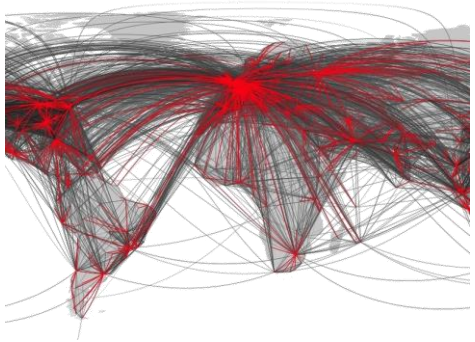
Graph Neural Networks: Models and Applications

Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang
Michigan State University

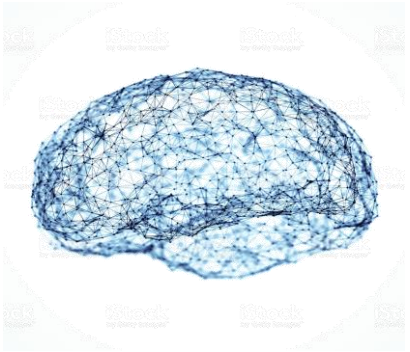
Tutorial website: Graph Neural Networks: Models and Applications

Data as Graphs - Explicit

Social Graphs

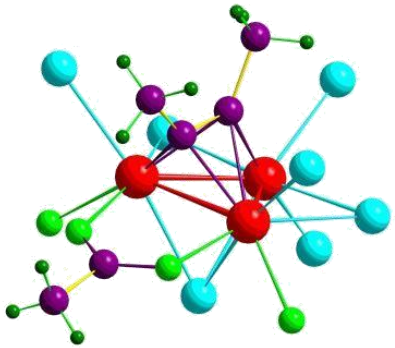


Transportation Graphs

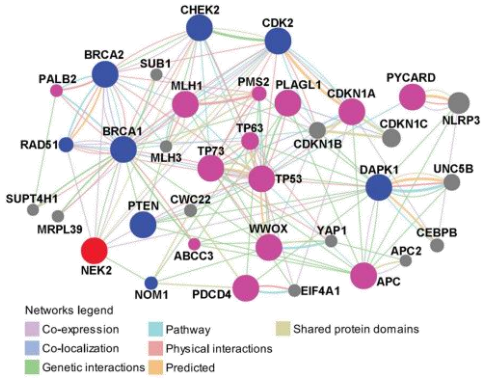


Brain Graphs

Web Graphs

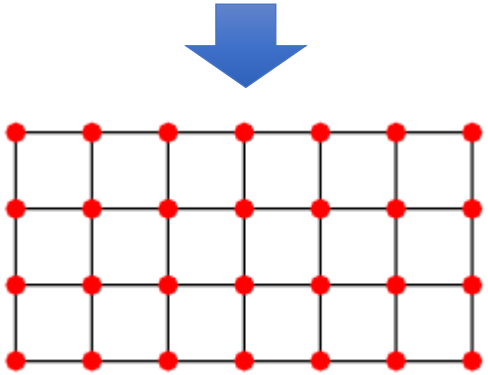


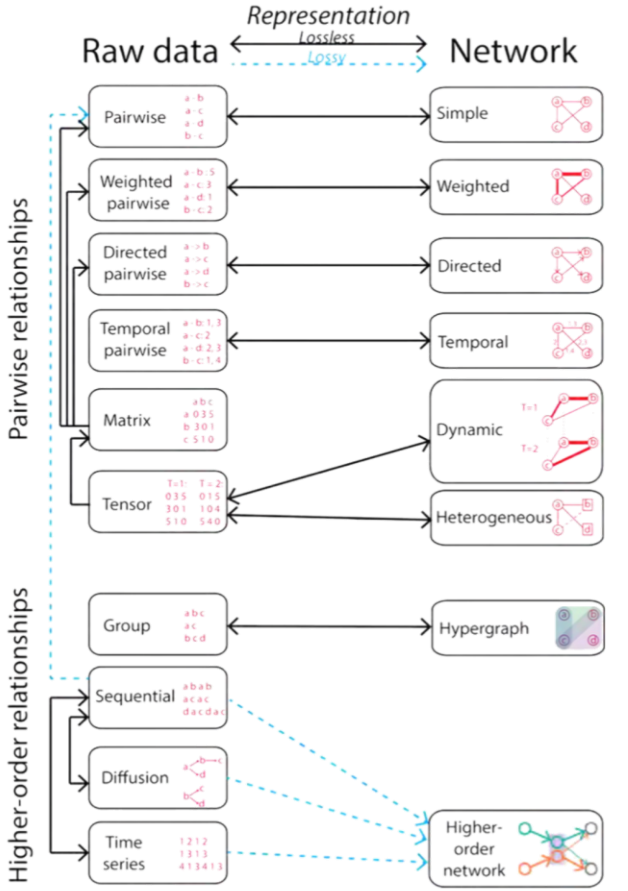
Molecular Graphs



Gene Graphs

Data as Graphs - Implicit





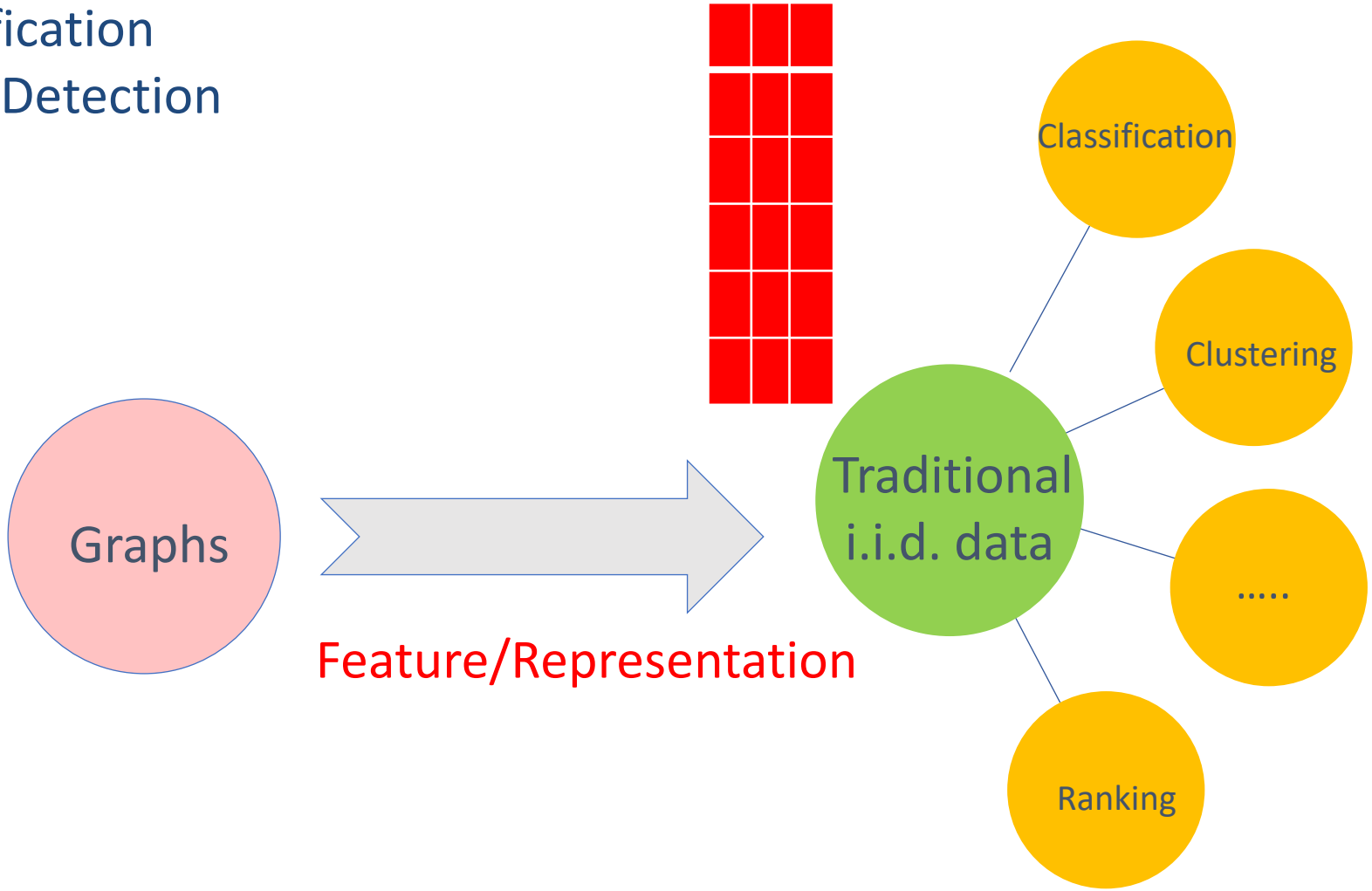
Jian Xu. Representing Big Data as Networks. PhD Dissertation, University of Notre Dame

ML on Graphs

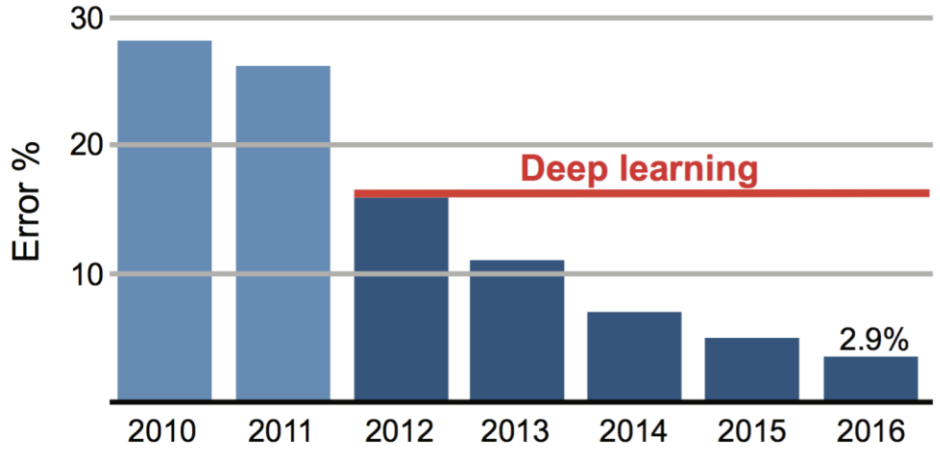
Numerous real-world problems can be summarized as a set of tasks on graphs

- Link prediction
- Node Classification
- Community Detection
- Ranking ….

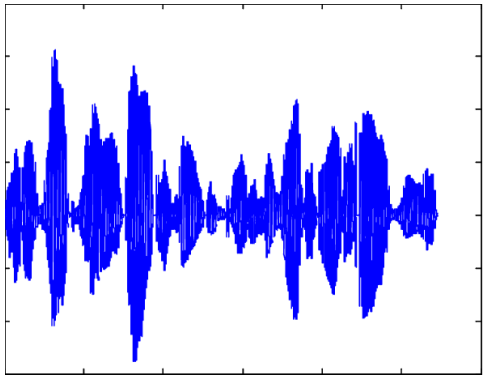
ML solutions



The Power of Deep Learning



IMAGENET

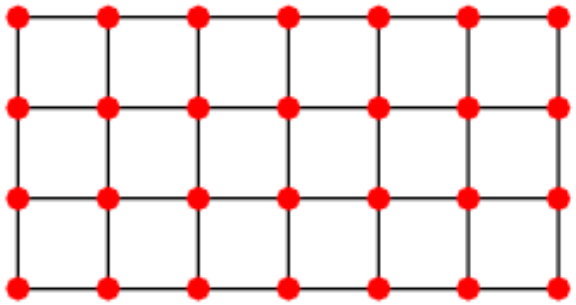


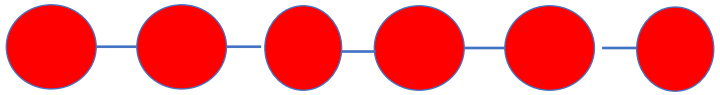
Acoustic model	Recog \ WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass –adapt	27.4	23.6
Deep Learning	1-pass –adapt	18.5	16.1

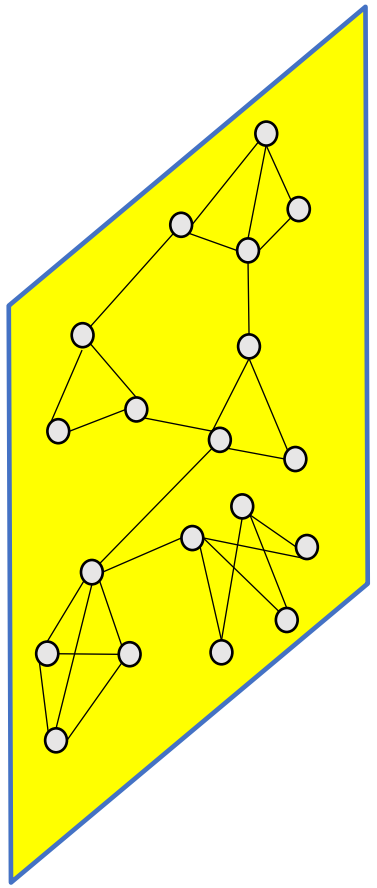
Deep Learning Meets Graphs: Challenges

Traditional DL is designed for simple grids or sequences

- CNNs for fixed-size images/grids
- RNNs for text/sequences



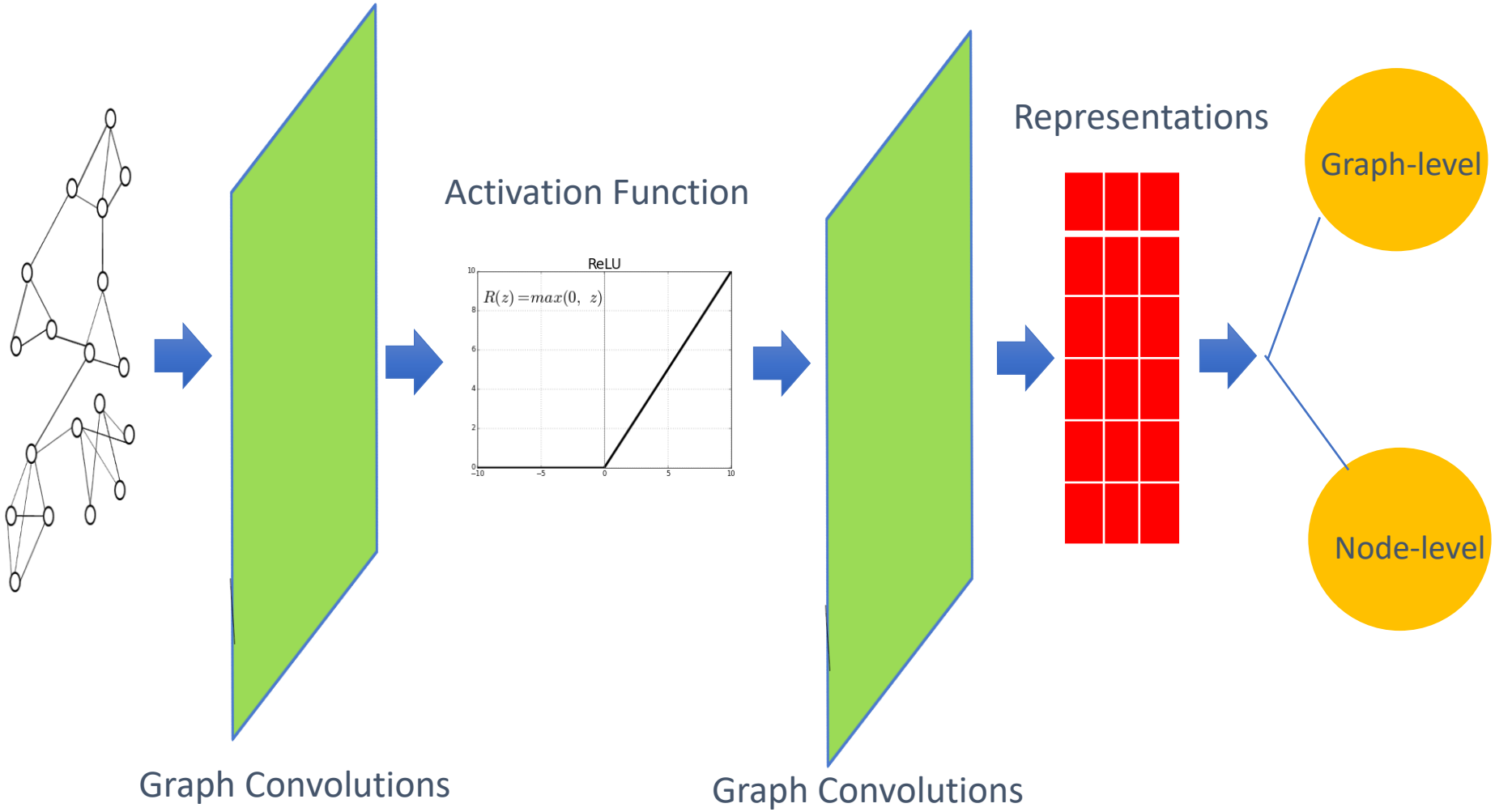




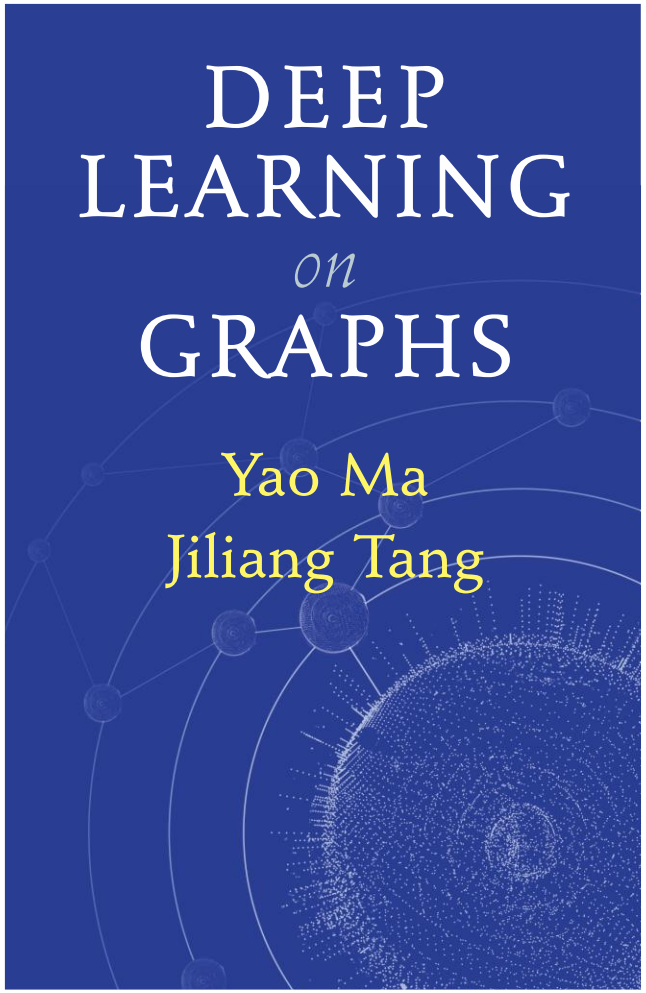
But nodes on graphs have different connections

- Arbitrary neighbor size
- Complex topological structure
- No fixed node ordering

Graph Neural Networks



Book: Deep Learning on Graphs

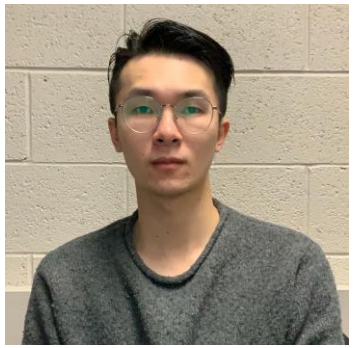


https://cse.msu.edu/~mayao4/dlg_book/

Chinese Version

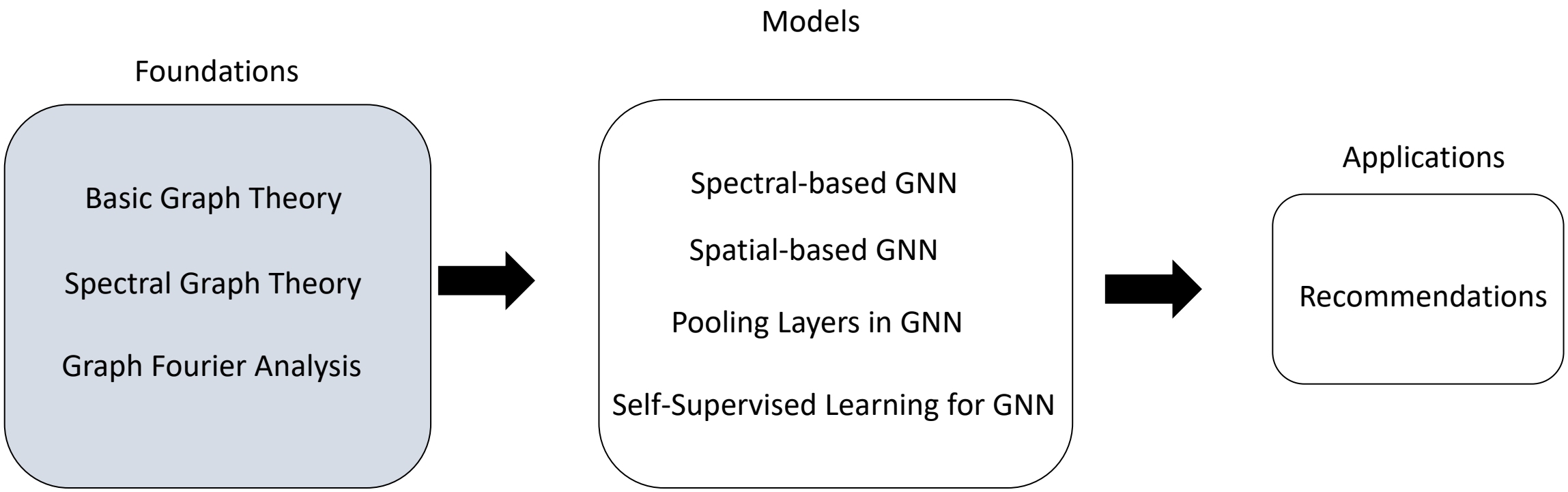
A Repository

GraphRobust: A repository about adversarial attacks and defenses on graph data

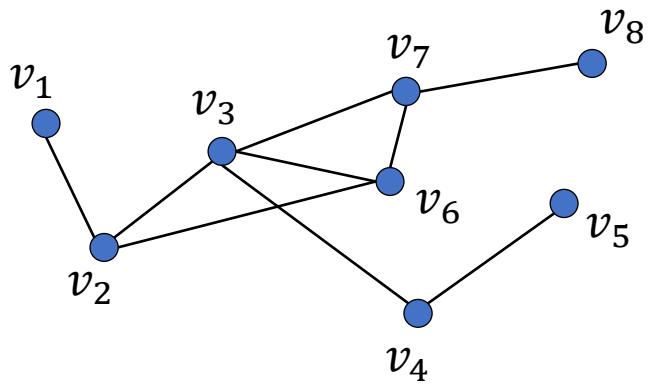


DeepRobust: A repository about general adversarial attacks and defenses

Tutorial Overview



Graphs and Graph Signals

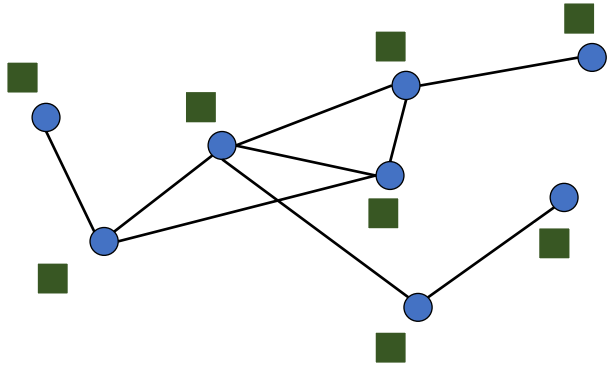


$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

Graphs and Graph Signals



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

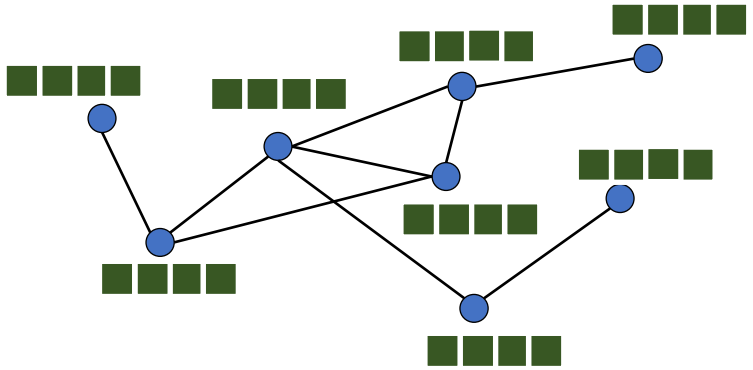
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

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

Graphs and Graph Signals



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

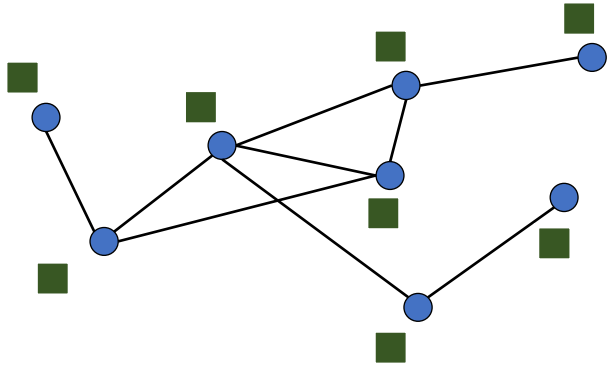
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

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

Graphs and Graph Signals



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

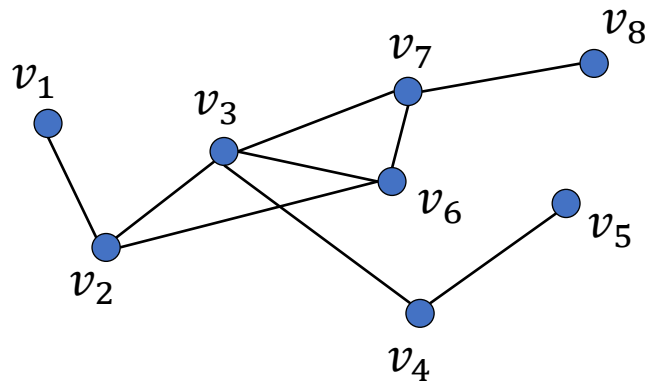
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

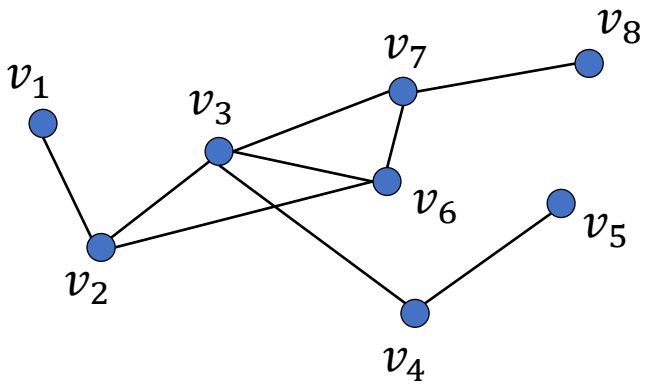
$$\mathcal{V} \longrightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

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

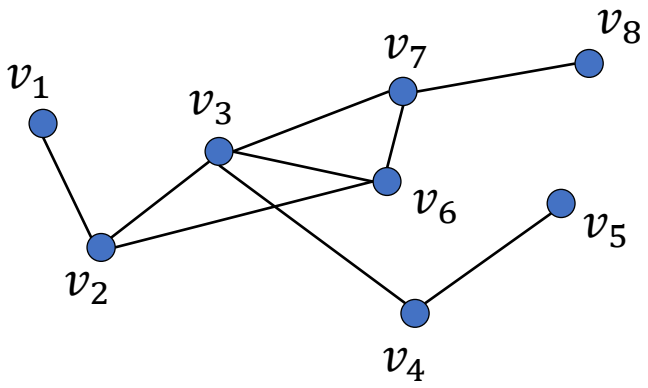
Adjacency Matrix

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

$\boldsymbol{A}$

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

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

$\boldsymbol{D}$

$-$

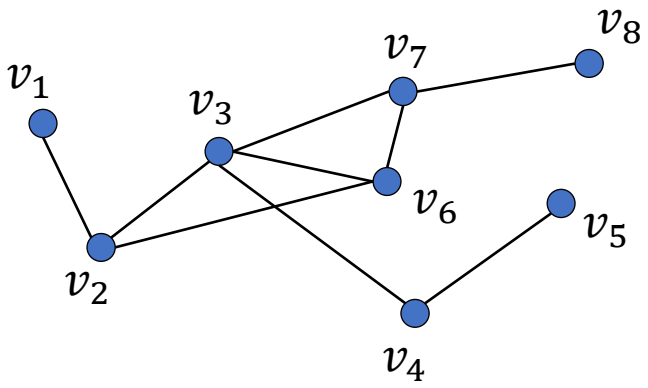
Adjacency Matrix

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

$\boldsymbol{A}$

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

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

$\boldsymbol{D}$

$-$

Adjacency Matrix

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

$\boldsymbol{A}$

$=$

Laplacian Matrix

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

$\boldsymbol{L}$

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:

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

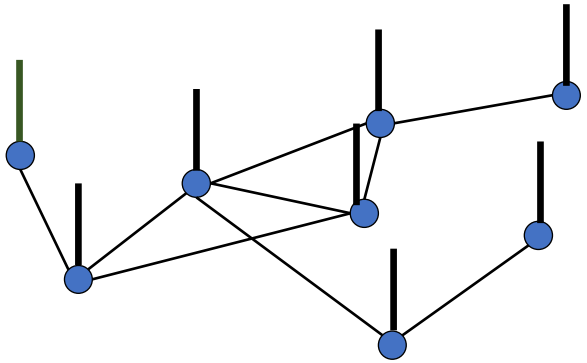
"Smoothness" or "Frequency" of the signal *f*

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:

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

"Smoothness" or "Frequency" of the signal *f*

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

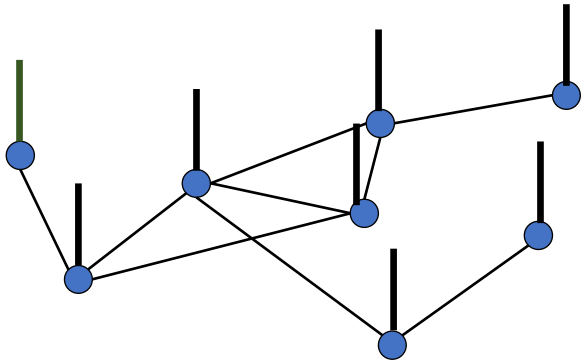
$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

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

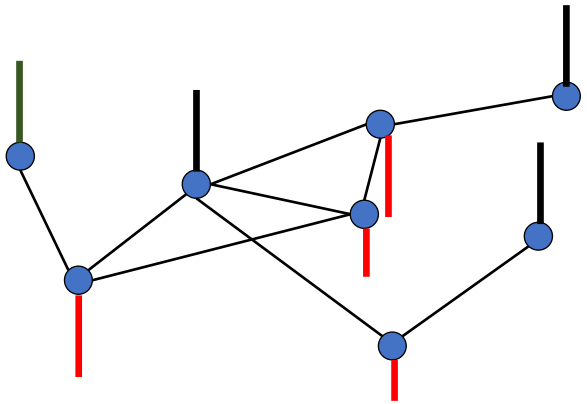
Laplacian quadratic form:

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

"Smoothness" or "Frequency" of the signal *f*



Low frequency graph signal



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:

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

Eigen-decomposition of Laplacian Matrix

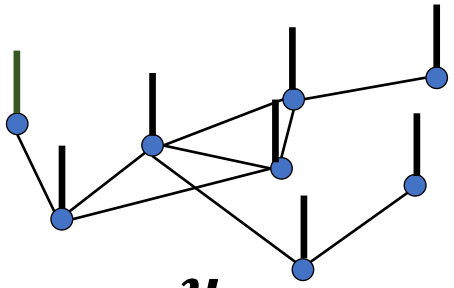
Laplacian matrix has a complete set of orthonormal eigenvectors:

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

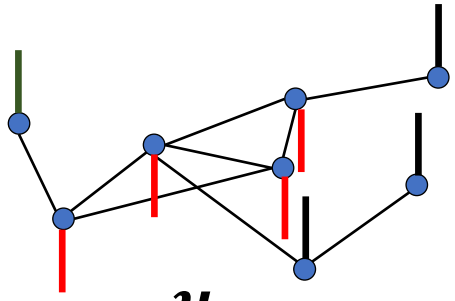
Eigenvalues are sorted non-decreasingly:

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

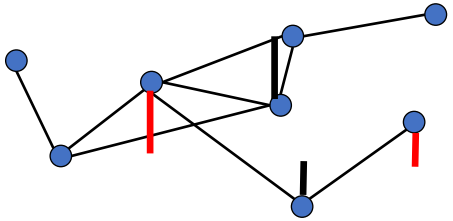
Eigenvectors as Graph Signals



$\boldsymbol{u}_0$



$\boldsymbol{u}_1$



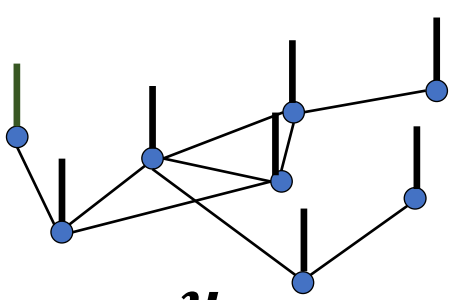
$\boldsymbol{u}_7$

Eigenvectors as Graph Signals

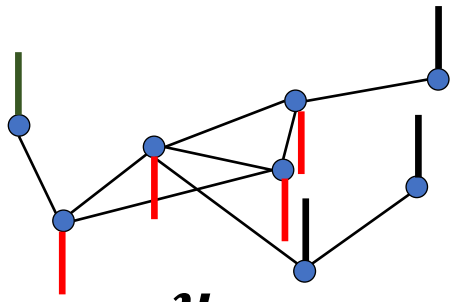
The frequency of an eigenvector of Laplacian matrix is its corresponding eigenvalue:

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

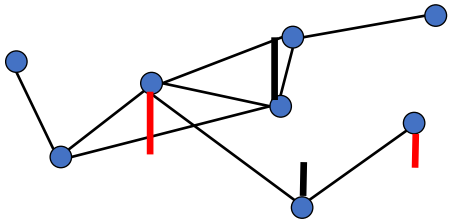
Frequency of the signal $\boldsymbol{u}_i$



$\boldsymbol{u}_0$



$\boldsymbol{u}_1$



$\boldsymbol{u}_7$

$\boldsymbol{u_0^T L u_0 = \lambda_0 = 0}$

$\boldsymbol{u_1^T L u_1 = \lambda_1}$

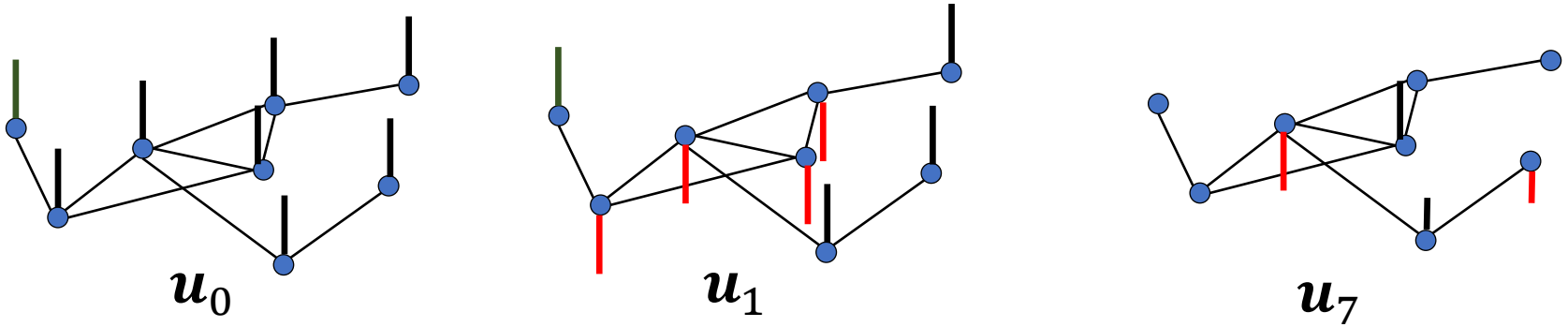
$\boldsymbol{u_7^T L u_7 = \lambda_7}$

Eigenvectors as Graph Signals

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

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

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



$\boldsymbol{u}_0^T \boldsymbol{L} \boldsymbol{u}_0 = \boldsymbol{\lambda}_0 = \mathbf{0}$ $\quad$ $\boldsymbol{u}_1^T \boldsymbol{L} \boldsymbol{u}_1 = \boldsymbol{\lambda}_1$ $\quad$ $\boldsymbol{u}_7^T \boldsymbol{L} \boldsymbol{u}_7 = \boldsymbol{\lambda}_7$

Graph Fourier Transform

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

A signal f can be written as graph Fourier series:

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

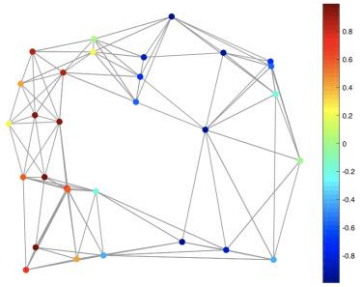
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients

Graph Fourier Transform (GFT)

A signal f can be written as graph Fourier series:

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

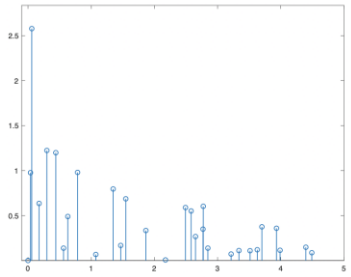
u_i: graph Fourier mode $\quad$ λ_i: frequency $\quad$ $\hat{f}_i$: graph Fourier coefficients



Spatial domain: $\boldsymbol{f}$

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

Decompose signal f



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

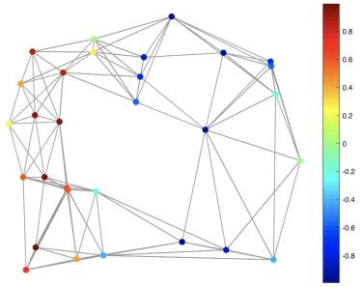
The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*

Inverse Graph Fourier Transform (IGFT)

A signal f can be written as graph Fourier series:

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

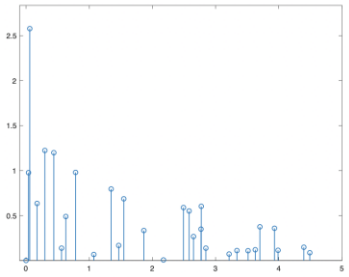
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients



Spatial domain: $\boldsymbol{f}$

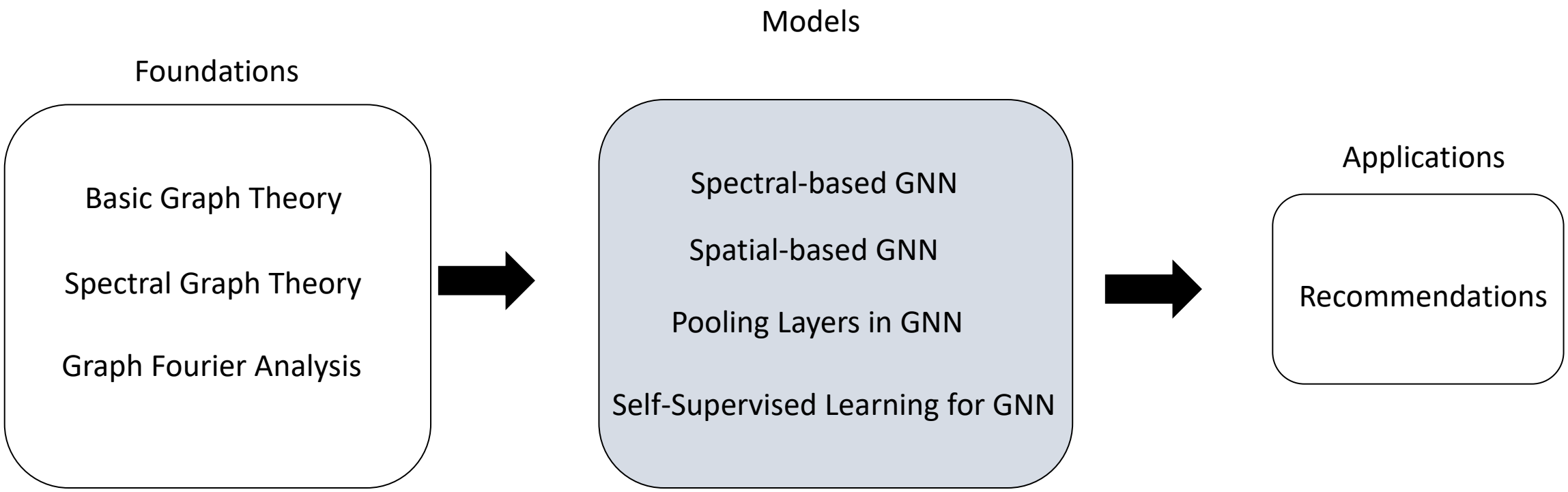
$\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$

Reconstruct signal $\boldsymbol{f}$



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

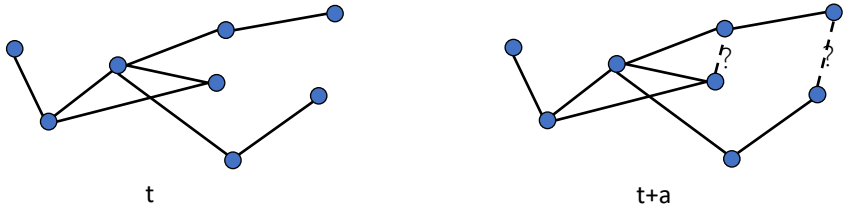
Tutorial Overview



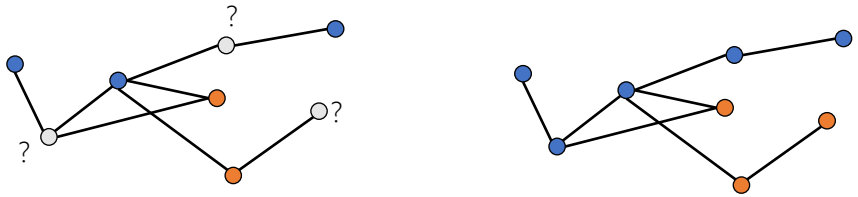
Tasks on Graph-Structured Data

Node-level

Link Prediction

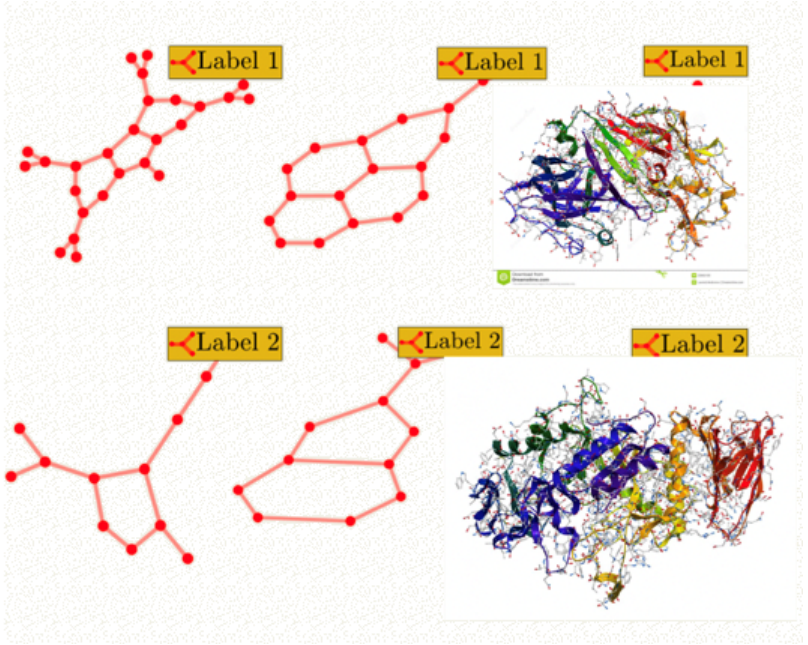


Node Classification



Graph-level

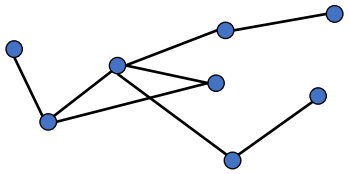
Graph Classification



Tasks on Graph-Structured Data

Node-level

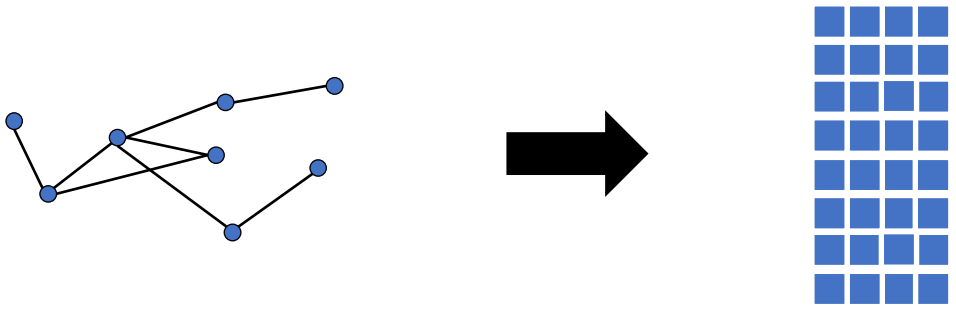
Graph-level



Tasks on Graph-Structured Data

Node-level

Graph-level

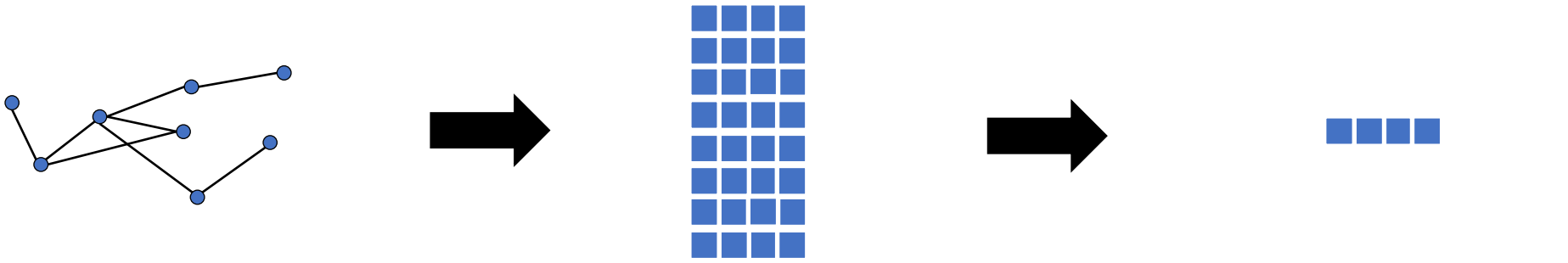


Node Representations

Tasks on Graph-Structured Data

Node-level

Graph-level



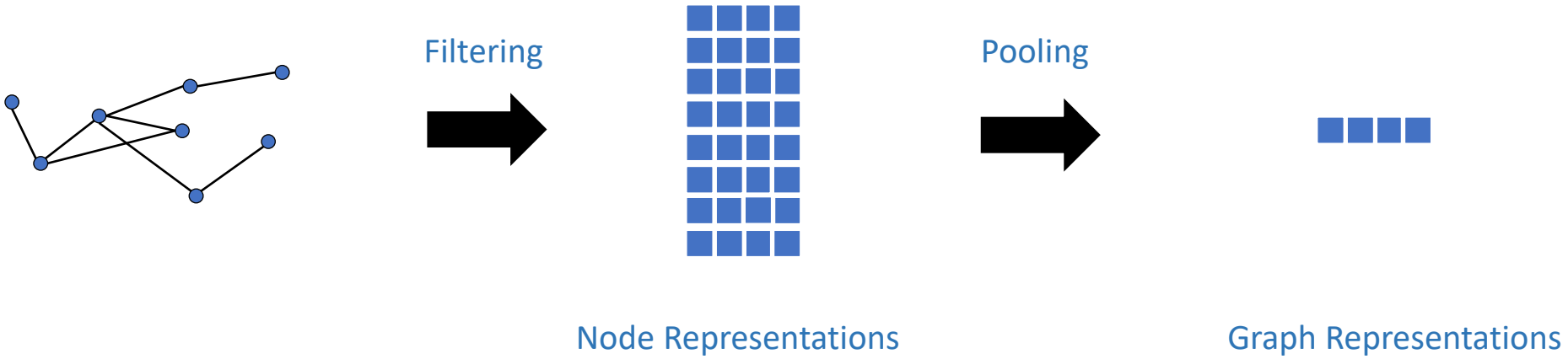
Node Representations

Graph Representation

Tasks on Graph-Structured Data

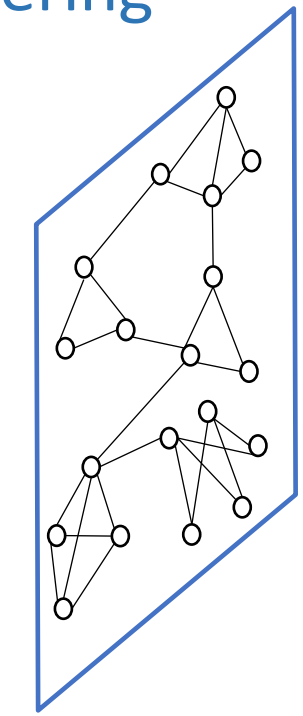
Node-level

Graph-level



Two Main Operations in GNN

Graph Filtering

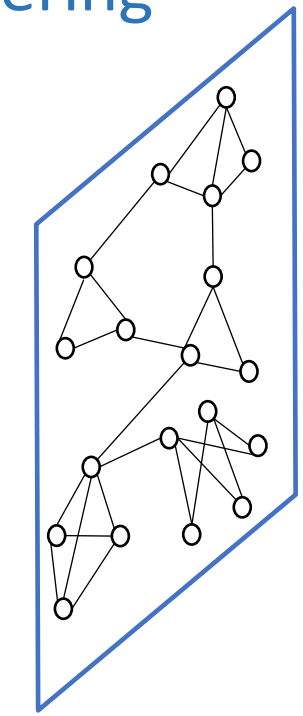


Graph Filtering

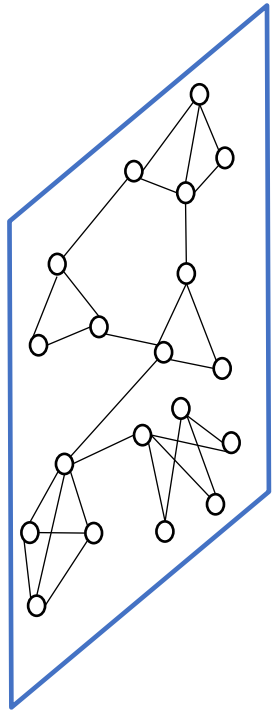
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

Two Main Operations in GNN

Graph Filtering



Graph Filtering

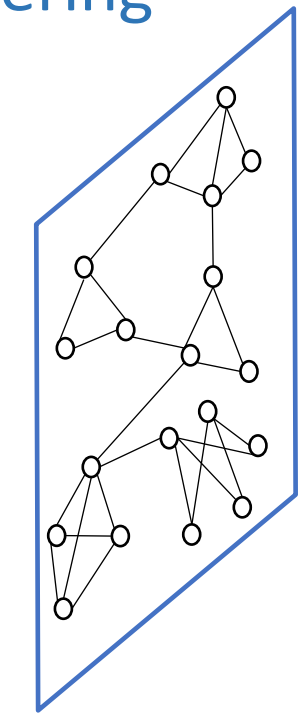


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

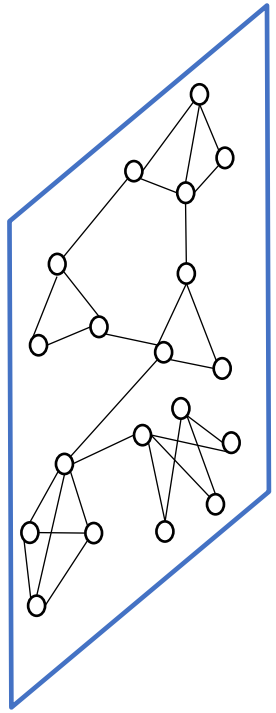
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X}_f \in \mathbb{R}^{n \times d_{new}}$$

Two Main Operations in GNN

Graph Filtering



Graph Filtering



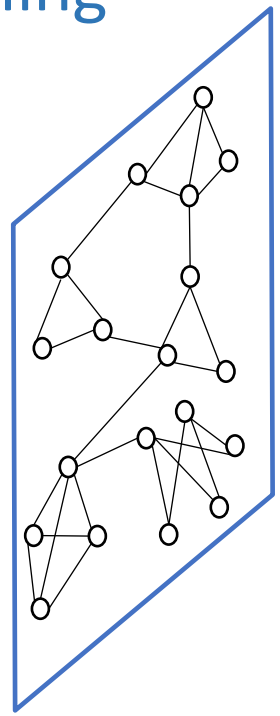
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Graph filtering refines the node features

Two Main Operations in GNN

Graph Pooling

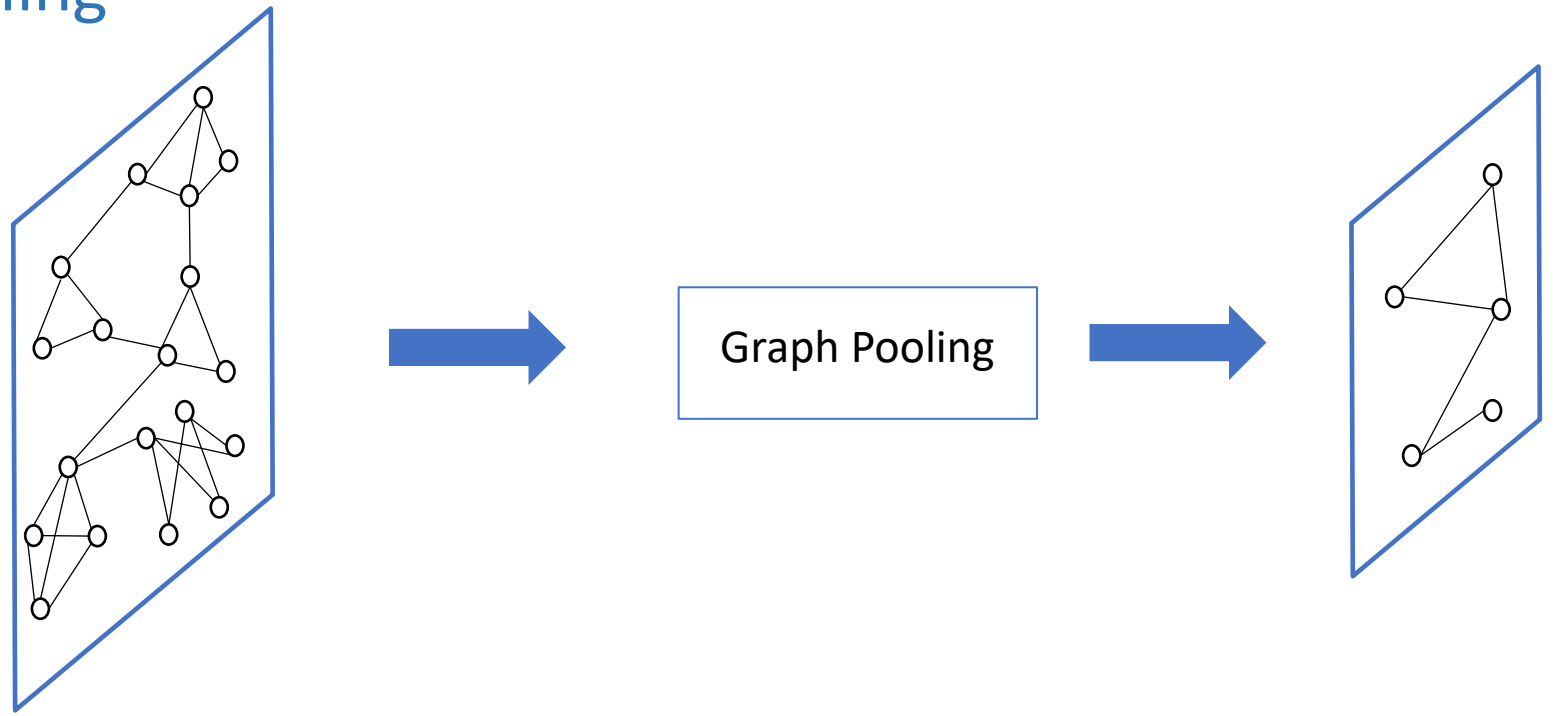


Graph Pooling

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

Two Main Operations in GNN

Graph Pooling

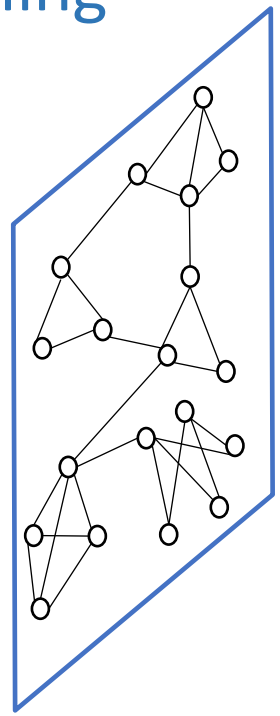


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$$

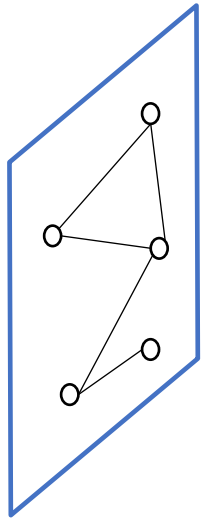
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{X}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Two Main Operations in GNN

Graph Pooling



Graph Pooling



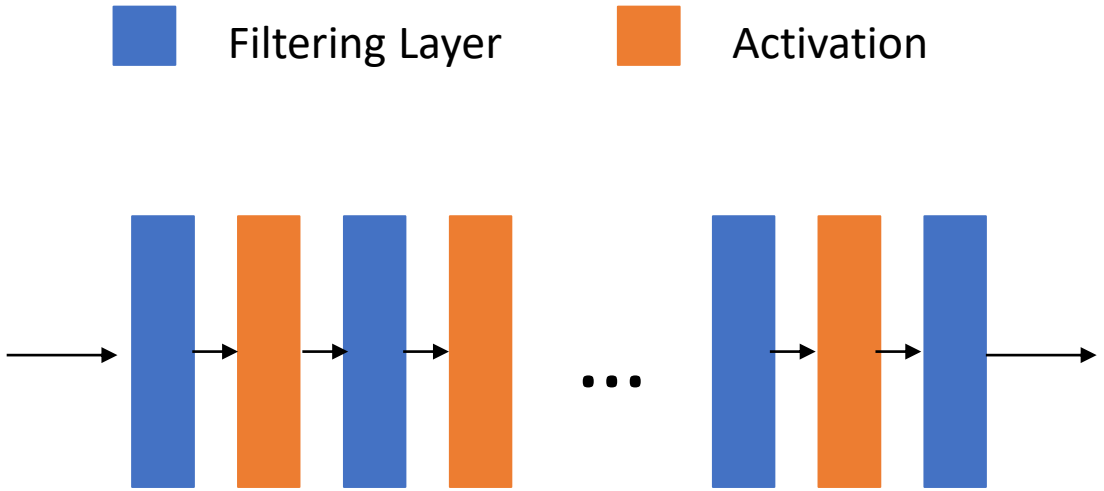
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Graph pooling generates a smaller graph

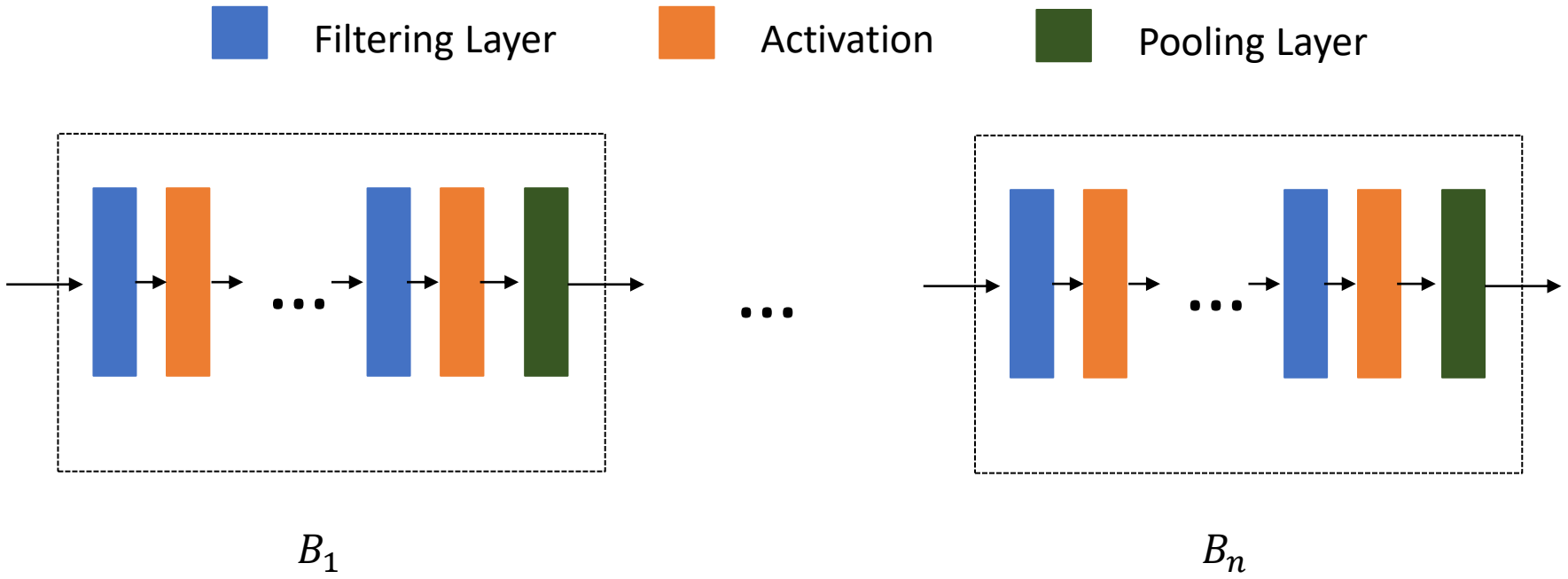
General GNN Framework

For node-level tasks

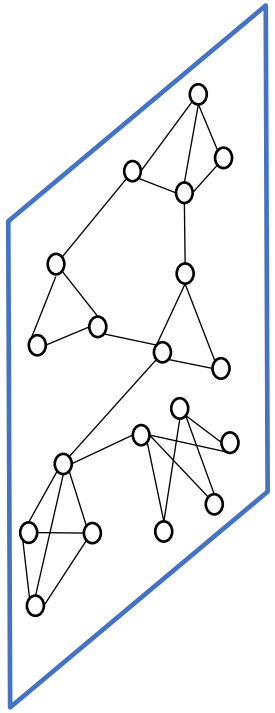


General GNN Framework

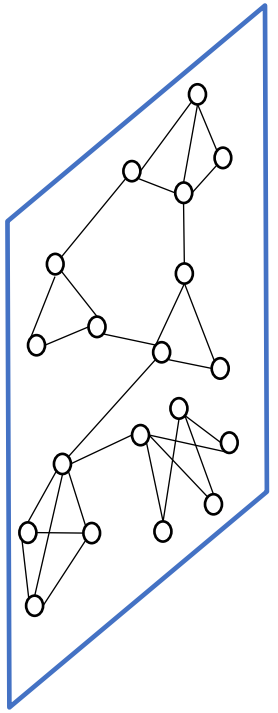
For graph-level tasks



Graph Filtering Operation



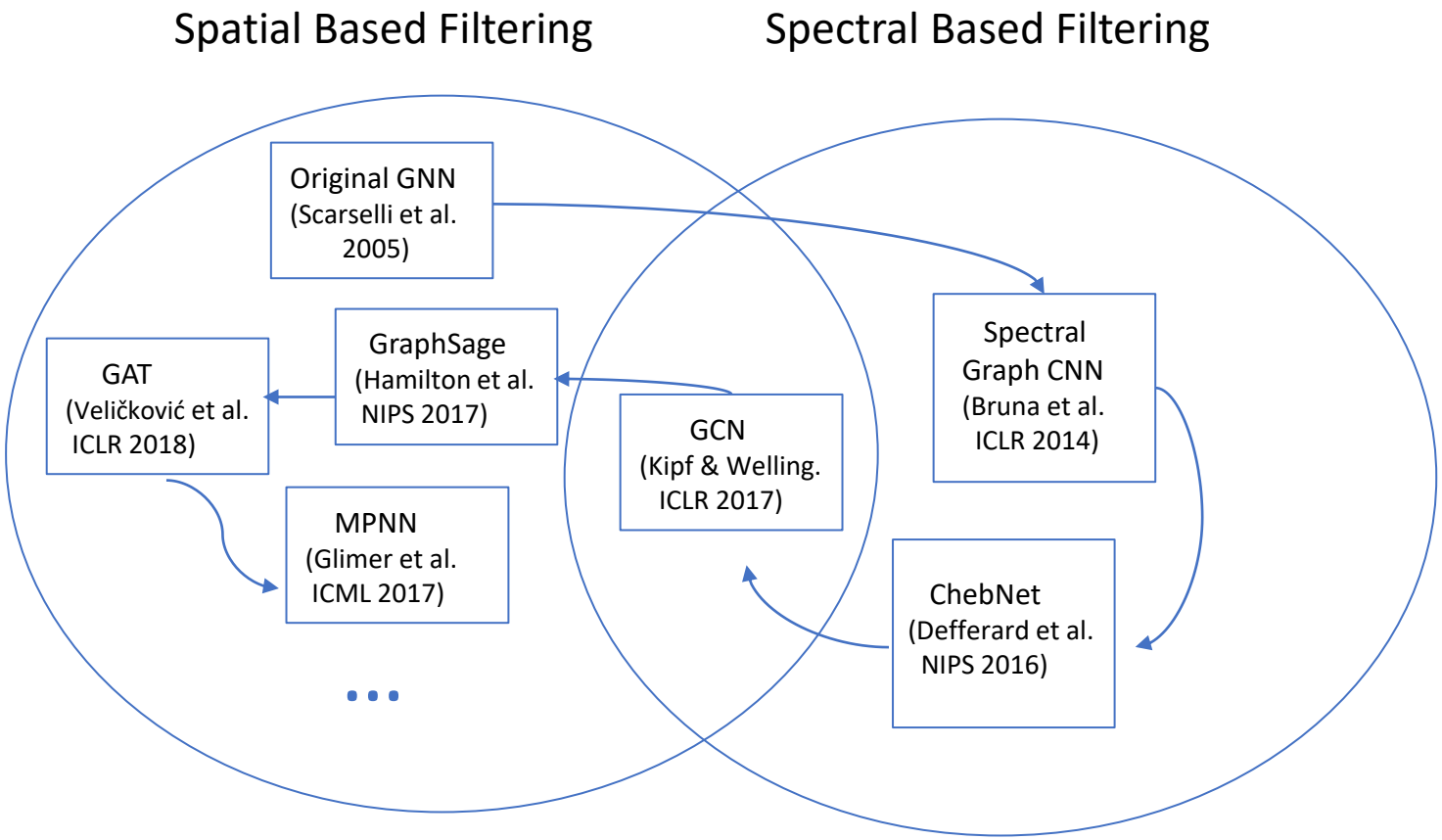
Graph Filtering



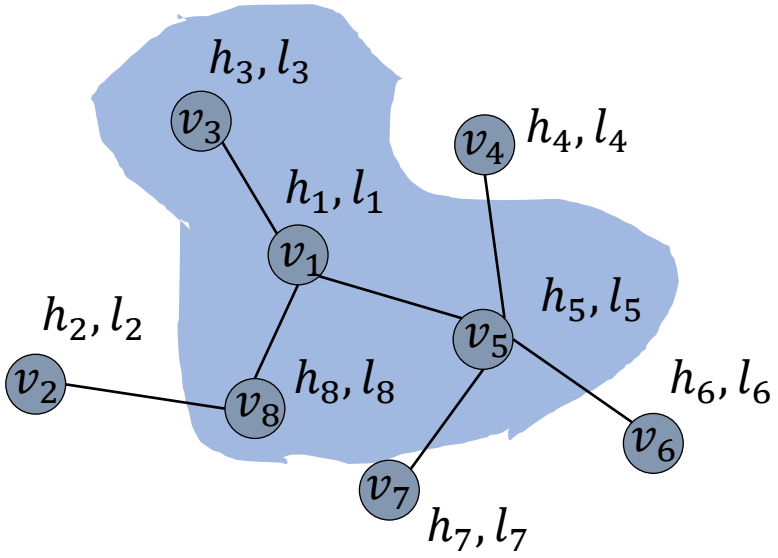
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Two Types of Graph Filtering Operation



Graph Filtering in the First GNN Paper

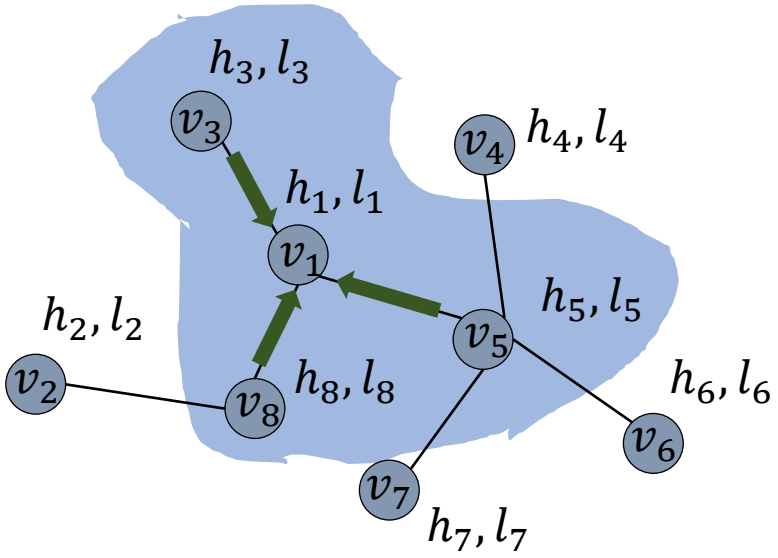


h_i: The hidden features

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.

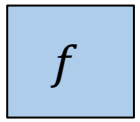
$f(\cdot)$: Feedforward neural network.

Graph Spectral Filtering for Graph Signal

Recall:

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

Filter a graph signal f:

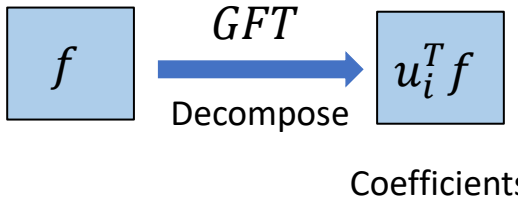


Graph Spectral Filtering for Graph Signal

Recall:

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

Filter a graph signal f:

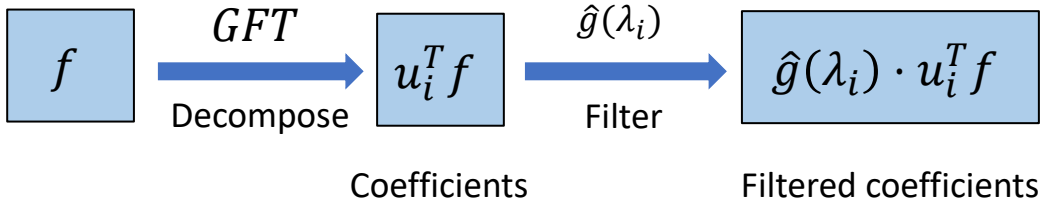


Graph Spectral Filtering for Graph Signal

Recall:

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

Filter a graph signal f:



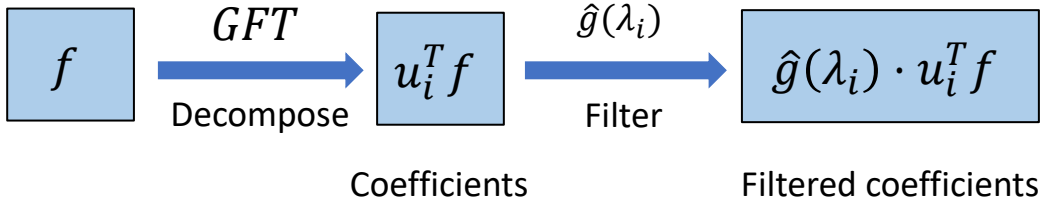
Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Graph Spectral Filtering for Graph Signal

Recall:

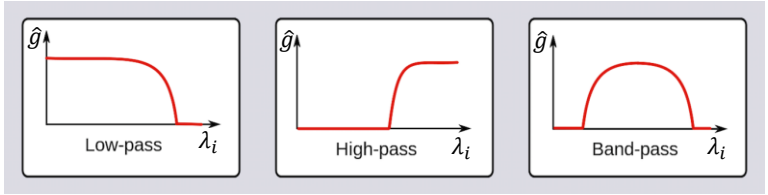
$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



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

Example:

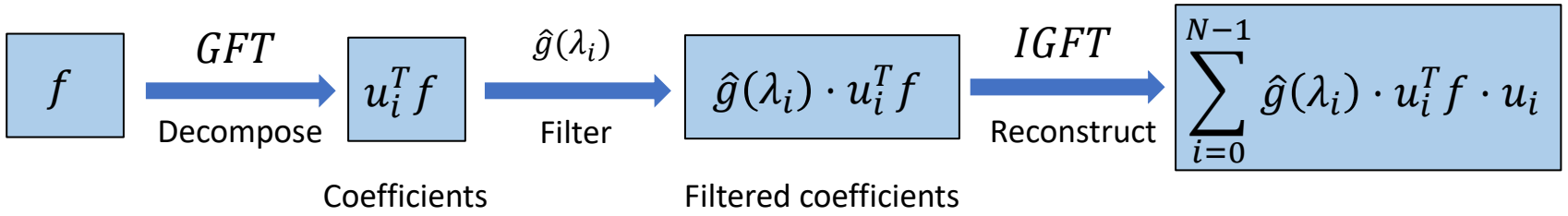


Graph Spectral Filtering for Graph Signal

Recall:

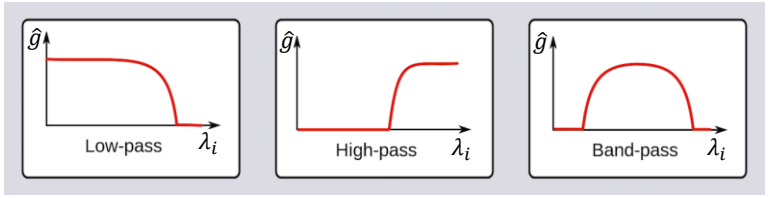
$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



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

Example:

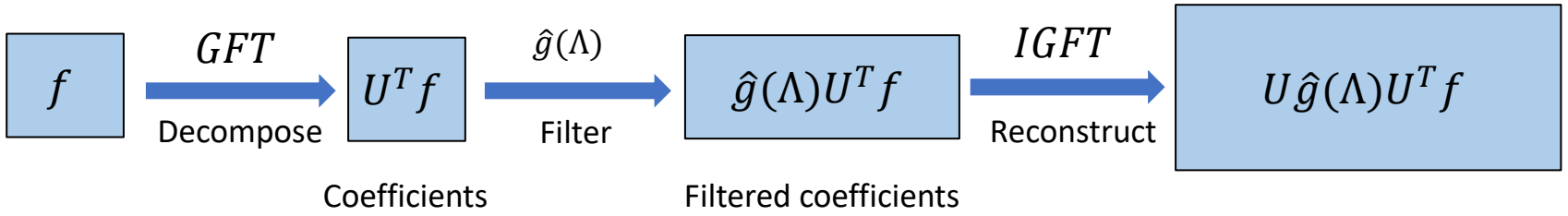


Graph Spectral Filtering for Graph Signal

Recall:

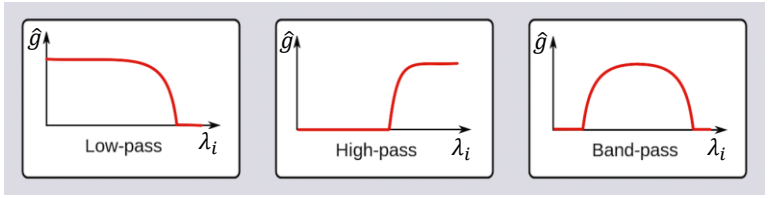
$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



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

Example:

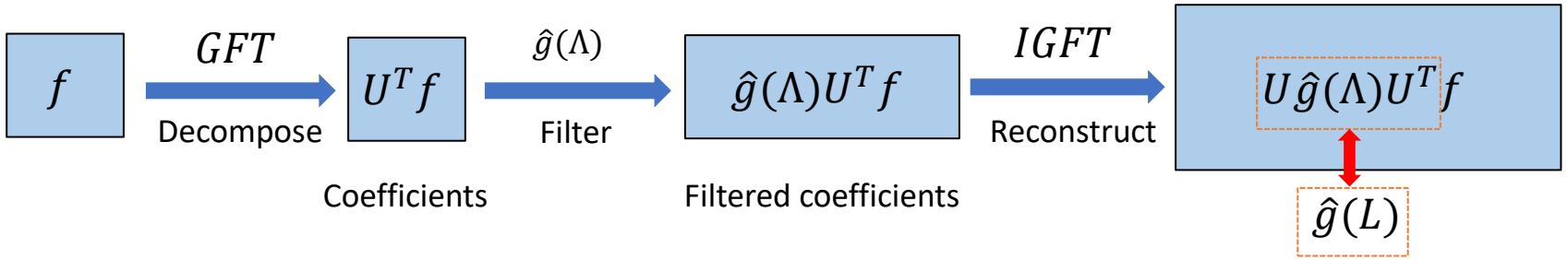


Graph Spectral Filtering for Graph Signal

Recall:

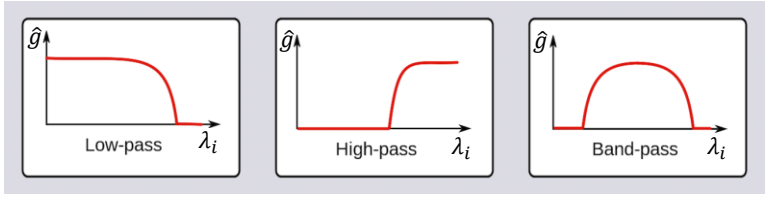
$$GFT: \hat{f} = U^T f \qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



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

Example:

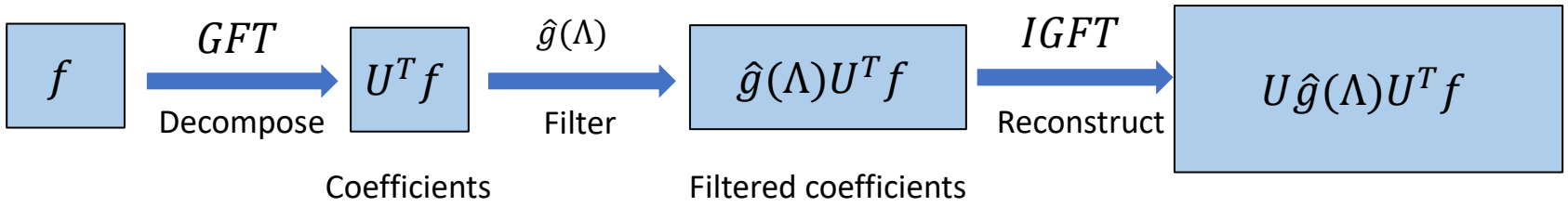


Graph Spectral Filtering for Graph Signal

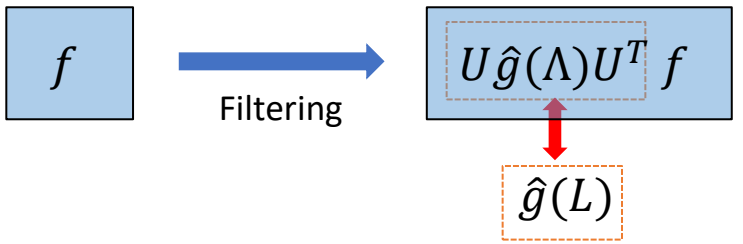
Recall:

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

Filter a graph signal f:



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



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!

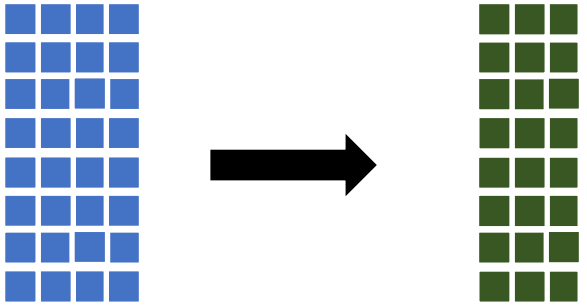
Graph Spectral Filtering for GNN

How to design the filter?

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

How to deal with multi-channel signals?

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



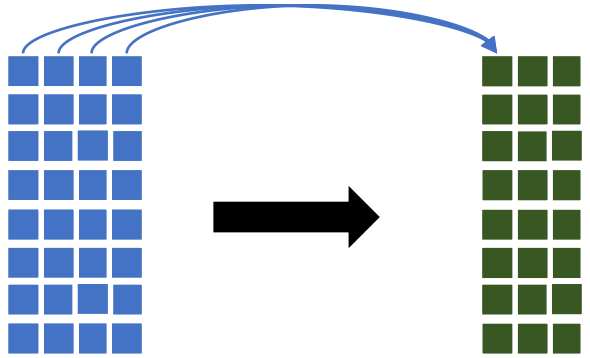
Graph Spectral Filtering for GNN

How to design the filter?

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

How to deal with multi-channel signals?

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



Each input channel contributes to each output channel

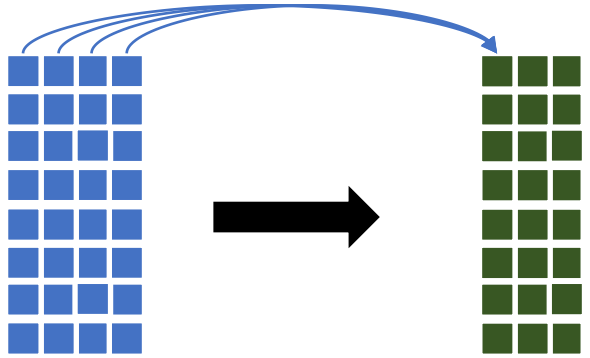
Graph Spectral Filtering for GNN

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

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

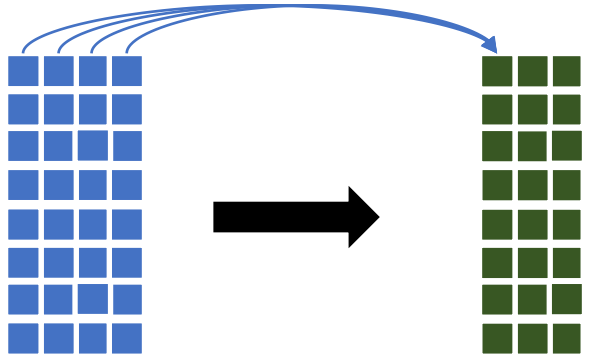
Graph Spectral Filtering for GNN

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

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

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

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

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

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

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

$d_2 \times d_1 \times K$ parameters

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

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

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

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

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

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 $\theta = \theta_0 = -\theta_1$

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

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

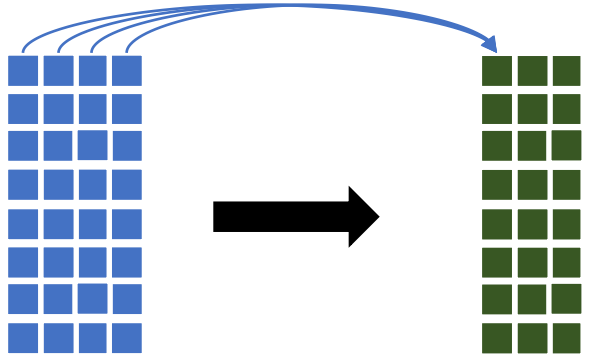
Apply a renormalization trick

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

GCN for Multi-channel Signal

Recall:

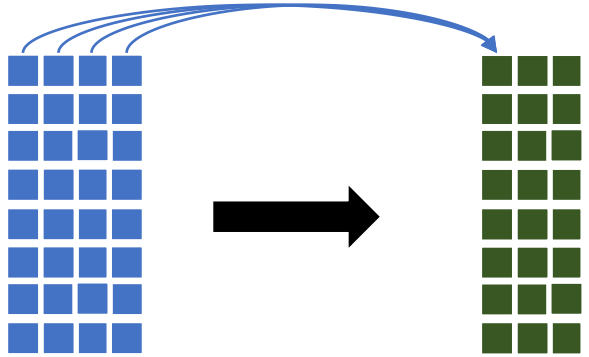
$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]}_{\text{Filter each input channel}} \quad i = 1, \dots d_2$$



GCN for Multi-channel Signal

Recall:

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



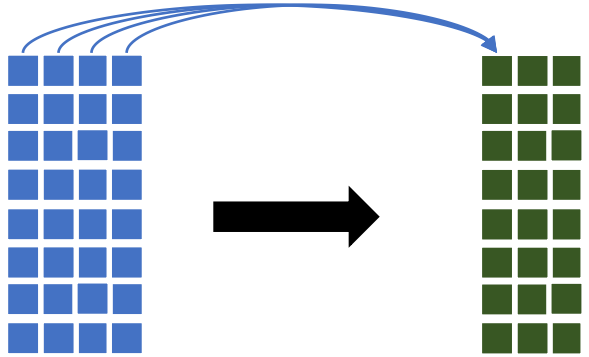
For GCN:

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

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:

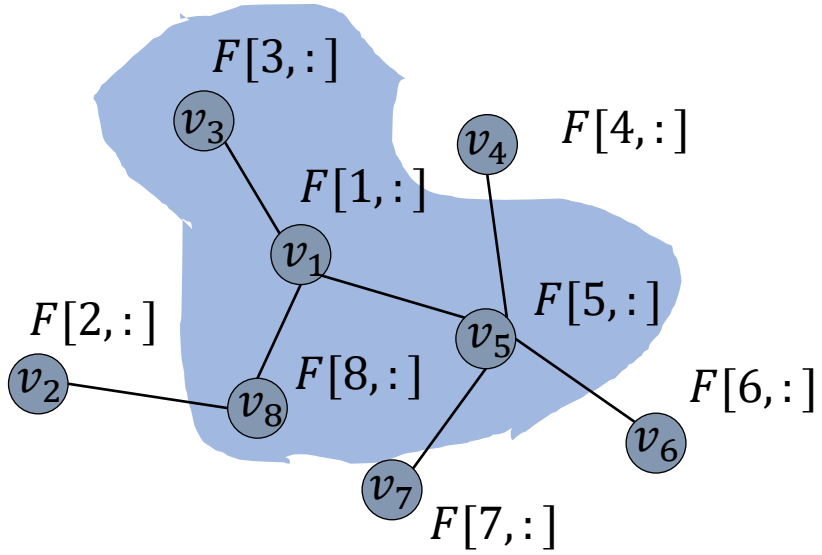
$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\theta_{ji}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})\boldsymbol{F}_{in}[:,j]}_{\text{GCN filter}} \quad i = 1, \dots d_2$$

In matrix form:

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

A Spatial View of GCN Filter

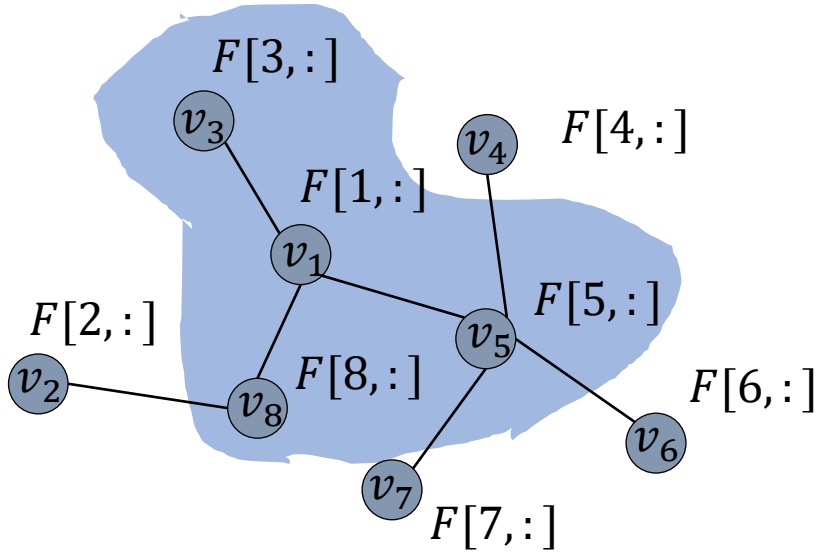
Denote $C = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$



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, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$



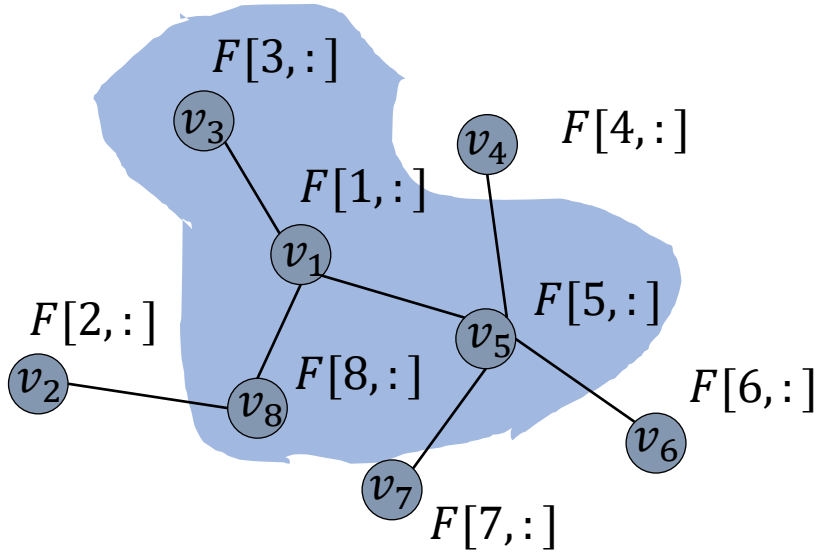
A Spatial View of GCN Filter

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 $v_j \notin N(v_i) \cup \{v_i\}$



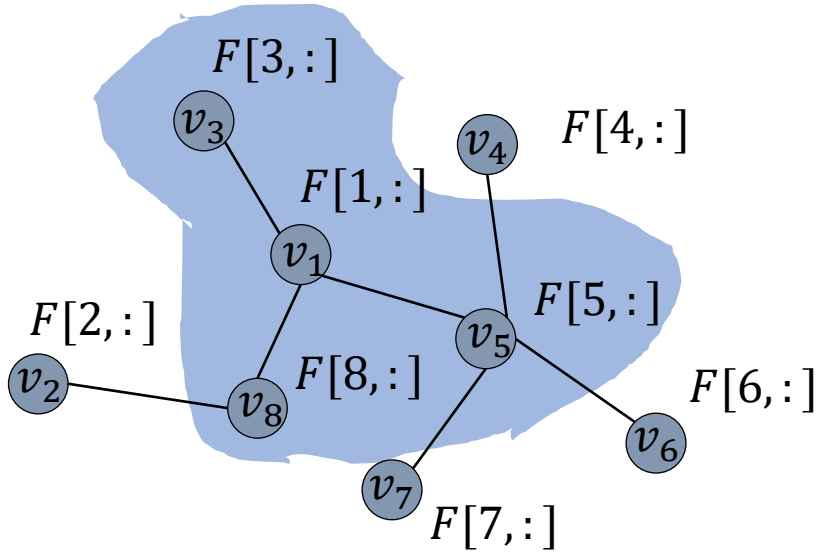
A Spatial View of GCN Filter

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,

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

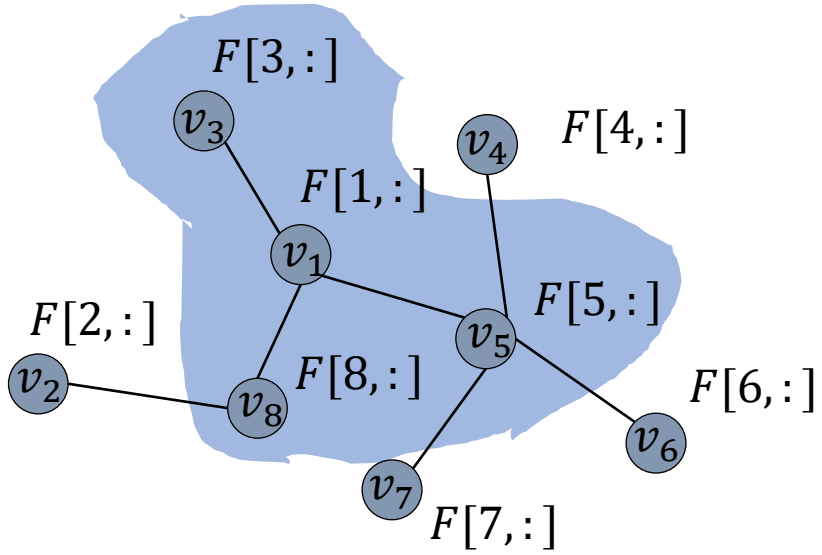
A Spatial View of GCN Filter

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,

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

Feature transformation

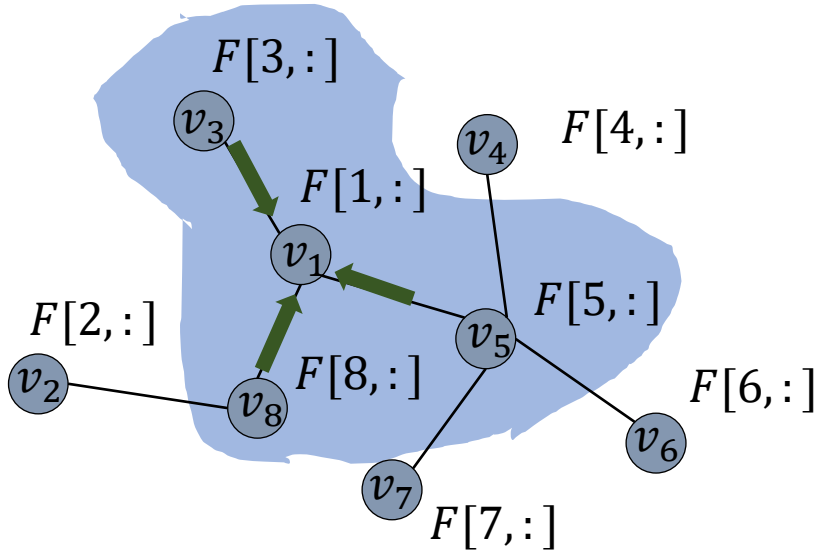
A Spatial View of GCN Filter

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:

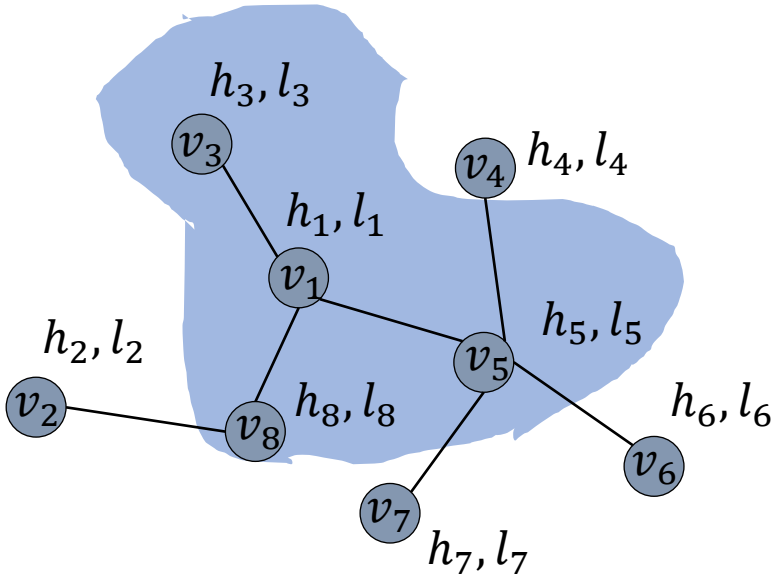
- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$



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



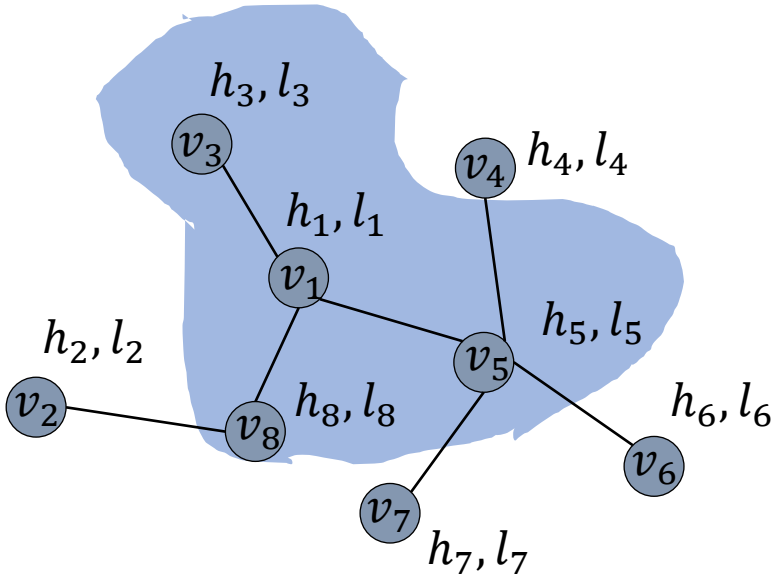
GCN: k-th layer

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

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



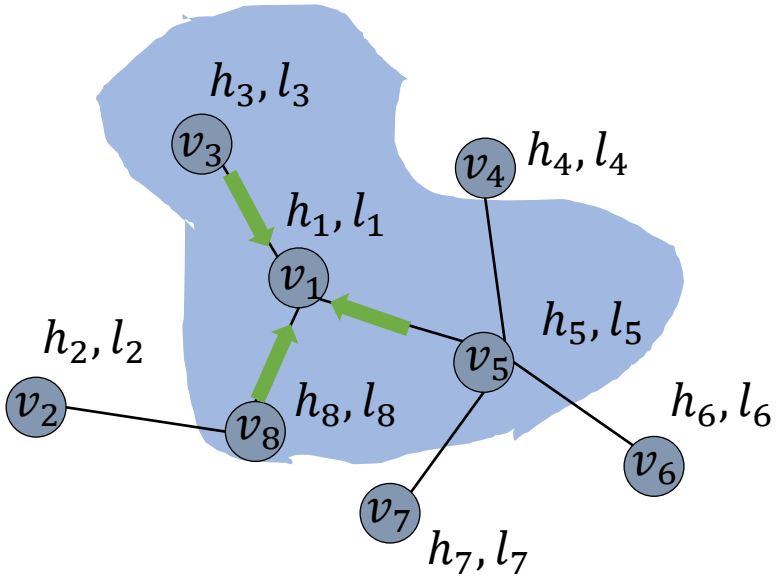
GCN: k-th layer

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

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



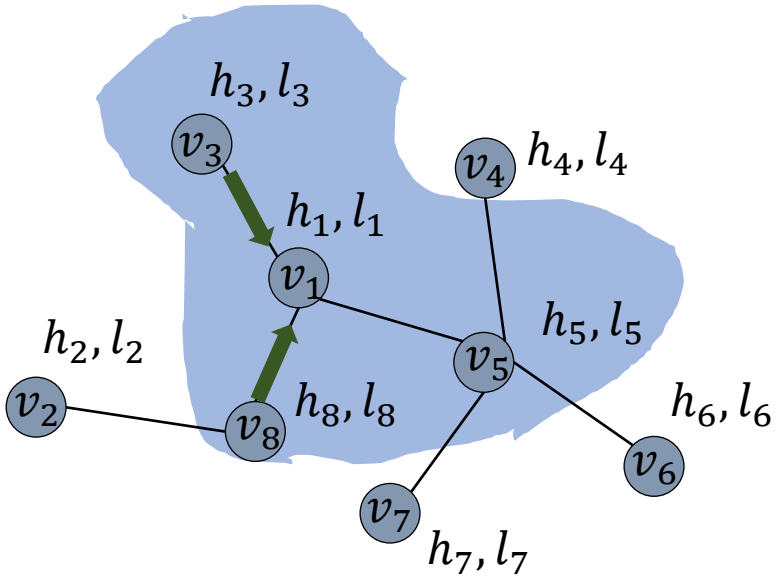
GCN: k-th layer

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

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 GraphSage

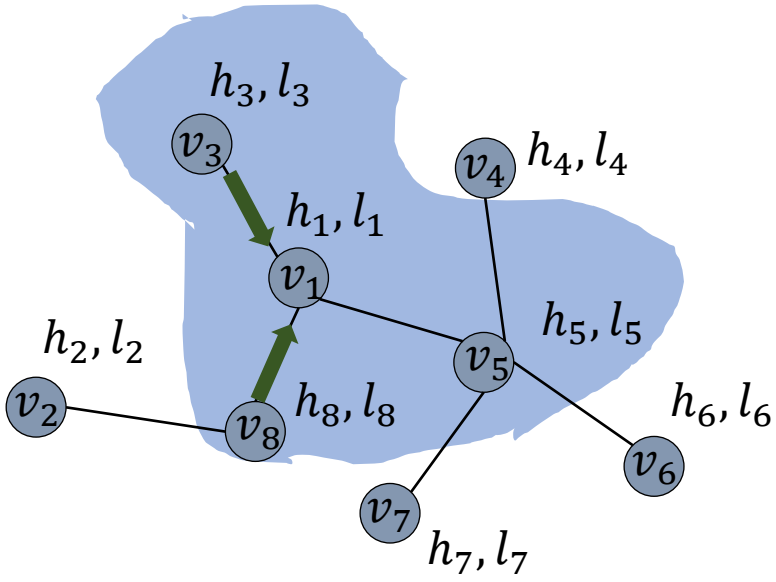


Neighbor Sampling

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

Inductive Representation Learning on Large Graphs. NIPS 2017.

Filter in GraphSage



Neighbor Sampling

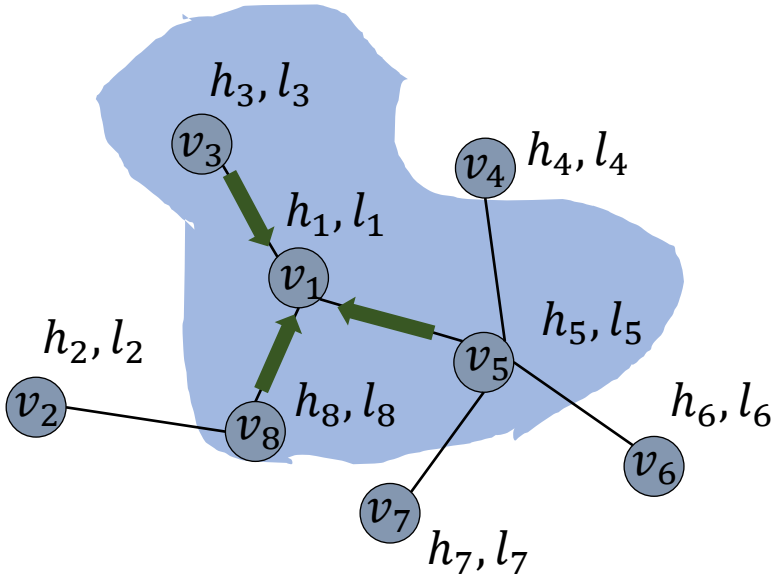
$$\mathcal{N}(v_i) \rightarrow \mathcal{N}_s(v_i)$$

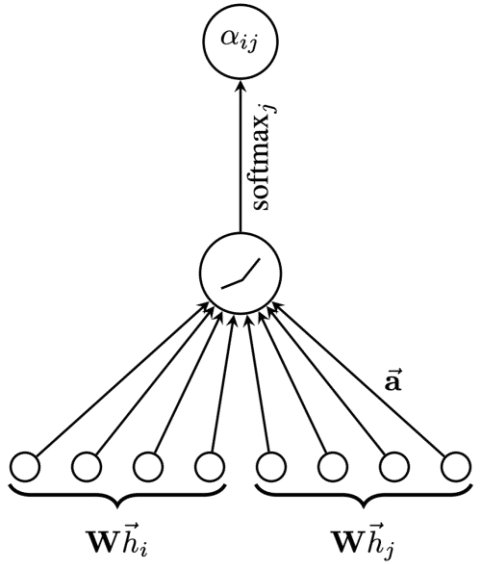
Aggregation

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

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

Filter in GAT

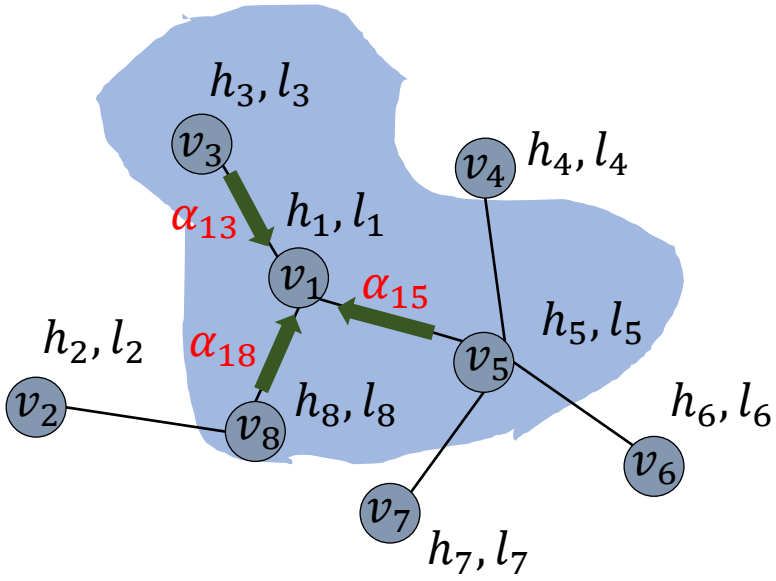


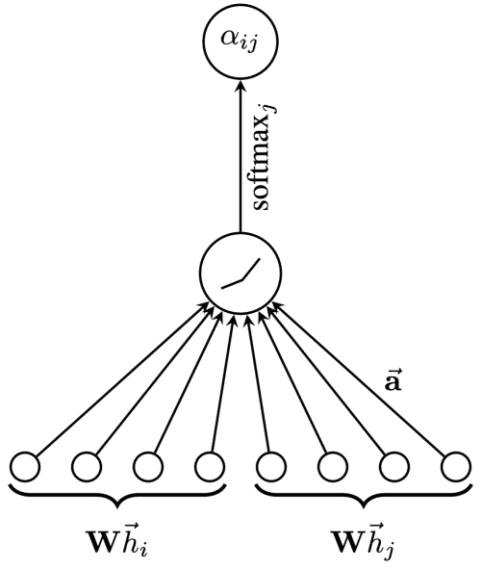


$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Graph Attention Networks. ICLR 2018.

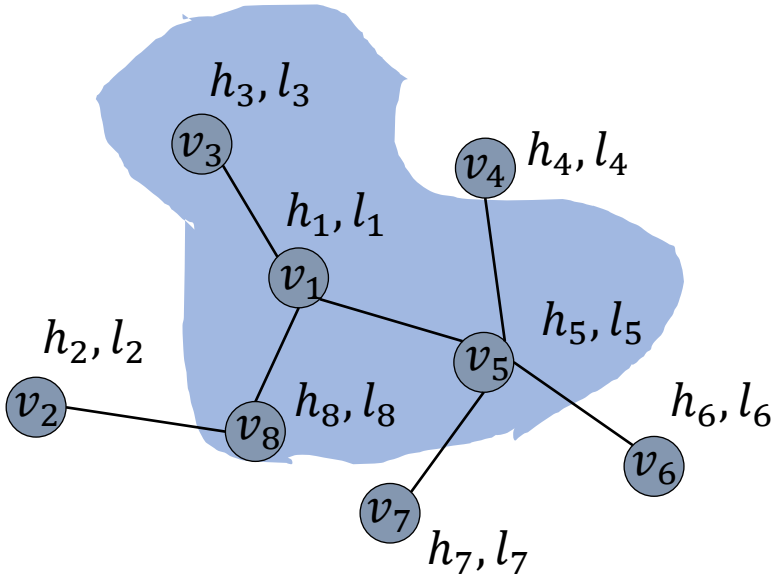
Filter in GAT





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

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

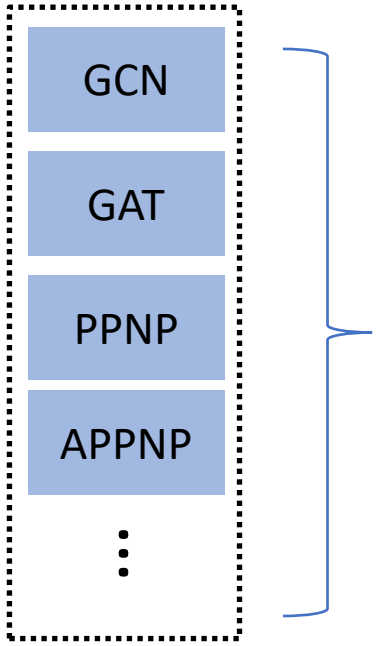
$$h_i^{(k+1)} = U_k\left(h_i^{(k)}, m_i^{(k+1)}\right)$$

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

Neural Message Passing for Quantum Chemistry. ICML 2017.

UGNN: A Unified Framework

Filtering Operations

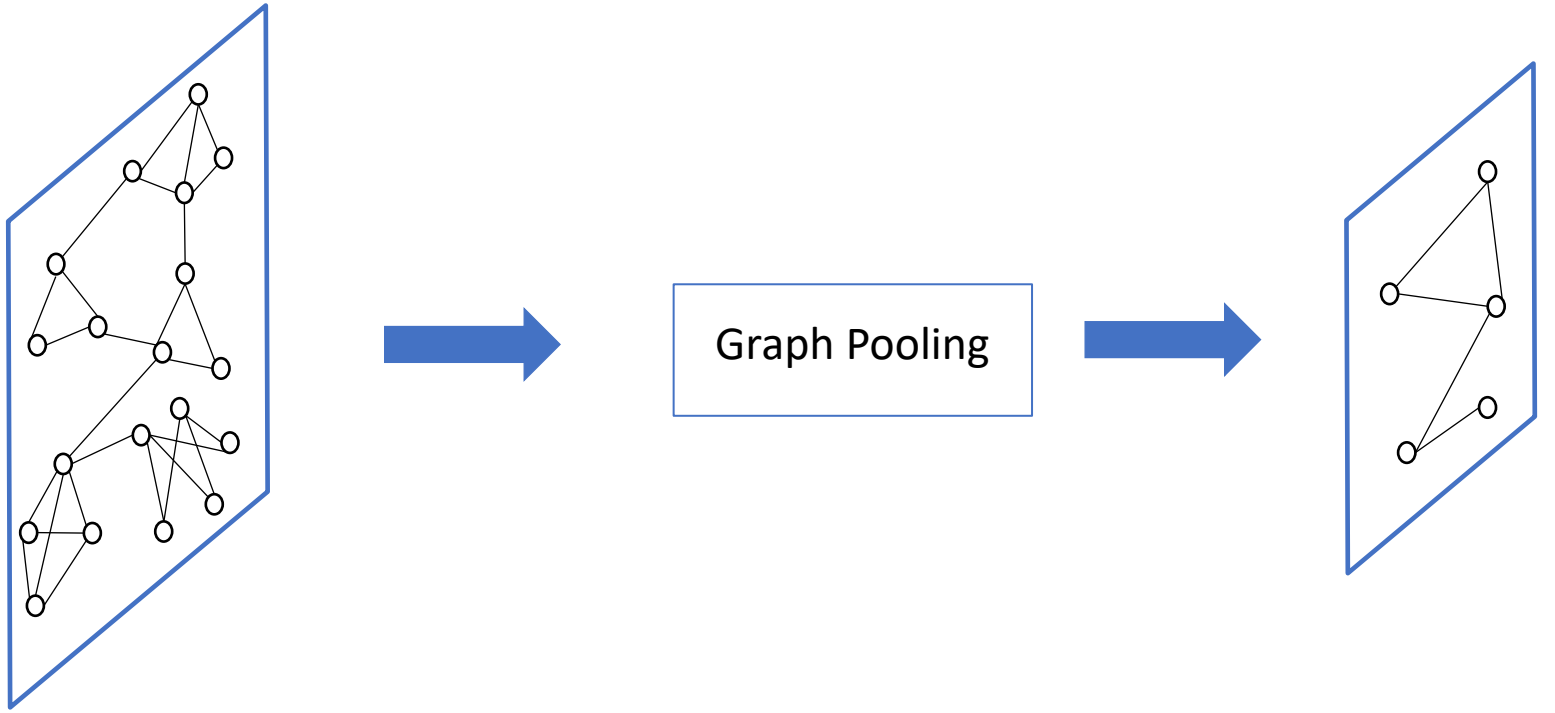


Graph Signal Denoising

$$\arg\min_{\mathbf{X}_f} \mathcal{L}(\mathbf{X}_f) = \|\mathbf{X}_f - \mathbf{X}'\|_F^2 + c \cdot \frac{1}{2} \sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|\mathbf{X}_f[i,:] - \mathbf{X}_f[j,:]\|_2^2$$

A Unified View on Graph Neural Networks as Graph Signal Denosing. arXiv, 2020.

Graph Pooling Operation

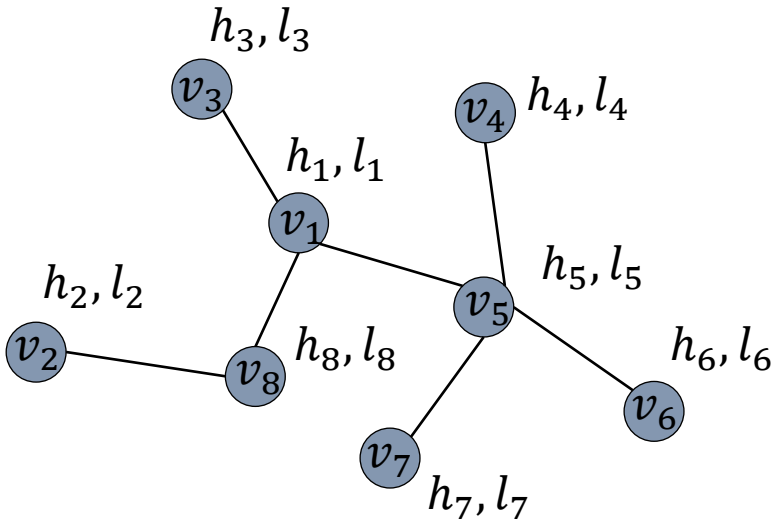


$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

gPool

Downsample by selecting the most importance nodes



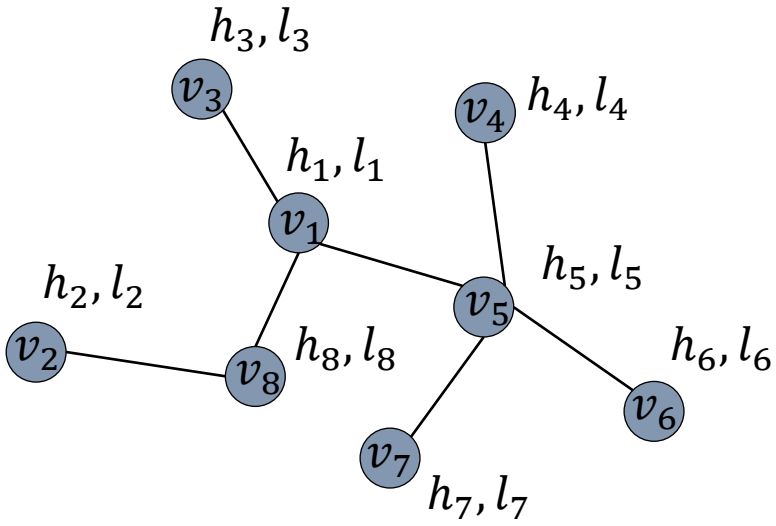
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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

Graph U-Nets. ICML 2019.

gPool

Downsample by selecting the most importance nodes



Importance Measure

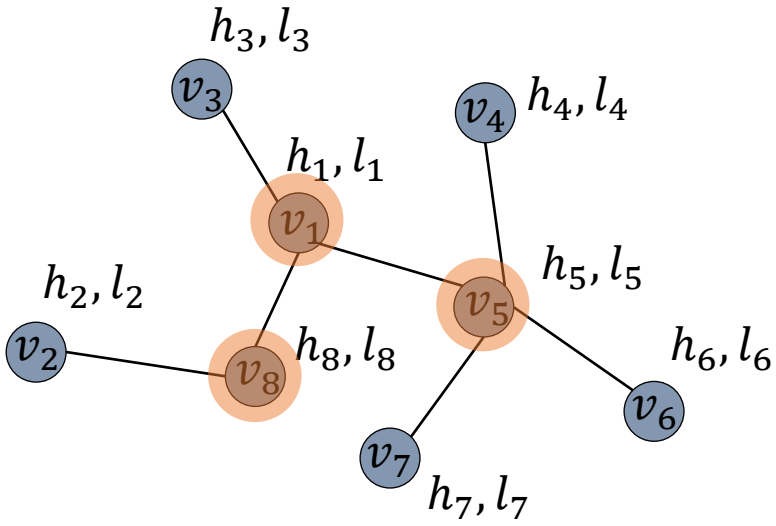
$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

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

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

gPool

Downsample by selecting the most importance nodes



Importance Measure

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

Select top the n_p nodes

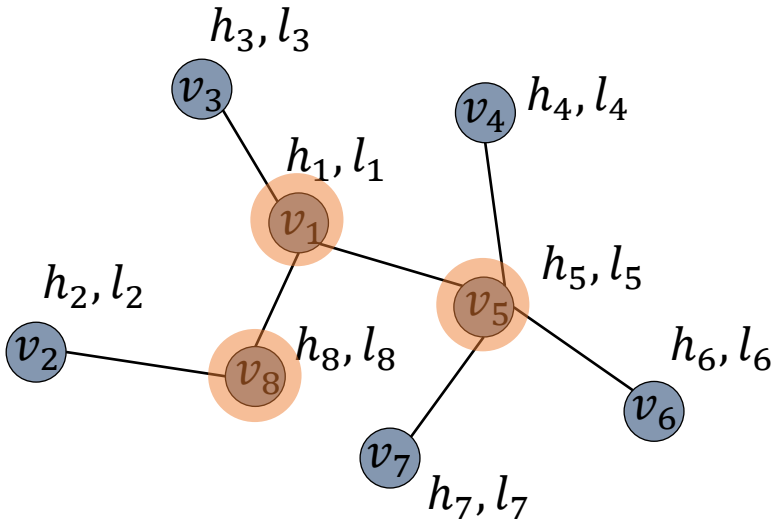
$$idx = rank(\boldsymbol{y}, n_p)$$

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

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

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

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

Select top the n_p nodes

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

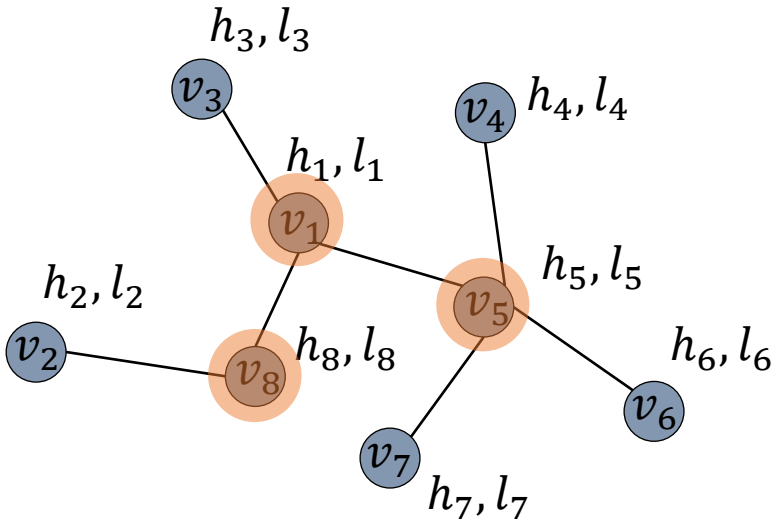
Generate A_p and intermediate H_{inter}

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

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

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

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

Select top the n_p nodes

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

Generate A_p and intermediate H_{inter}

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

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

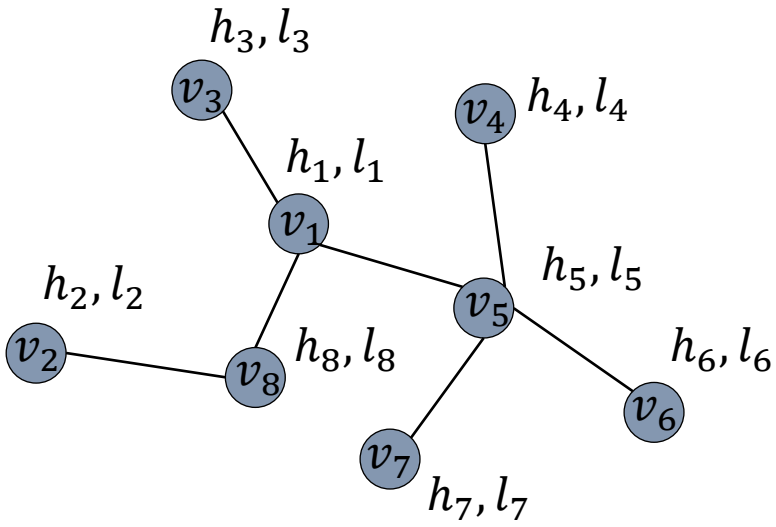
Generate H_p

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

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

DiffPool

Downsample by clustering the nodes using GNN



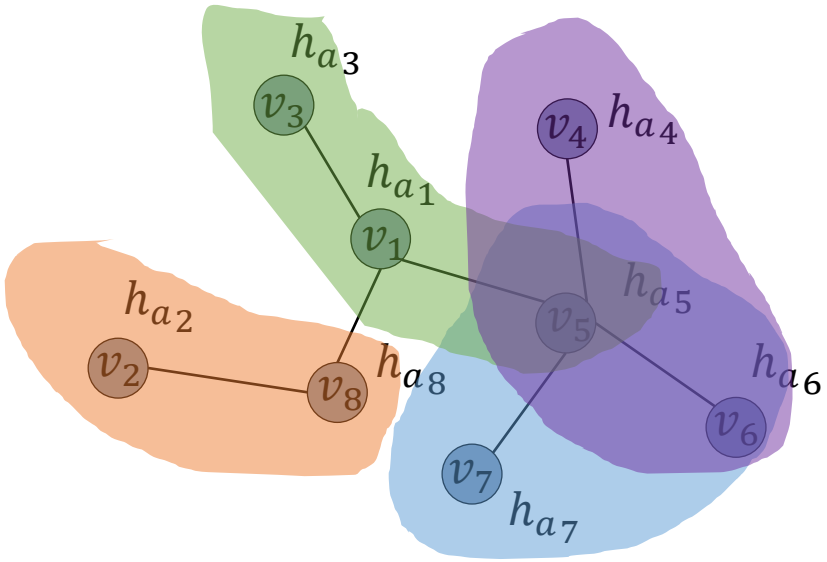
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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

Hierarchical Graph Representation Learning with Differentiable Pooling. NeurIPS 2018.

DiffPool

Downsample by clustering the nodes using GNN



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

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

Filter1:

Generate a soft-assign matrix

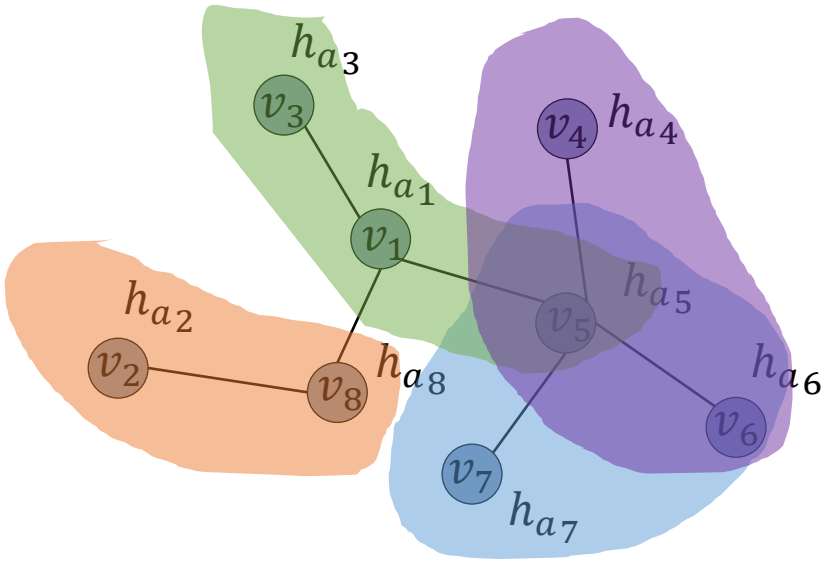
2 filters

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

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

DiffPool

Downsample by clustering the nodes using GNN



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

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

Filter1:

Generate a soft-assign matrix

2 filters

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

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

Filter2:

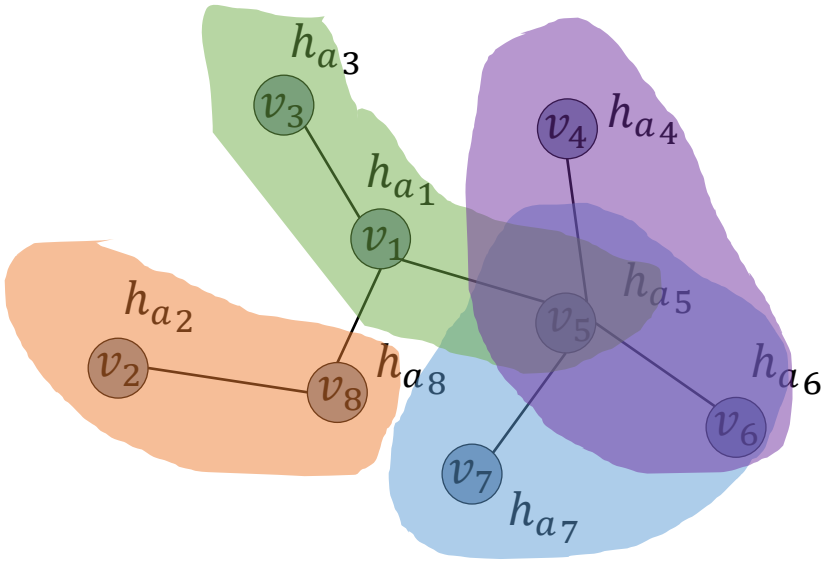
Generate new features

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

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

DiffPool

Downsample by clustering the nodes using GNN



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

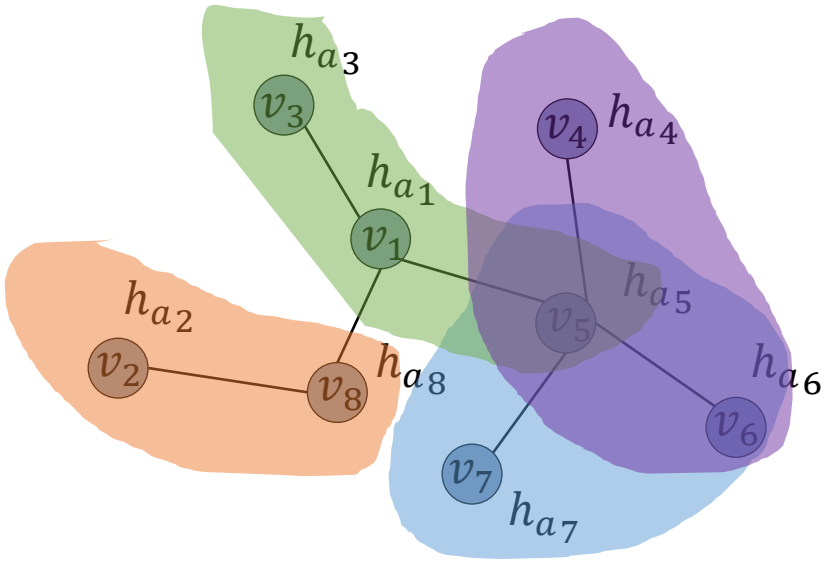
Generated new features $\mathbf{H}_f \in \mathbb{R}^{n \times d_{new}}$

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

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

DiffPool

Downsample by clustering the nodes using GNN



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

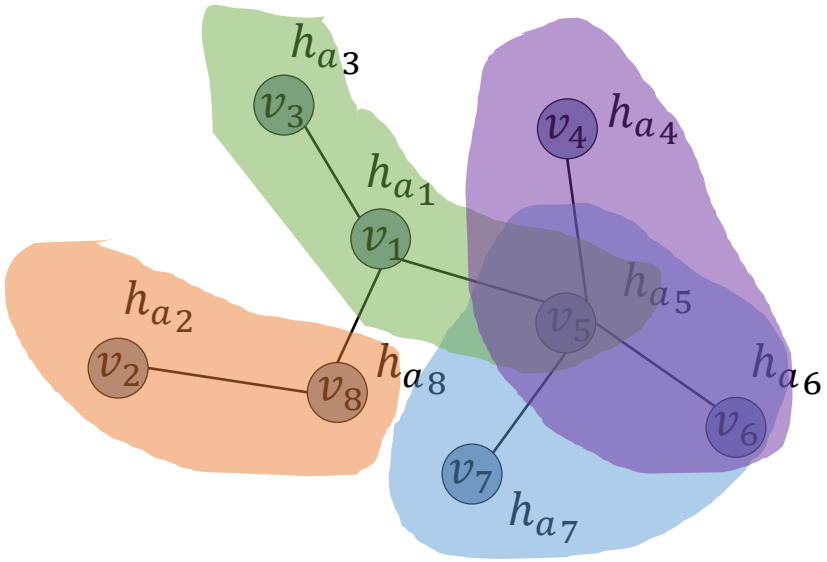
$$\mathbf{A}_p = \mathbf{H}_a^T \mathbf{A} \mathbf{H}_a$$

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

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

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

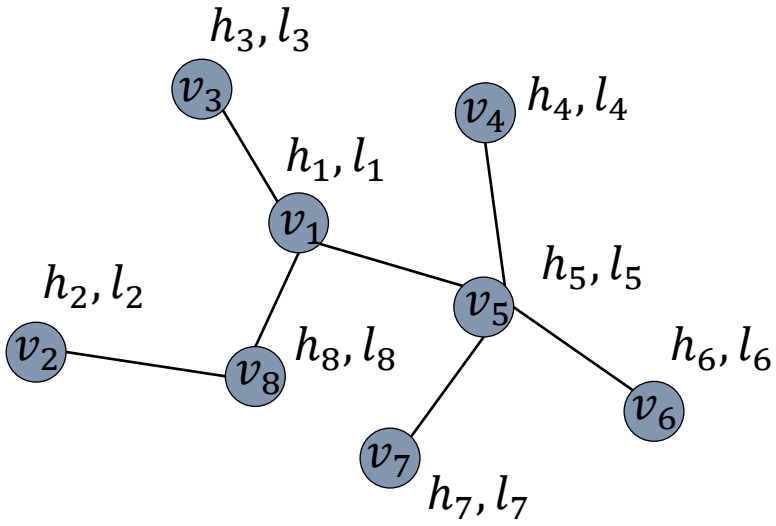
Generate A_p

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

Generate H_p

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

Eigenpooling

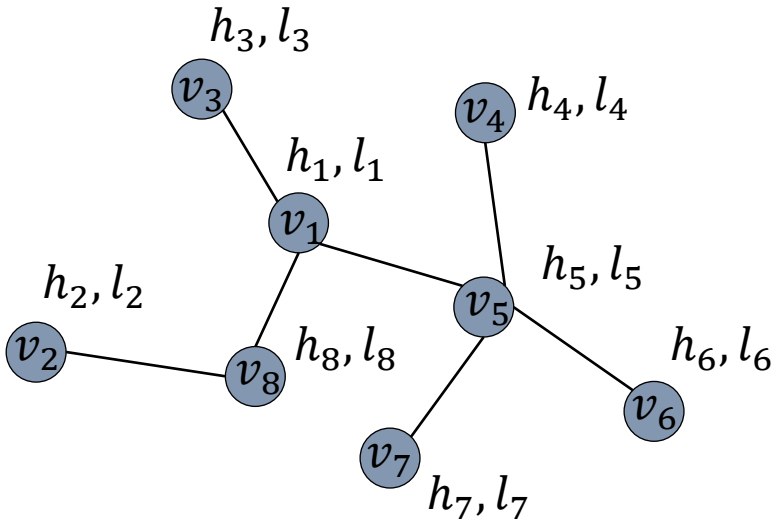


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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

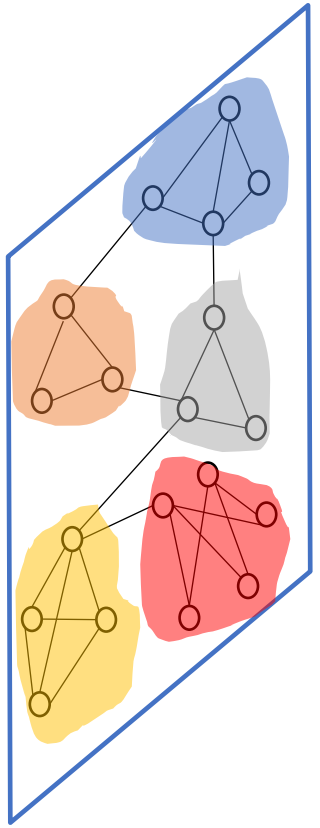
Graph Convolutional Networks with EigenPooling. KDD 2019.

Eigenpooling



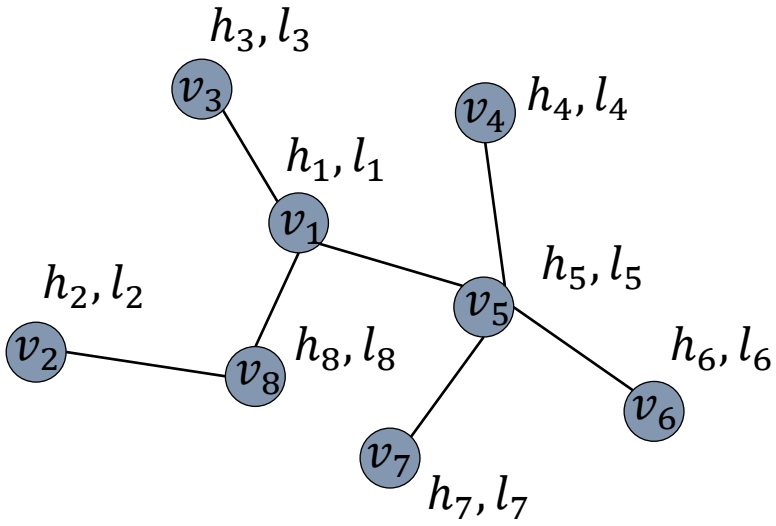
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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



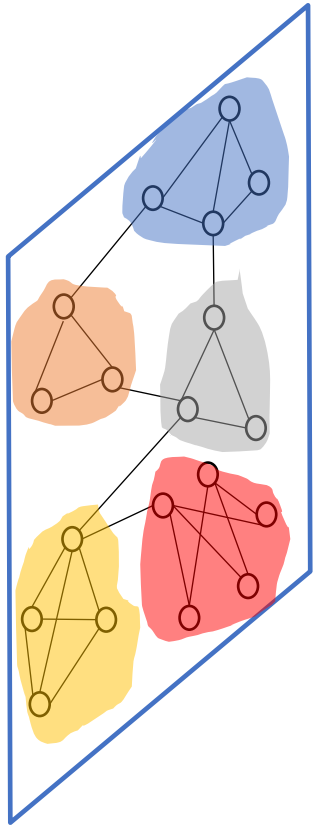
Learn A_p using clustering methods

Eigenpooling



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

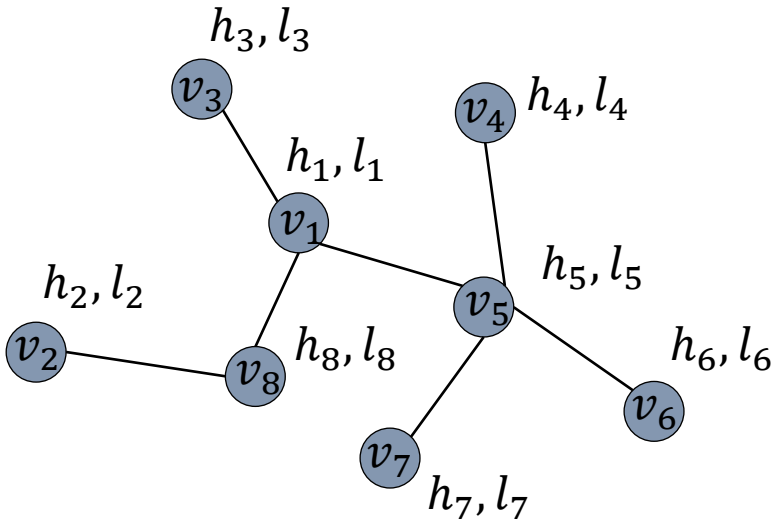
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$



Learn A_p using clustering methods

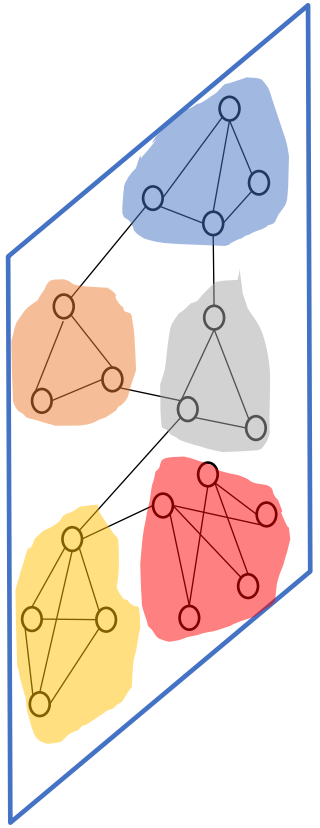
Focus on learning better H_p

Eigenpooling



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

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



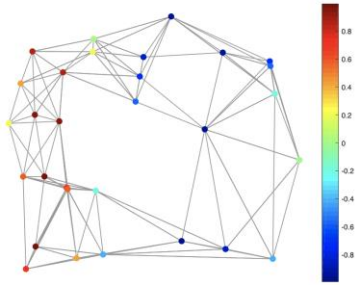
Learn A_p using clustering methods

Focus on learning better H_p

Capture both feature and graph structure

Going Back to Graph Spectral Theory

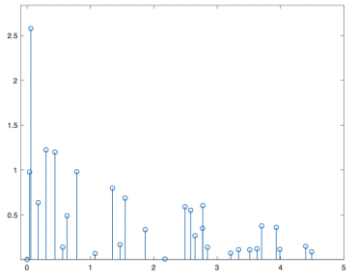
Recall:



Spatial domain: $\boldsymbol{f}$

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

Decompose signal f

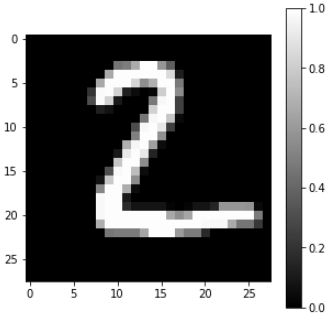


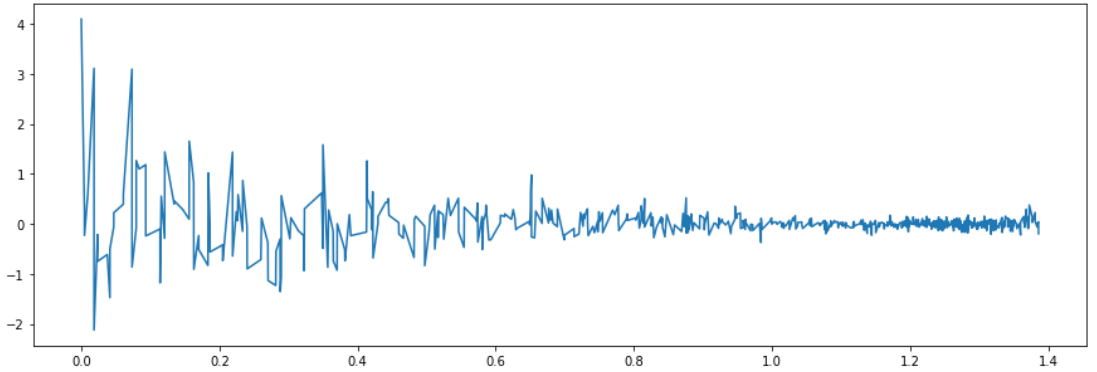
Spectral domain: $\hat{\boldsymbol{f}}$

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

Going Back to Graph Spectral Theory

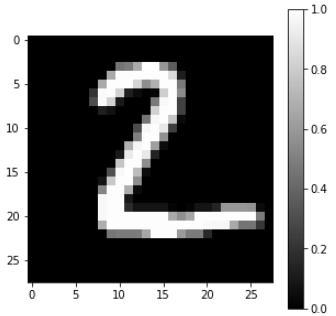
Do we need all the coefficients to reconstruct a “good” signal?

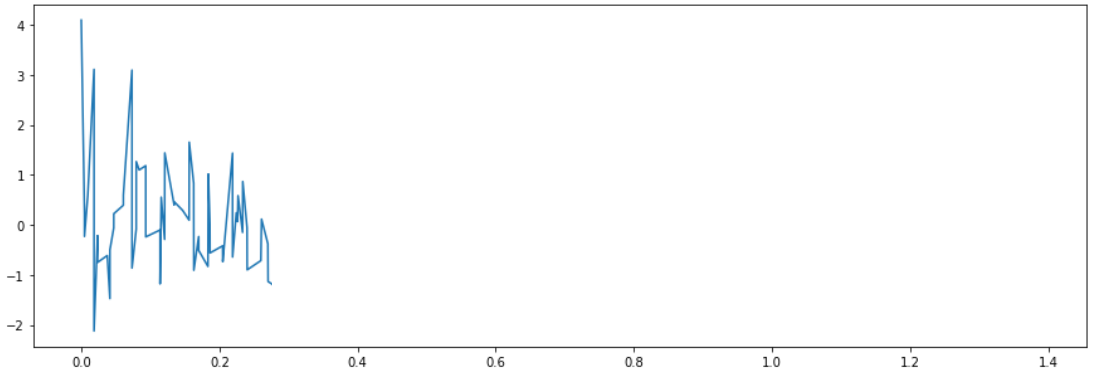




Going Back to Graph Spectral Theory

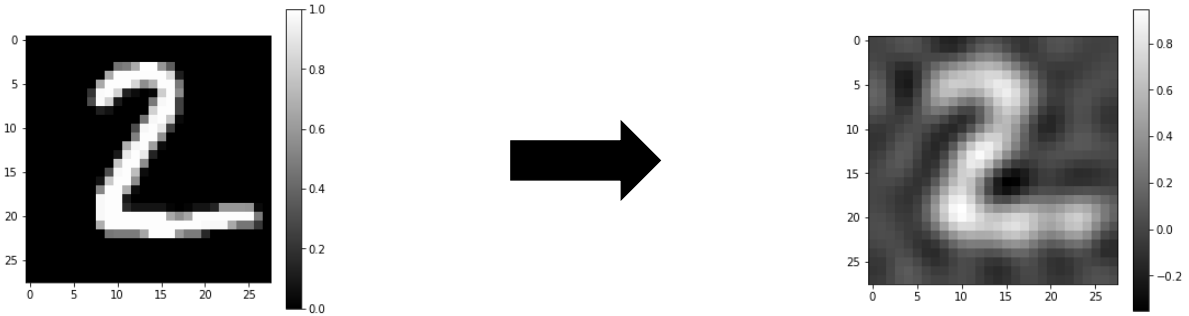
Do we need all the coefficients to reconstruct a “good” signal?

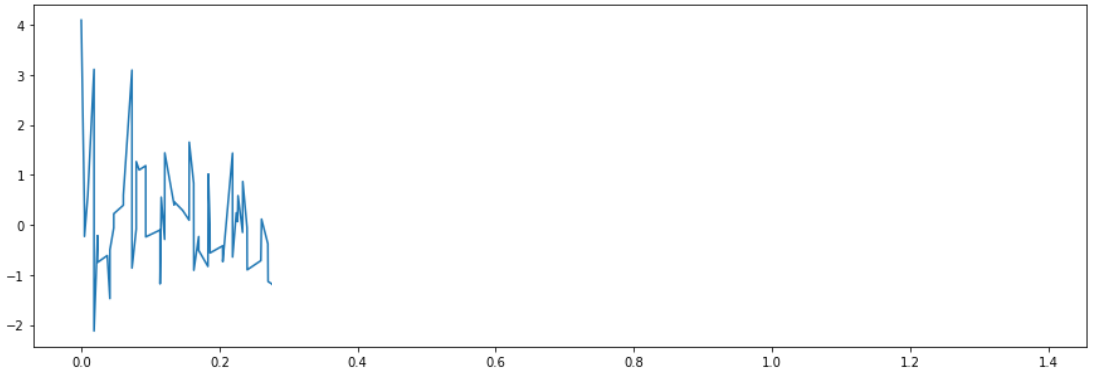




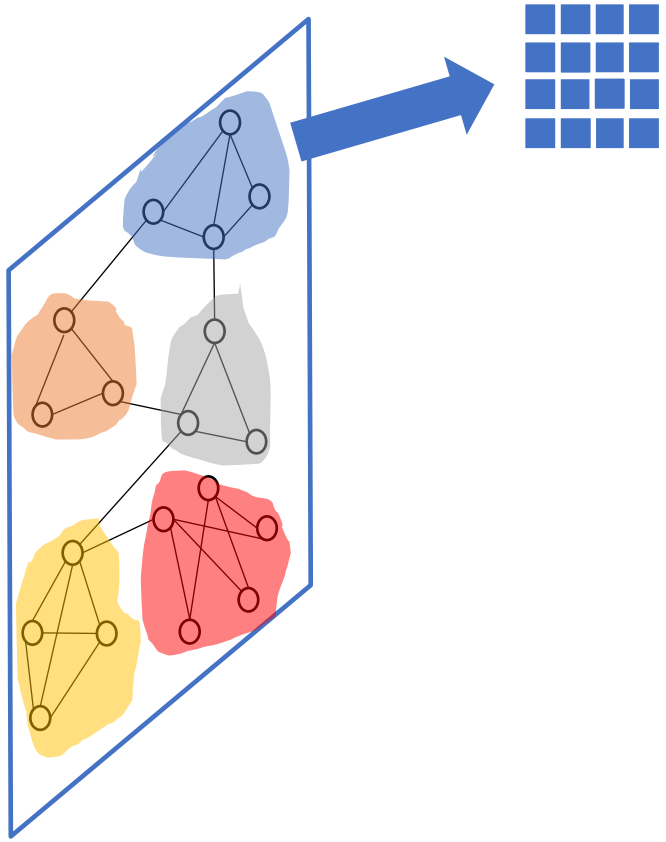
Going Back to Graph Spectral Theory

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



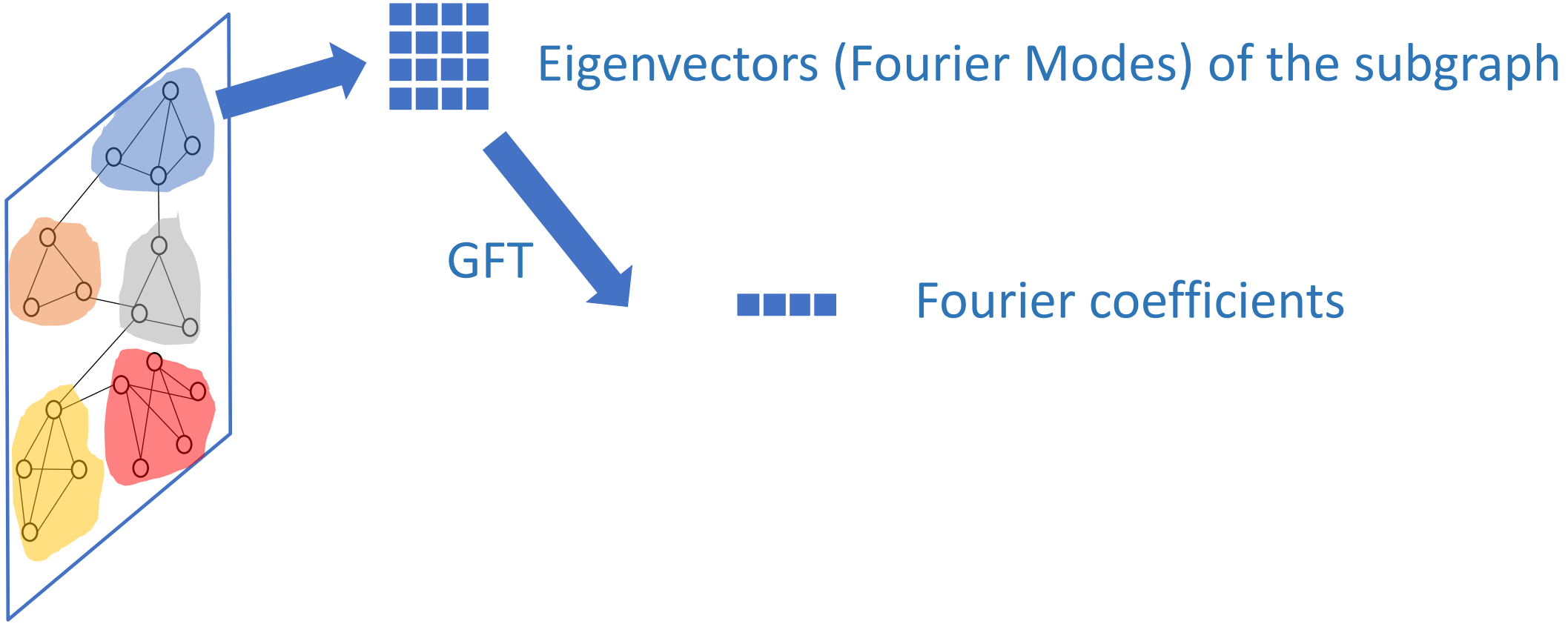


Eigenpooling: Truncated Fourier Coefficients

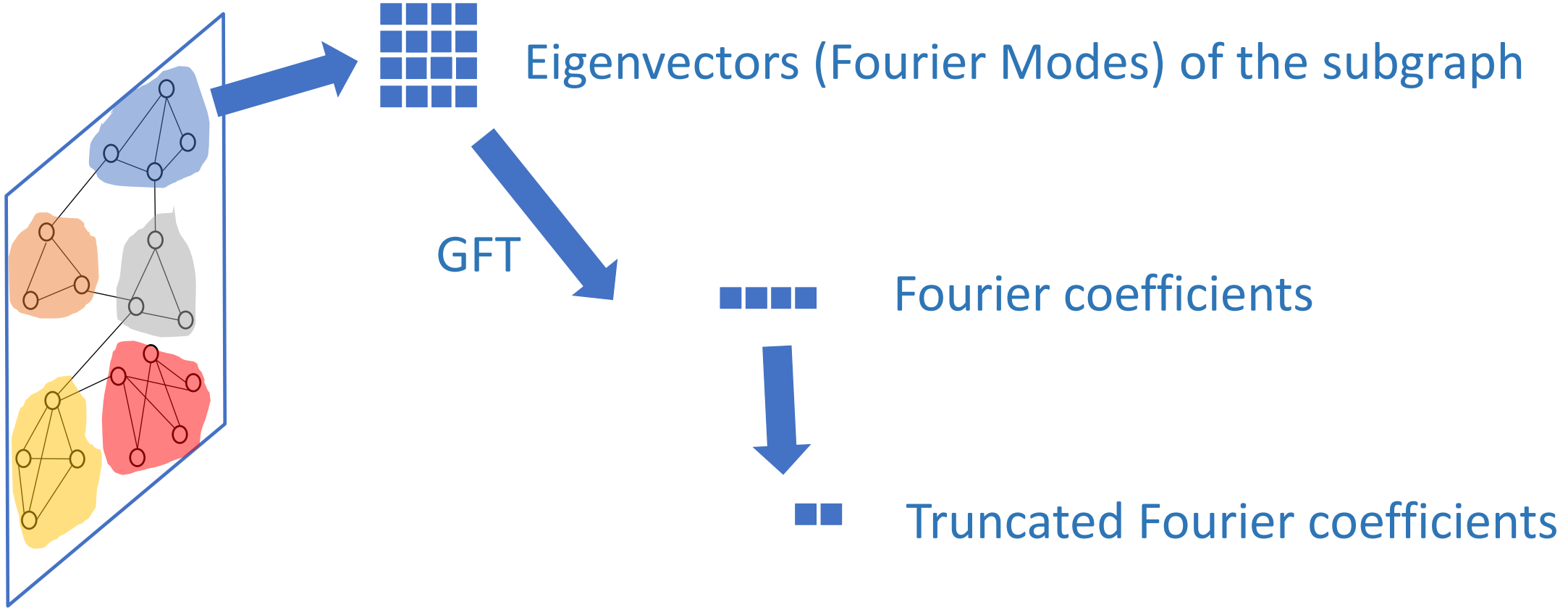


Eigenvectors (Fourier Modes) of the subgraph

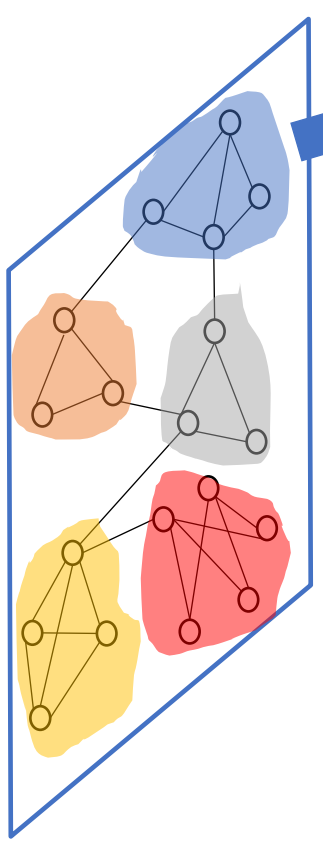
Eigenpooling: Truncated Fourier Coefficients

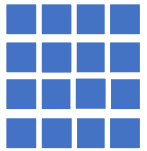


Eigenpooling: Truncated Fourier Coefficients



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)

Self-Supervised Learning for Graph Neural Networks

Weakness of Deep learning Models

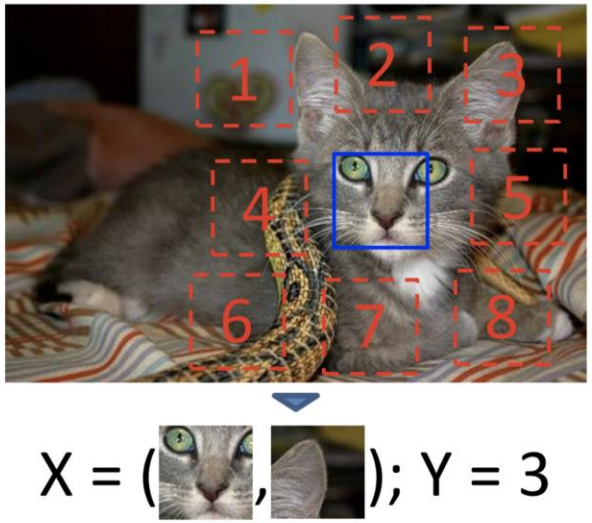
Require strong supervision

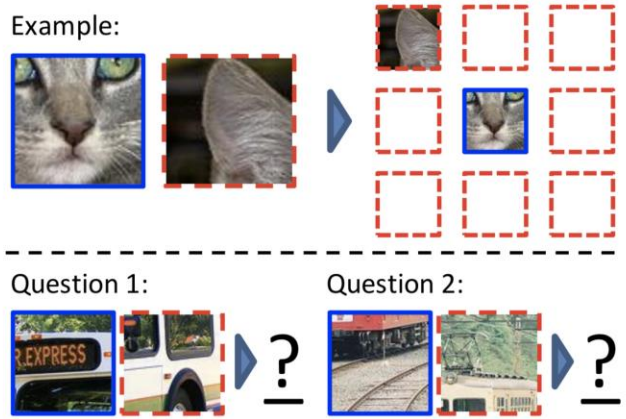
- Massive labeled data
- Expensive to obtain

Are there alternatives to strong supervision for training?

- Self-supervised learning!

Self-Supervised Learning



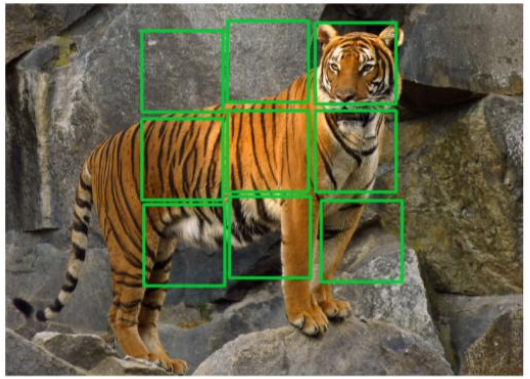


Relative position pretext task

Doersch et al., 2015

Jigsaw puzzle pretext task

Noroozi and Favaro, 2016



(a)

(b) Shuffled

(c) Solved

Applying SSL to Graphs

Similarities to Image and Text Domains

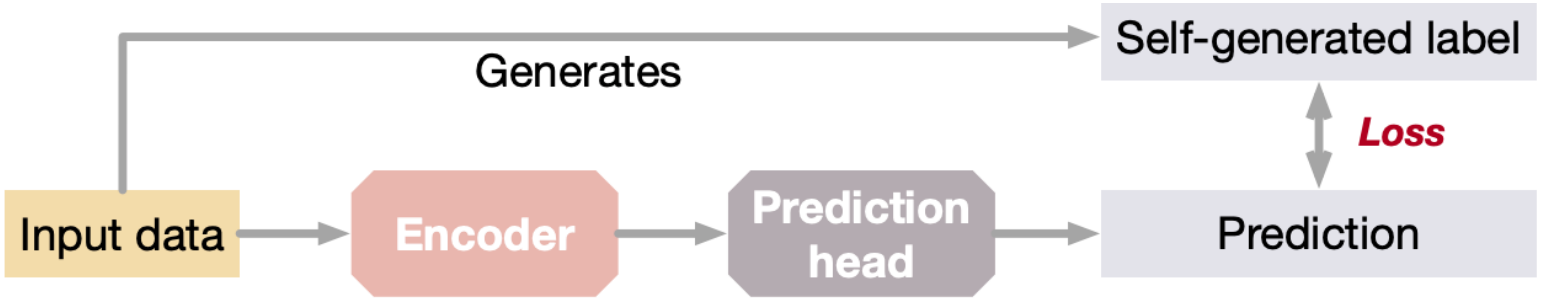
- Nodes have features like images or text
  → Pretext tasks using *attribute information*

Fundamental Differences Found in Graph Domain

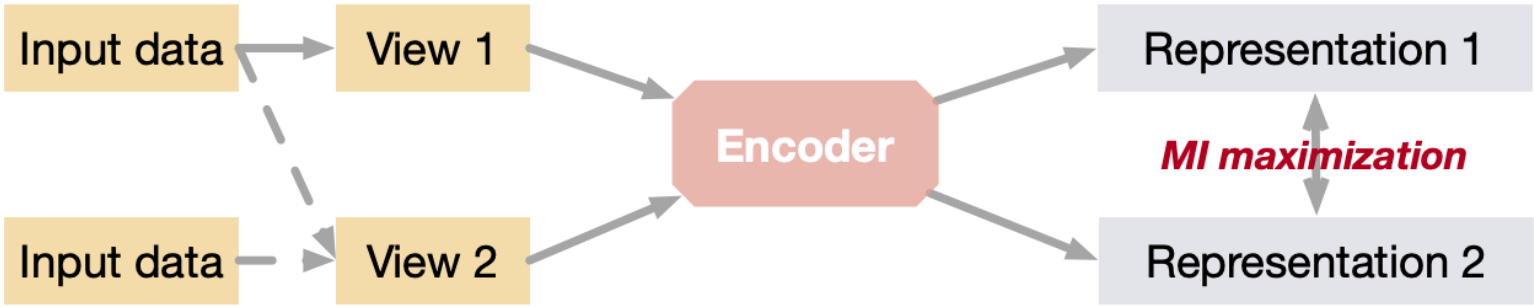
- Nodes are connected and dependent
  → Pretext tasks using *node pairs* or even sets
- (Node classification) Unlabeled nodes have structural relations to labeled nodes
  → Pretext tasks using *label information*
- (Graph classification) The graph can also express some semantic meaning
  → Pretext tasks using graph-level information

Methods Applying SSL to Graphs

- Predictive Methods



- Contrastive Methods



Self-Supervised Learning of Graph Neural Networks: A Unified Review. ArXiv 2021.

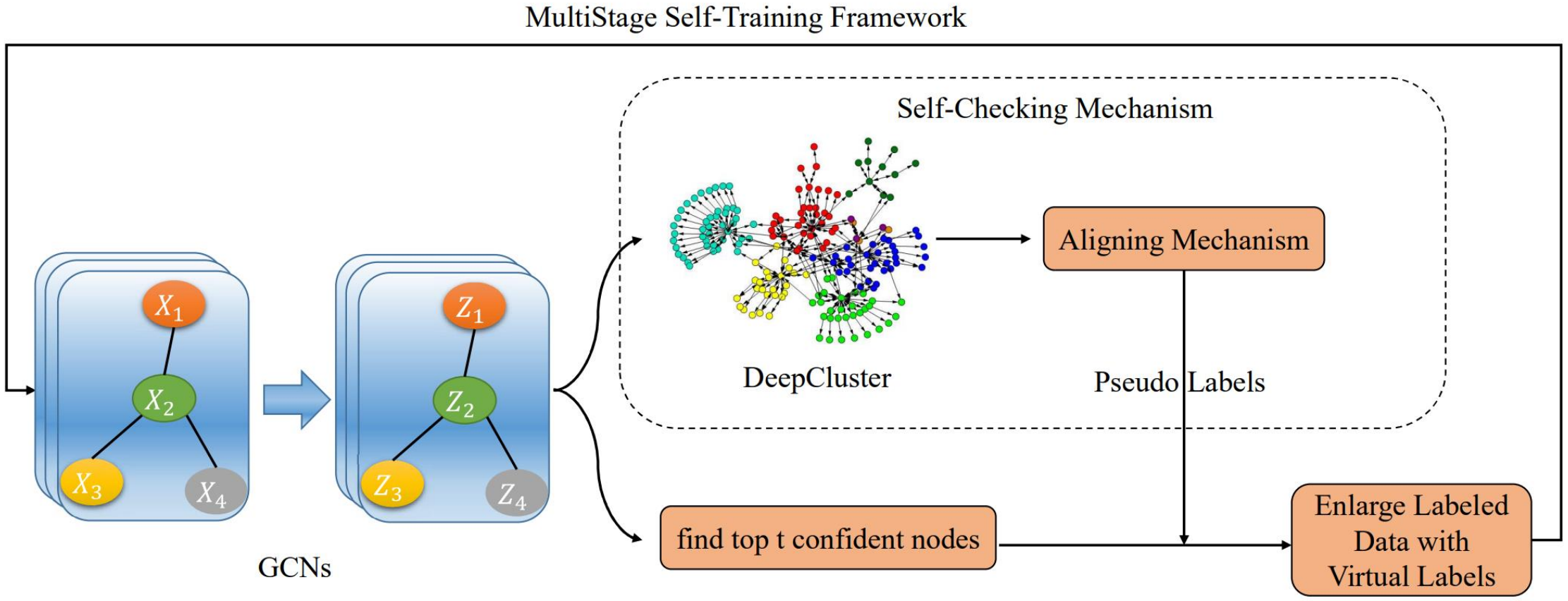
Methods Applying SSL to Graphs

- Predictive Methods
  - M3S
  - SS-GCN
  - SelfTask
  - Pre-Training GNNs
- Contrastive Methods
  - MVGRL
  - GCC
  - GraphCL
  - GCA

Methods Applying SSL to Graphs

- Predictive Methods
  - M3S
  - SS-GCN
  - SelfTask
  - Pre-Training GNNs
- Contrastive Methods
  - MVGRL
  - GCC
  - GraphCL
  - GCA

M3S: Multi-Stage Self-Training GNNs

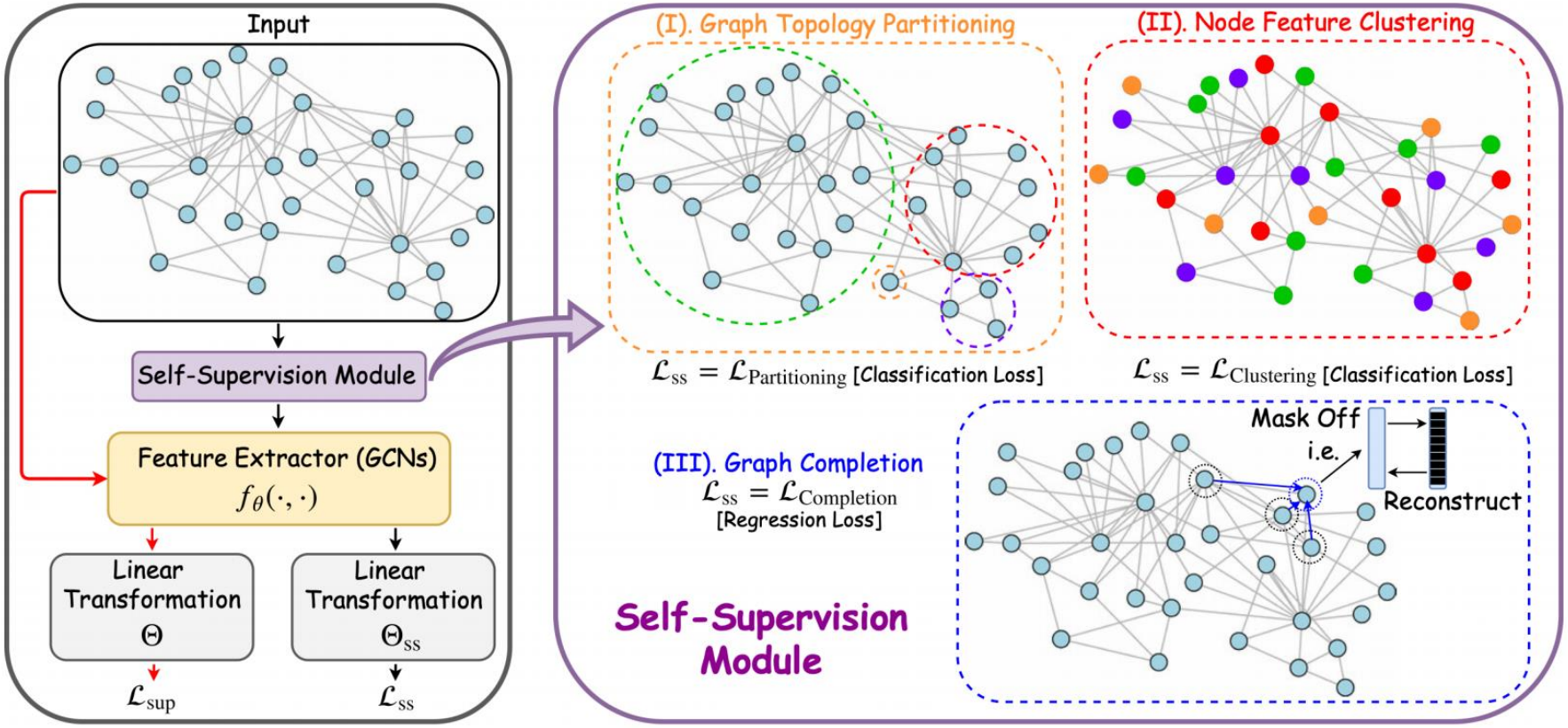


Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labels. AAAI. 2020.

SS-GCN: When Does SSL Help GCNs?

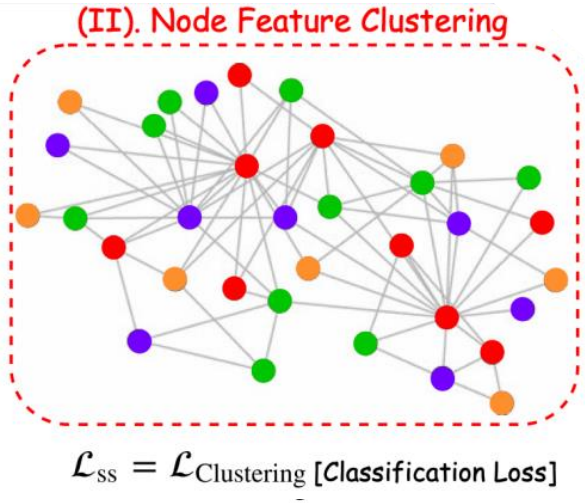
Three ways to include SSL tasks:

- Pretraining/ Finetuning
- Self-training
- **Multi-task learning**



When Does Self-Supervision Help Graph Convolutional Networks? ICML. 2020.

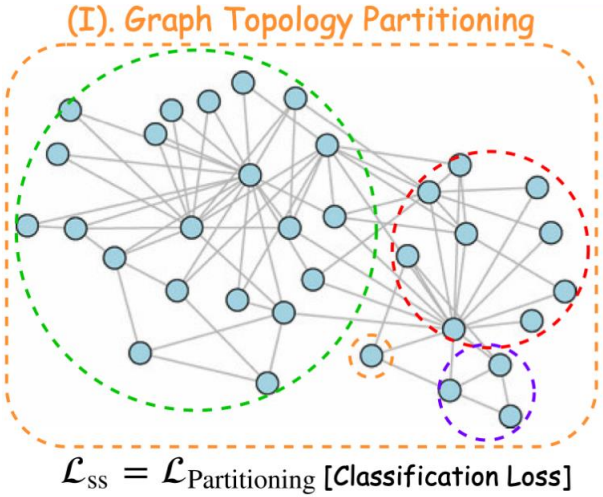
SS-GCN: Node Clustering Pretext Task



- Features used: Nodes
- Assumptions: Feature similarity
- Loss Type: Classification

- Follows the ideas of clustering from M3S
- Each node is given a cluster based on node features
- This cluster label is assigned the self-supervised label to predict

SS-GCN: Partitioning Pretext Task

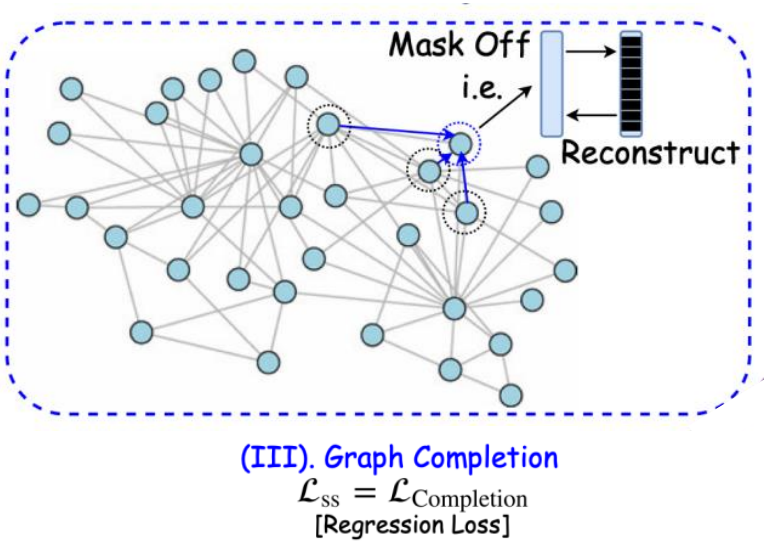


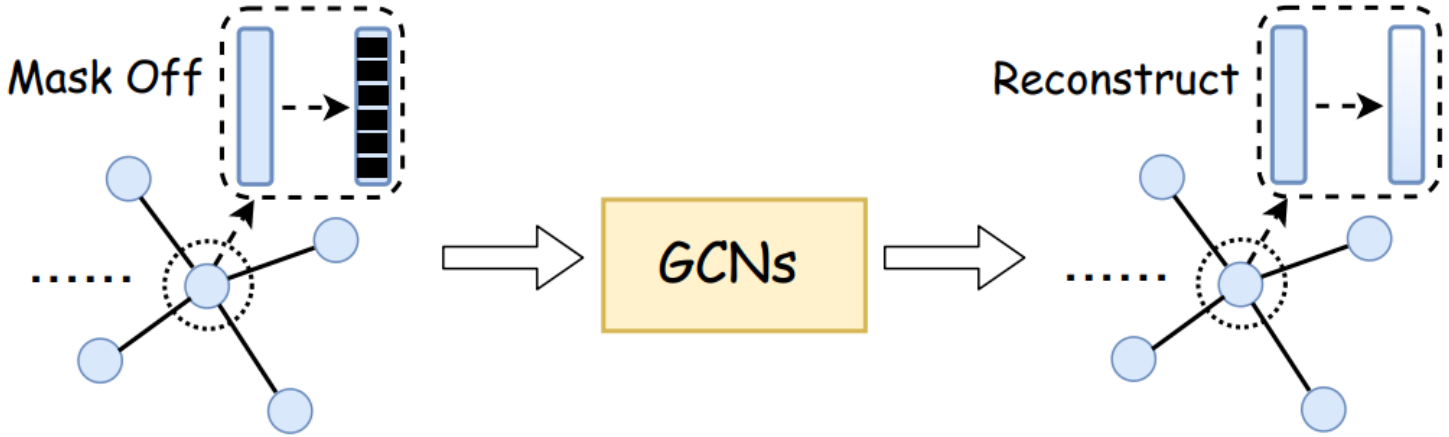
- Features used: Edges
- Assumptions: Connection density
- Loss Type: Classification

- Rather than clustering the node features, instead they partition the network based on the structure
- Similarly, partition indices are the self-supervised label to predict

SS-GCN: Graph Completion Pretext Task

- Features used: Nodes & Edges
- Assumptions: Context based Representation
- Loss Type: Regression





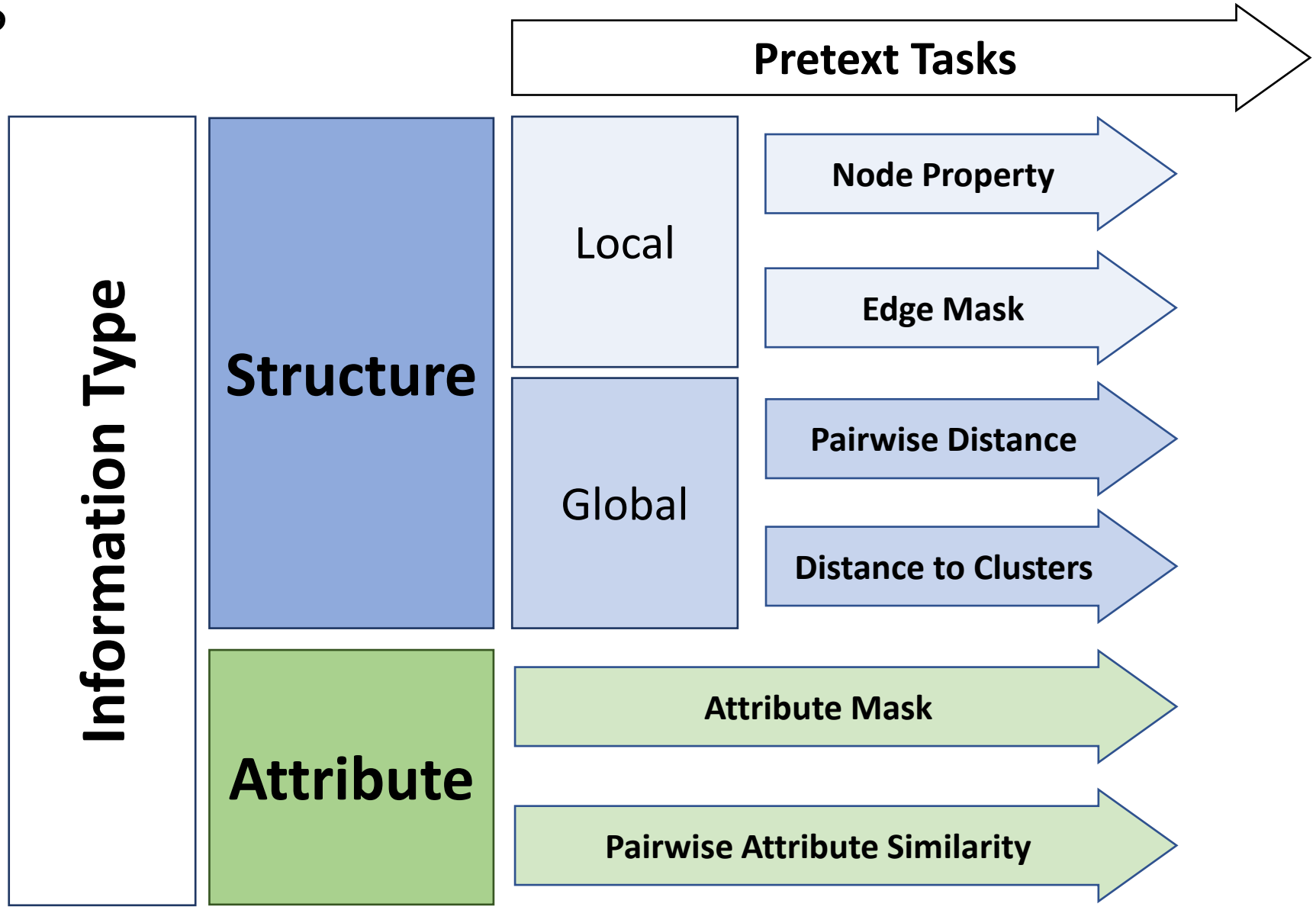
SelfTask: When and Why SSL Works on GNNs

- Presents a set of basic pretext tasks using structure and attribute information

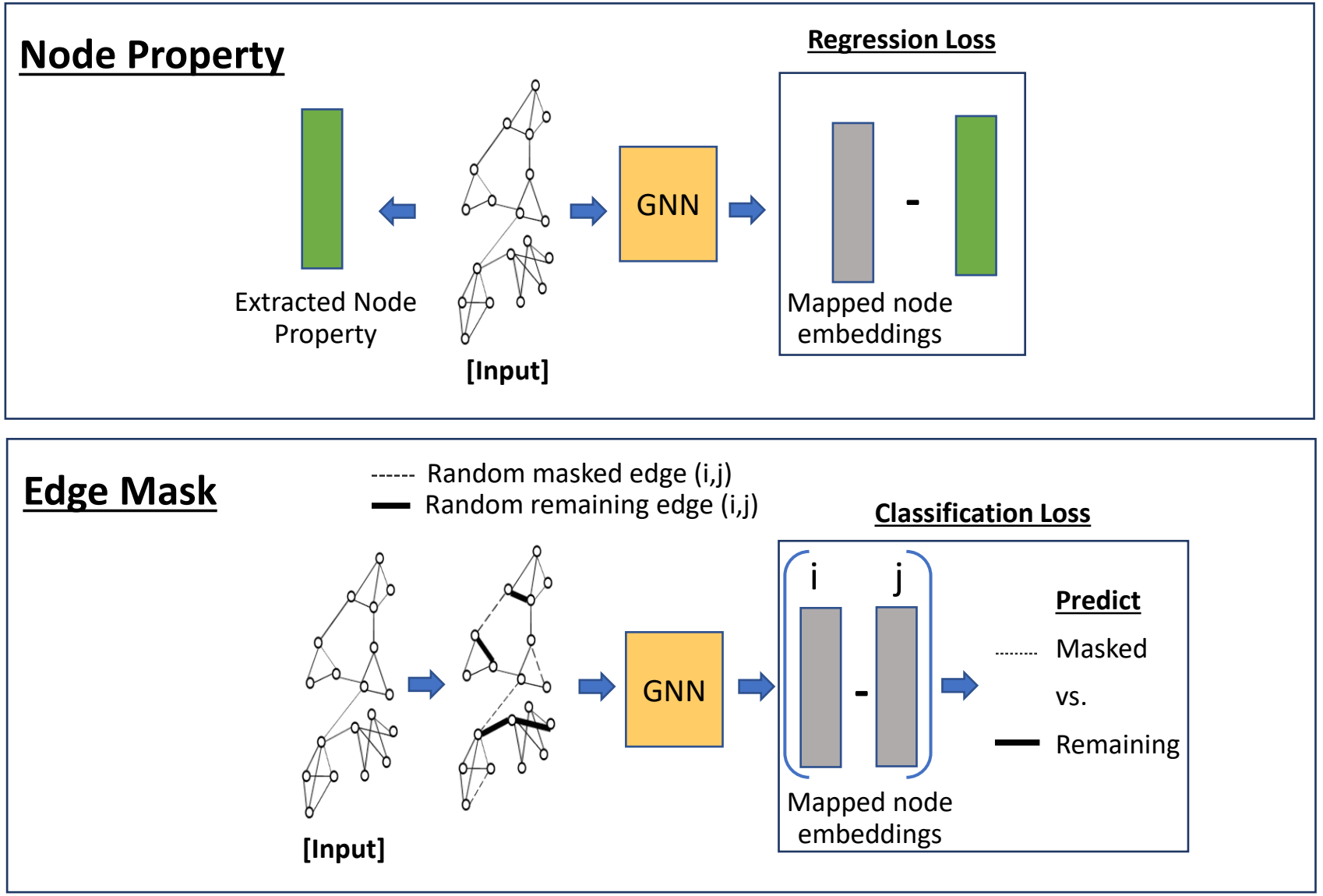
- Insights gained on:
  - Which strategy to harness SSL on GNNs?
  - Why do some pretext tasks work other others do not on GNNs?
  - How to construct advanced pretext tasks beyond basic structure and attributes?

Self-Supervised Learning on Graphs: Deep Insights and New Directions. arXiv. 2020.

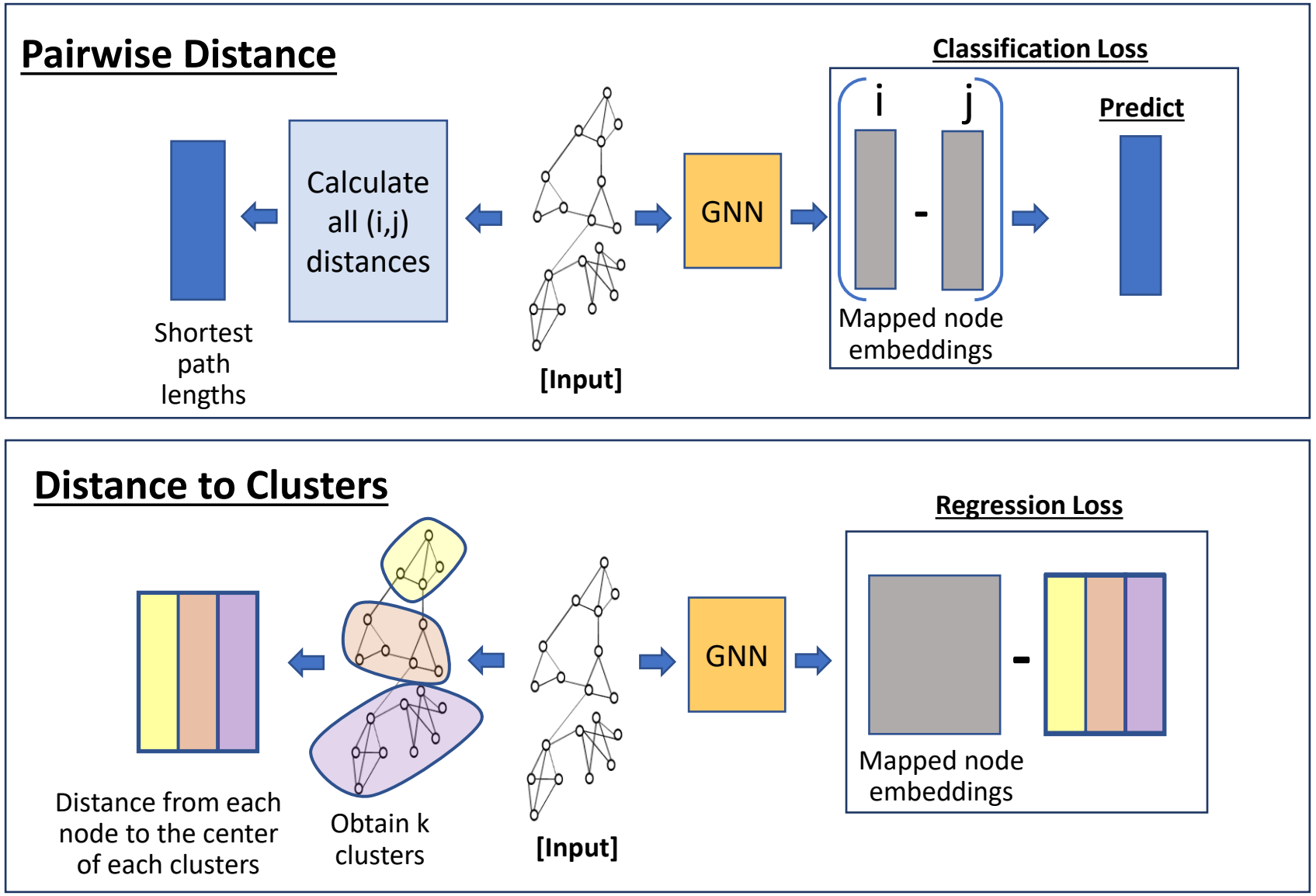
SelfTask: Basic Pretext Tasks on Graphs



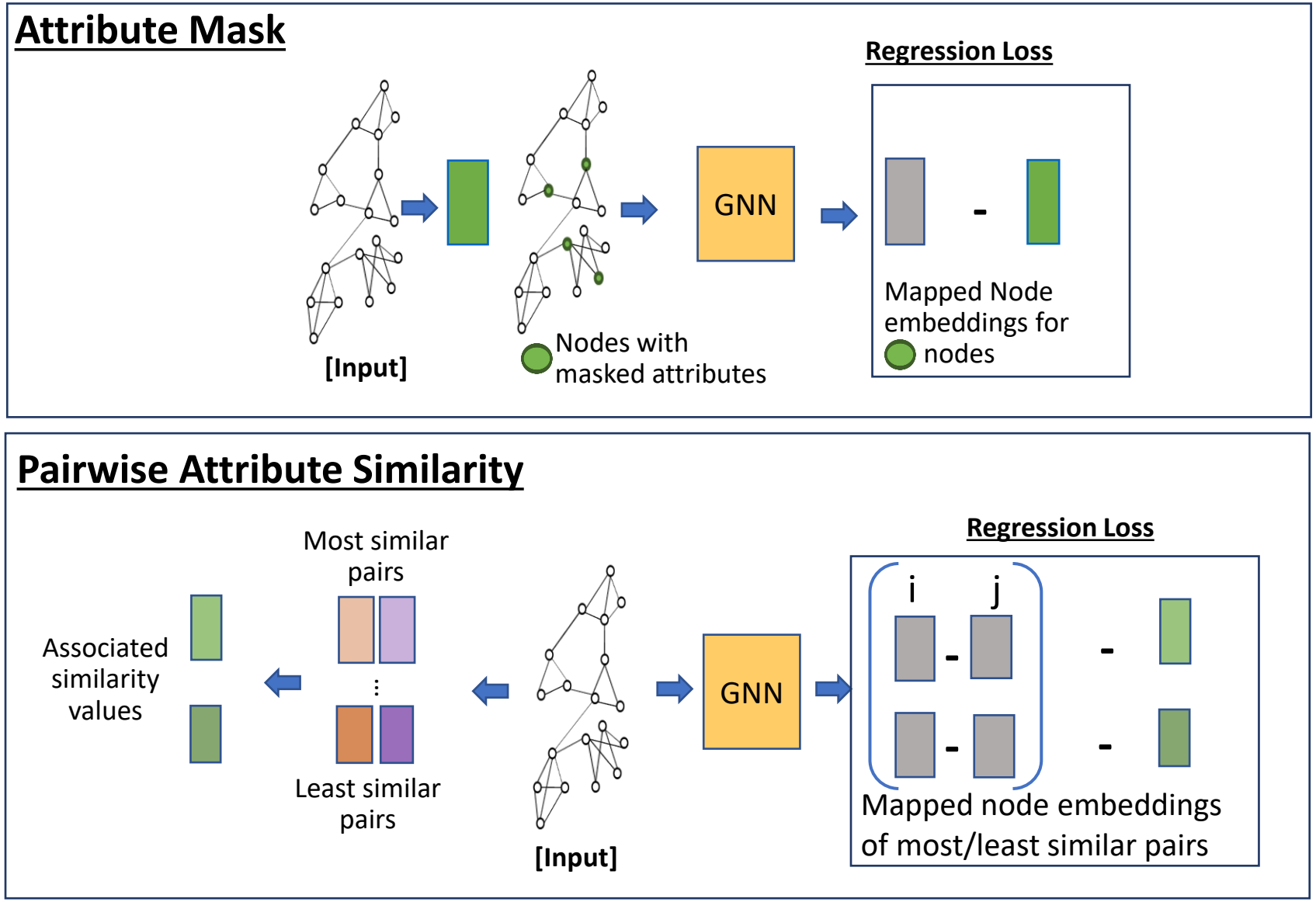
SelfTask: Local Structure Pretext Tasks



SelfTask: Global Structure Pretext Tasks



SelfTask: Attribute Pretext Tasks



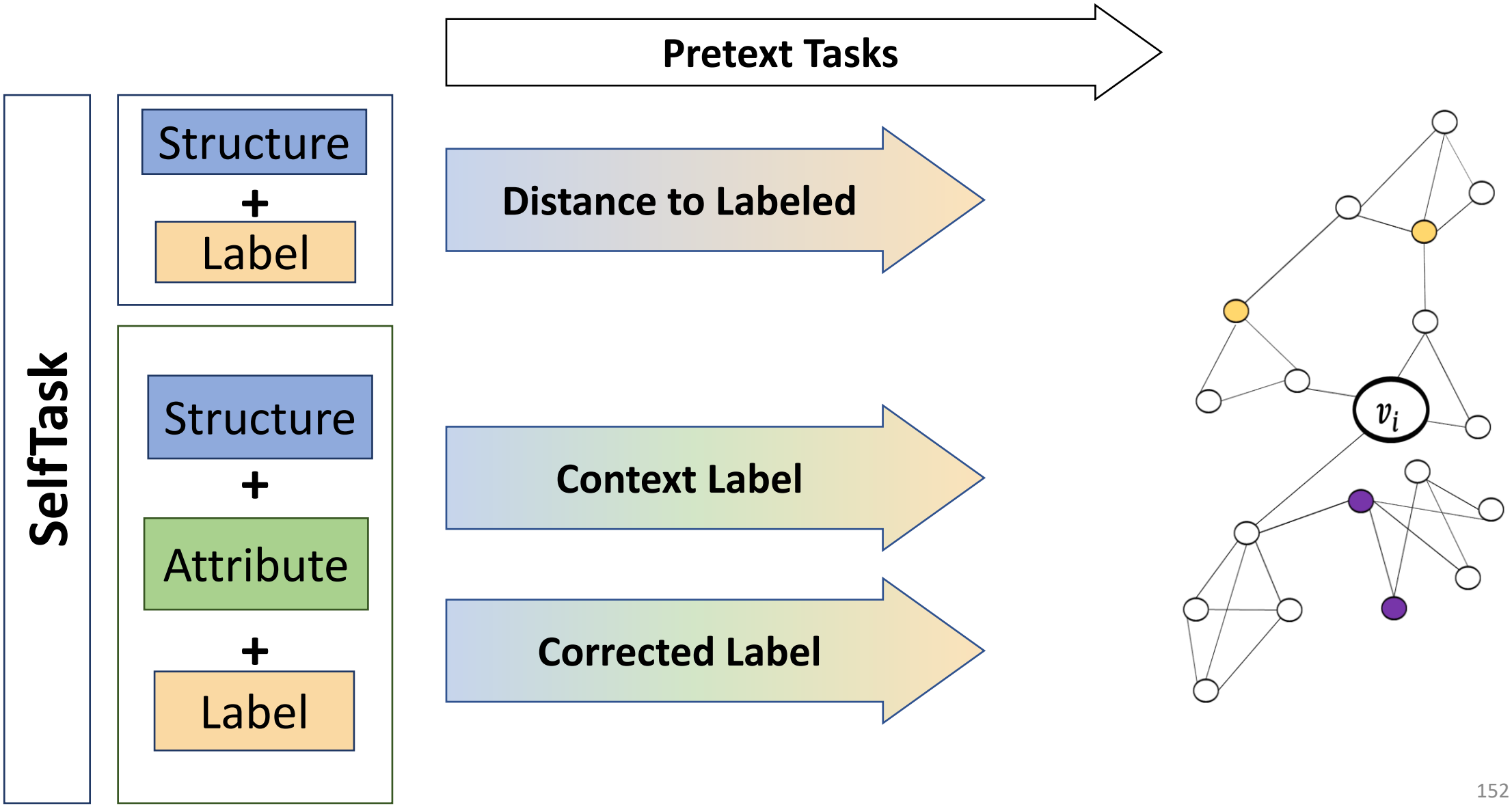
SelfTask: Empirical Study of Basic Pretext Tasks

	Model	Joint Training			Two-stage Training		
		Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
	GCN	81.32	71.53	79.28	81.32	71.53	79.28
	GCN-DroppedGraph	81.03	71.29	79.28	81.03	71.29	79.26
	GCN-PCA	81.74	70.38	78.83	81.74	70.38	78.83
Local Structure	NodeProperty	81.94	71.60	79.44	81.59	71.69	79.24
	EdgeMask	81.69	71.51	78.90	81.44	71.57	79.33
Global Structure	PairwiseNodeDistance	83.11	71.90	80.05	82.39	72.02	79.57
	Distance2Cluster	83.55	71.44	79.88	81.80	71.55	79.51
Attribute	AttributeMask	81.47	70.57	78.88	81.31	70.40	78.72
	PairwiseAttrSim	83.05	71.67	79.45	81.57	71.74	79.42

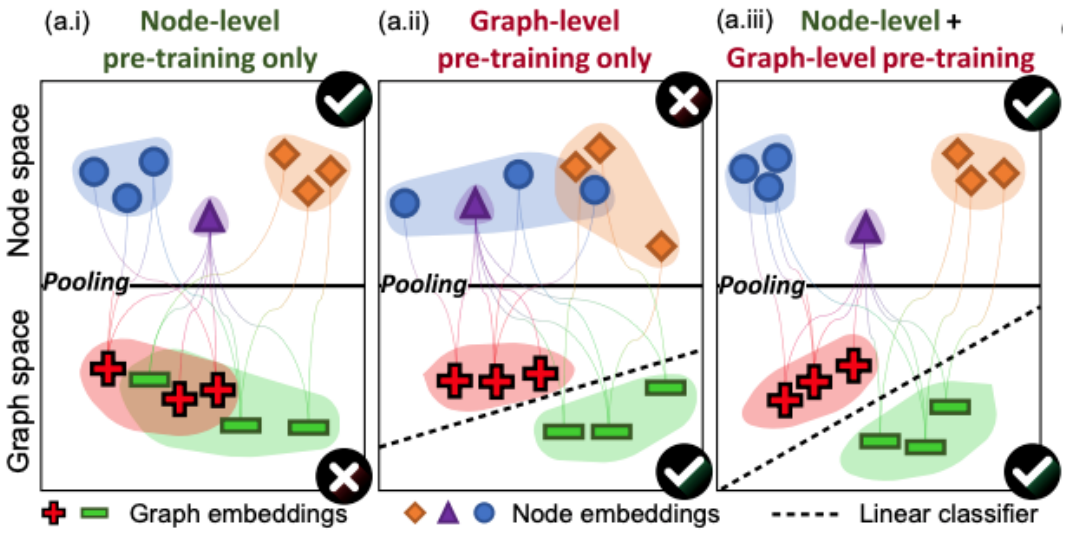
Insights:

- In general joint/multi-task training outperforms pre-training/two-stage training
- Global structure generally outperforms local structure
- Is there a way to further combine and improve these basic methods?

SelfTask: Advanced Pretext Tasks on Graphs



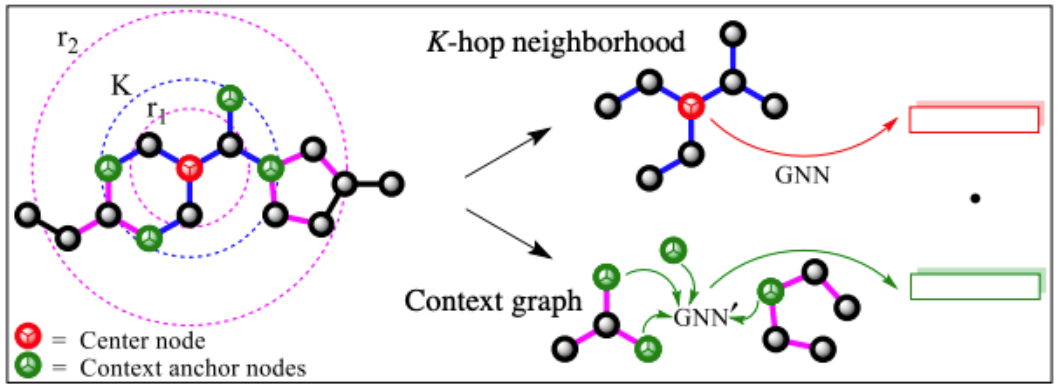
Pre-Training GNNs



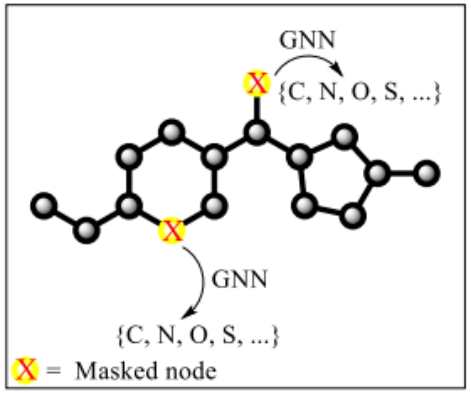
Need for both node-level and graph-level pretraining

Node-level:

- Context Prediction

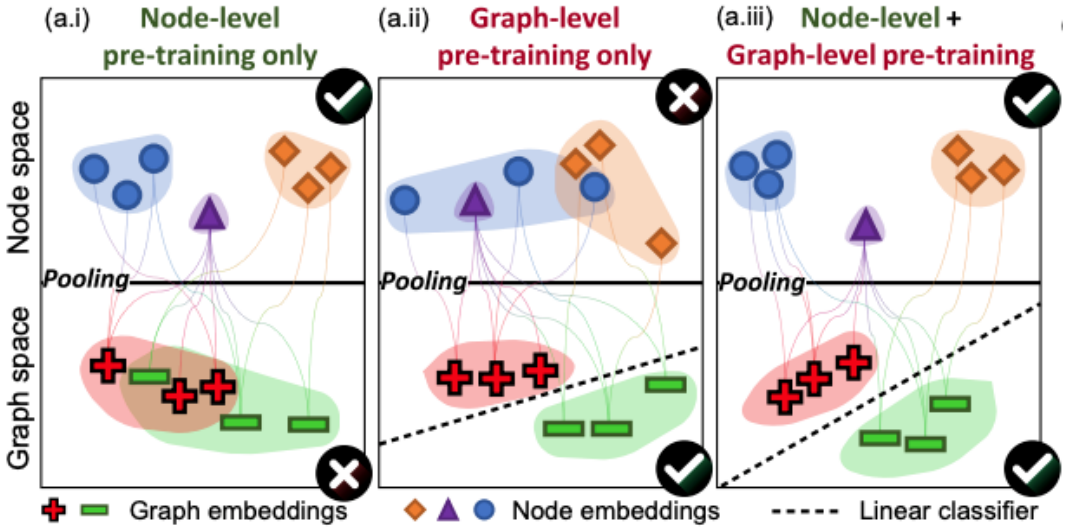


- Attribute Masking



Strategies for Pre-Training Graph Neural Networks. ICLR. 2020.

Pre-Training GNNs



Need for both node-level and graph-level pretraining

Graph-level:

- Domain specific knowledge prediction at the graph level
  - E.g., molecular property prediction in a multi-task format
- Global structural similarity
  - E.g., graph edit distance (although this is time consuming)

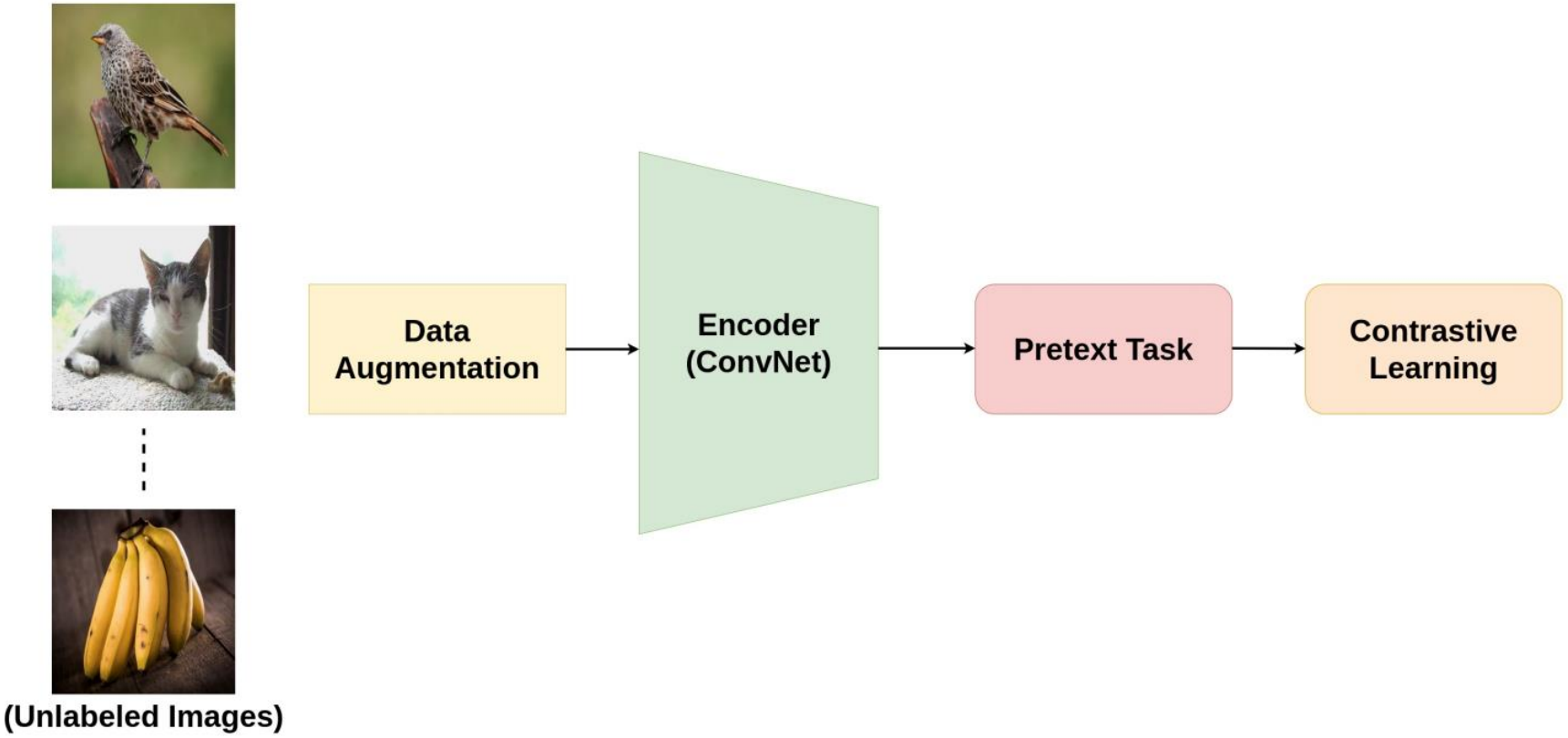
Strategies for Pre-Training Graph Neural Networks. ICLR. 2020.

Methods Applying SSL to Graphs

- Predictive Methods
  - M3S
  - SS-GCN
  - SelfTask
  - Pre-Training GNNs

- Contrastive Methods
  - GraphCL
  - GCC
  - MVGRL
  - GCA

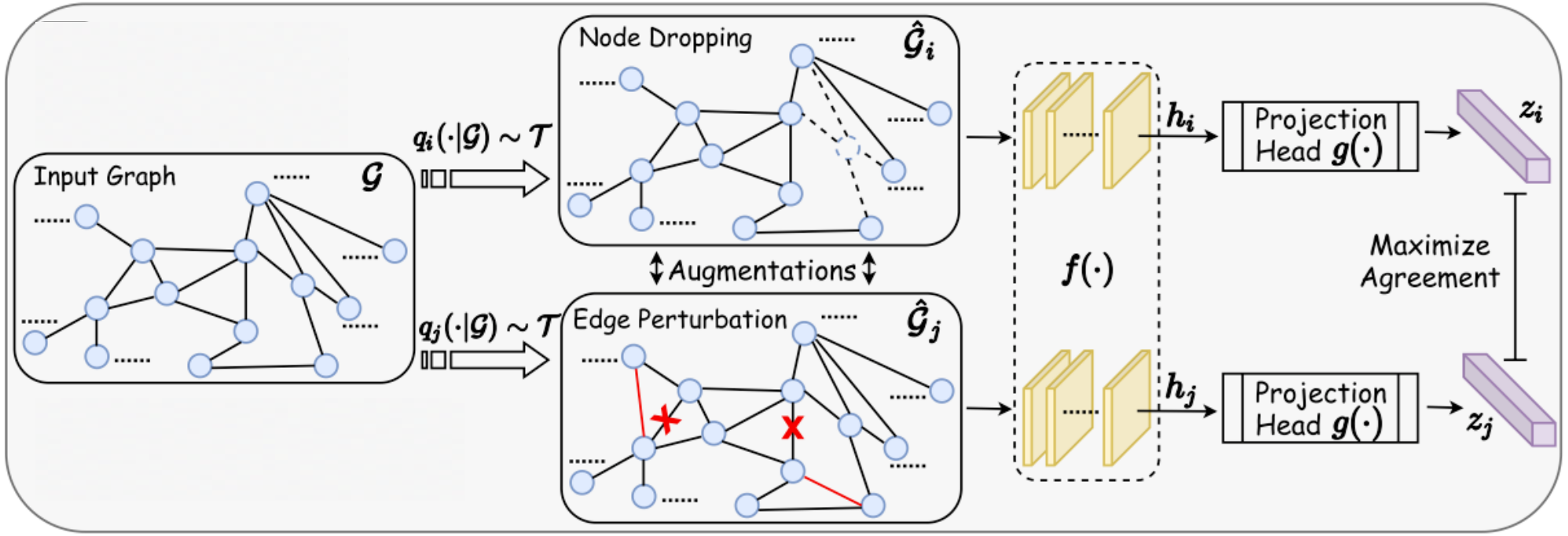
Contrastive Learning in Computer Vision



A typical pretext task in contrastive learning to is predict whether two augmented samples from the same original sample

A Survey on Contrastive Self-supervised Learning. ArXiv. 2020.

GraphCL: Graph vs Graph



- Graph data augmentation
- GNN-based encoder $f(\cdot)$
- Projection head $g(\cdot)$
- Contrastive loss

Graph Contrastive Learning with Augmentations. NeurIPS. 2020.

GraphCL: Graph vs Graph

Data augmentation

Data augmentation	Type
Node dropping	Nodes, edges
Edge perturbation	Edges
Attribute masking	Nodes
Subgraph	Nodes, edges

GNN-based encoder $f(\cdot)$

Projection head $g(\cdot)$

Contrastive loss

GraphCL: Graph vs Graph

Data augmentation

GNN-based encoder $f(\cdot)$

Projection head $g(\cdot)$

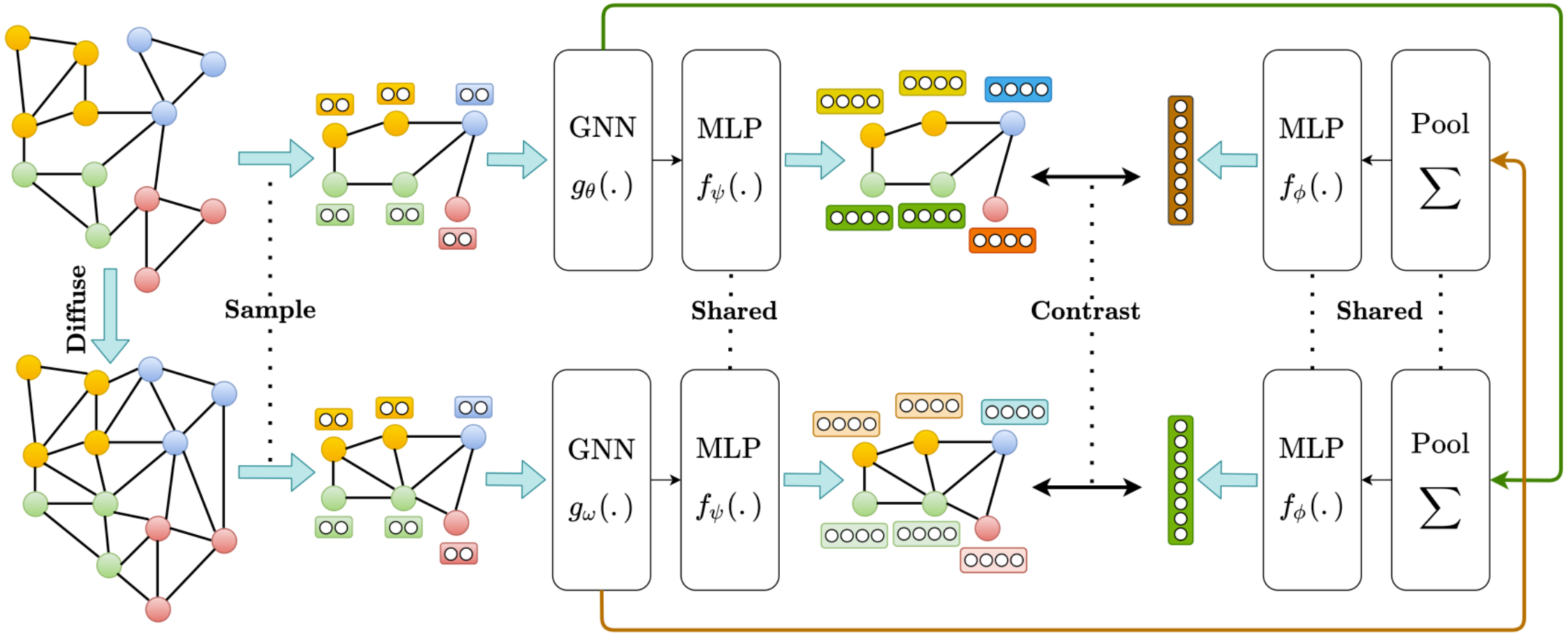
Contrastive loss

For the n-th graph in the minibatch,

$$\ell_n = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_{n,i}, \boldsymbol{z}_{n,j})/\tau)}{\sum_{n'=1, n' \neq n}^{N} \exp(\text{sim}(\boldsymbol{z}_{n,i}, \boldsymbol{z}_{n',j})/\tau)}$$

which is termed as the normalized temperature-scaled cross entropy loss (NT-Xent)

MVGRL: Node vs Subgraph



Contrastive Multi-View Representation Learning on Graphs. ICML. 2020.

MVGRL: Node vs Subgraph

Diffusion view (data augmentation)

Diffusion matrix measures a more global relationship between nodes:

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k \in \mathbb{R}^{N \times N}$$

$$\text{By setting } \mathbf{T} = \mathbf{A}\mathbf{D}^{-1} \text{ and } \theta_k = \alpha(1-\alpha)^k,$$

$$\mathbf{S}^{\text{PPR}} = \alpha \left(\mathbf{I}_n - (1-\alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \right)^{-1}$$

MVGRL: Node vs Subgraph

Objective function

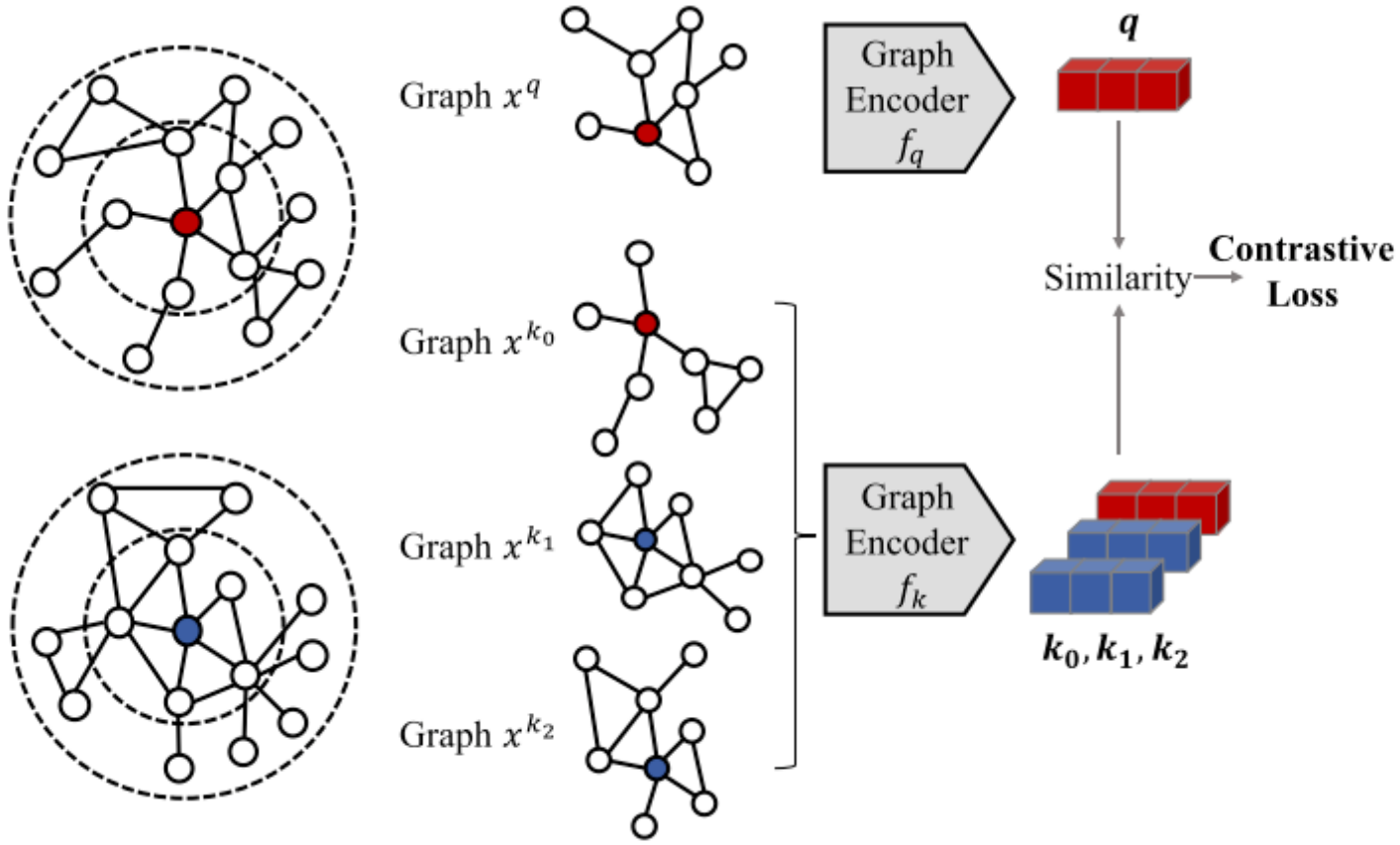
For two views α and β:

$$\max \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \left(\frac{1}{|g|} \sum_{i=1}^{|g|} \mathrm{MI}(\mathbf{h}_i^{\alpha}, \mathbf{h}_g^{\beta}) + \mathrm{MI}(\mathbf{h}_i^{\beta}, \mathbf{h}_g^{\alpha}) \right)$$

where

$$\mathbf{h}_g = \sigma\left(\|_{l=1}^{L} \left(\sum_{i=1}^{n} \mathbf{h}_i^l \right) \mathbf{W} \right) \in \mathbb{R}^{h_d}$$

GCC: Subgraph vs Subgraph



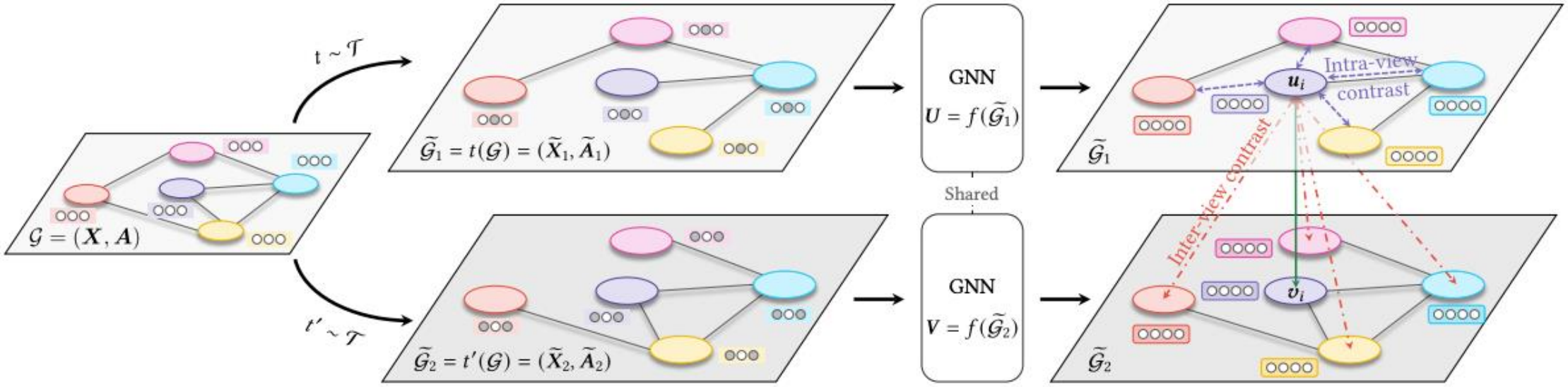
Contrast subgraph embedding!

Positive pairs: The subgraphs augmented from the same r-ego network

Negative pairs: The subgraphs augmented from different r-ego networks

GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training KDD. 2020.

GCA: Node vs Node



Different views:

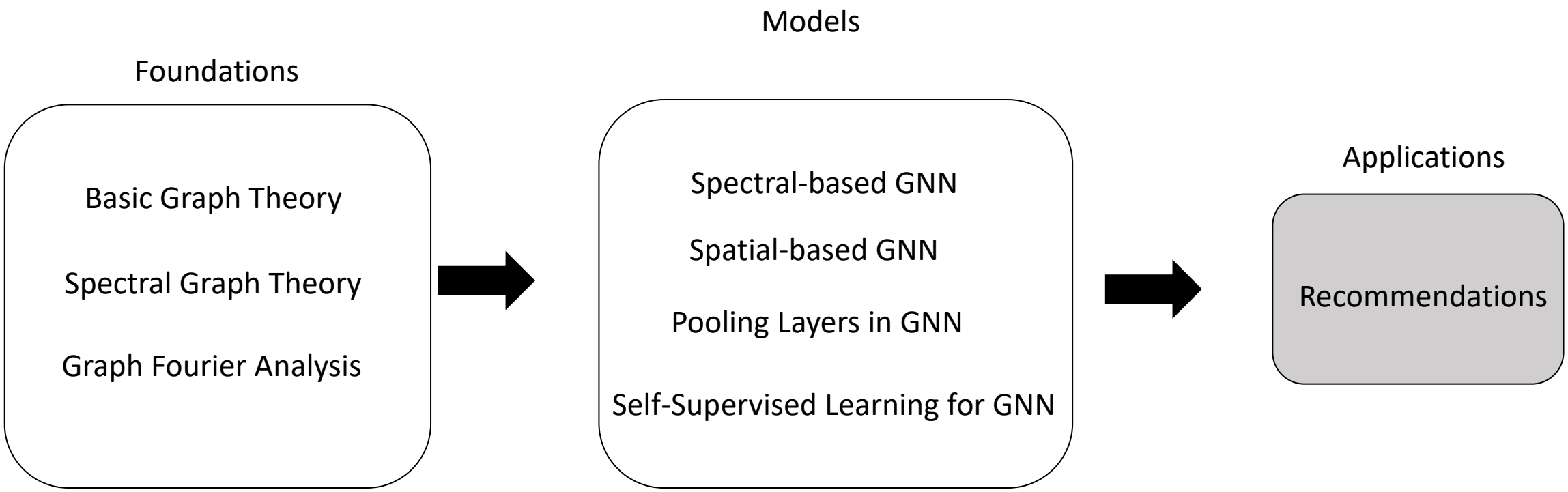
1. Topology-level: Dropping unimportant edges
2. Attribute-level: Dropping unimportant feature dimensions

Graph Contrastive Learning with Adaptive Augmentation WWW. 2021.

Summary of SSL for GNNs

- SSL for GNNs is still in the early stages but seen rapid growth/interest
- The key idea of SSL for GNNs is to design suitable pretext tasks
- Methods have taken a pre-training, self-training, or multi-task training approaches
- Further analysis both theoretically and empirically are desired to better understand when/why/how SSL for GNNs can work

Tutorial Overview



Recommender Systems

Recommender Systems

Suggest relevant items to users

Recommender Systems

Collaborative Filtering

Similar users may share similar preference

Modeling user's preference based on past interactions

Suggest relevant items to users

Recommender Systems

Suggest relevant items to users

Collaborative Filtering

Similar users may share similar preference

Modeling user's preference based on past interactions

users \ items				
	1	0	1	1
	0	1	0	0
	1	1	0	0
	1	0	0	1

0/1 Interaction matrix

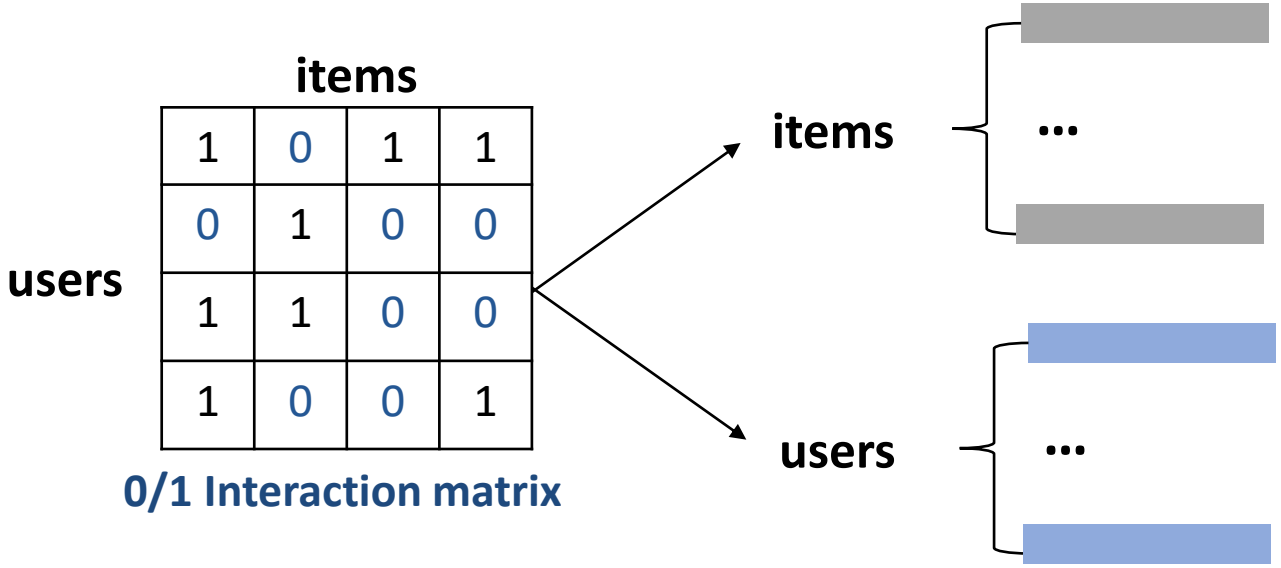
Recommender Systems

Suggest relevant items to users

Collaborative Filtering

Similar users may share similar preference

Modeling user's preference based on past interactions

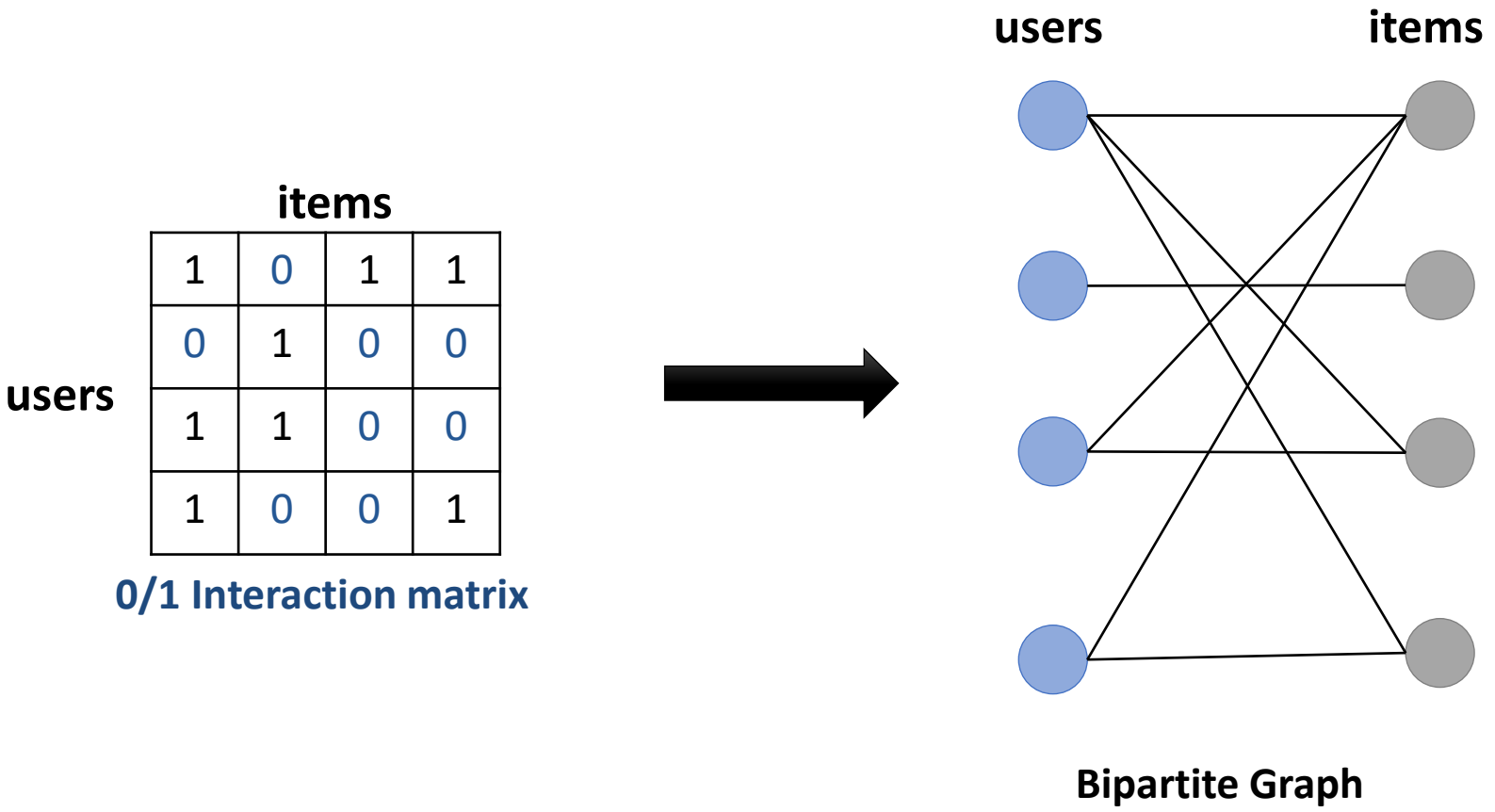


Interactions as Bipartite Graph

users \ items				
	1	0	1	1
	0	1	0	0
	1	1	0	0
	1	0	0	1

0/1 Interaction matrix

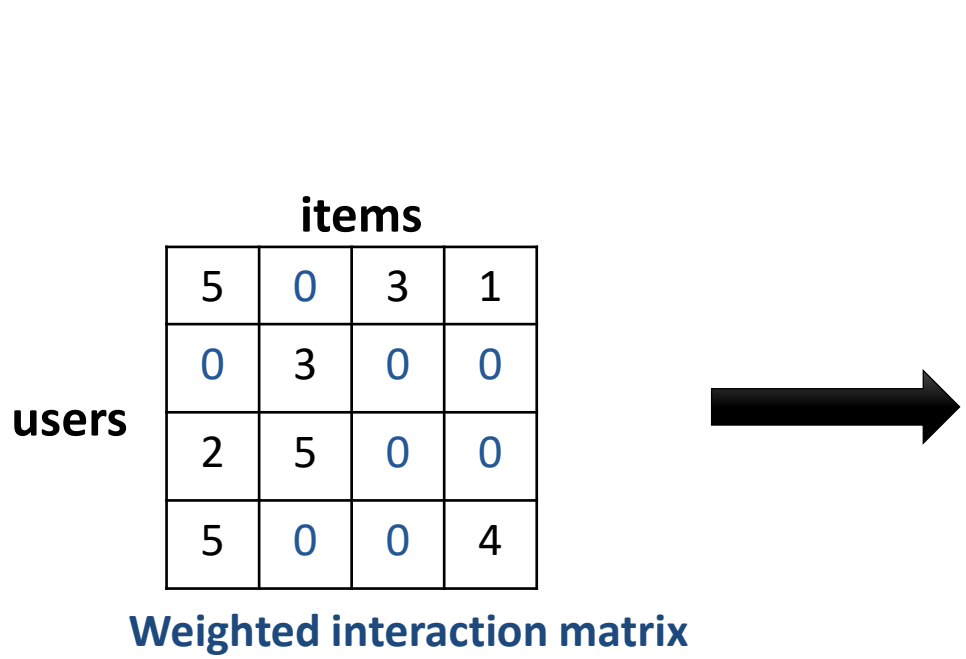
Interactions as Bipartite Graph



Interactions as Bipartite Graph

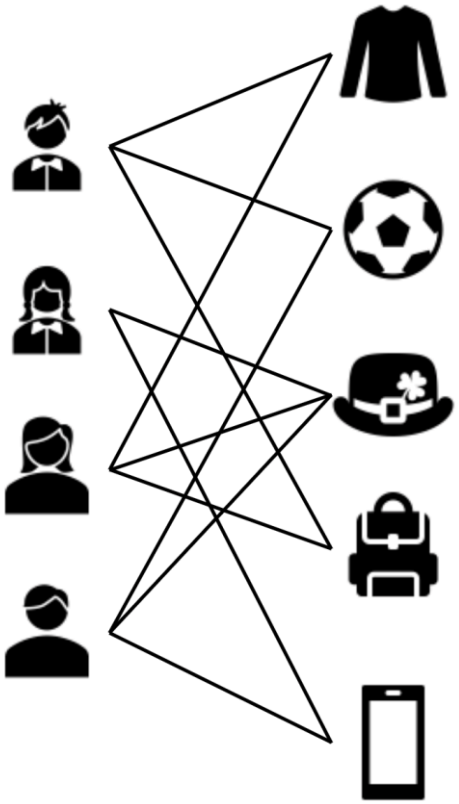
users \ items				
	5	0	3	1
	0	3	0	0
	2	5	0	0
	5	0	0	4

Weighted interaction matrix



Bipartite Graph

Side Information as Graphs



Interaction Graph

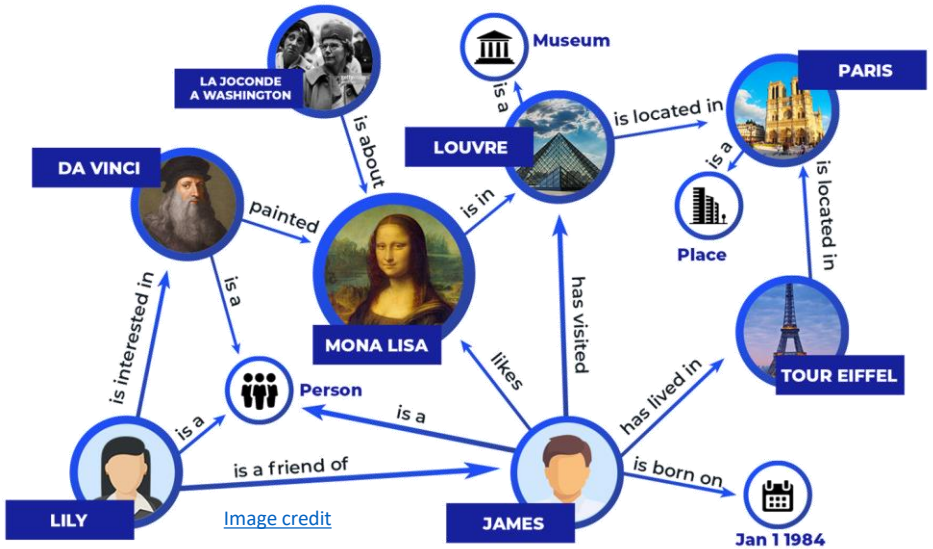
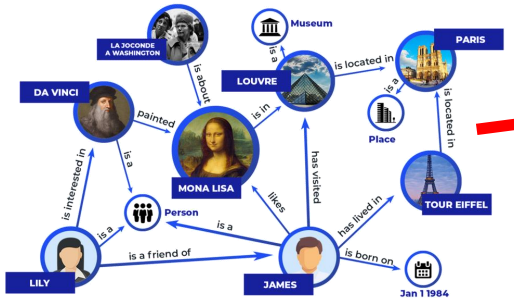


Image credit

Knowledge Graph

Social Networks

GNN based Recommender System

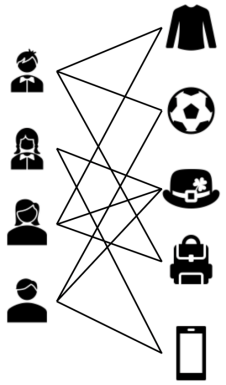


Without Side Information

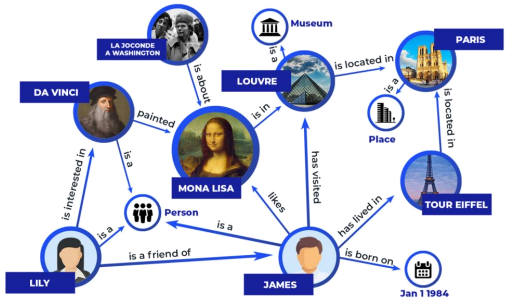
With Side Information about Items

With Side Information about Users

GNN based Recommender System



Without Side Information



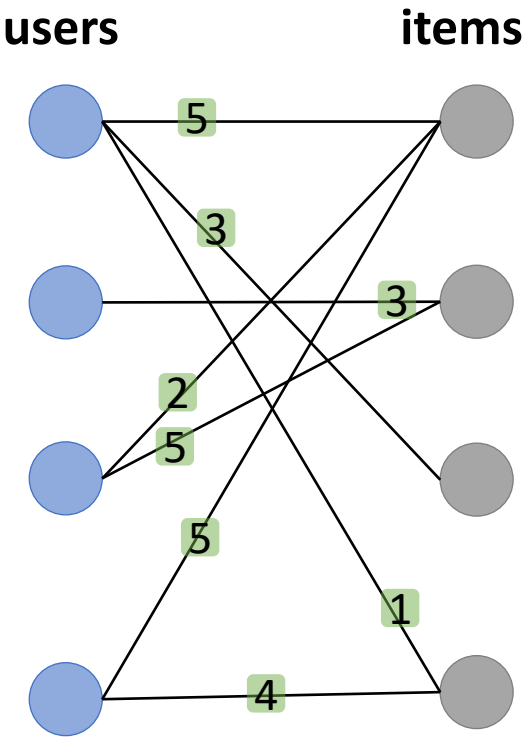
With Side Information about Items

With Side Information about Users

Graph Convolution Matrix Completion

User representation learning

Aggregate for each rating: $\mu_{i,r} = \sum_{j \in \mathcal{N}_{i,r}} \frac{1}{c_{ij}} W_r x_j$



Bipartite Graph

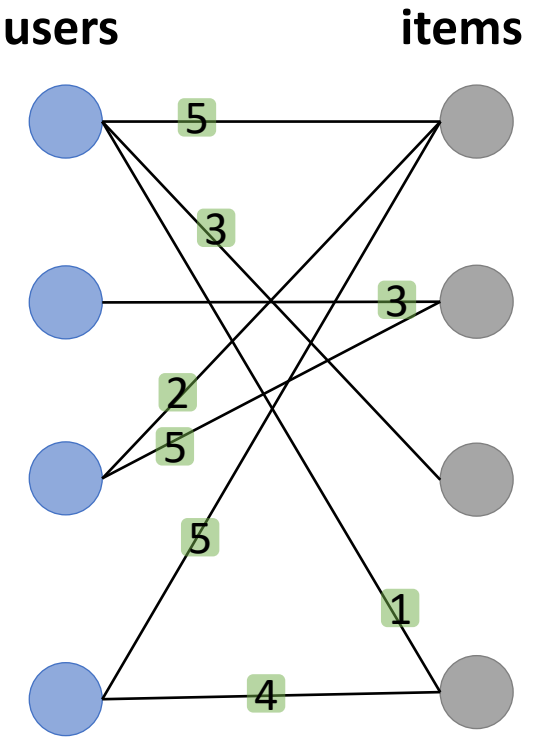
Graph Convolutional Matrix Completion. 2017.

Graph Convolution Matrix Completion

User representation learning

Aggregate for each rating: $\mu_{i,r} = \sum_{j \in \mathcal{N}_{i,r}} \frac{1}{c_{ij}} W_r x_j$

$$u_i = \mathbf{W} \cdot \sigma(accum(u_{i,1}, \ldots, u_{i,R}))$$



Bipartite Graph

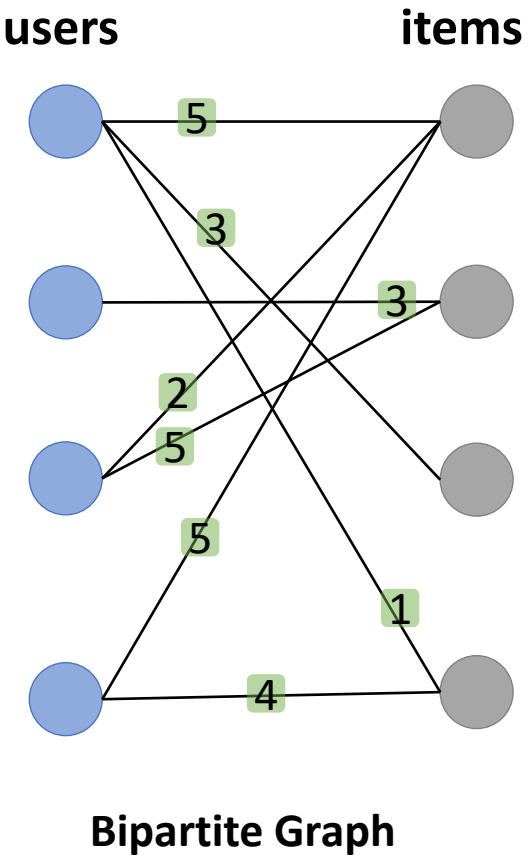
Graph Convolution Matrix Completion

User representation learning

Aggregate for each rating: $\mu_{i,r} = \sum_{j \in \mathcal{N}_{i,r}} \frac{1}{c_{ij}} W_r x_j$

$$u_i = \mathbf{W} \cdot \sigma(accum(u_{i,1}, \ldots, u_{i,R}))$$

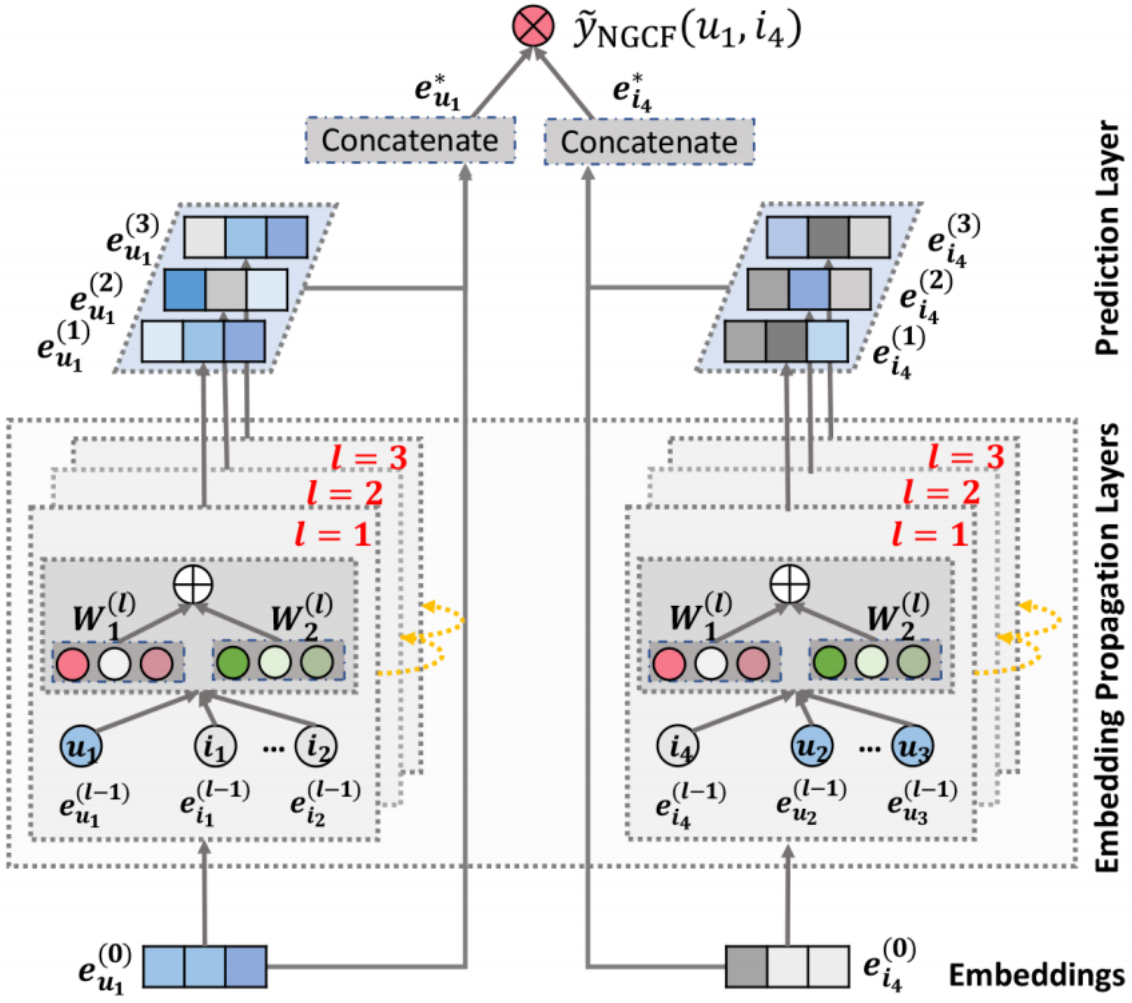
Item representation learning in a similar way



Bipartite Graph

NGCN

Refining the user and item embeddings via GNN



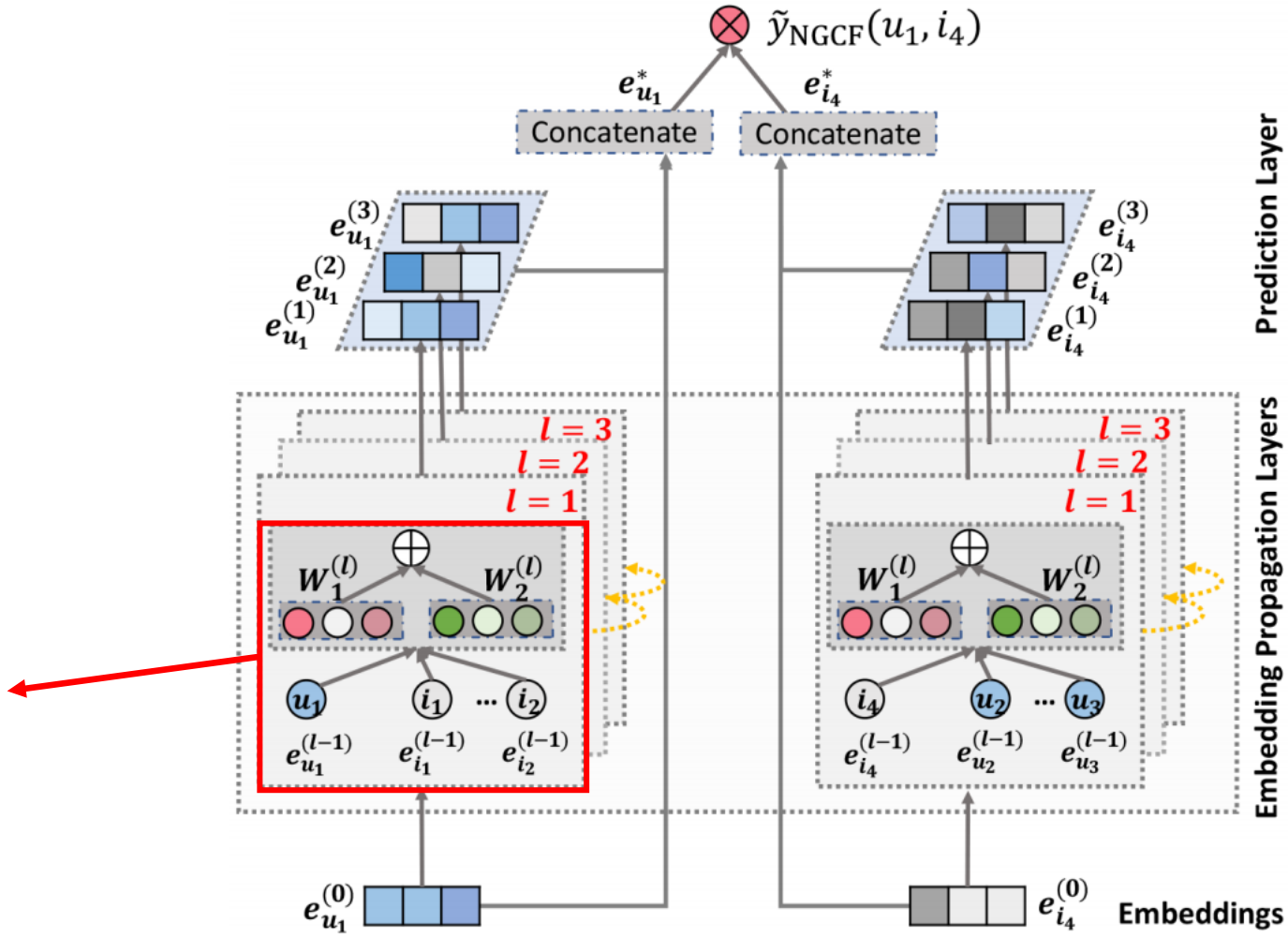
Neural Graph Collaborative Filtering. SIGIR 2019.

NGCN

Refining the user and item embeddings via GNN

$$\begin{cases} \mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right), \\ \mathbf{m}_{u\leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases}$$

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}^{(l)}\right),$$



Neural Graph Collaborative Filtering. SIGIR 2019.

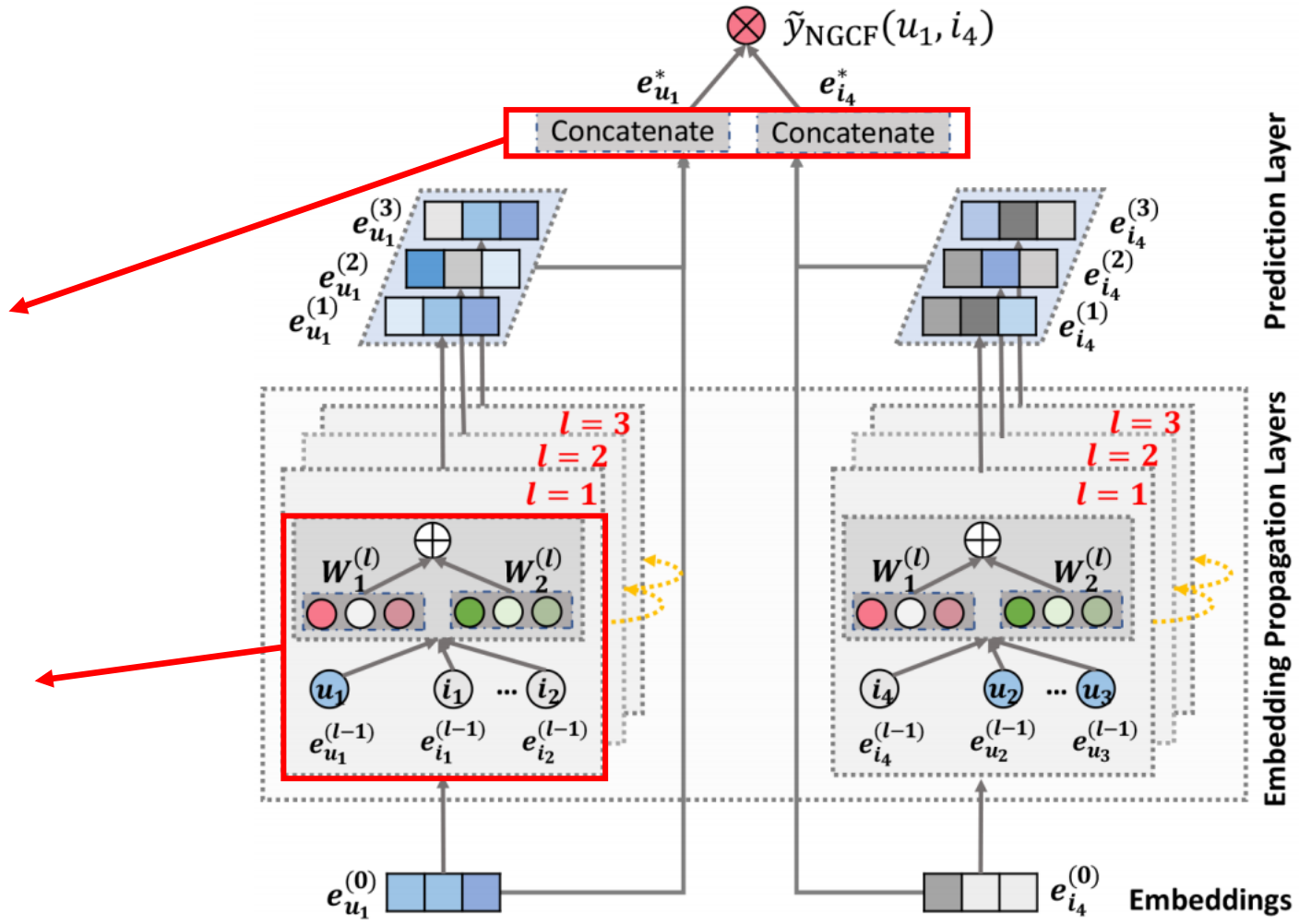
NGCN

Refining the user and item embeddings via GNN

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \| \cdots \| \mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)} \| \cdots \| \mathbf{e}_i^{(L)},$$

$$\begin{cases} \mathbf{m}_{u \leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right), \\ \mathbf{m}_{u \leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases}$$

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u \leftarrow u}^{(l)} + \sum_{i \in \mathcal{N}_u} \mathbf{m}_{u \leftarrow i}^{(l)}\right),$$



Neural Graph Collaborative Filtering. SIGIR 2019.

NGCN

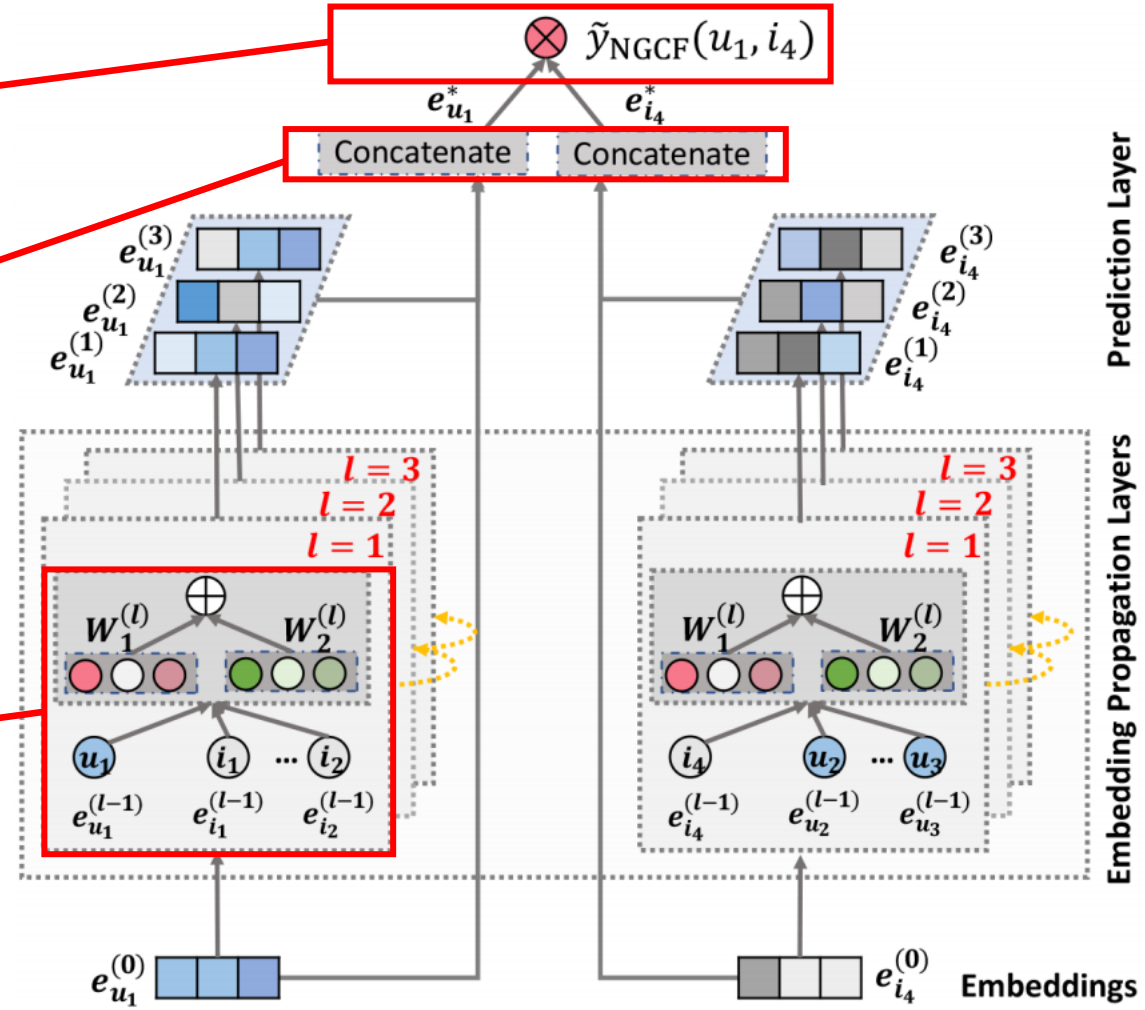
Refining the user and item embeddings via GNN

$$\hat{y}_{\text{NGCF}}(u, i) = \mathbf{e}_u^{*\top} \mathbf{e}_i^*.$$

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \| \cdots \| \mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)} \| \cdots \| \mathbf{e}_i^{(L)},$$

$$\begin{cases} \mathbf{m}_{u \leftarrow i}^{(l)} = p_{ui} \left(\mathbf{W}_1^{(l)} \mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)} (\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)}) \right), \\ \mathbf{m}_{u \leftarrow u}^{(l)} = \mathbf{W}_1^{(l)} \mathbf{e}_u^{(l-1)}, \end{cases}$$

$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u \leftarrow u}^{(l)} + \sum_{i \in \mathcal{N}_u} \mathbf{m}_{u \leftarrow i}^{(l)} \right),$$



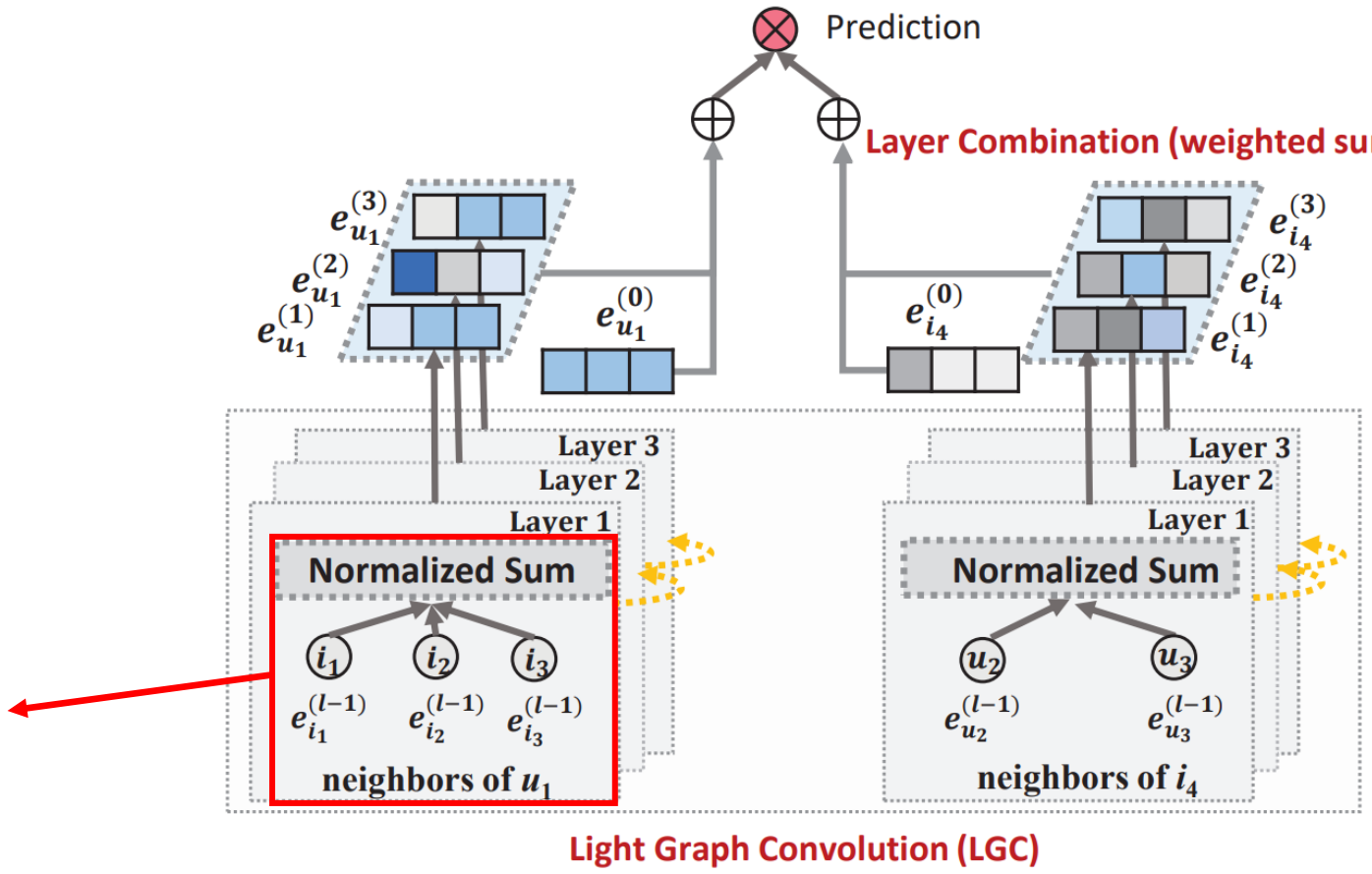
Neural Graph Collaborative Filtering. SIGIR 2019.

LightGCN

Simplifying GCN to make it more concise and appropriate for recommendation

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}.$$



LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. SIGIR 2020.

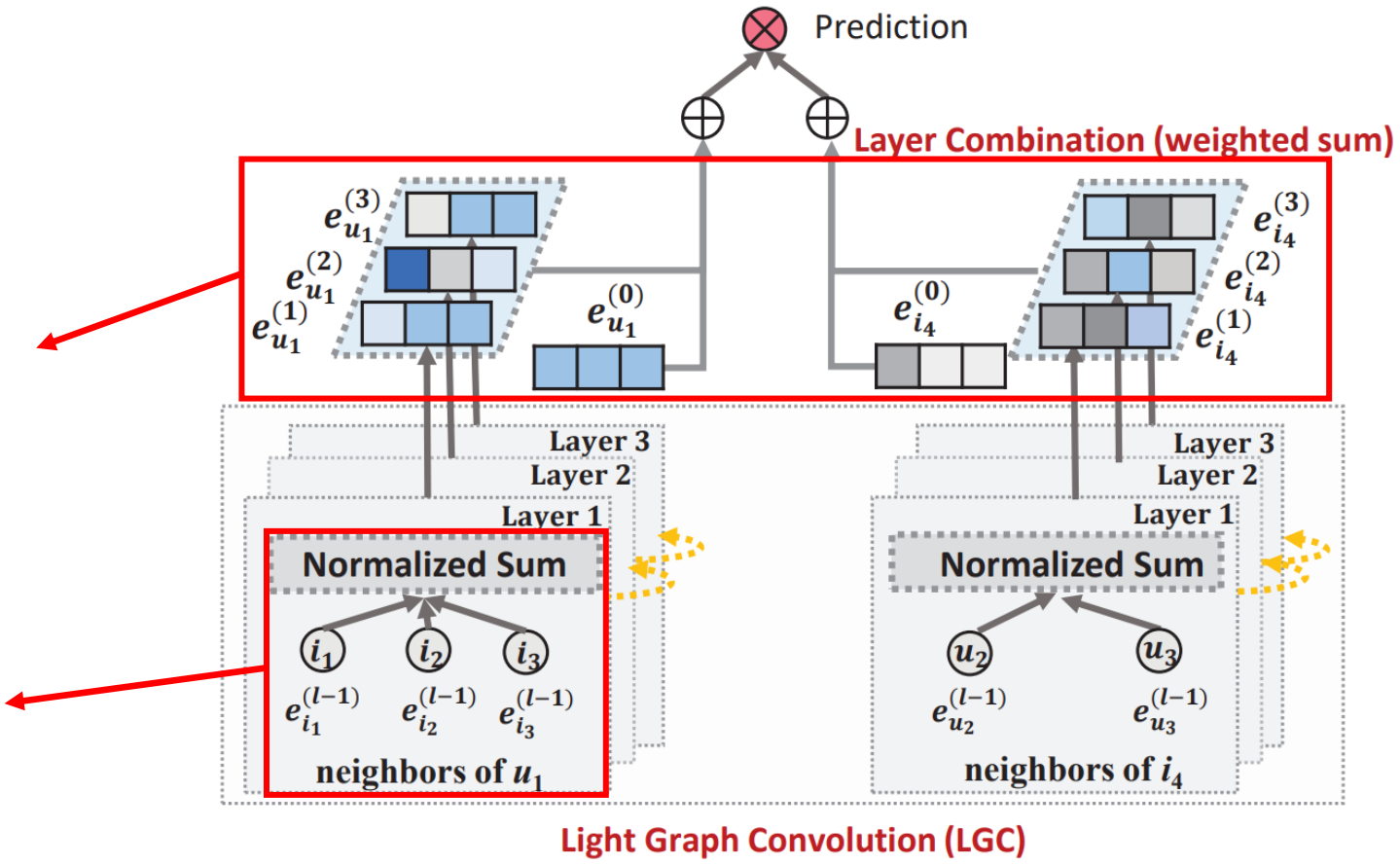
LightGCN

Simplifying GCN to make it more concise and appropriate for recommendation

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)},$$

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}.$$



LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. SIGIR 2020.

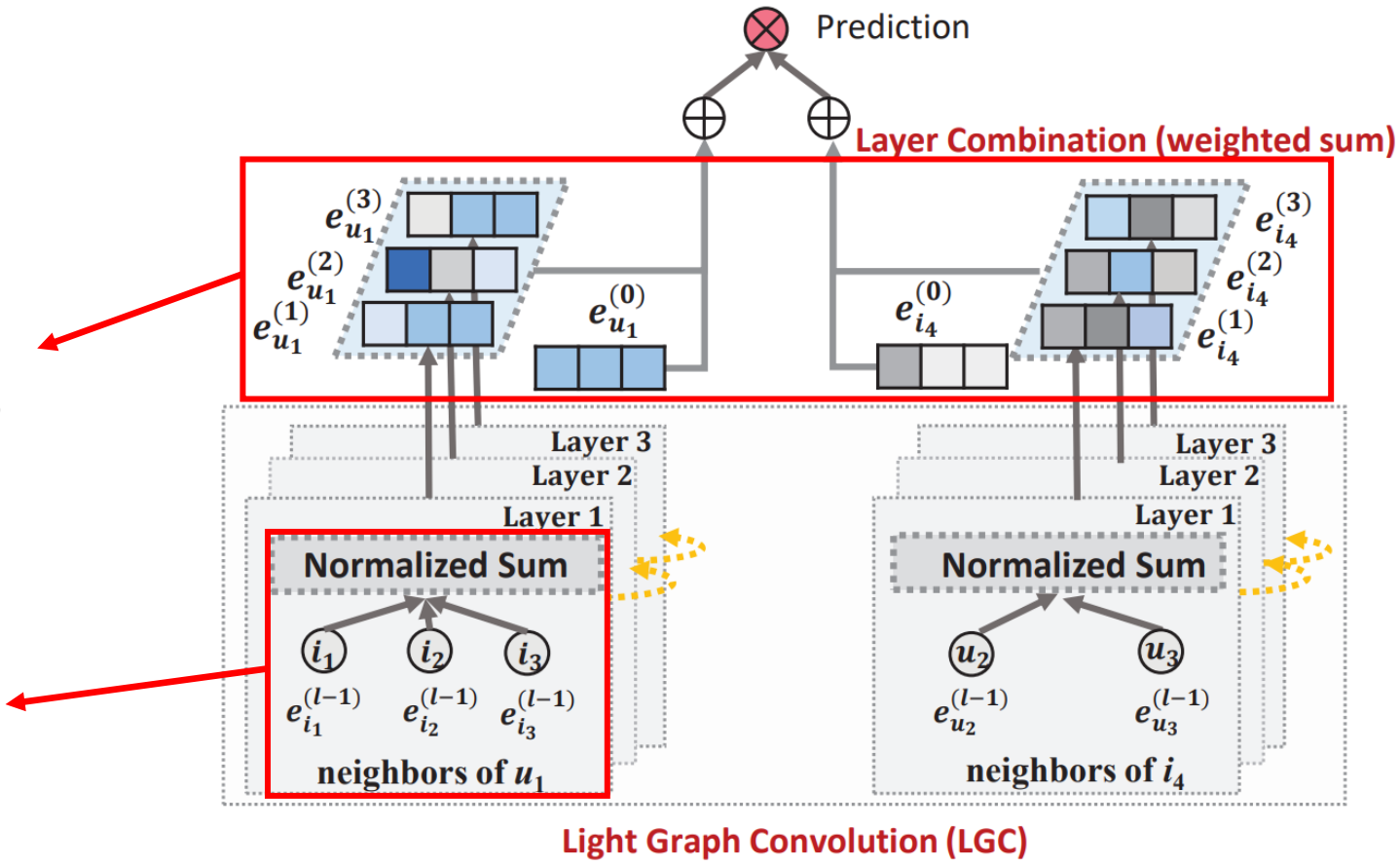
LightGCN

Simplifying GCN to make it more concise and appropriate for recommendation

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)},$$

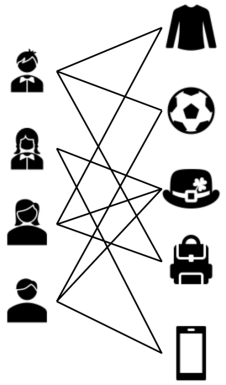
$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)},$$

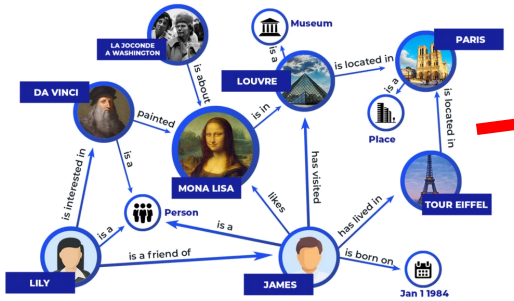
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}.$$



LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. SIGIR 2020.

GNN based Recommender System





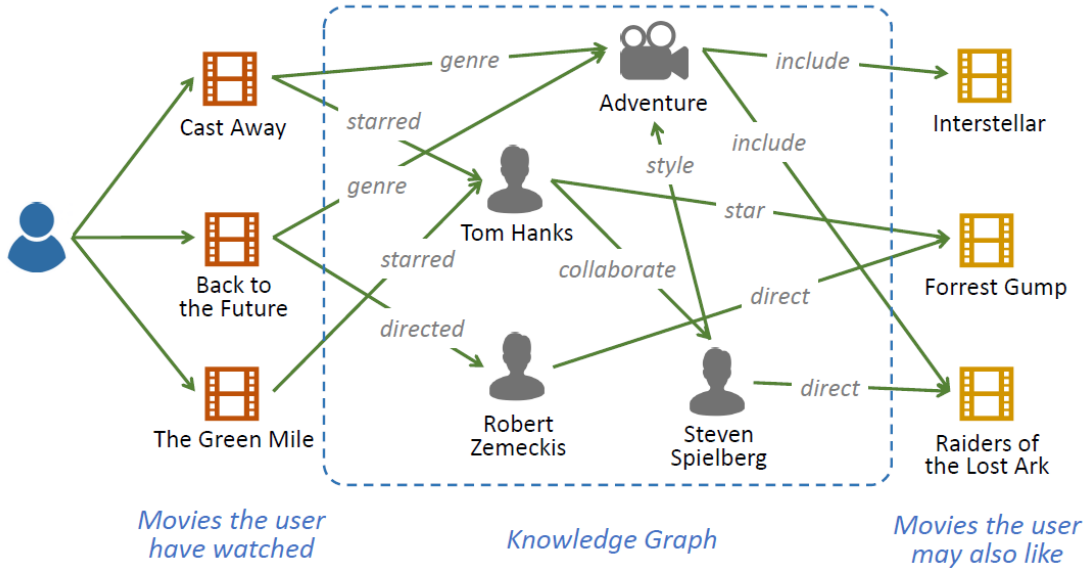
Without Side Information

With Side Information about Items

With Side Information about Users

Knowledge-graph-aware Recommendation

Side information about items: Knowledge graph (KG)

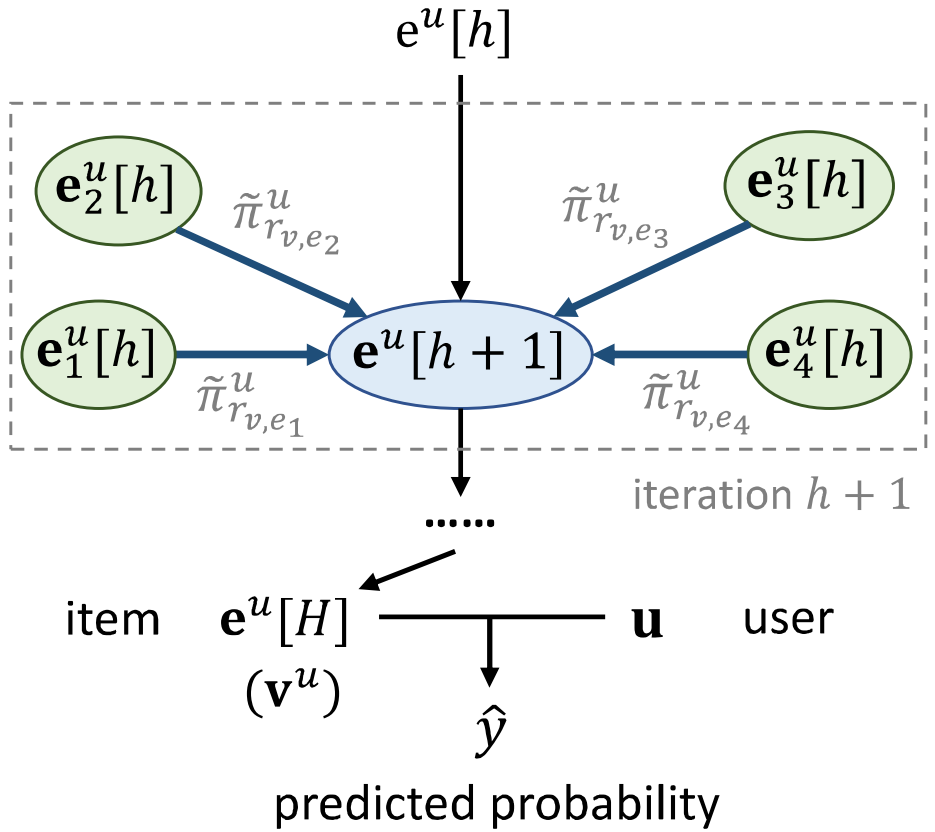


KG can help in learning item representations.

Triples:(head, relation, tail)

KGCN

Learning item representations by aggregation over KG

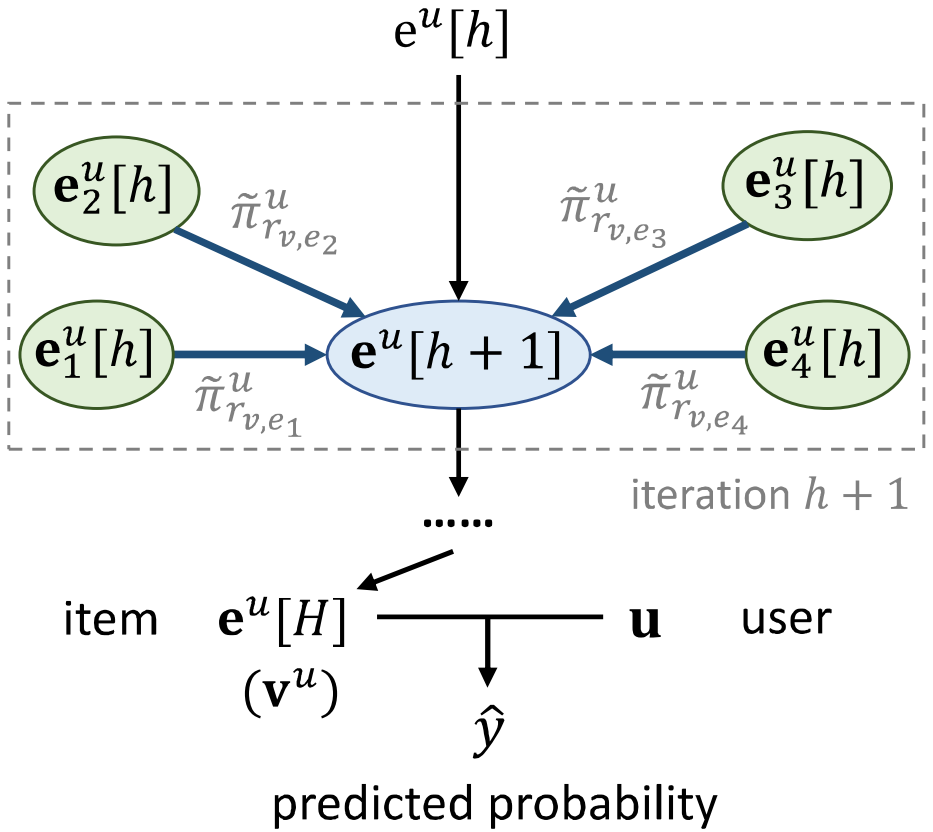


Knowledge Graph Convolutional Networks for Recommender Systems. WWW 2019.

KGCN

Learning item representations by aggregation over KG

$\pi_r^u = g(\mathbf{u}, \mathbf{r})$ user-specific relation

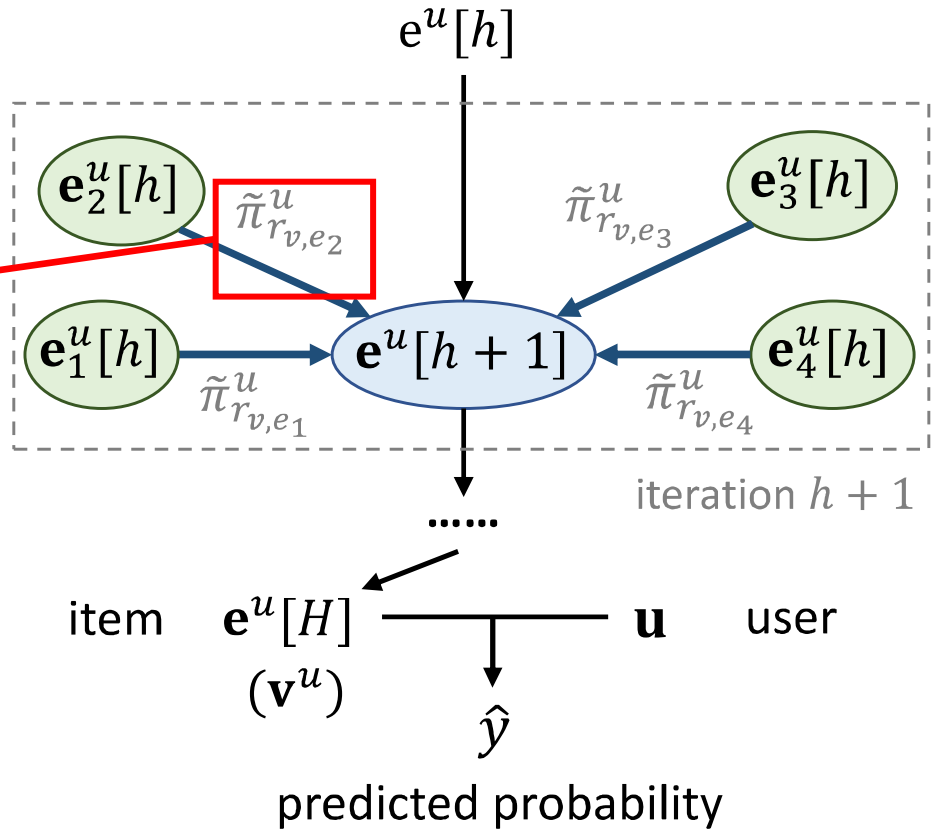


KGCN

Learning item representations by aggregation over KG

$$\pi_r^u = g(\mathbf{u}, \mathbf{r})$$ user-specific relation

$$\tilde{\pi}^u_{r_{v,e}} = \frac{\exp\left(\pi^u_{r_{v,e}}\right)}{\sum_{e \in \mathcal{N}(v)} \exp\left(\pi^u_{r_{v,e}}\right)}$$



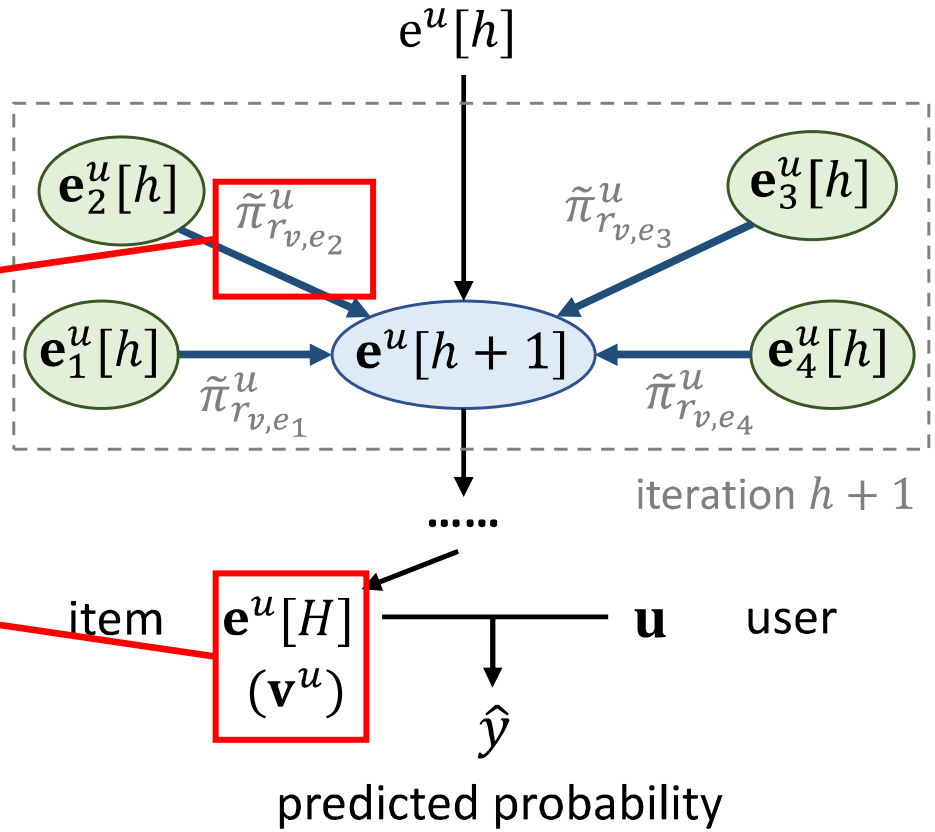
KGCN

Learning item representations by aggregation over KG

$$\pi_r^u = g(\mathbf{u}, \mathbf{r})$$ user-specific relation

$$\tilde{\pi}^u_{r_{v,e}} = \frac{\exp\left(\pi^u_{r_{v,e}}\right)}{\sum_{e \in \mathcal{N}(v)} \exp\left(\pi^u_{r_{v,e}}\right)}$$

$$\mathbf{v}^u_{\mathcal{N}(v)} = \sum_{e \in \mathcal{N}(v)} \tilde{\pi}^u_{r_{v,e}} \mathbf{e}$$



KGCN

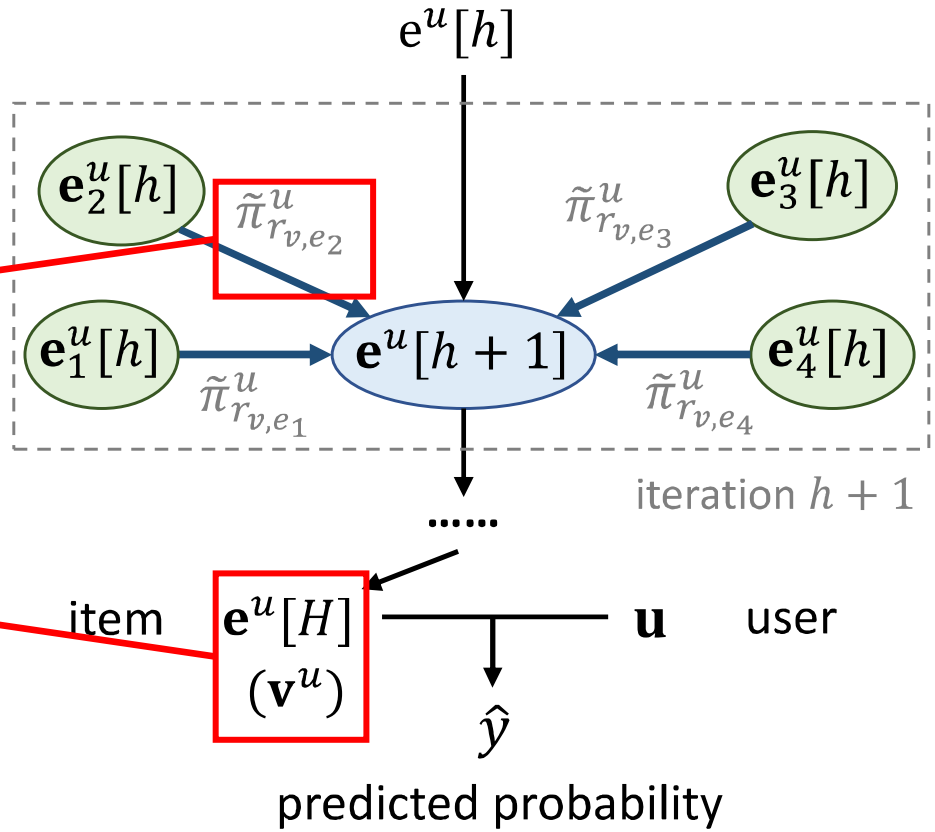
Learning item representations by aggregation over KG

$$\pi_r^u = g(\mathbf{u}, \mathbf{r})$$ user-specific relation

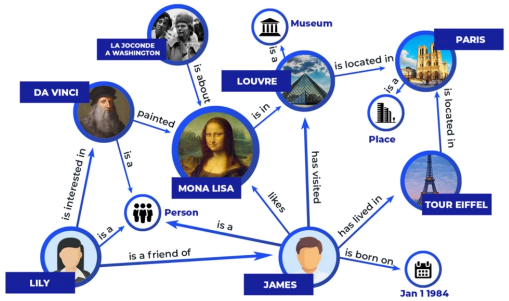
$$\tilde{\pi}^u_{r_{v,e}} = \frac{\exp\left(\pi^u_{r_{v,e}}\right)}{\sum_{e \in \mathcal{N}(v)} \exp\left(\pi^u_{r_{v,e}}\right)}$$

$$\mathbf{v}^u_{\mathcal{N}(v)} = \sum_{e \in \mathcal{N}(v)} \tilde{\pi}^u_{r_{v,e}} \mathbf{e}$$

$$\mathbf{v}^u = f(\mathbf{v}, \mathbf{v}^u_{\mathcal{N}(v)})$$



GNN based Recommender System



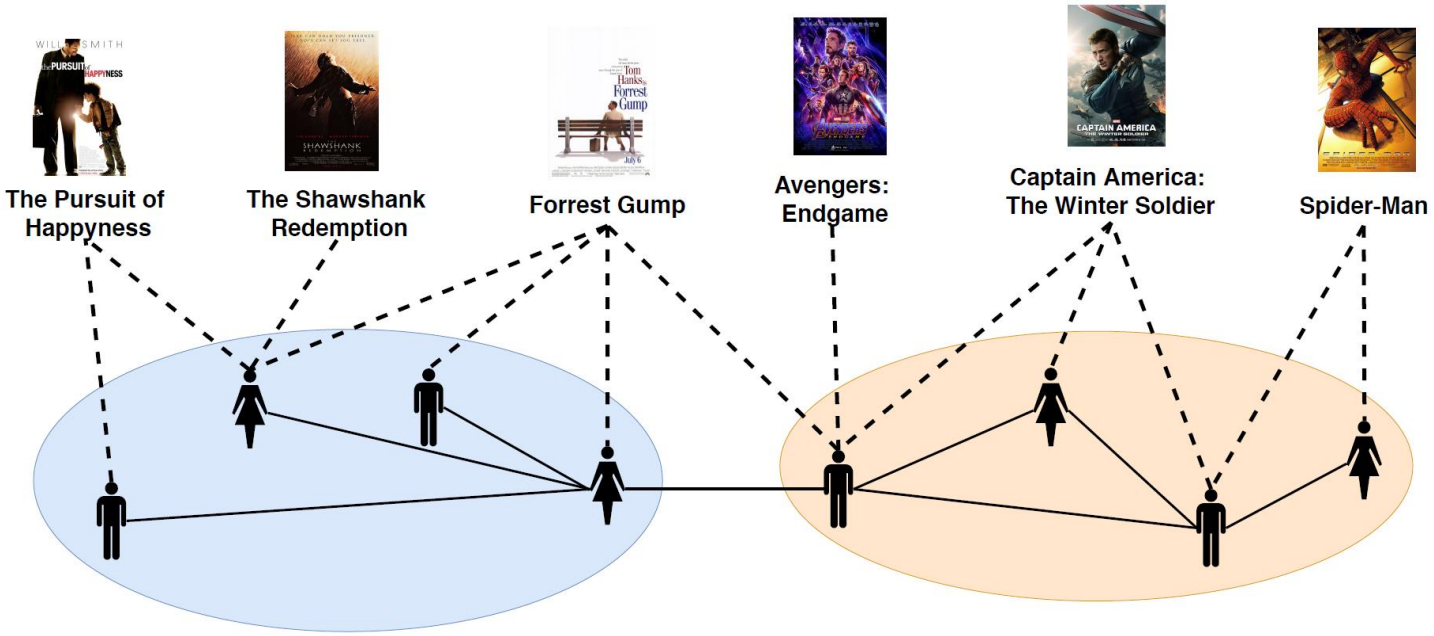
Without Side Information

With Side Information about Items

With Side Information about Users

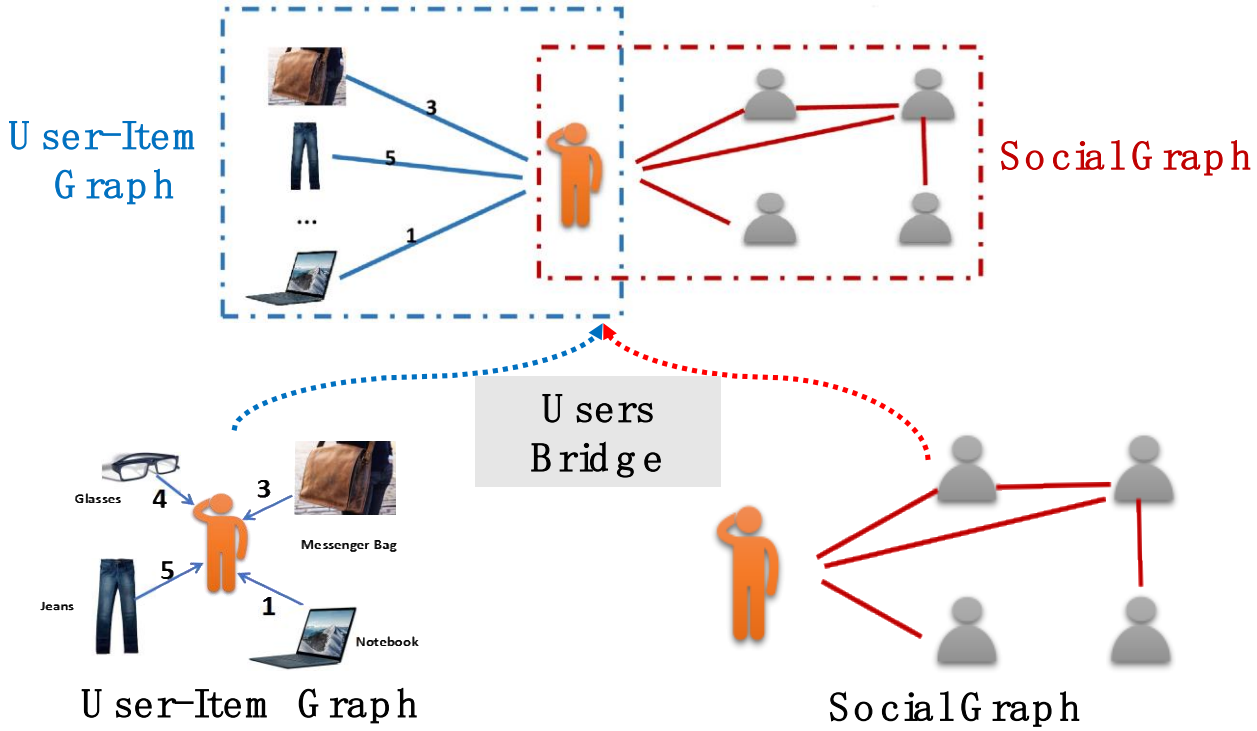
Social Recommendation

Side information about users: social networks



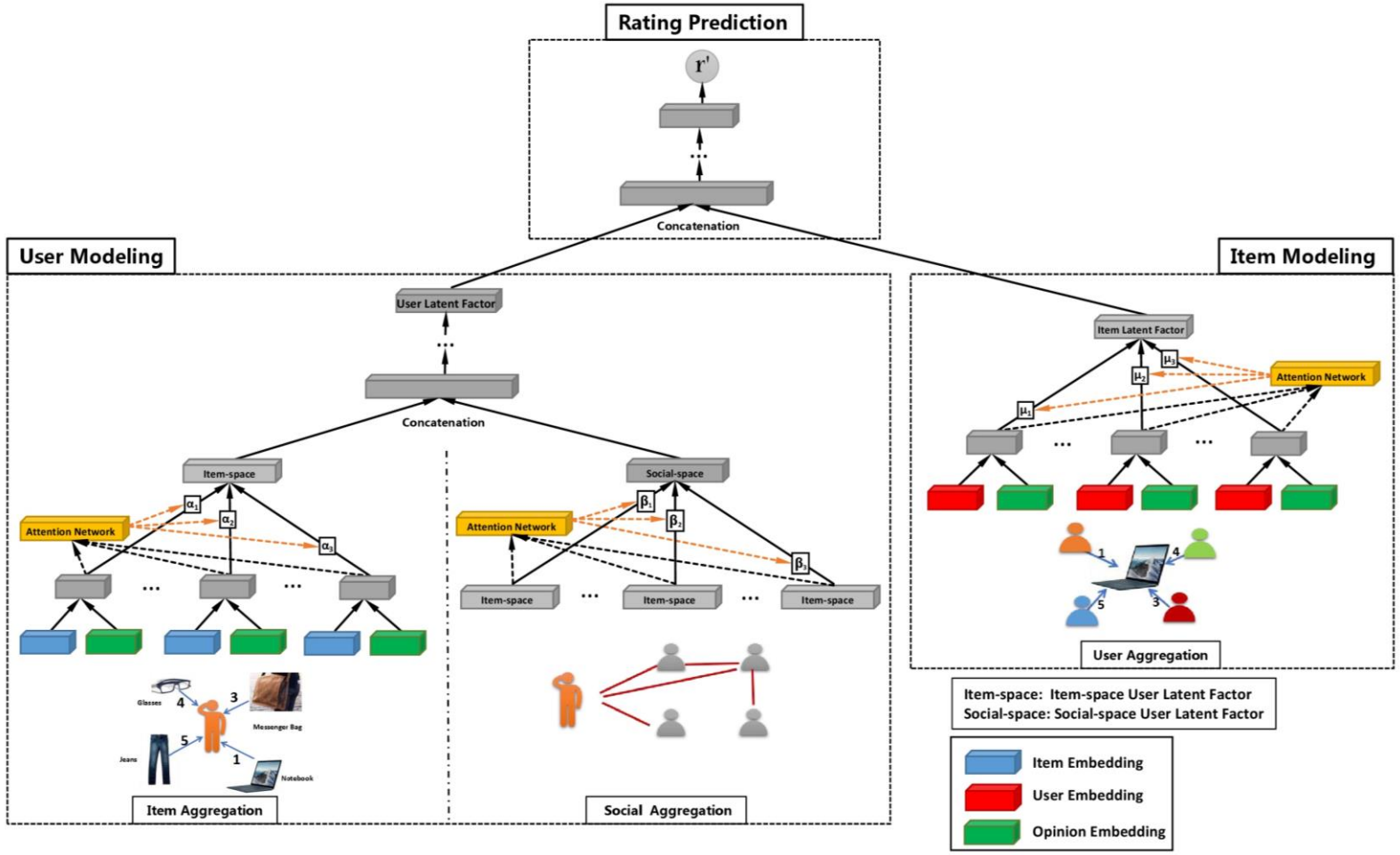
GraphRec

Utilizing both graphs for item and user representation learning

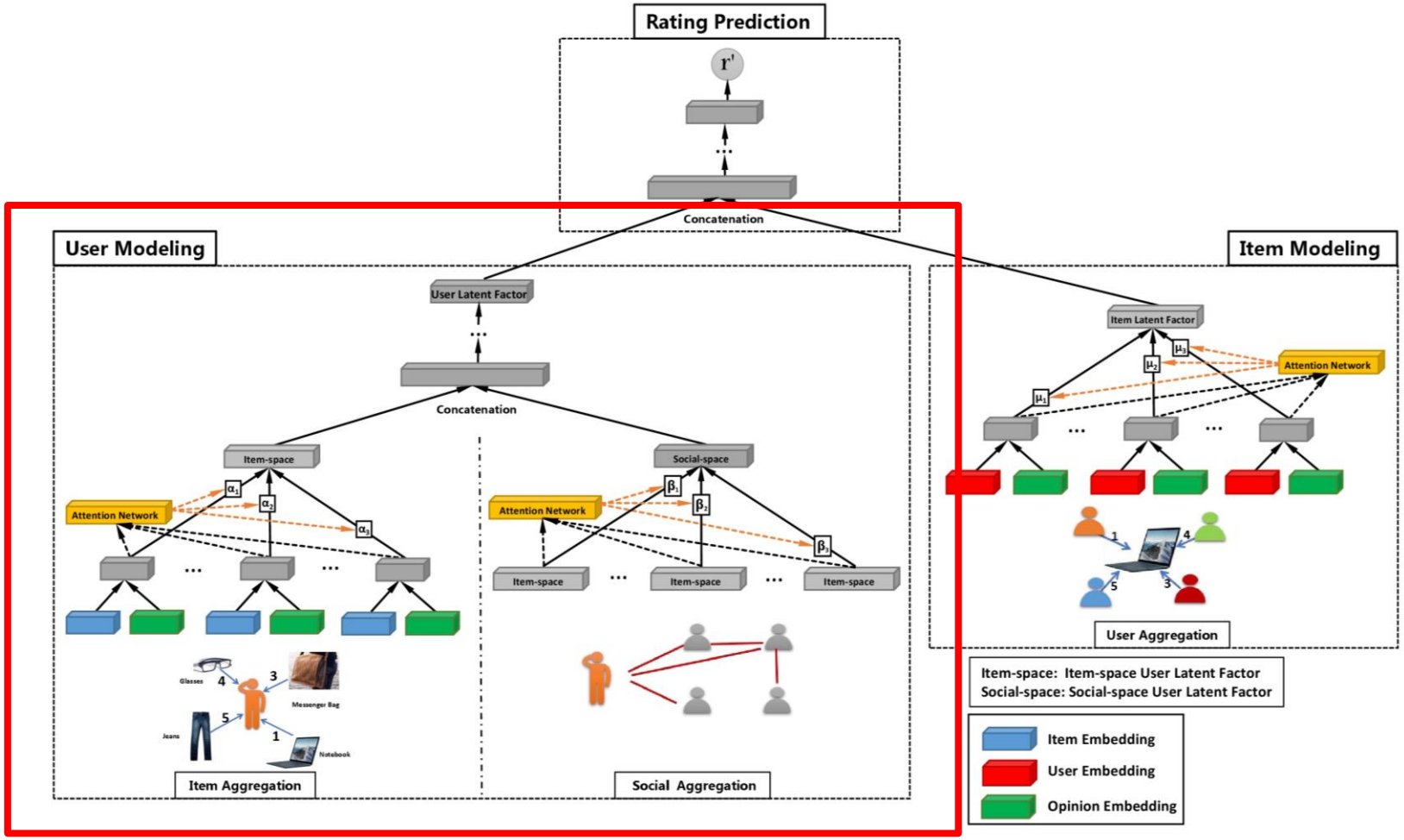


Graph Neural Networks for Social Recommendation. WWW 2019.

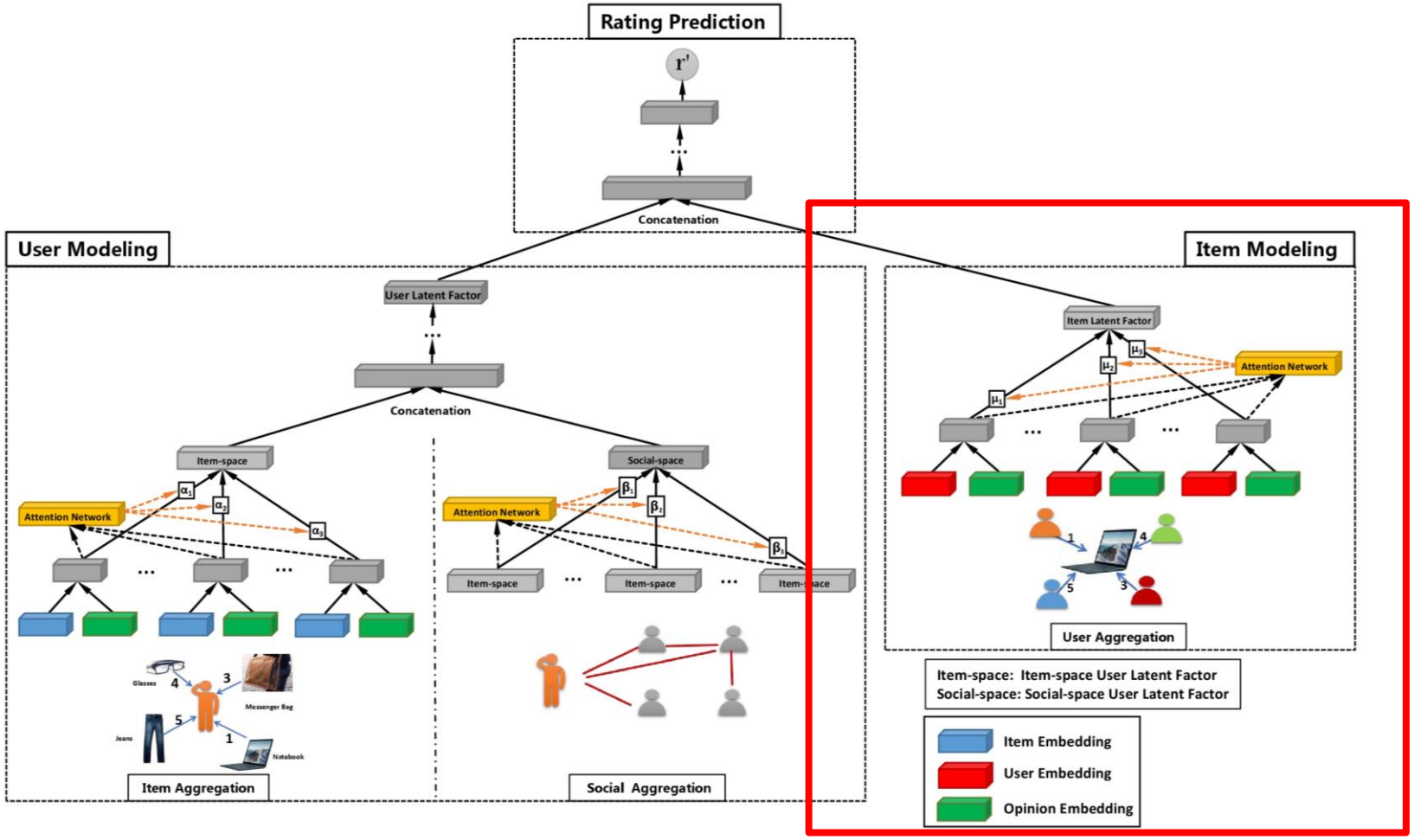
GraphRec


Rating Prediction
Concatenation
User Modeling
Item Modeling
User Latent Factor
Concatenation
Item-space
Social-space
Attention Network
Attention Network
Item-space
Item-space
Item-space
Item Latent Factor
Attention Network
User Aggregation
Item Aggregation
Social Aggregation
Item-space: Item-space User Latent Factor
Social-space: Social-space User Latent Factor
Item Embedding
User Embedding
Opinion Embedding

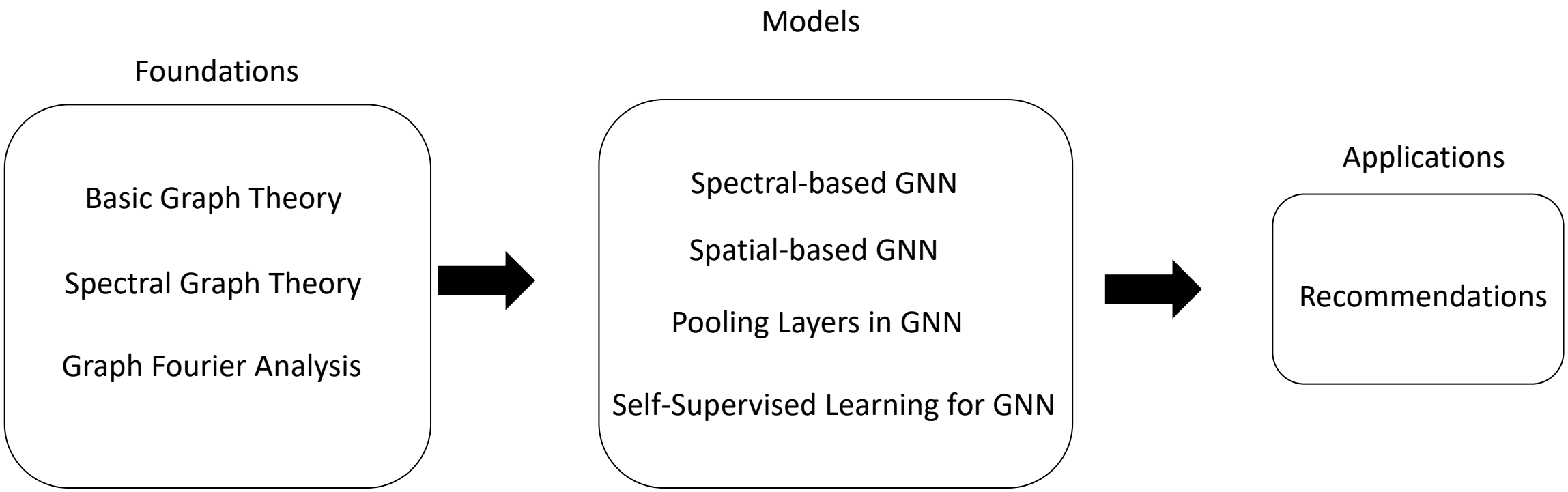
GraphRec


Rating Prediction
Concatenation
User Modeling
User Latent Factor
Concatenation
Item-space
Social-space
Attention Network
Attention Network
Item-space
Item-space
Item-space
Item Aggregation
Social Aggregation
Item Modeling
Item Latent Factor
Attention Network
User Aggregation
Item-space: Item-space User Latent Factor
Social-space: Social-space User Latent Factor
Item Embedding
User Embedding
Opinion Embedding

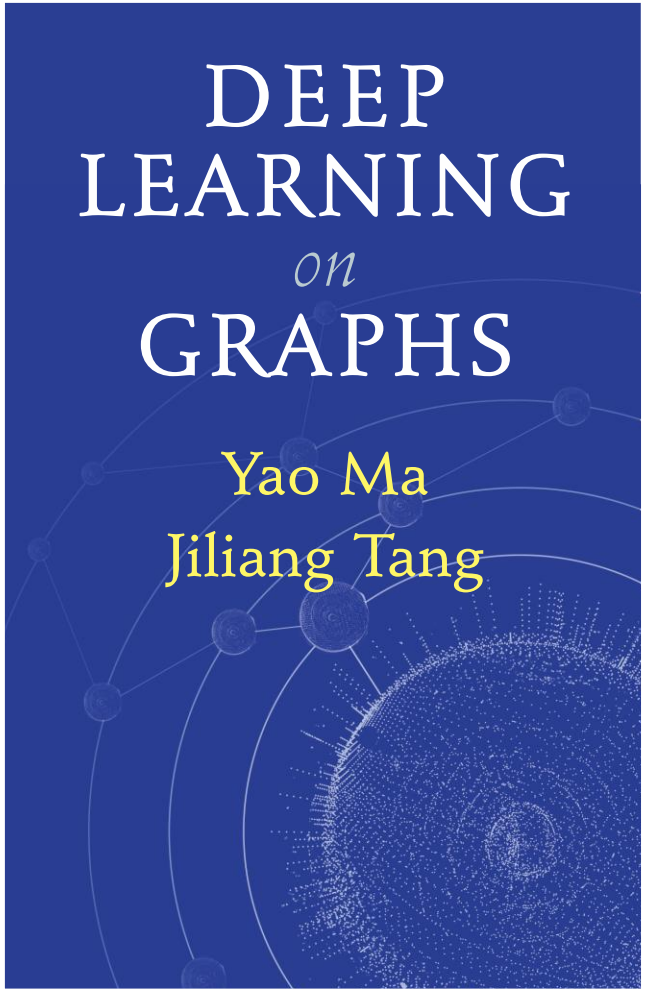
GraphRec


Rating Prediction
Concatenation
User Modeling
User Latent Factor
Concatenation
Item-space
Social-space
Attention Network
Attention Network
Item-space
Item-space
Item Aggregation
Social Aggregation
Item Modeling
Item Latent Factor
Attention Network
User Aggregation
Item-space: Item-space User Latent Factor
Social-space: Social-space User Latent Factor
Item Embedding
User Embedding
Opinion Embedding

Tutorial Overview



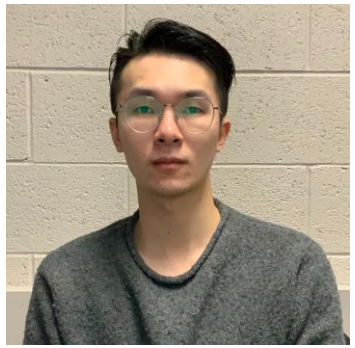
Book: Deep Learning on Graphs



https://cse.msu.edu/~mayao4/dlg_book/

A Repository

GraphRobust: A repository about adversarial attacks and defenses on graph data



DeepRobust: A repository about general adversarial attacks and defenses