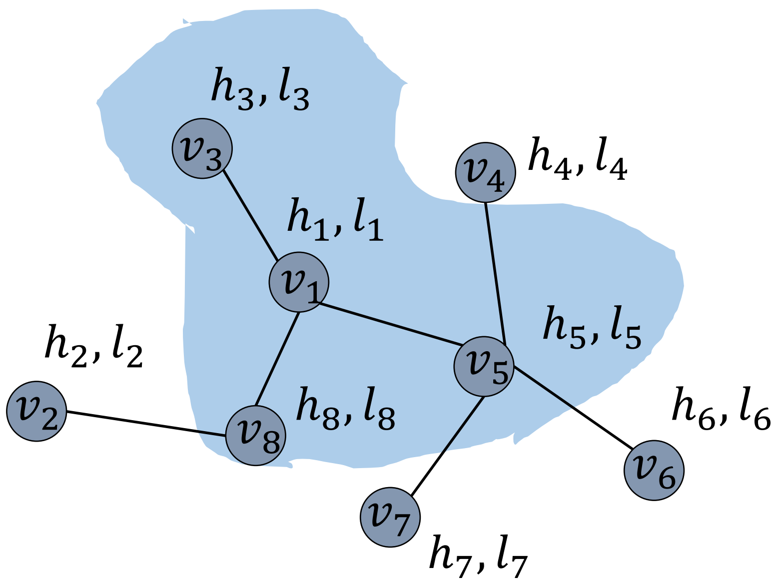
- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$

Hence,



$$\mathbf{F}_{out}[i,:] = \underbrace{\sum_{v_j \in \mathcal{N}(v_i)\cup\{v_i\}}}_{\text{Aggregation}} C[i,j]\underbrace{\mathbf{F}_{in}[j,:]\Theta}_{\text{Feature transformation}}$$

Filter in GCN VS Filter in the First GNN



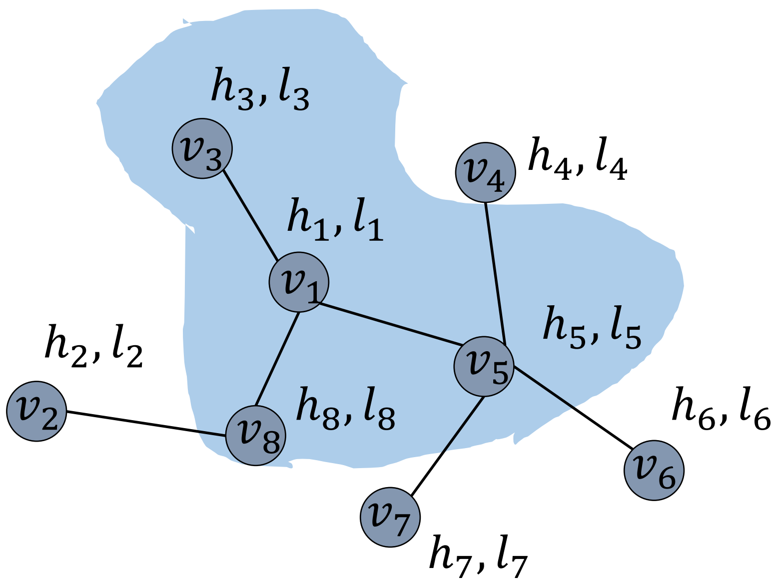
GCN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} C[i,j]\mathbf{h}_j^{(k)}\Theta, \forall v_i \in \mathcal{V}$$

The first GNN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i)} f(l_i, \mathbf{h}_j^{(k)}, l_j), \forall v_i \in \mathcal{V}$$

Filter in GCN VS Filter in the First GNN



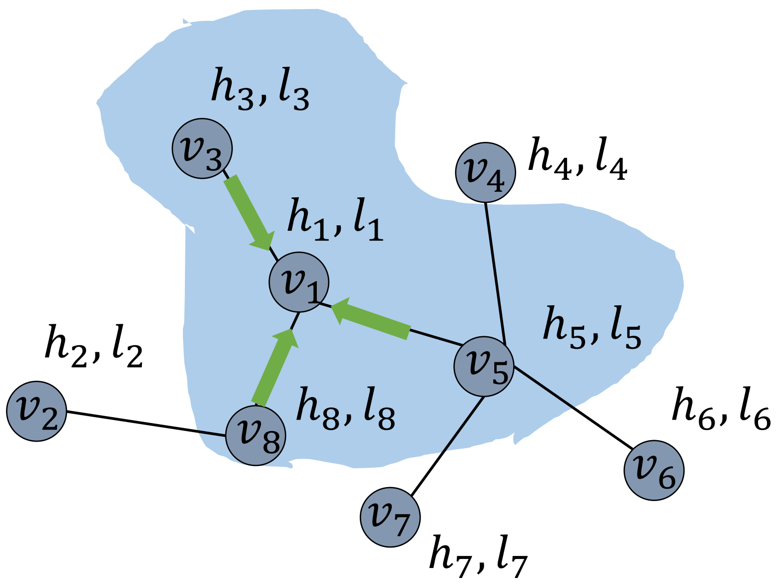
GCN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} C[i,j]\underline{\mathbf{h}_j^{(k)}\Theta}, \forall v_i \in \mathcal{V}$$

The first GNN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \sum_{v_j \in \mathcal{N}(v_i)} \underline{f(l_i, \mathbf{h}_j^{(k)}, l_j)}, \forall v_i \in \mathcal{V}$$

Filter in GCN VS Filter in the First GNN



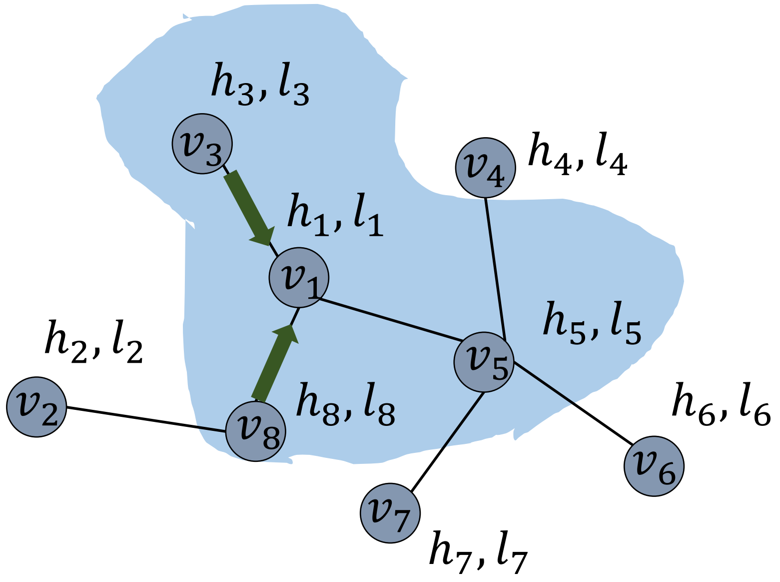
GCN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \underline{\sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}}} C[i,j]\underline{\mathbf{h}_j^{(k)}}\Theta, \forall v_i \in \mathcal{V}$$

The first GNN: k-th layer

$$\mathbf{h}_i^{(k+1)} = \underline{\sum_{v_j \in \mathcal{N}(v_i)}} \underline{f(l_i, \mathbf{h}_j^{(k)}, l_j)}, \forall v_i \in \mathcal{V}$$

Filter in GraphSage

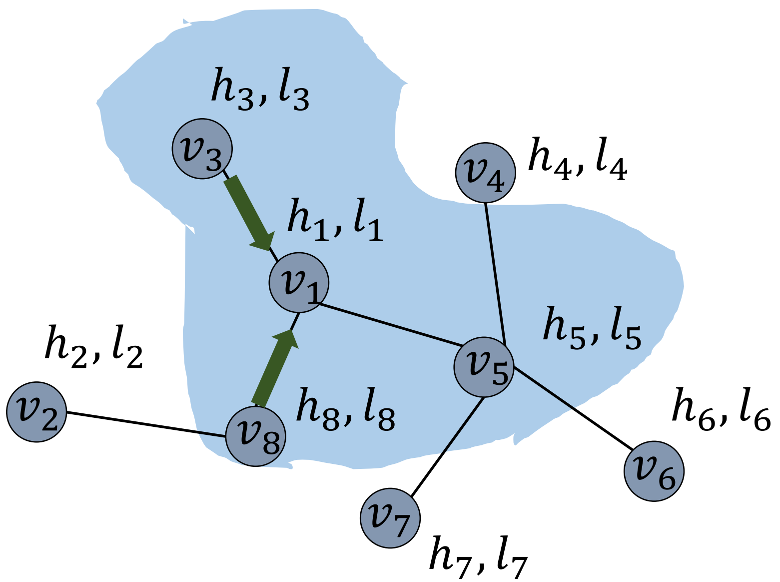


Neighbor Sampling

$$\mathcal{N}(v_i) \rightarrow \mathcal{N}_s(v_i)$$

Inductive Representation Learning on Large Graphs. NIPS 2017.

Filter in GraphSage



Neighbor Sampling

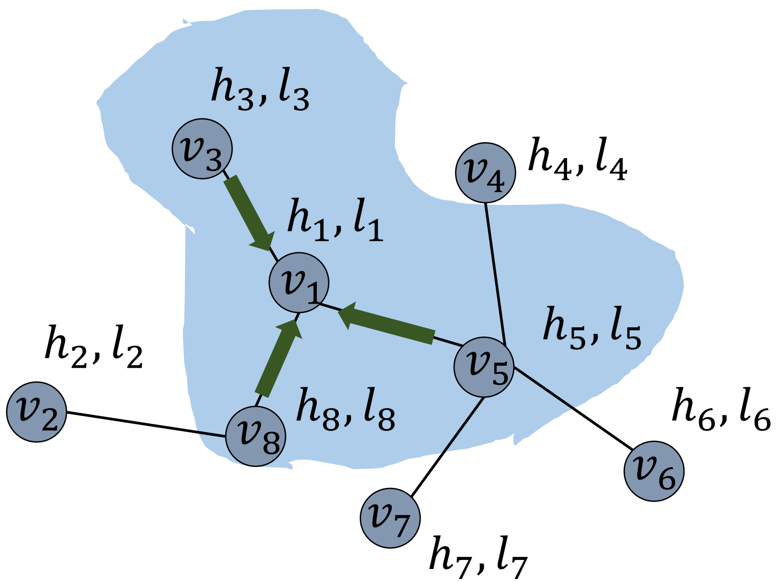
$$\mathcal{N}(v_i) \rightarrow \mathcal{N}_s(v_i)$$

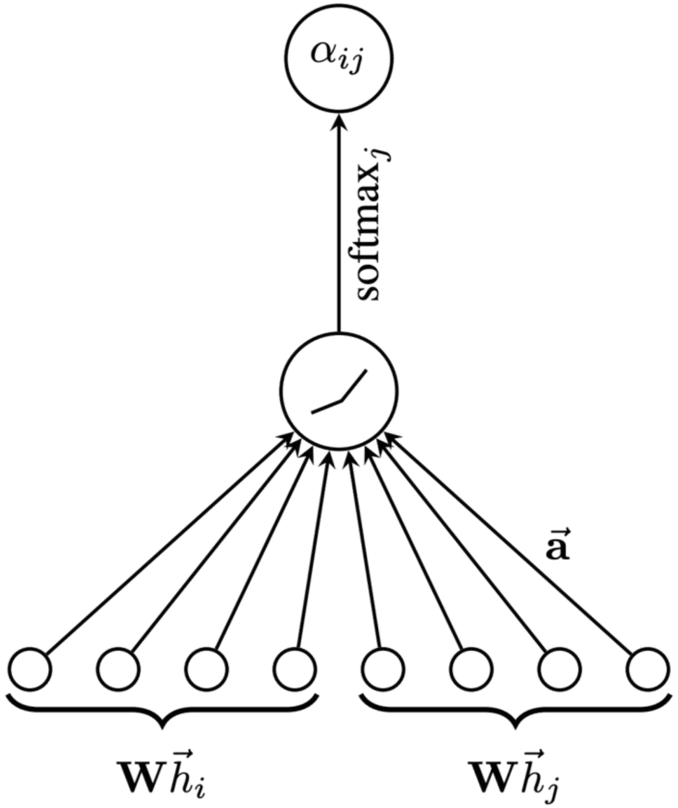
Aggregation

$$\mathbf{h}_{\mathcal{N}_s(v_i)}^{(k+1)} = \underline{AGG}(\{\mathbf{h}_i^{(k)}, v_j \in \mathcal{N}_s(v_i)\})$$

$$\mathbf{h}_i^{(k+1)} = \sigma(\Theta[\mathbf{h}_i^{(k)}, \mathbf{h}_{\mathcal{N}_s(v_i)}^{(k+1)}])$$

Filter in GAT

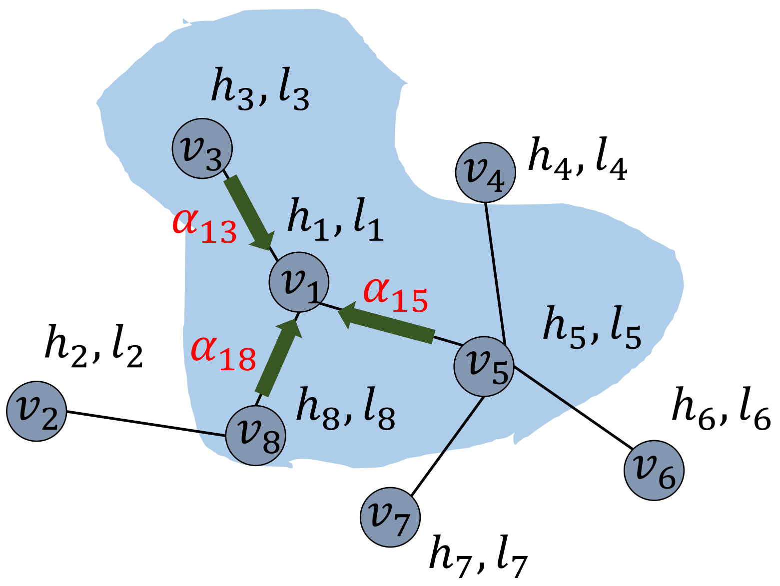


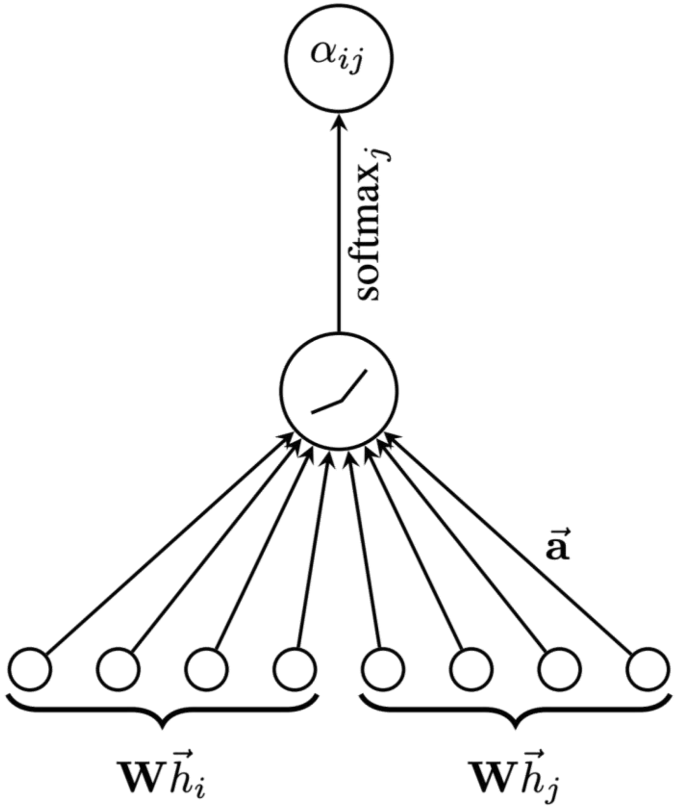


$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Graph Attention Networks. ICLR 2018.

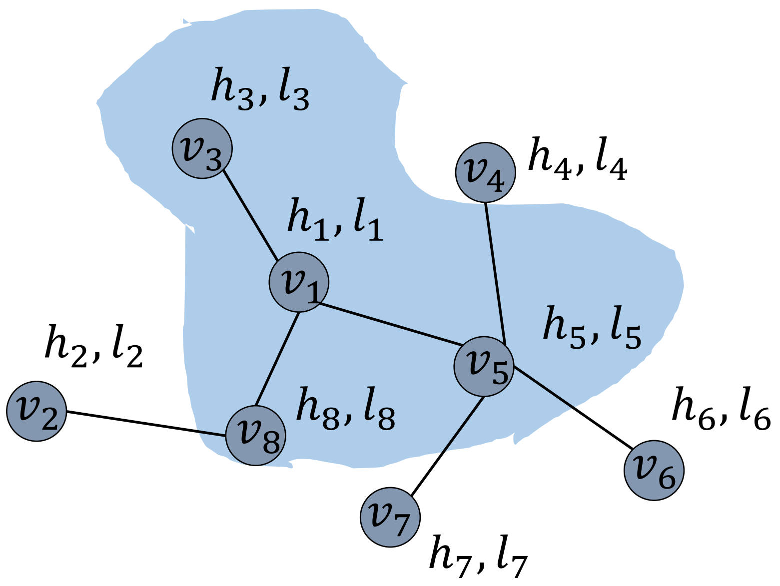
Filter in GAT





$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Filter in MPNN



Message Passing

$$m_i^{(k+1)} = \sum_{v_j \in N(v_i)} M_k\left(h_i^{(k)}, h_j^{(k)}, e_{ij}\right)$$

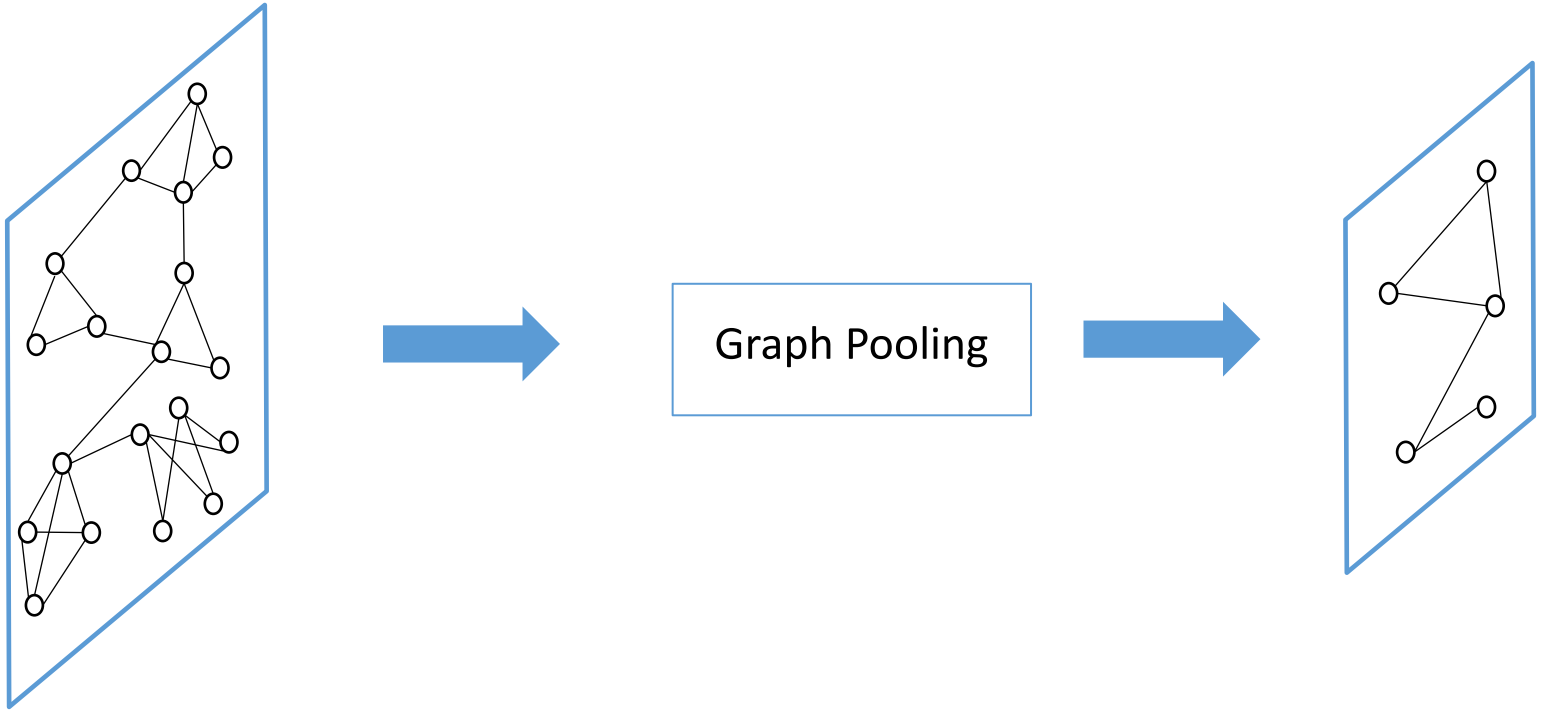
Feature Updating

$$h_i^{(k+1)} = U_k\left(h_i^{(k)}, m_i^{(k+1)}\right)$$

$M_k()$ and $U_k()$ are functions to be designed

Neural Message Passing for Quantum Chemistry. ICML 2017.

Graph Pooling Operation

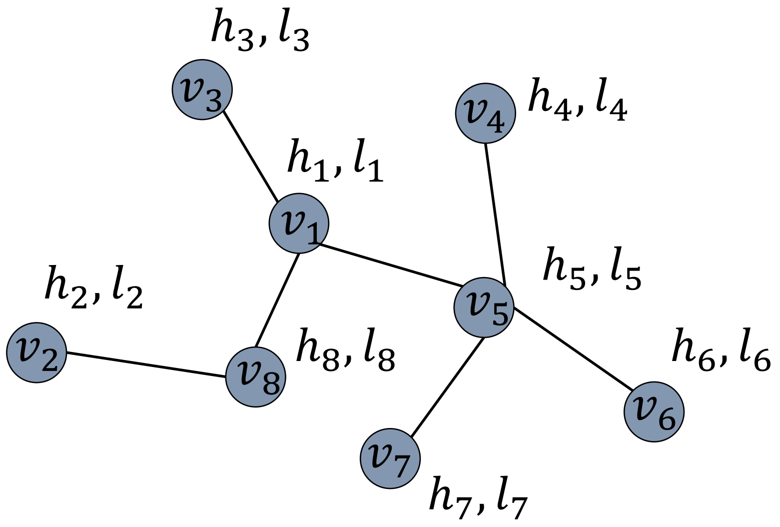


$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$

gPool

Downsample by selecting the most importance nodes



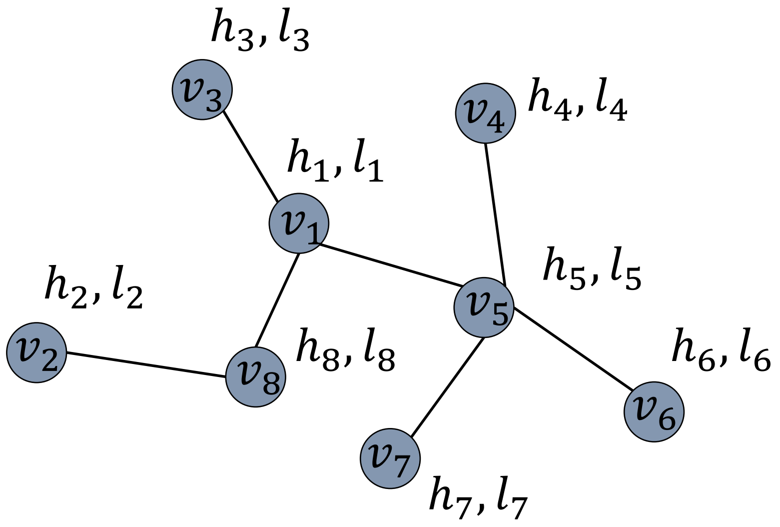
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Graph U-Nets. ICML 2019.

gPool

Downsample by selecting the most importance nodes



Importance Measure

$v_i \to y_i \qquad y_i = \frac{h_i^T p}{||p||}$

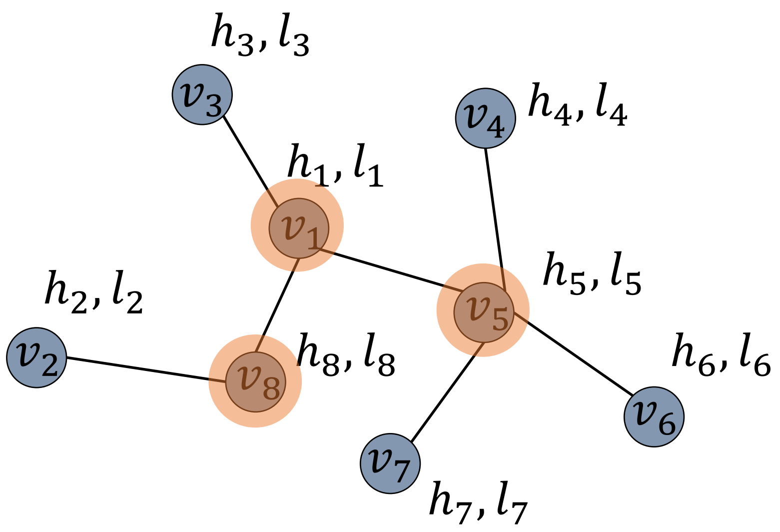
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\downarrow$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

gPool

Downsample by selecting the most importance nodes



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\Downarrow$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Importance Measure

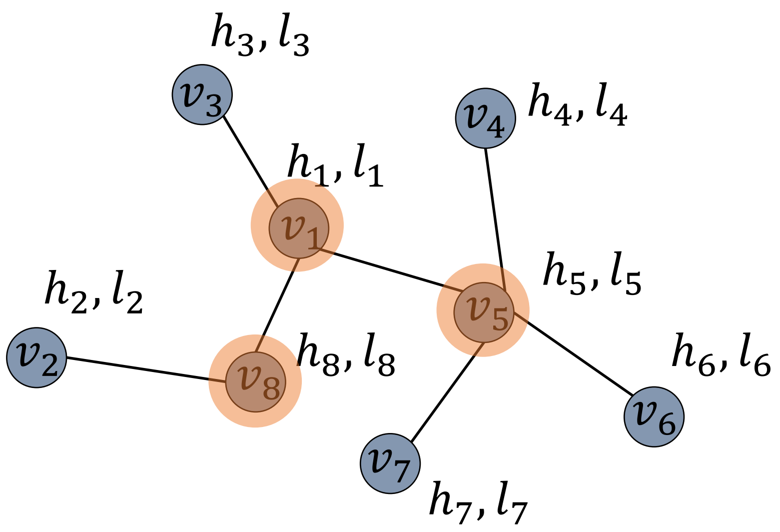
$$v_i \to y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

$$idx = rank(\boldsymbol{y}, n_p)$$

gPool

Downsample by selecting the most importance nodes



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\downarrow$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Importance Measure

$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

$$idx = rank(\boldsymbol{y}, n_p)$$

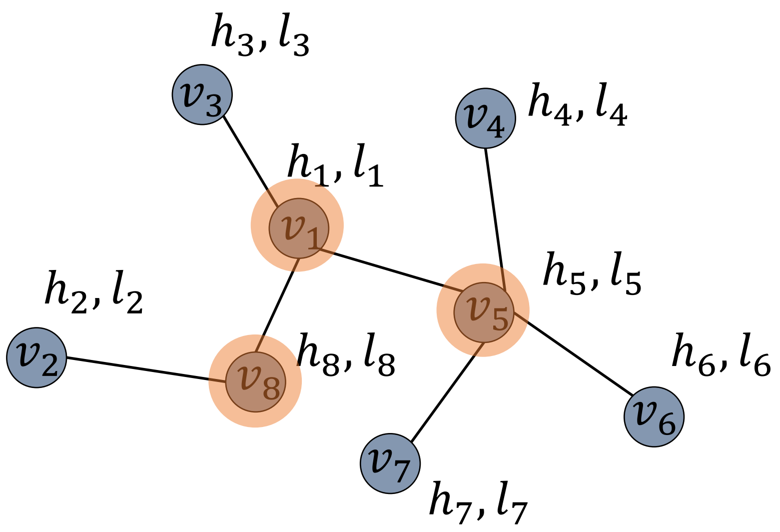
Generate A_p and intermediate H_{inter}

$$A_p = A[idx, idx]$$

$$H_{inter} = H[idx, :]$$

gPool

Downsample by selecting the most importance nodes



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Importance Measure

$$v_i \to y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

$$idx = rank(\boldsymbol{y}, n_p)$$

Generate A_p and intermediate H_{inter}

$$A_p = A[idx, idx]$$

$$H_{inter} = H[idx, :]$$

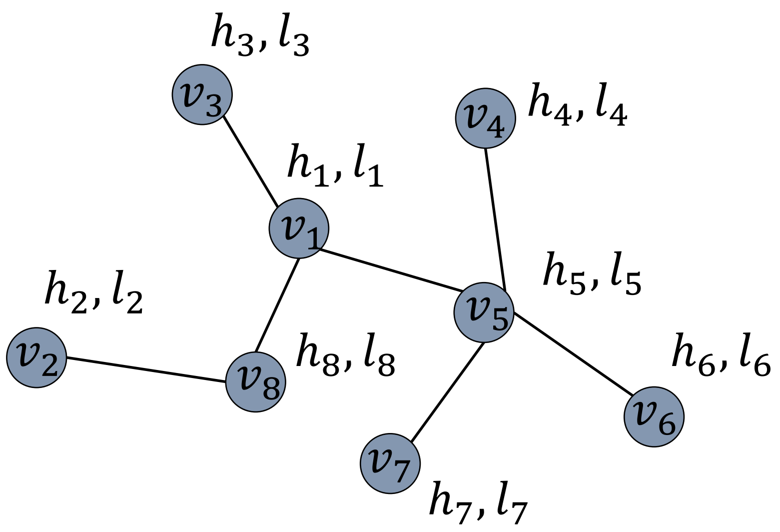
Generate H_p

$$\tilde{y} = sigmoid(y[idx])$$

$$H_p = H_{inter} \odot \tilde{y}$$

DiffPool

Downsample by clustering the nodes using GNN



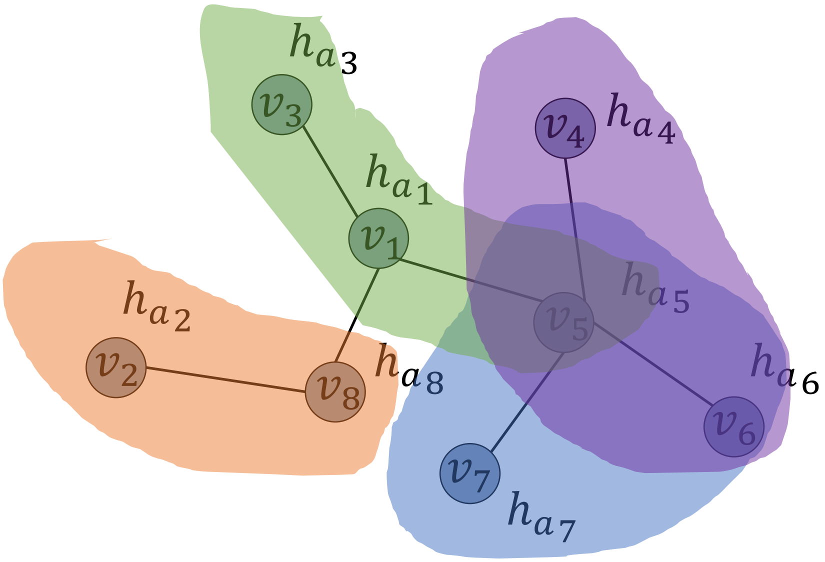
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Hierarchical Graph Representation Learning with Differentiable Pooling. NeurIPS 2018.

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

$$\mathbf{A}_p \in \{0,1\}^{n_p\times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p\times d_{new}}, n_p < n$$

Filter1:

Generate a soft-assign matrix

2 filters

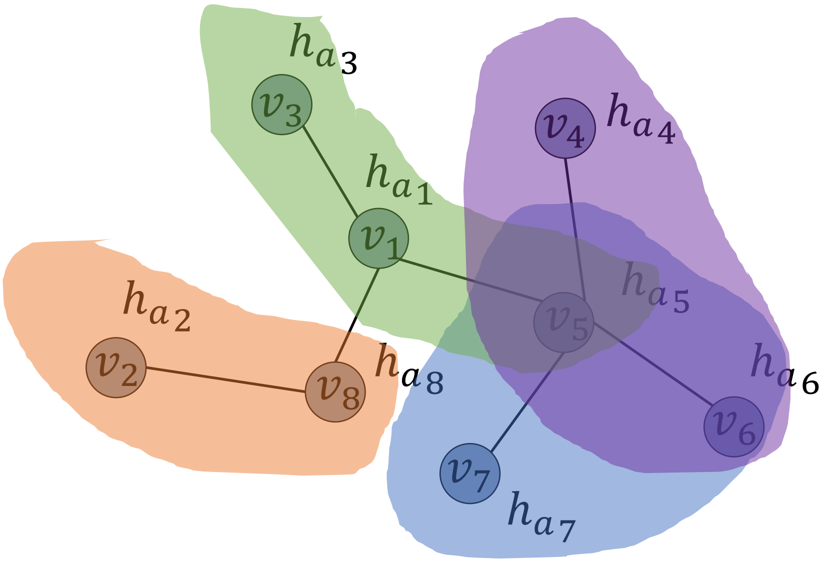
$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H}_a \in \mathbb{R}^{n\times n_p}$$

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

$$\mathbf{A}_p \in \{0,1\}^{n_p\times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p\times d_{new}}, n_p < n$$

2 filters

Filter1:

Generate a soft-assign matrix

$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H}_a \in \mathbb{R}^{n\times n_p}$$

Filter2:

Generate new features

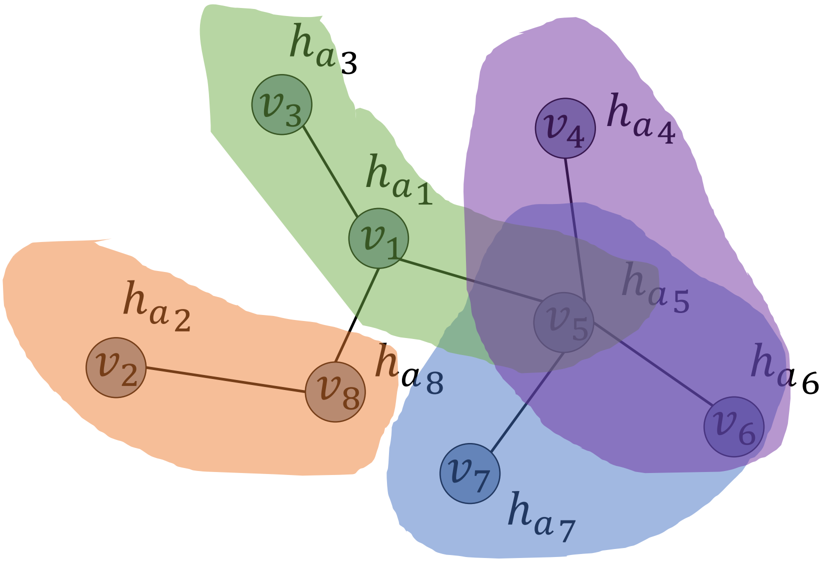
$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H}_f \in \mathbb{R}^{n\times d_{new}}$$

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

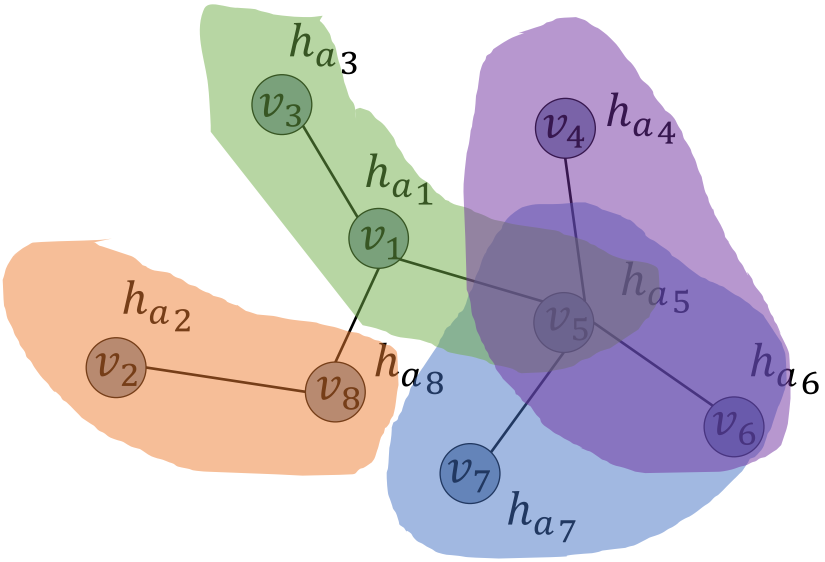
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Generated soft-assign matrix $\mathbf{H}_a \in \mathbb{R}^{n \times n_p}$

Generated new features $\mathbf{H}_f \in \mathbb{R}^{n \times d_{new}}$

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p\times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p\times d_{new}}, n_p < n$$

Generated soft-assign matrix $\mathbf{H}_a \in \mathbb{R}^{n\times n_p}$

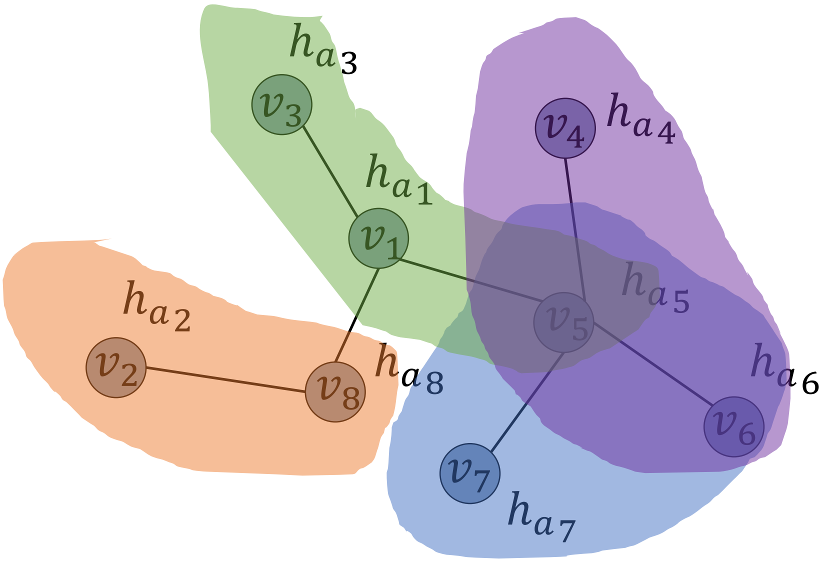
Generated new features $\mathbf{H}_f \in \mathbb{R}^{n\times d_{new}}$

Generate A_p

$$\mathbf{A}_p = \mathbf{H}_a^T \mathbf{A} \mathbf{H}_a$$

DiffPool

Downsample by clustering the nodes using GNN



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Generated soft-assign matrix $\mathbf{H}_a \in \mathbb{R}^{n \times n_p}$

Generated new features $\mathbf{H}_f \in \mathbb{R}^{n \times d_{new}}$

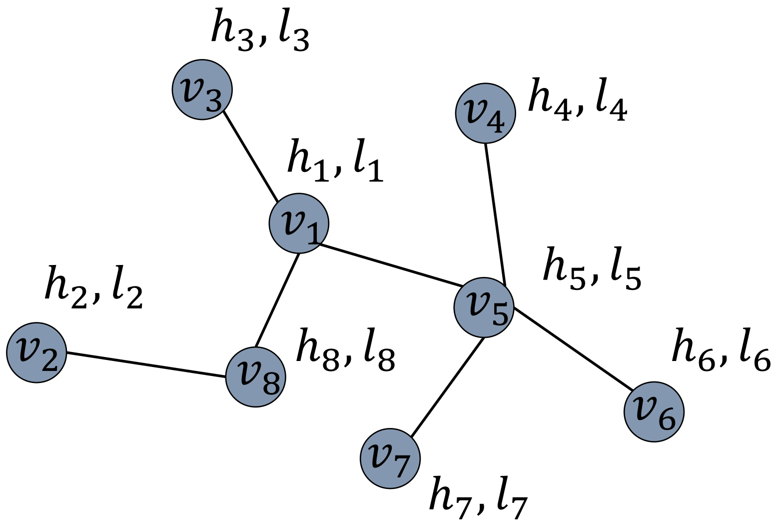
Generate A_p

$$\mathbf{A}_p = \mathbf{H}_a^T \mathbf{A} \mathbf{H}_a$$

Generate H_p

$$\mathbf{H}_p = \mathbf{H}_a^T \mathbf{H}_f$$

Eigenpooling

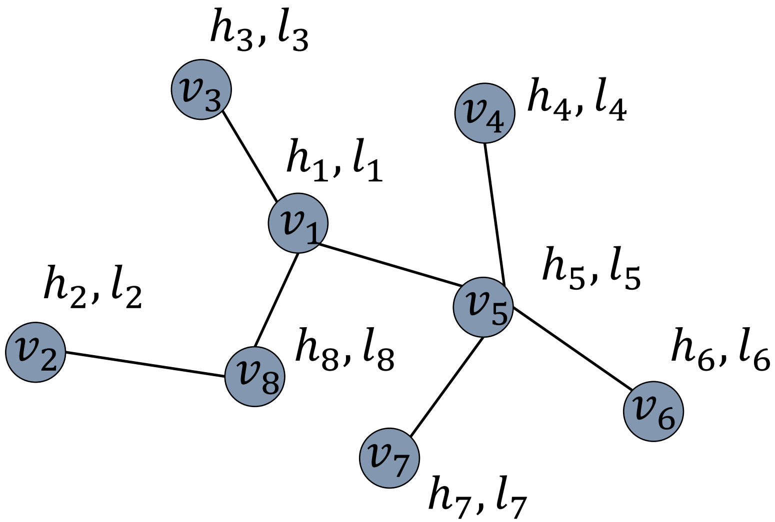


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

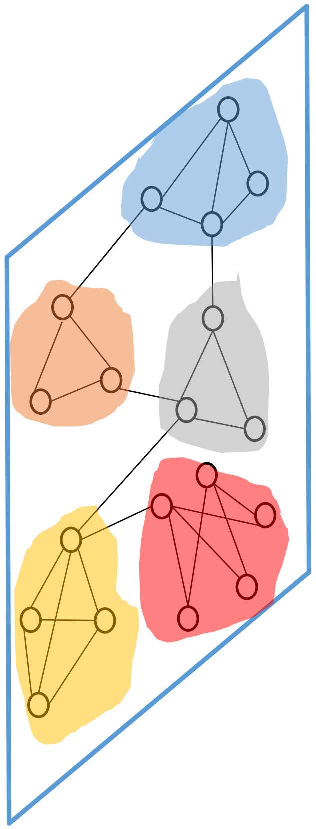
Graph Convolutional Networks with EigenPooling. KDD 2019.

Eigenpooling



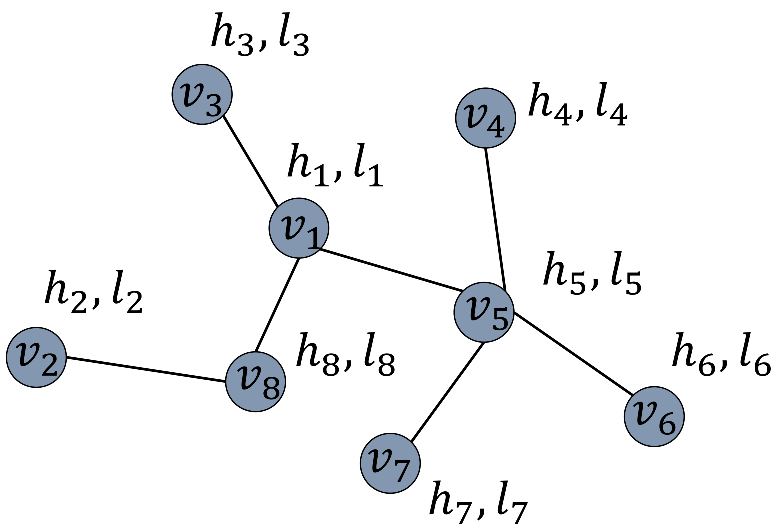
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$



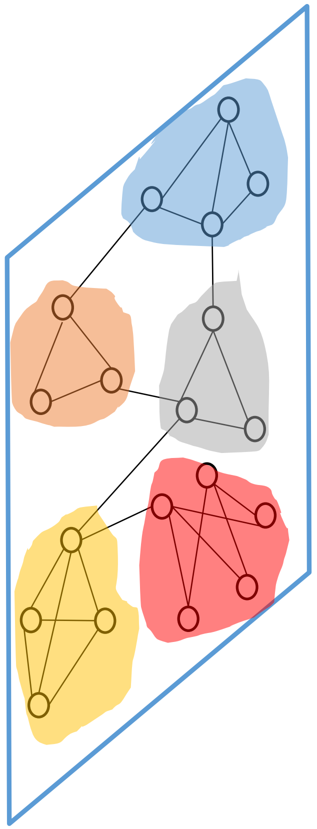
Learn A_p using clustering methods

Eigenpooling



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

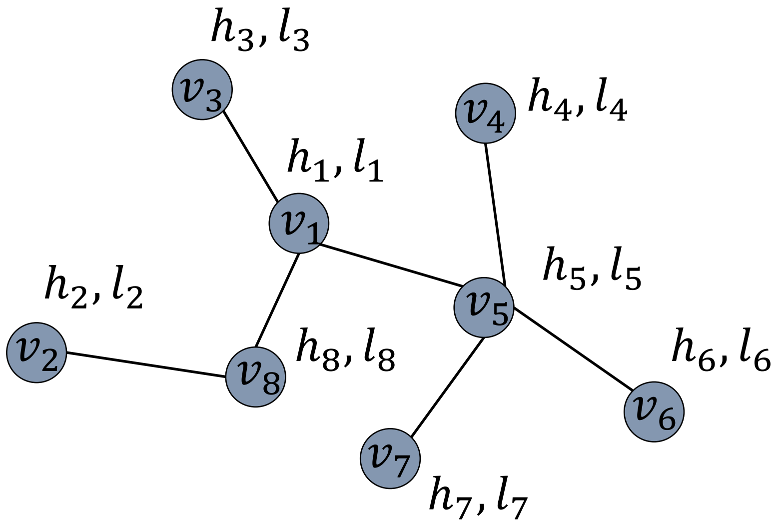
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$



Learn A_p using clustering methods

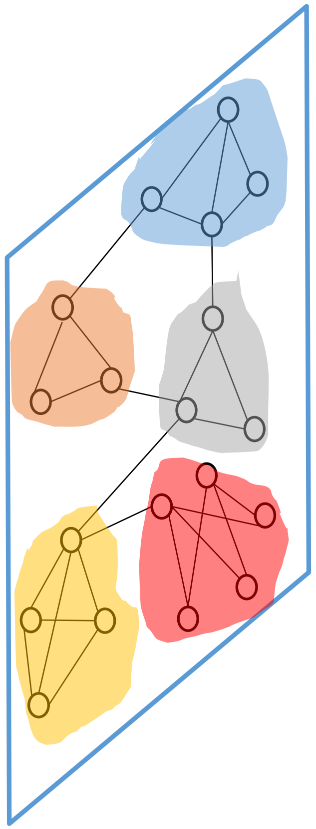
Focus on learning better H_p

Eigenpooling



$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$



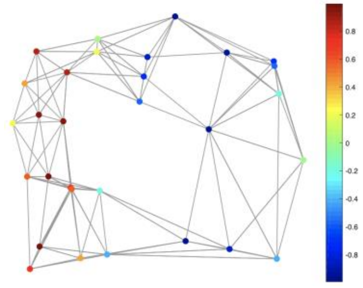
Learn A_p using clustering methods

Focus on learning better H_p

Capture both feature and graph structure

Going Back to Graph Spectral Theory

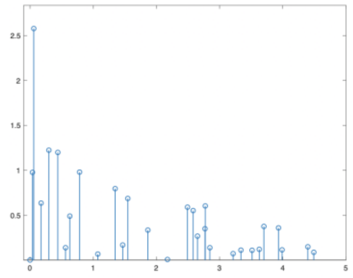
Recall:



Spatial domain: $\boldsymbol{f}$

$$\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$$

Decompose signal f

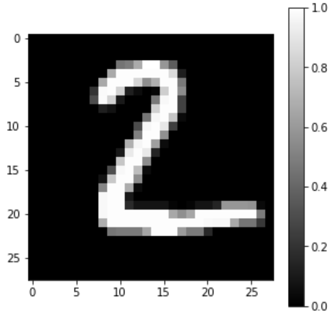


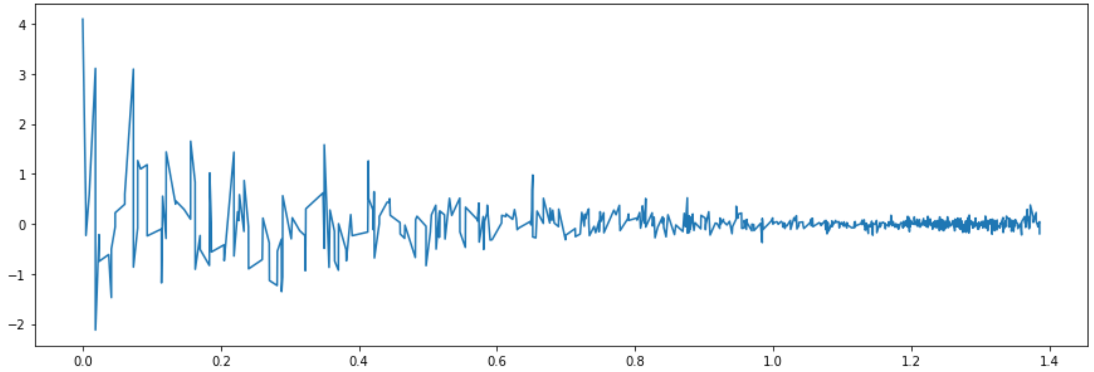
Spectral domain: $\hat{\boldsymbol{f}}$

$$\mathbf{f} = \hat{f}_0 u_0 + \hat{f}_1 u_1 + \dots \hat{f}_{N-1} u_{N-1}$$

Going Back to Graph Spectral Theory

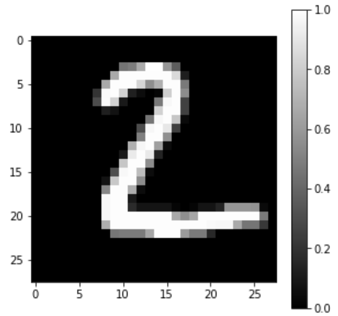
Do we need all the coefficients to reconstruct a "good" signal?

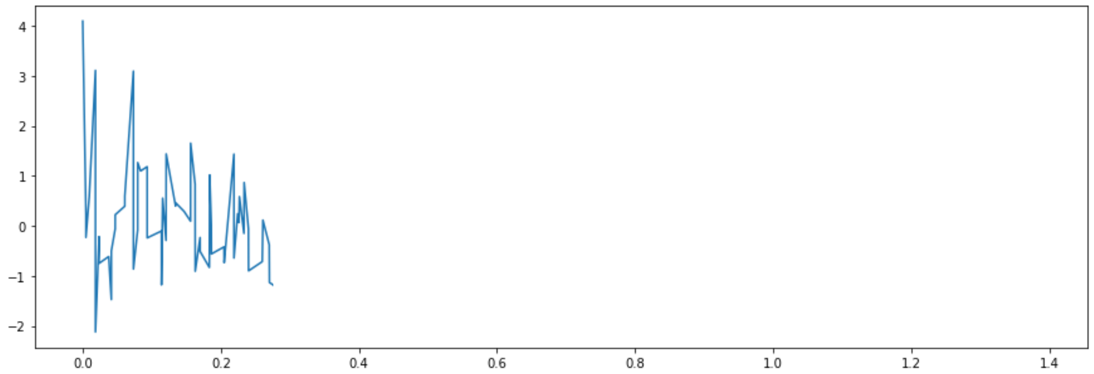




Going Back to Graph Spectral Theory

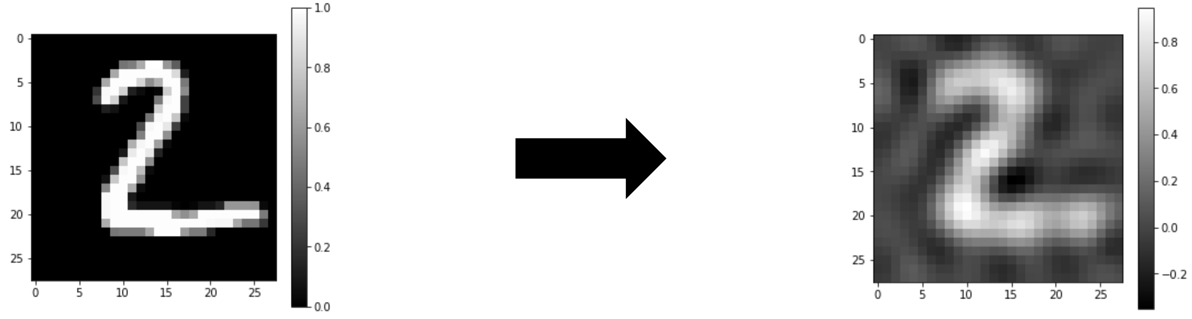
Do we need all the coefficients to reconstruct a “good” signal?

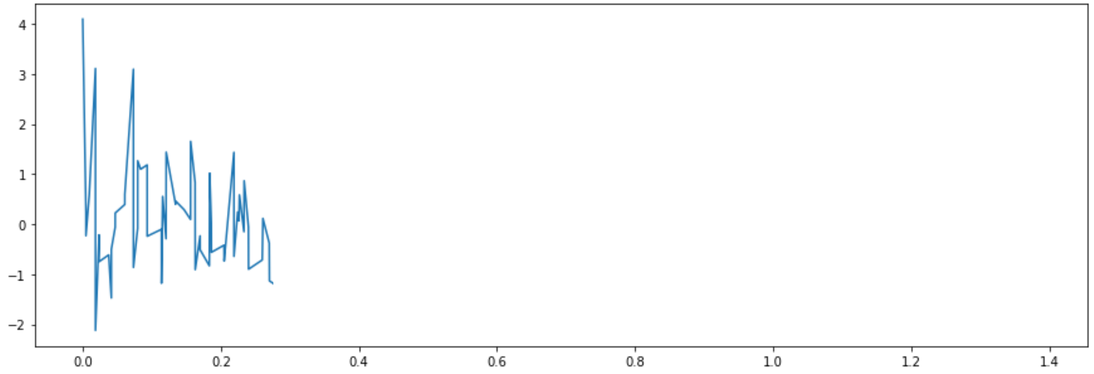




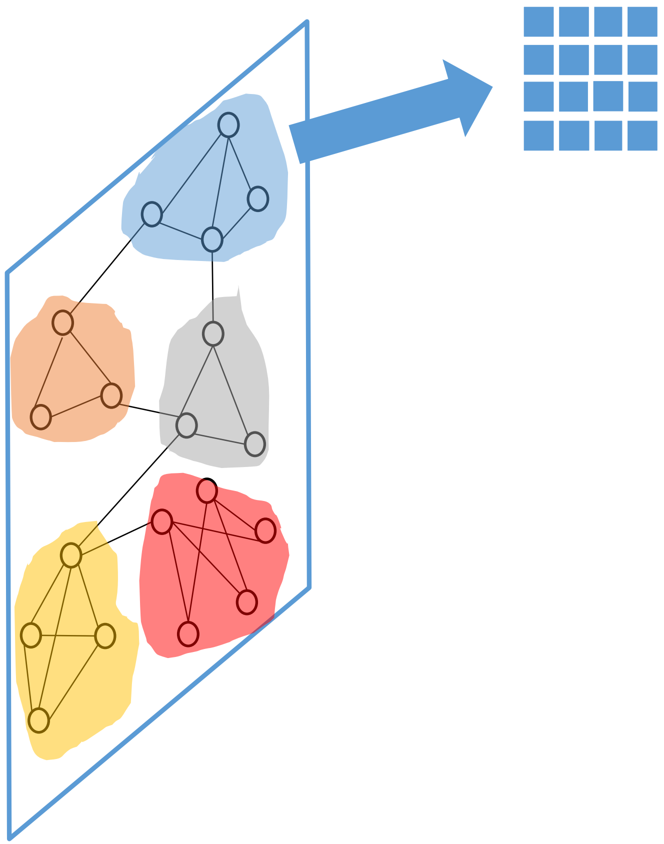
Going Back to Graph Spectral Theory

Do we need all the coefficients to reconstruct a "good" signal?



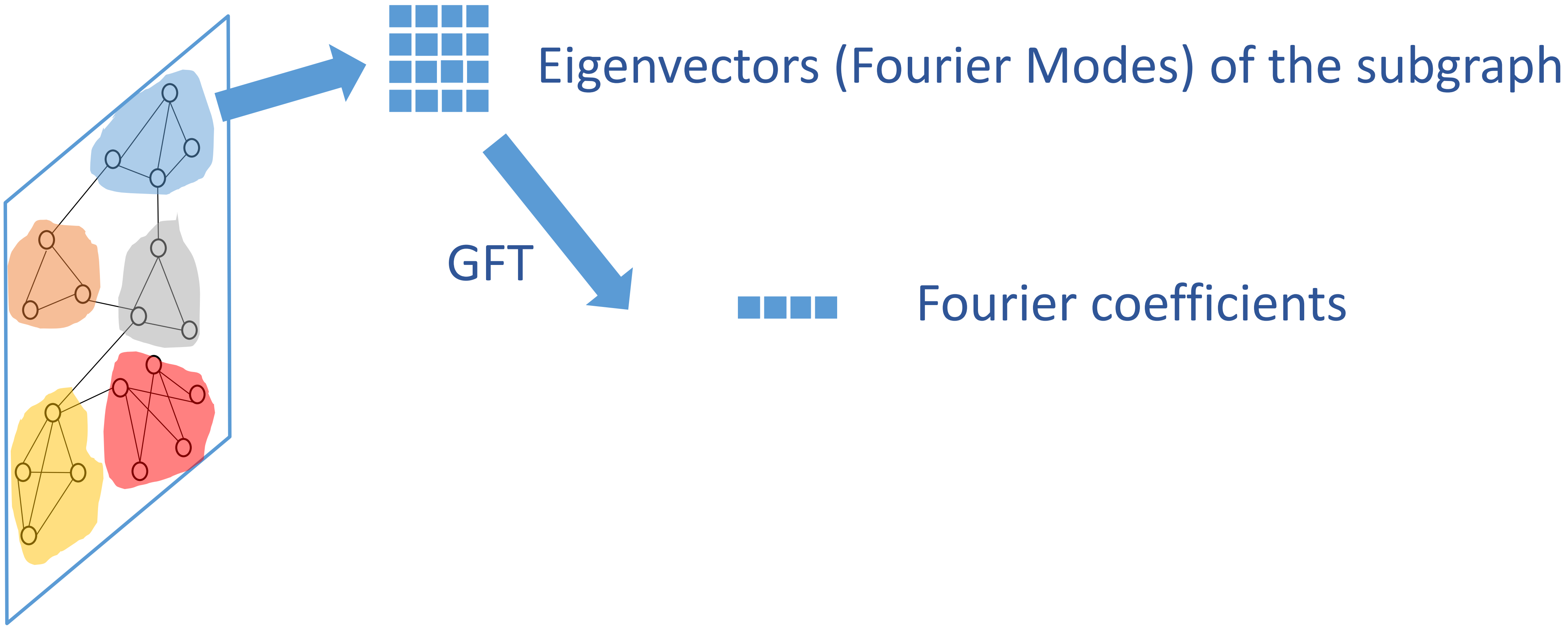


Eigenpooling: Truncated Fourier Coefficients

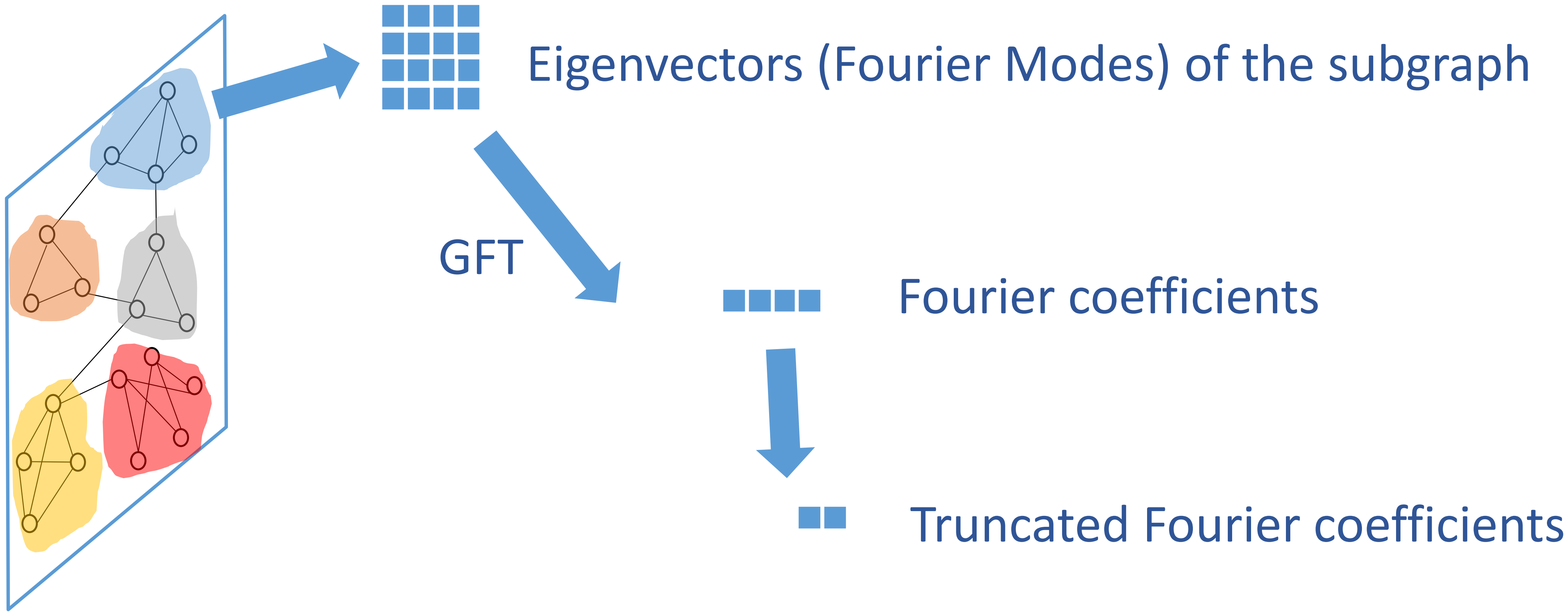


Eigenvectors (Fourier Modes) of the subgraph

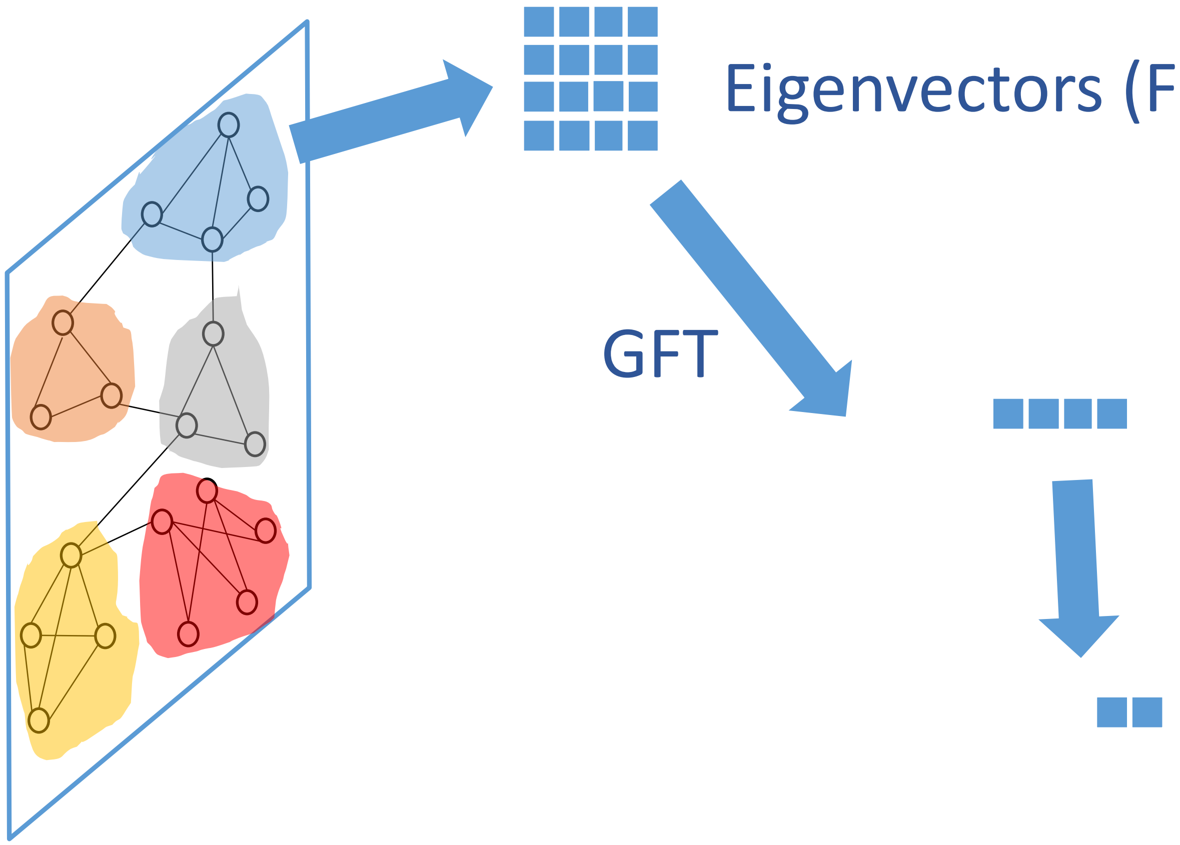
Eigenpooling: Truncated Fourier Coefficients



Eigenpooling: Truncated Fourier Coefficients



Eigenpooling: Truncated Fourier Coefficients



Eigenvectors (Fourier Modes) of the subgraph

GFT

Fourier coefficients

Truncated Fourier coefficients

New features for the subgraph (a node in the smaller graph)