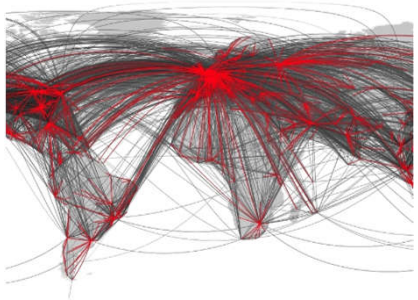
Graph Neural Networks: Models and Applications

Yao Ma, Wei Jin, and Jiliang Tang, Michigan State University
Lingfei Wu and Tengfei Ma, IBM Research

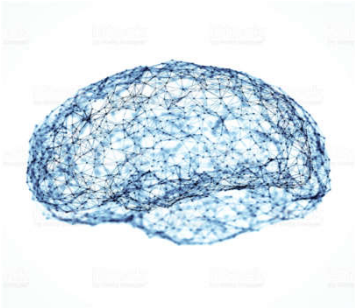
Tutorial website: http://cse.msu.edu/~mayao4/tutorials/aaai2020/

Data as Graphs - Explicit

Social Graphs

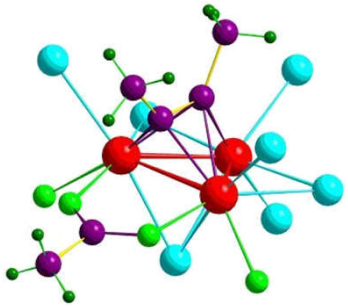


Transportation Graphs

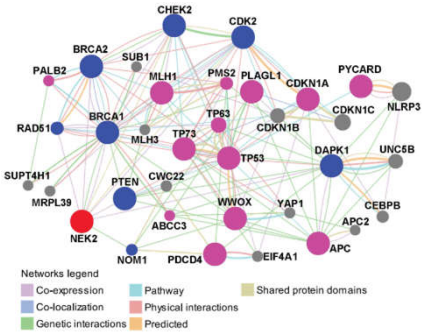


Brain Graphs

Web Graphs

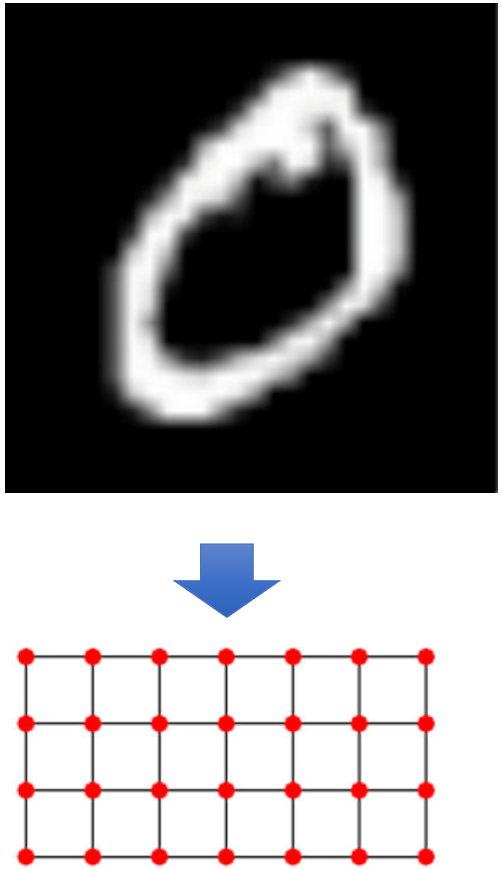


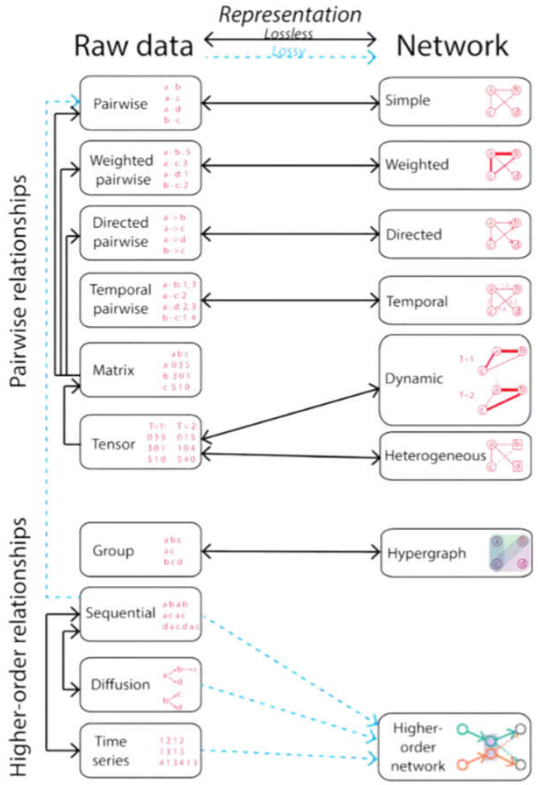
Molecular Graphs



Gene Graphs

Data as Graphs - Implicit





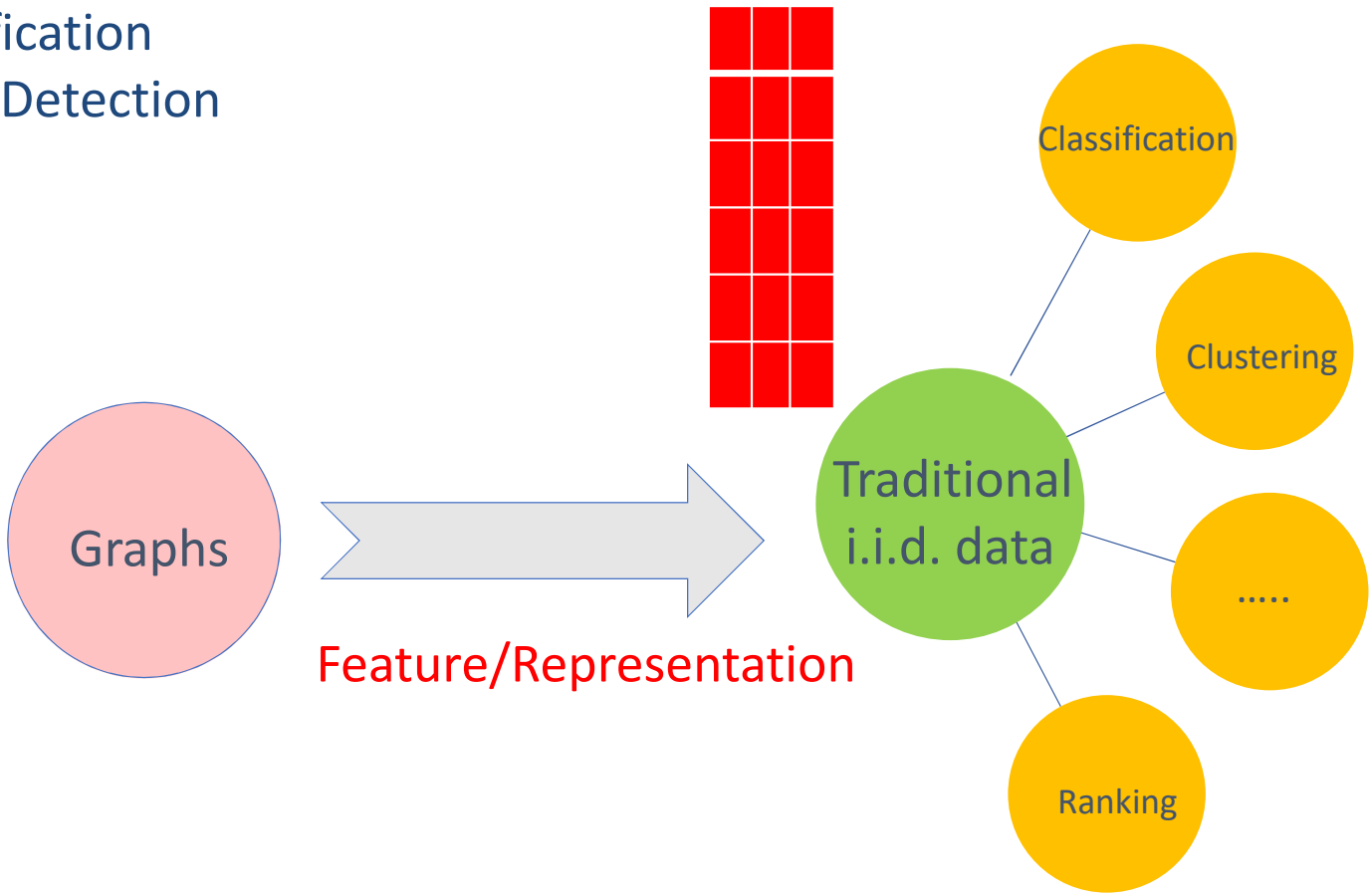
Jian Xu. Representing Big Data as Networks. PhD Dissertation, University of Notre Dame

ML on Graphs

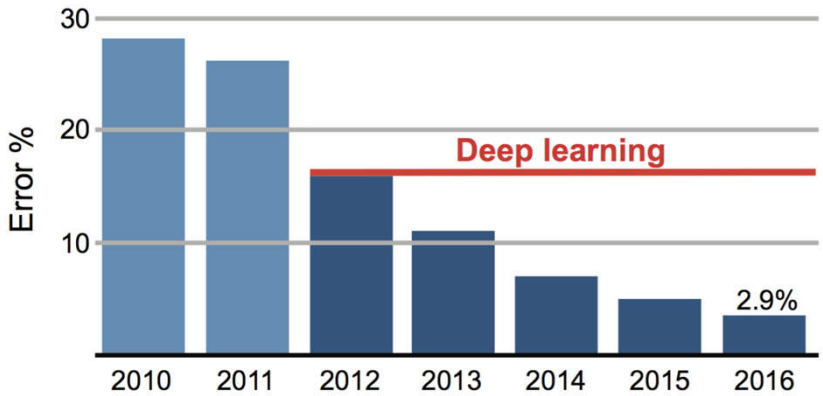
Numerous real-world problems can be summarized as a set of tasks on graphs

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

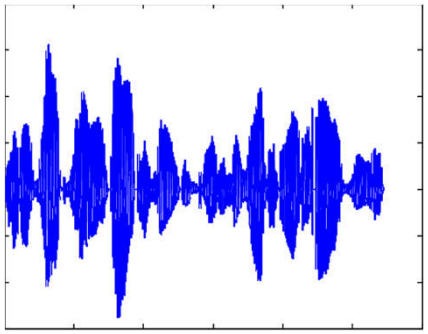
ML solutions



The Power of Deep Learning



IMAGENET

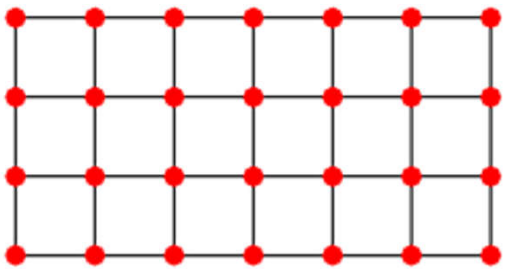


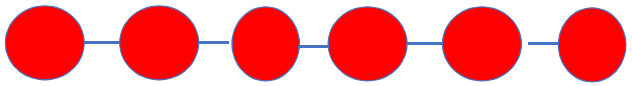
Acoustic model	Recog \ WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass –adapt	27.4	23.6
Deep Learning	1-pass –adapt	18.5	16.1

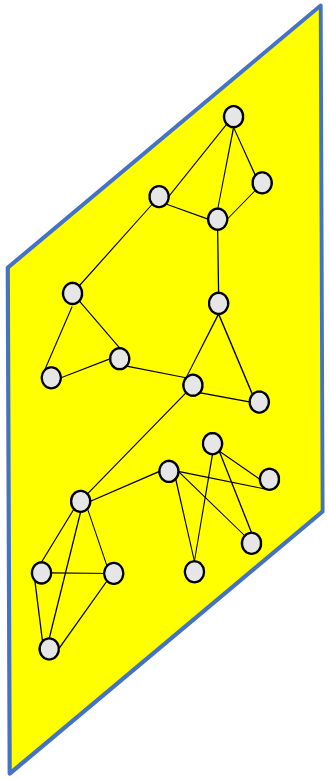
Deep Learning Meets Graphs: Challenges

Traditional DL is designed for simple grids or sequences

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



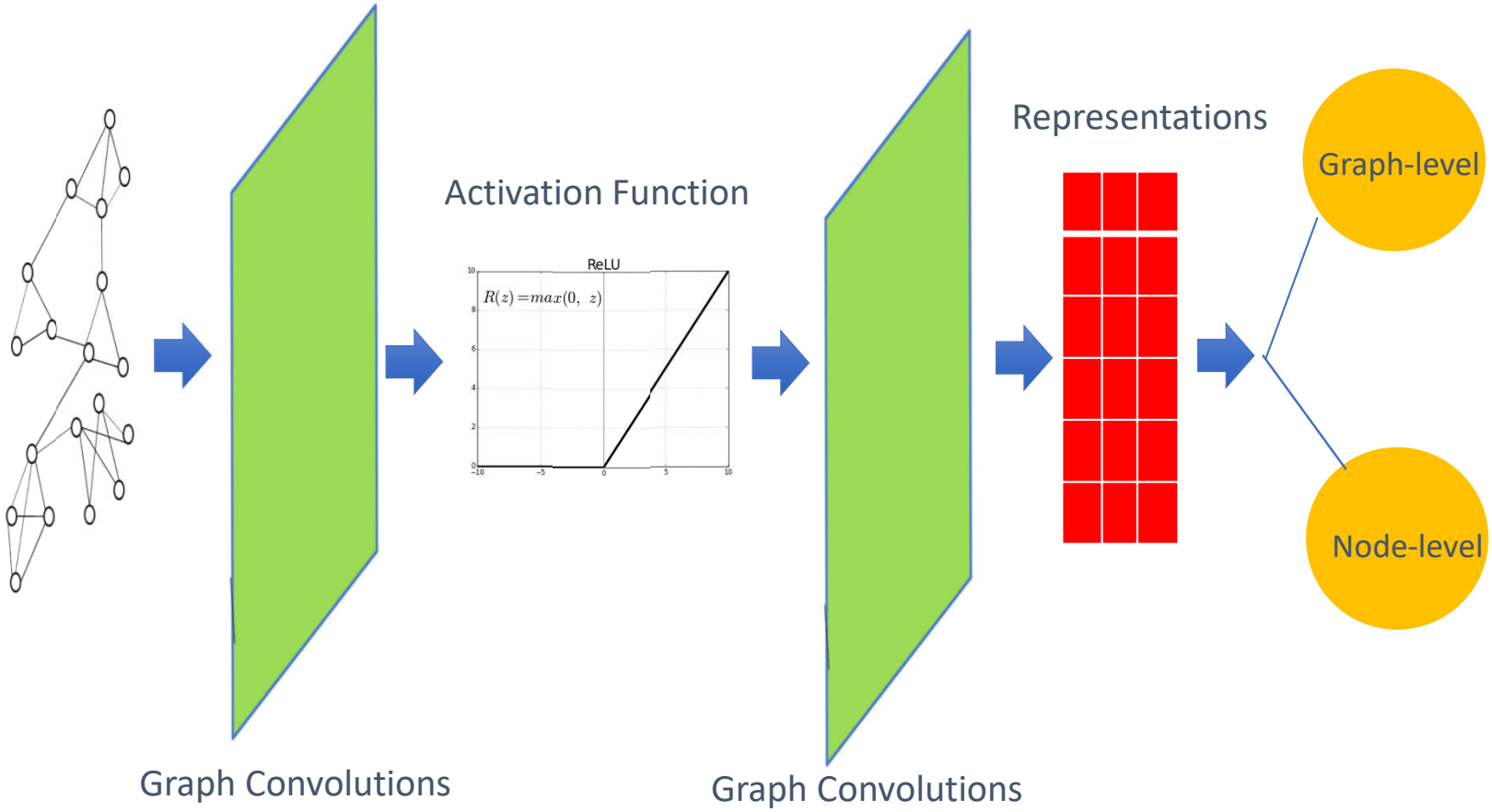




But nodes on graphs have different connections

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

Graph Neural Networks



Tutorial Overview

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

→

Models

- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

→

Applications

- Healthcare
- Natural Language Processing
- Recommendations

Two More Things

A Book

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

Models

- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

Applications

- Healthcare
- Natural Language Processing
- Recommendations

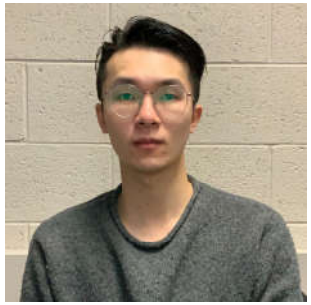
Feature Selection on Graphs
Graph Embedding

More advances in Models
Other Deep models on Graphs
GNNs for Complex Graphs

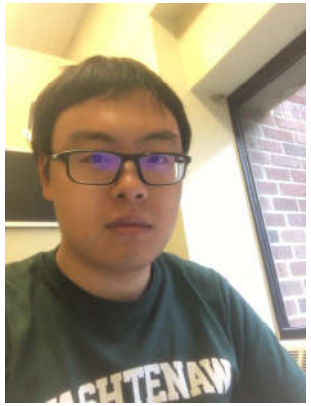
Data Mining
Computer Vision
Biology, Chemistry and Physics

A Repository

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



DeepRobust: A repository about general adversarial attacks and defenses



Tutorial Overview

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

→

Models

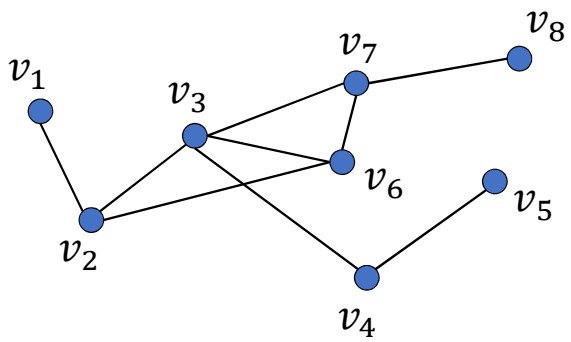
- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

→

Applications

- Healthcare
- Natural Language Processing
- Recommendations

Graphs and Graph Signals

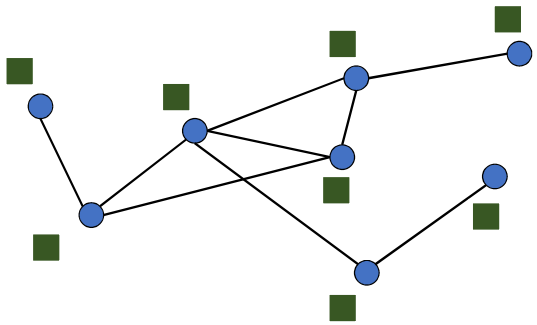


$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

Graphs and Graph Signals



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

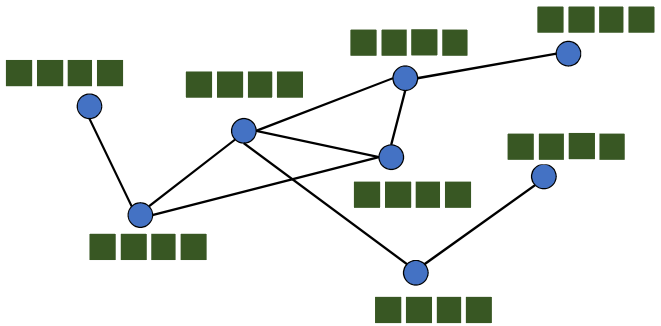
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

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

Graphs and Graph Signals



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

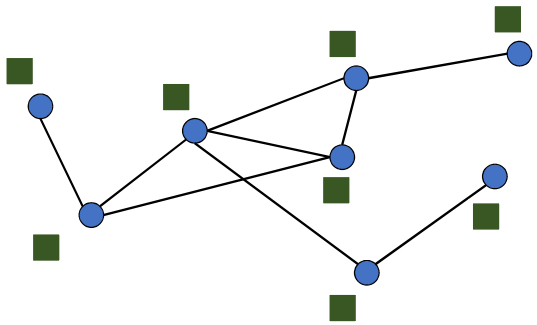
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

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

Graphs and Graph Signals



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

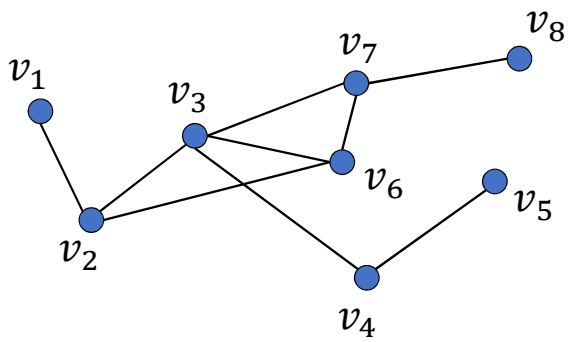
$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

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

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

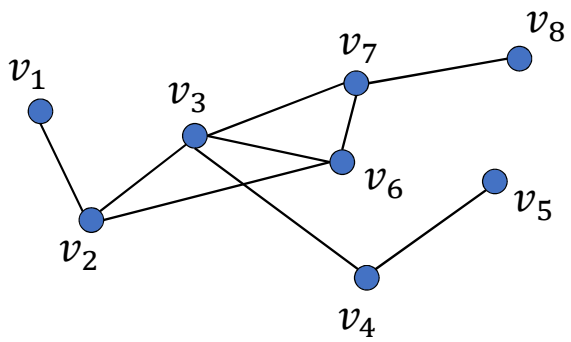
$$\mathcal{V} \longrightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

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

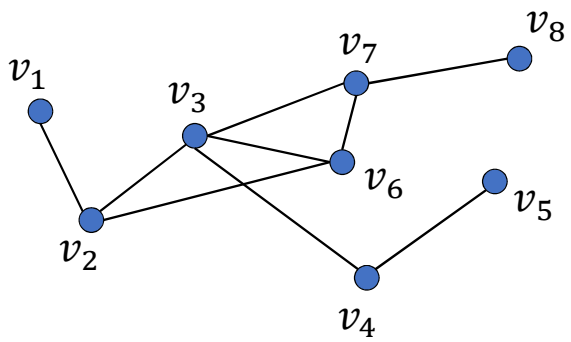
Adjacency Matrix

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

$\boldsymbol{A}$

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

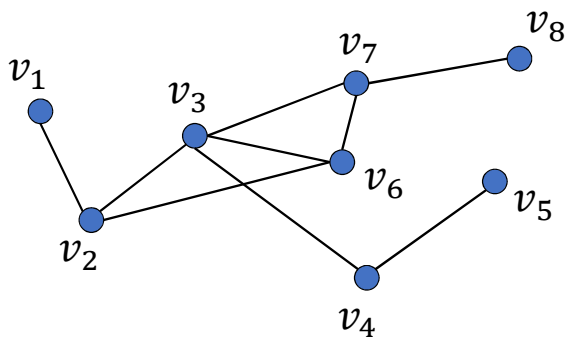
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix}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

$\boldsymbol{D}$ $\qquad$ $\boldsymbol{A}$

Adjacency Matrix

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:

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

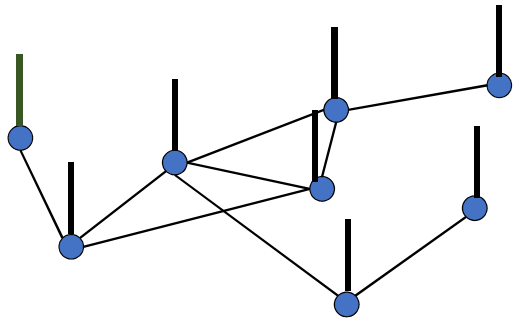
"Smoothness" or "Frequency" of the signal f

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

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

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



Low frequency graph signal

Laplacian quadratic form:

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

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

Laplacian Matrix as an Operator

Laplacian matrix is a difference operator:

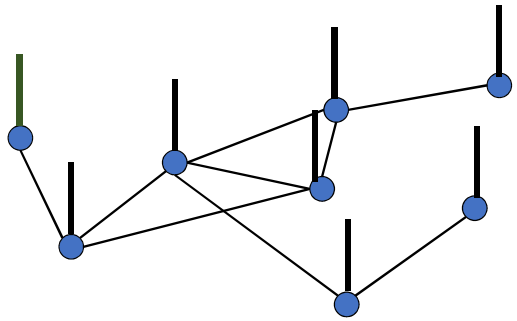
$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

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

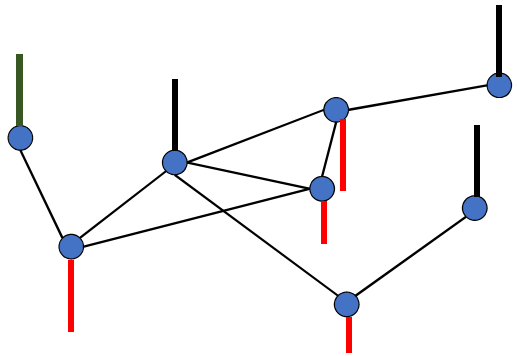
Laplacian quadratic form:

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

"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:

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

Eigen-decomposition of Laplacian Matrix

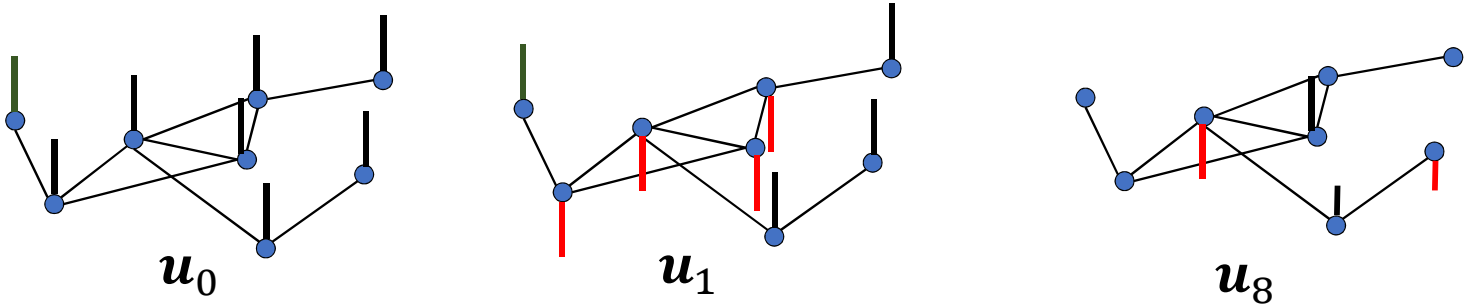
Laplacian matrix has a complete set of orthonormal eigenvectors:

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

Eigenvalues are sorted non-decreasingly:

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

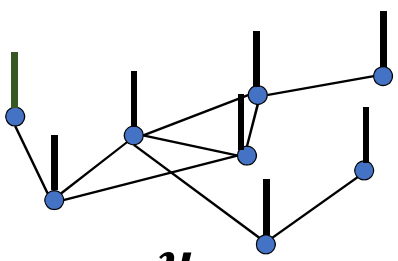
Eigenvectors as Graph Signals



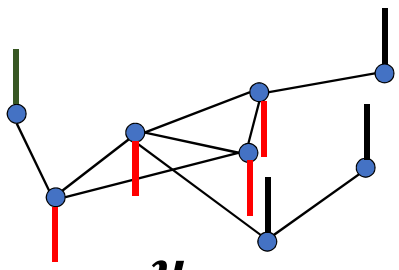
Eigenvectors as Graph Signals

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

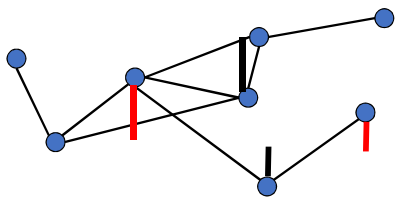
$$\underbrace{\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i}_{\text{Frequency of the signal } \boldsymbol{u}_i} = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i$$



$\boldsymbol{u}_0$



$\boldsymbol{u}_1$



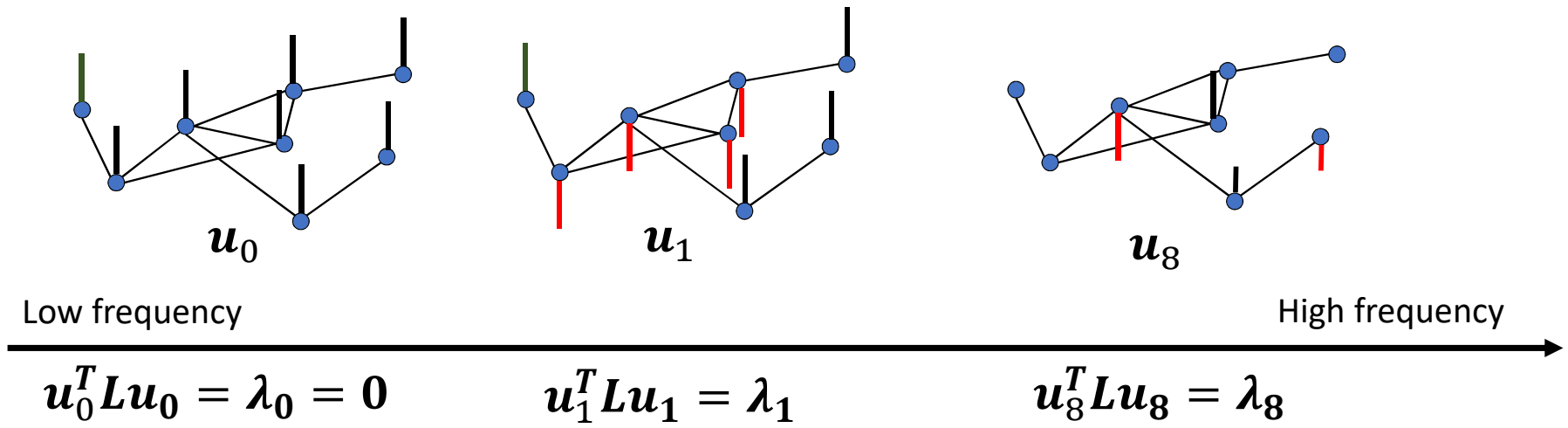
$\boldsymbol{u}_8$

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

Eigenvectors as Graph Signals

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

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



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:

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

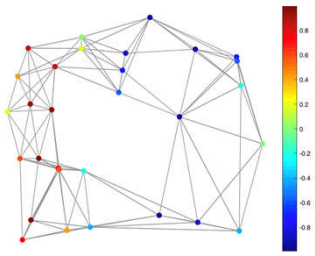
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients

Graph Fourier Transform (GFT)

A signal f can be written as graph Fourier series:

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

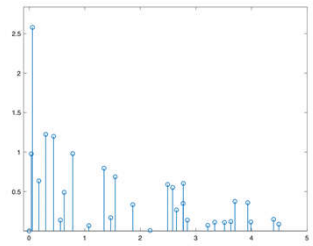
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients



Spatial domain: $\boldsymbol{f}$

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

Decompose signal f



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

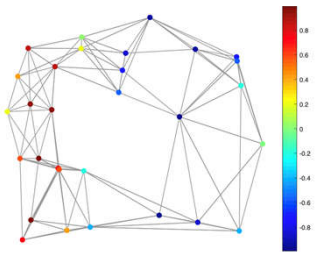
The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*

Inverse Graph Fourier Transform (IGFT)

A signal f can be written as graph Fourier series:

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

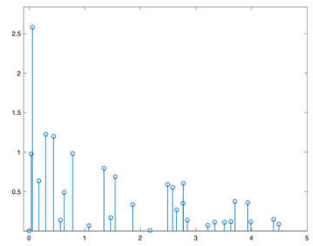
u_i: graph Fourier mode λ_i: frequency $\hat{f}_i$: graph Fourier coefficients



Spatial domain: $\boldsymbol{f}$

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

Reconstruct signal $\boldsymbol{f}$



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

Tutorial Overview

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

→

Models

- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

→

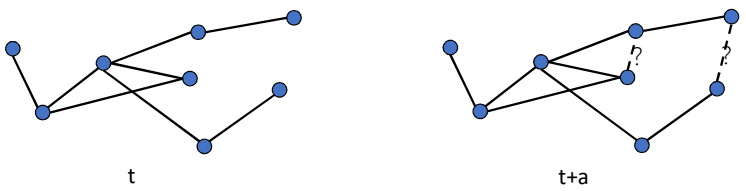
Applications

- Healthcare
- Natural Language Processing
- Recommendations

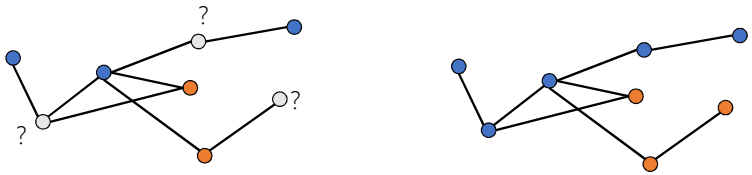
Tasks on Graph-Structured Data

Node-level

Link Prediction

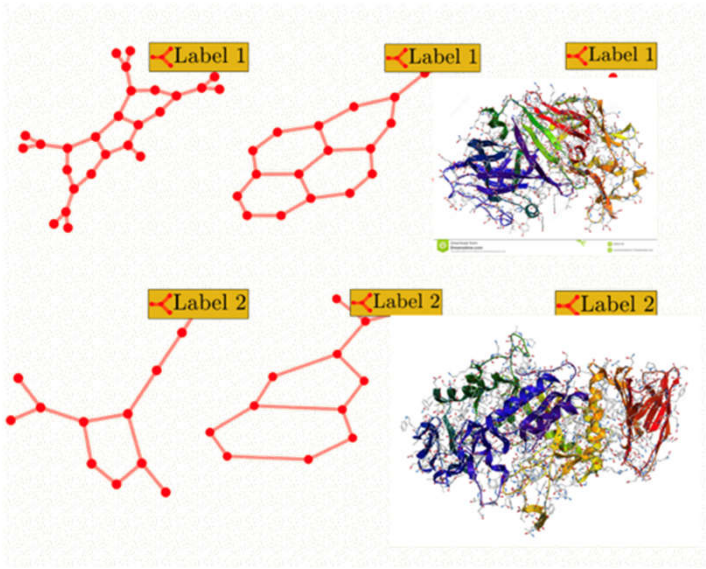


Node Classification



Graph-level

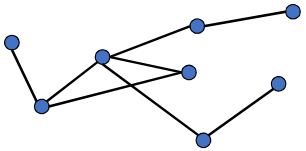
Graph Classification



Tasks on Graph-Structured Data

Node-level

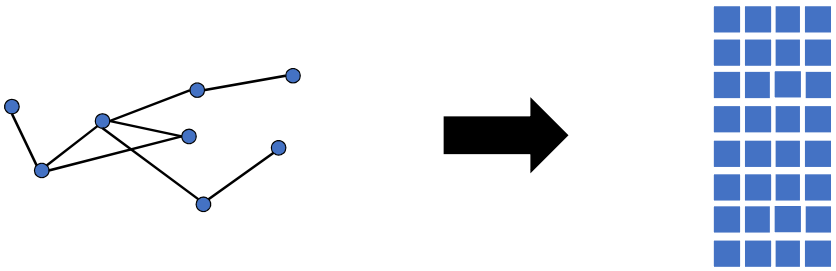
Graph-level



Tasks on Graph-Structured Data

Node-level

Graph-level

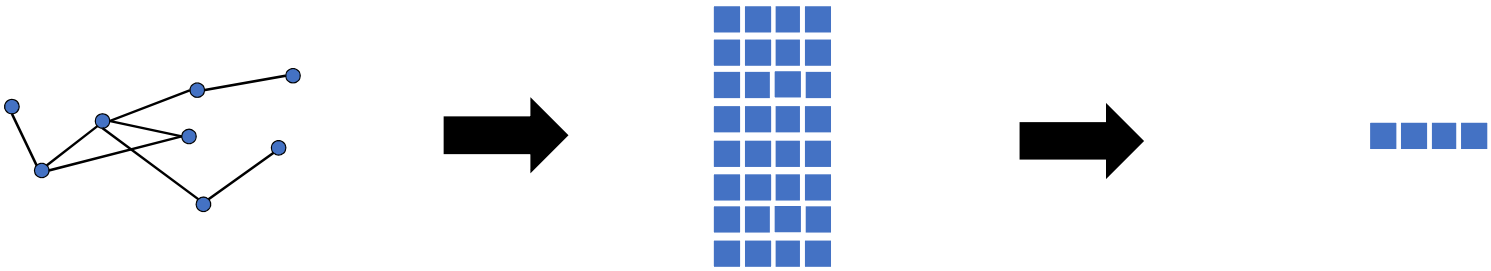


Node Representations

Tasks on Graph-Structured Data

Node-level

Graph-level



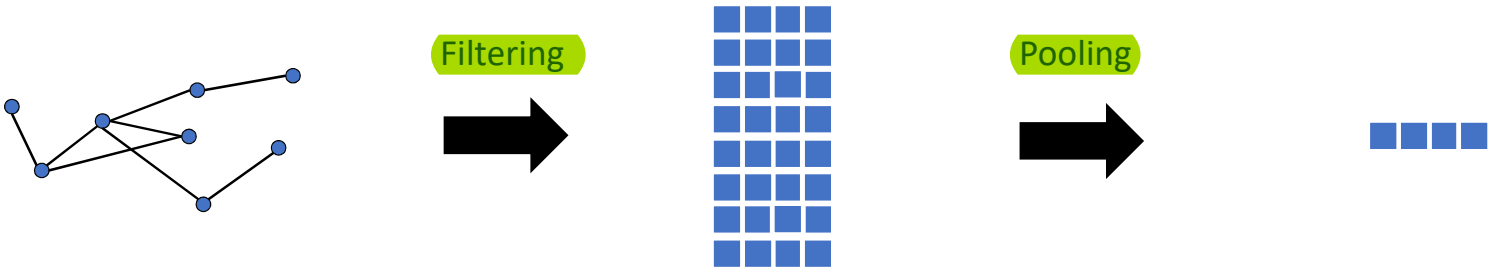
Node Representations

Graph Representation

Tasks on Graph-Structured Data

Node-level

Graph-level

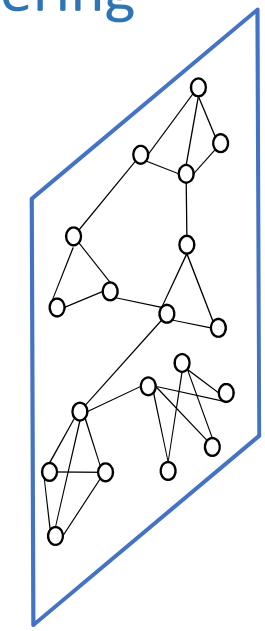


Node Representations

Graph Representations

Two Main Operations in GNN

Graph Filtering

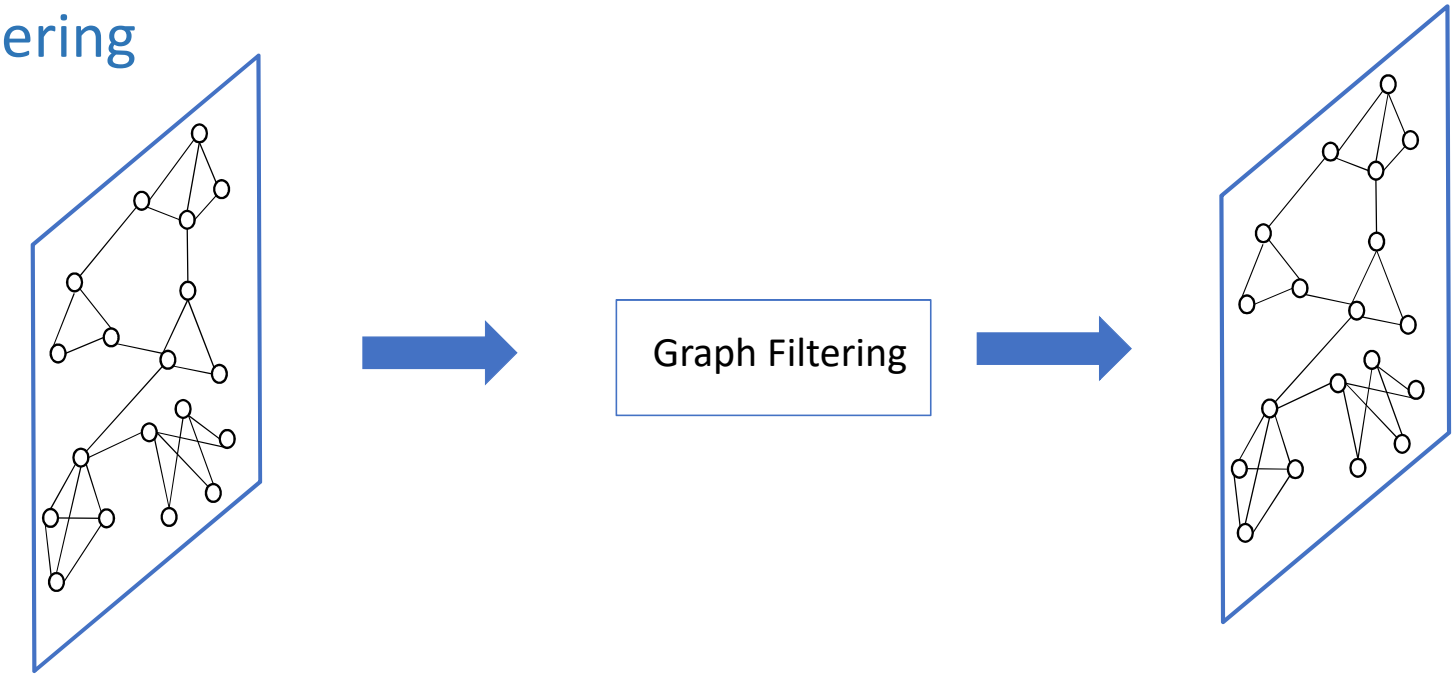


Graph Filtering

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

Two Main Operations in GNN

Graph Filtering

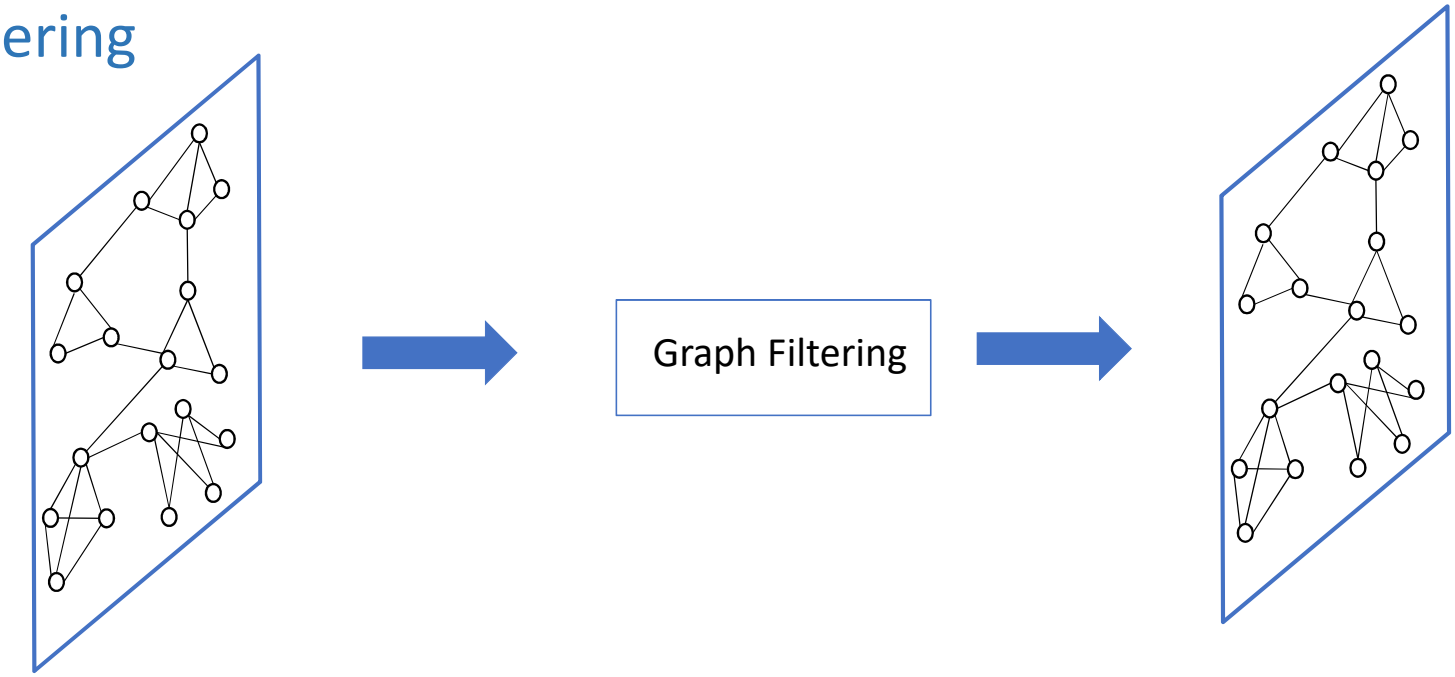


$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Two Main Operations in GNN

Graph Filtering



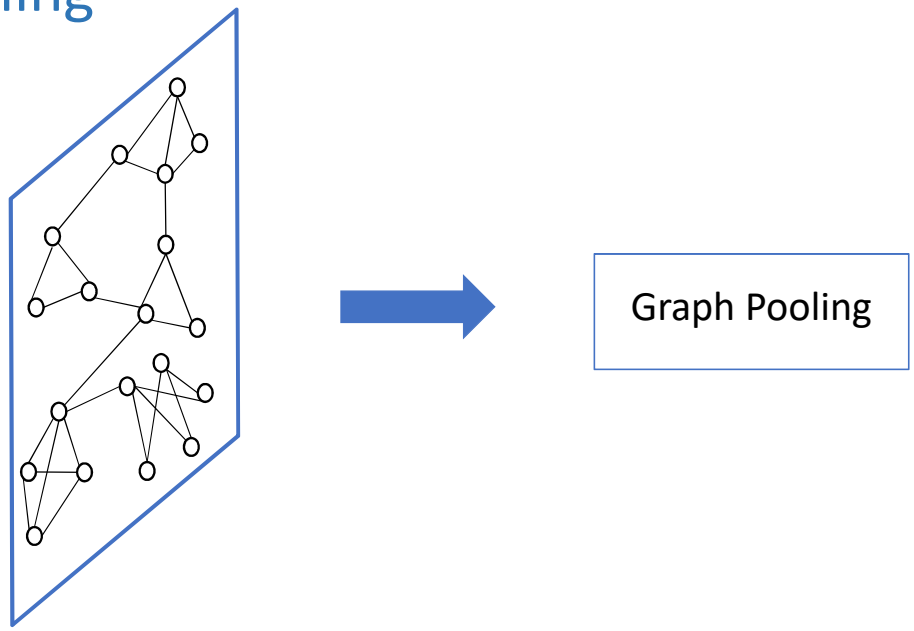
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Graph filtering refines the node features

Two Main Operations in GNN

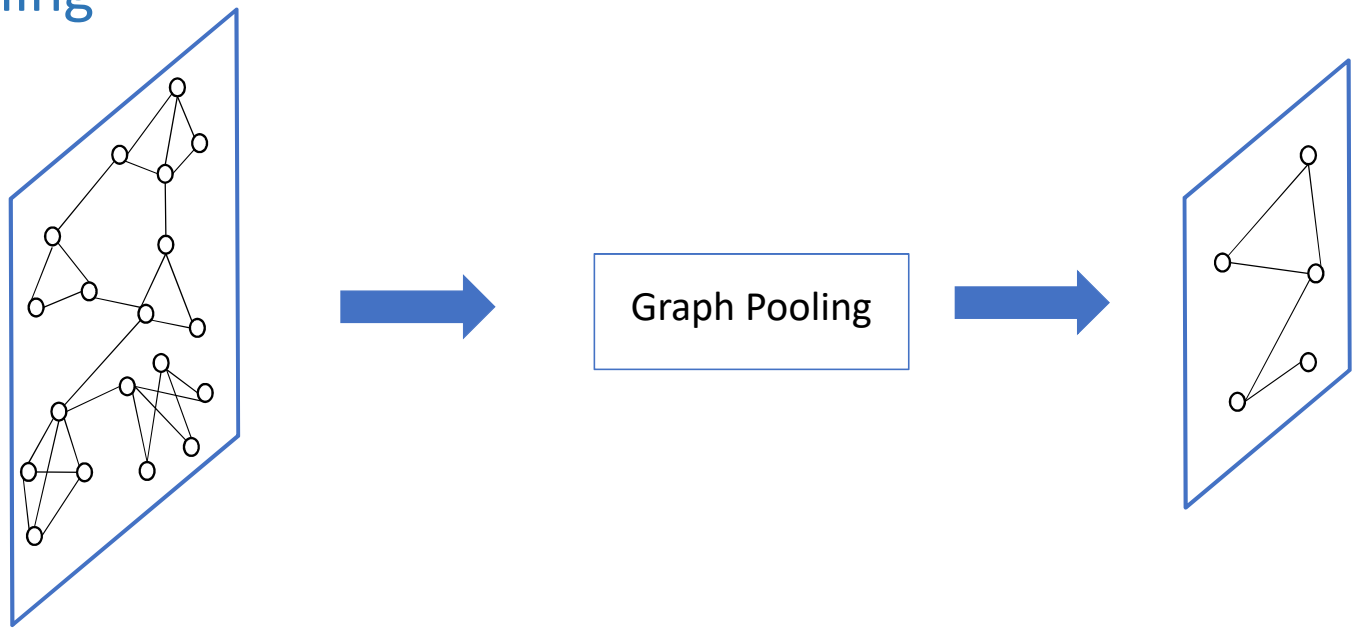
Graph Pooling



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

Two Main Operations in GNN

Graph Pooling

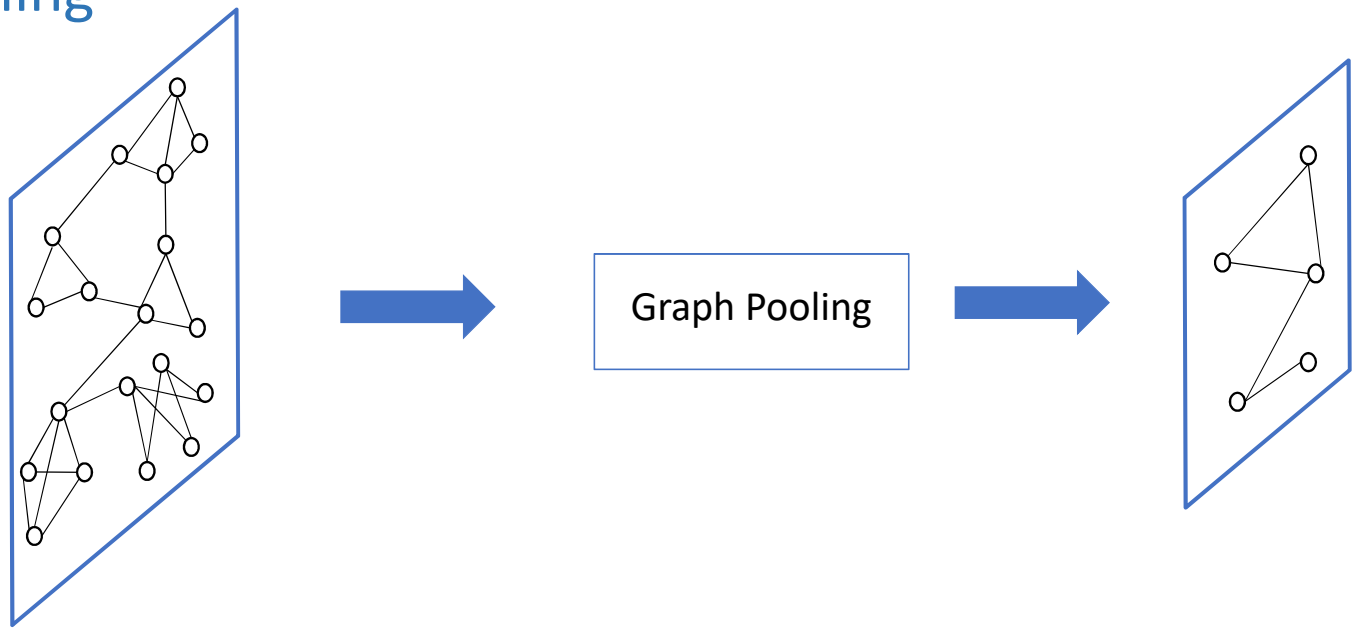


$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Two Main Operations in GNN

Graph Pooling



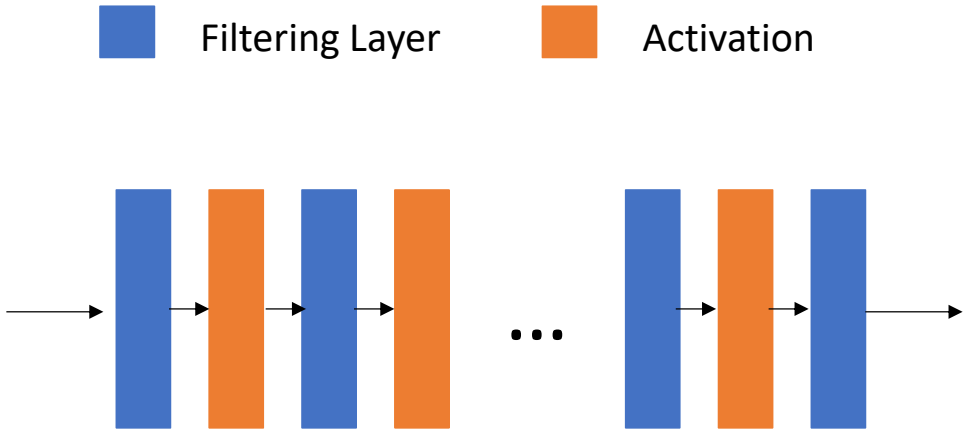
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Graph pooling generates a smaller graph

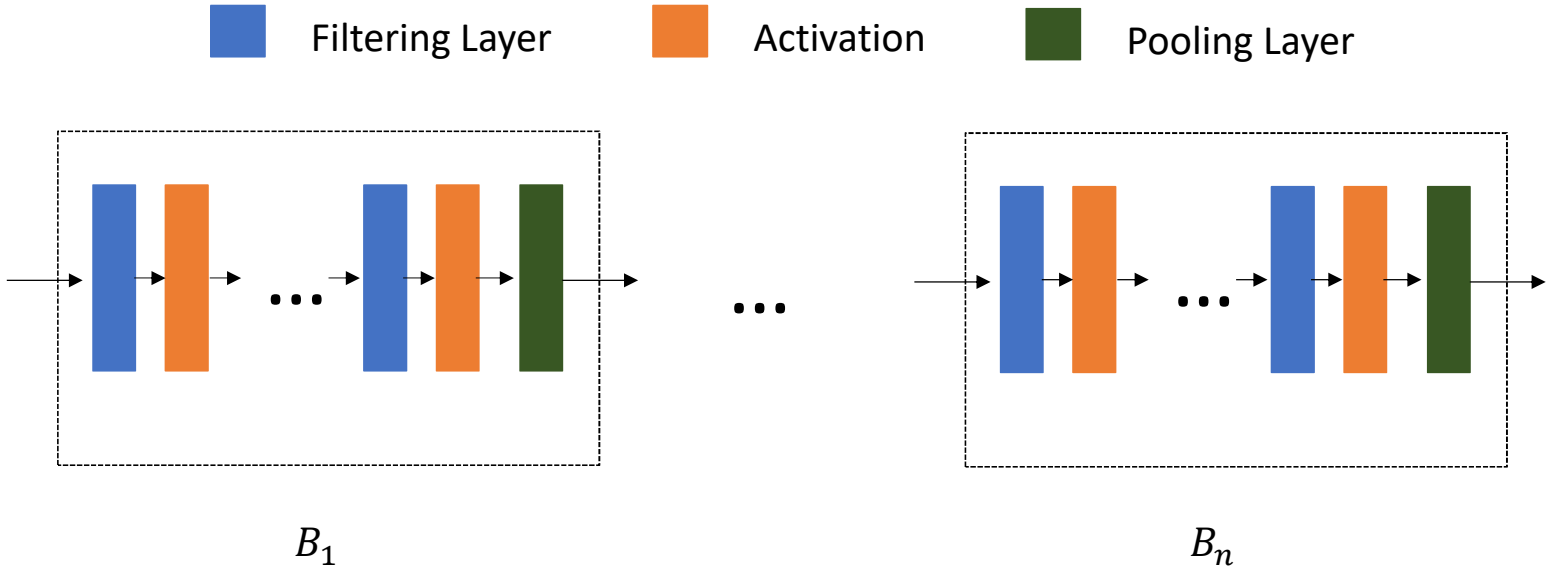
General GNN Framework

For node-level tasks

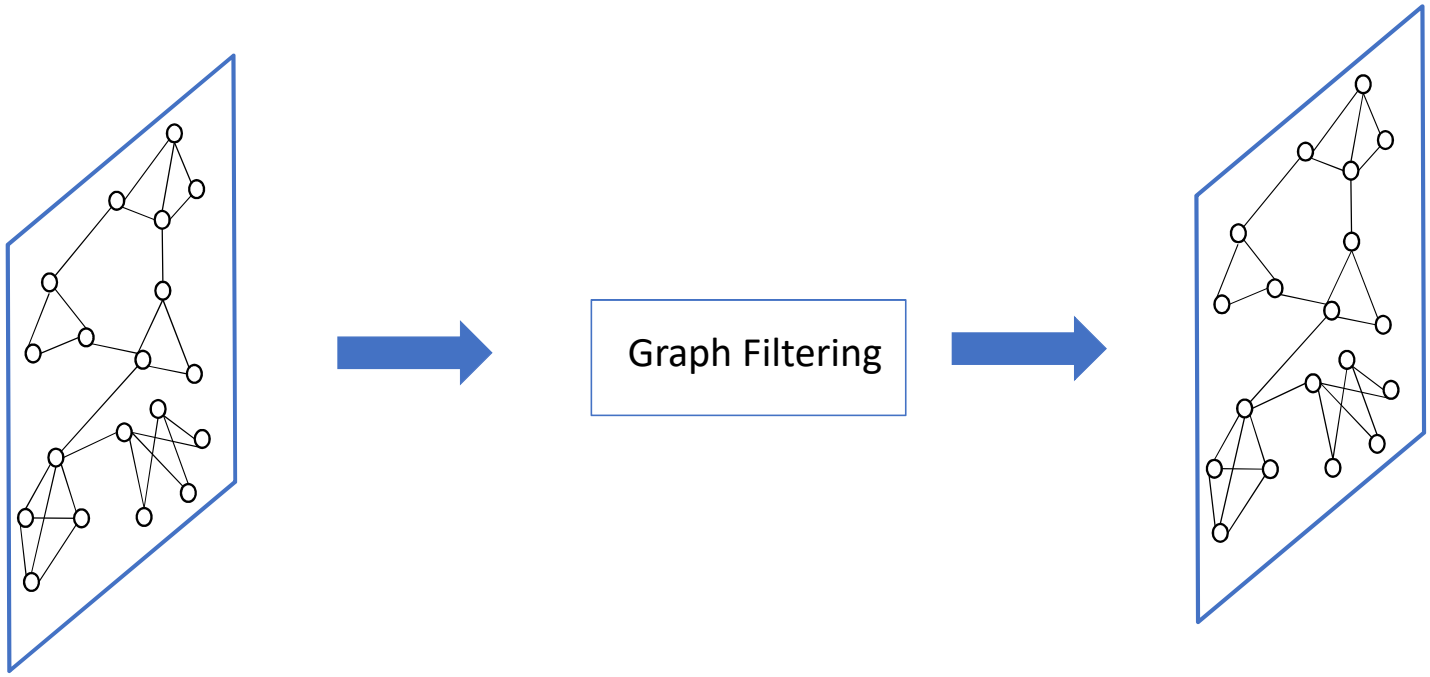


General GNN Framework

For graph-level tasks



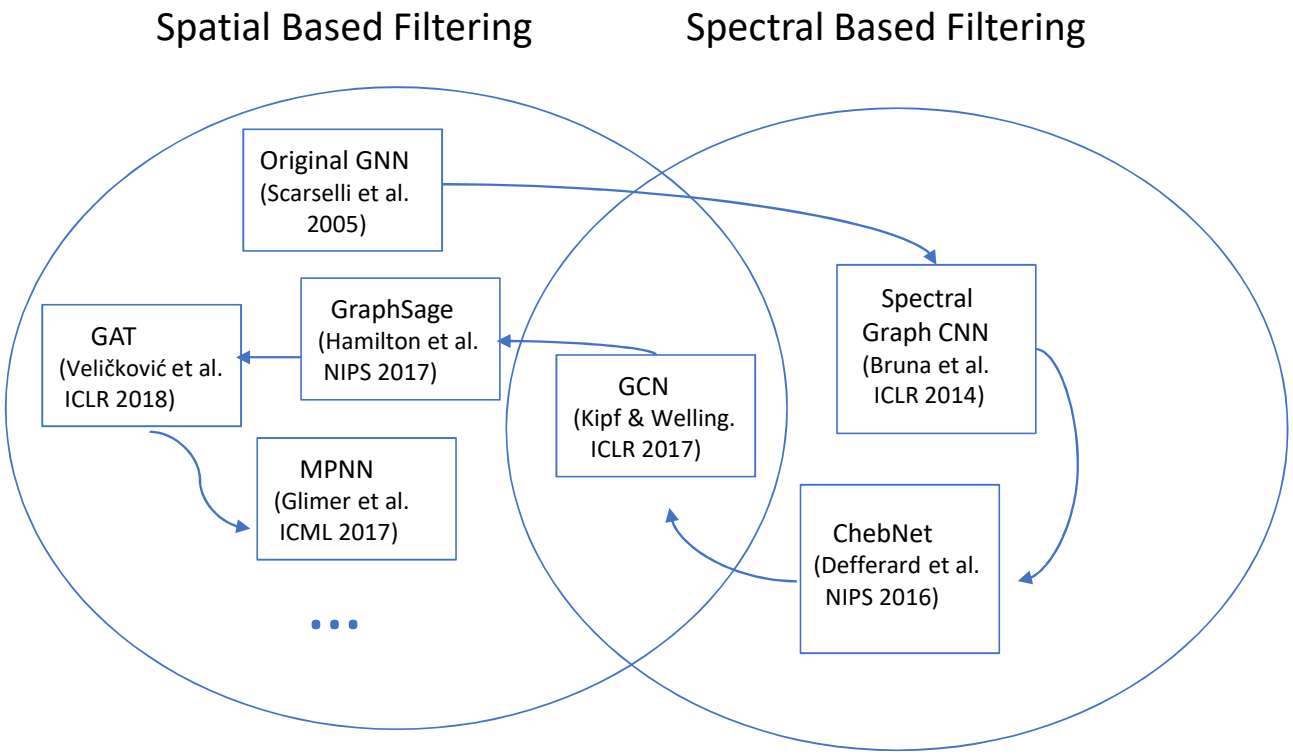
Graph Filtering Operation



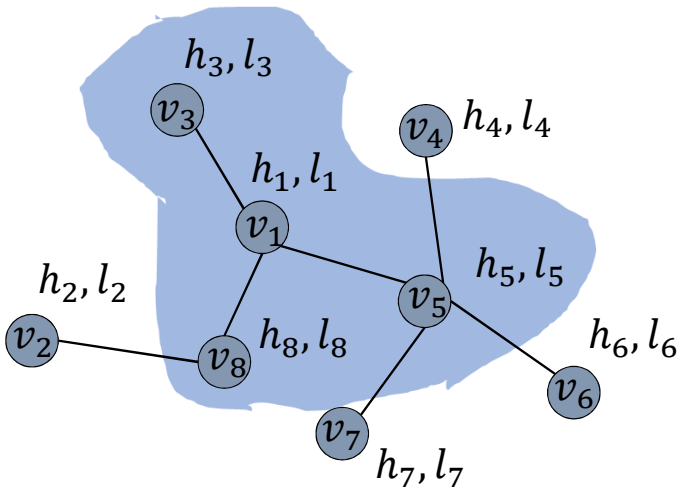
$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

Two Types of Graph Filtering Operation



Graph Filtering in the First GNN Paper

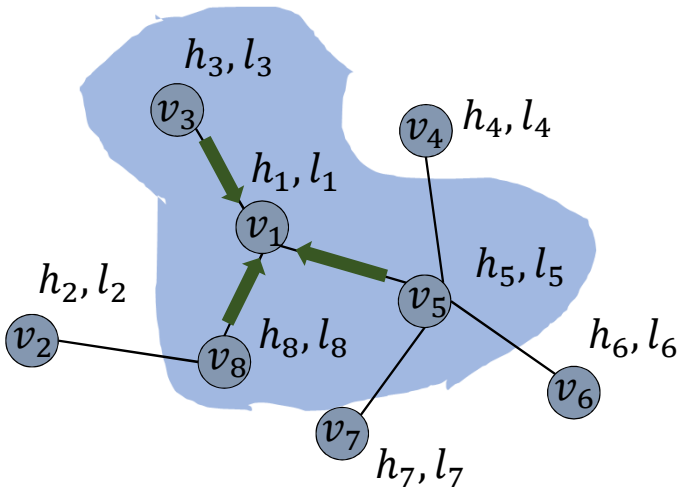


h_i: The hidden features

l_i: The input features

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

Graph Filtering in the First GNN Paper



h_i: The hidden features

l_i: The input features

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

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

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

Graph Spectral Filtering for Graph Signal

Recall:

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

Filter a graph signal f:

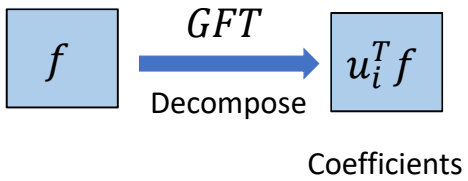
f

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:

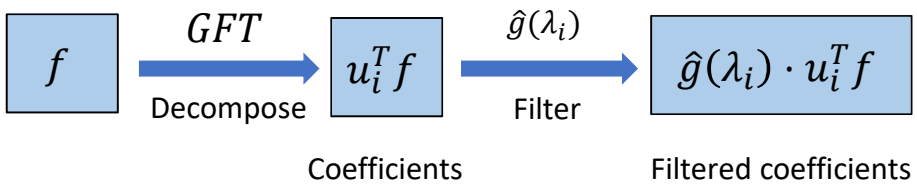


Graph Spectral Filtering for Graph Signal

Recall:

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

Filter a graph signal f:



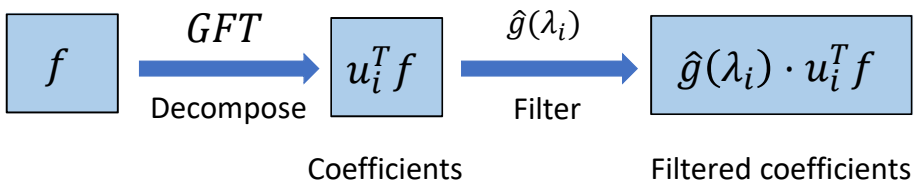
Filter $\hat{g}(\lambda_i)$: Modulating the frequency

Graph Spectral Filtering for Graph Signal

Recall:

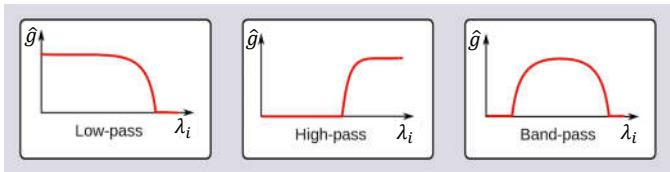
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



Example:

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



Graph Spectral Filtering for Graph Signal

Recall:

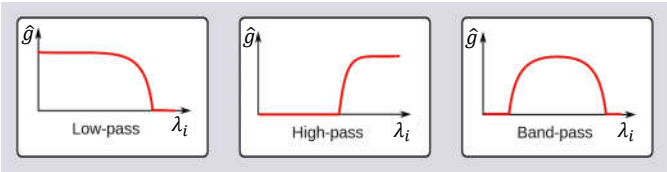
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:

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

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

Example:

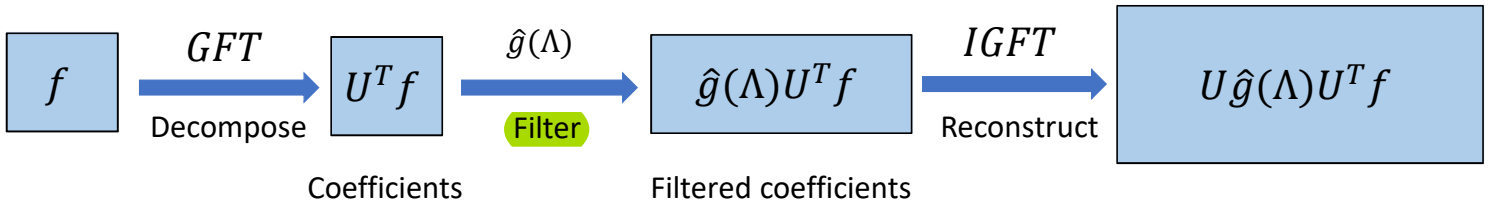


Graph Spectral Filtering for Graph Signal

Recall:

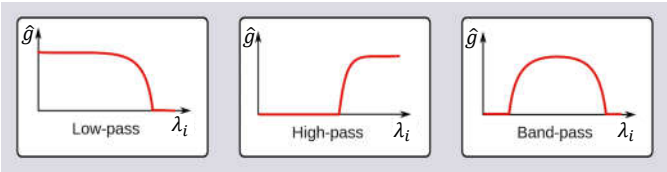
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

Filter a graph signal f:



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

Example:

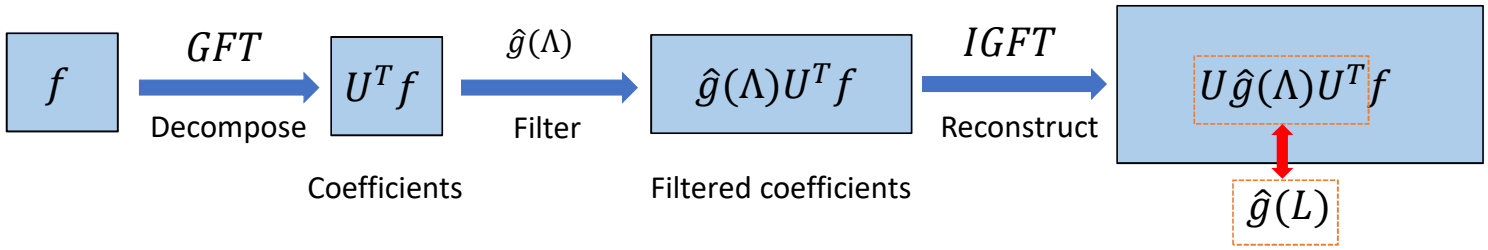


Graph Spectral Filtering for Graph Signal

Recall:

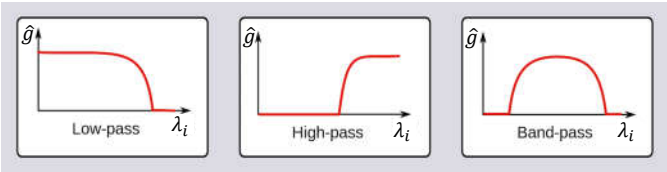
$$GFT: \hat{f} = U^T f \qquad\qquad IGFT: f = U\hat{f}$$

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:

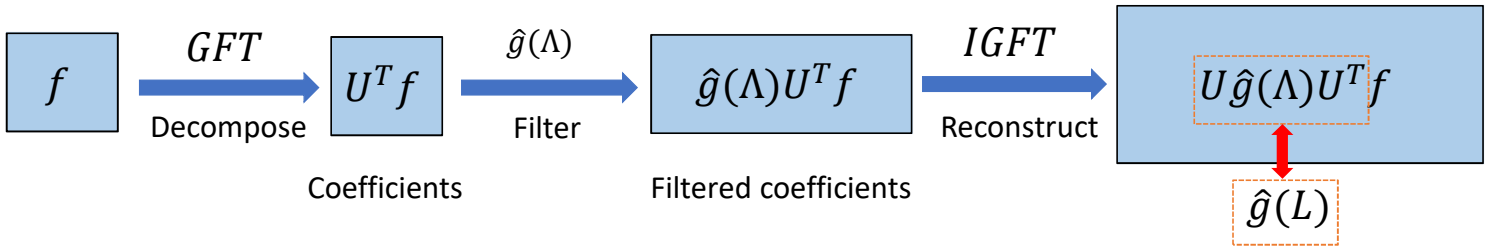


Graph Spectral Filtering for Graph Signal

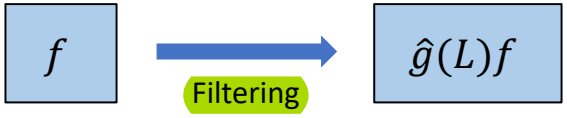
Recall:

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

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!

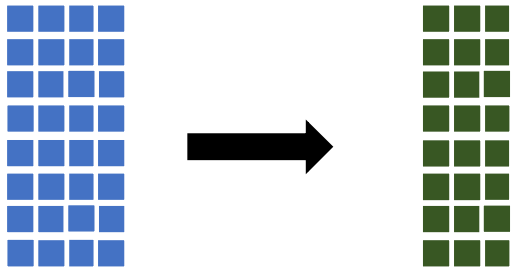
Graph Spectral Filtering for GNN

How to design the filter?

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

How to deal with multi-channel signals?

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



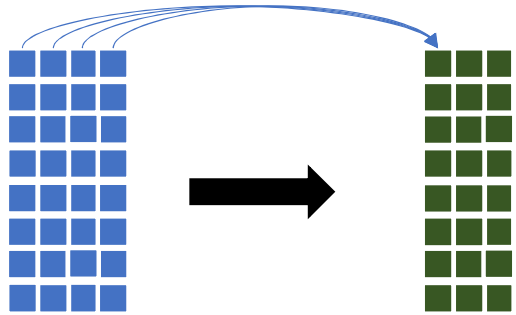
Graph Spectral Filtering for GNN

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

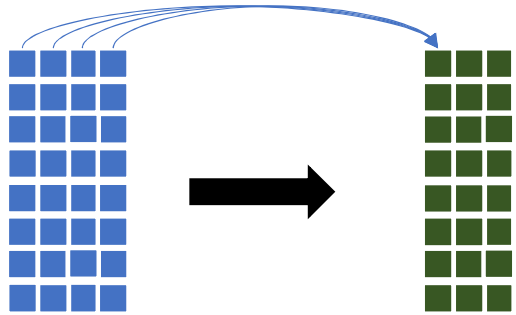
Graph Spectral Filtering for GNN

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

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

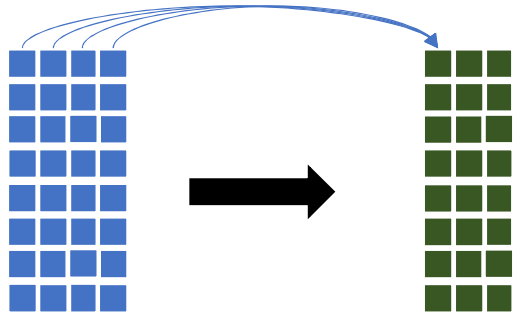
Graph Spectral Filtering for GNN

How to design the filter?

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

How to deal with multi-channel signals?

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



Each input channel contributes to each output channel

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

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

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

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 $\theta = \theta_0 = -\theta_1$

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

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

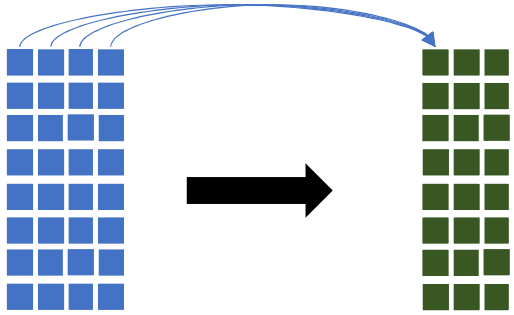
Apply a renormalization trick

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

GCN for Multi-channel Signal

Recall:

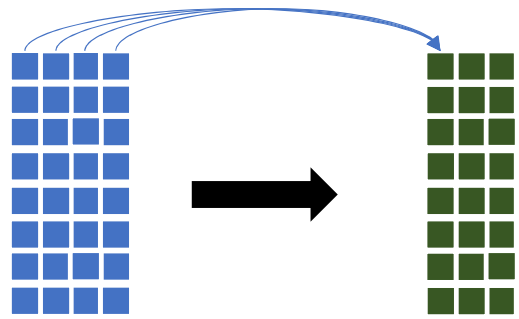
$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{in}[:,j]}_{\text{Filter each input channel}} \quad i = 1, \dots d_2$$



GCN for Multi-channel Signal

Recall:

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



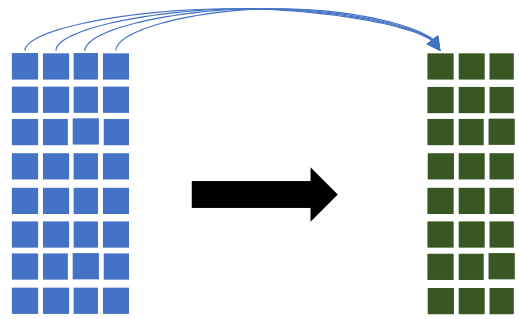
For GCN:

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

GCN for Multi-channel Signal

Recall:

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



For GCN:

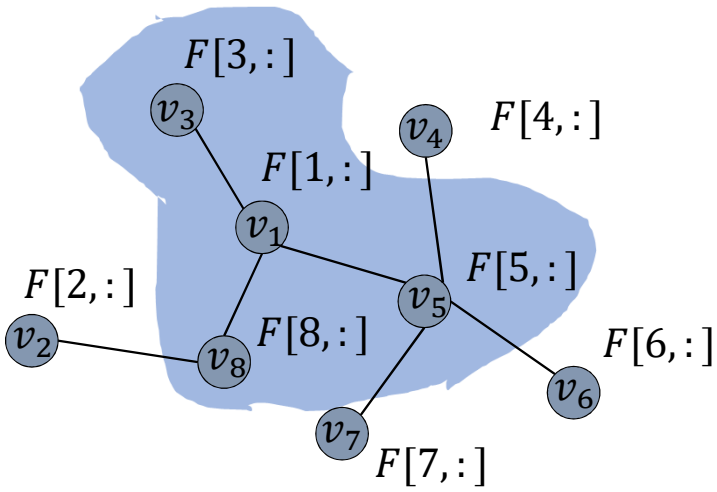
$$\boldsymbol{F}_{out}[:,i] = \sum_{j=1}^{d_1} \underbrace{\theta_{ji}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})}_{\text{GCN filter}}\boldsymbol{F}_{in}[:,j] \quad i = 1, \dots d_2$$

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

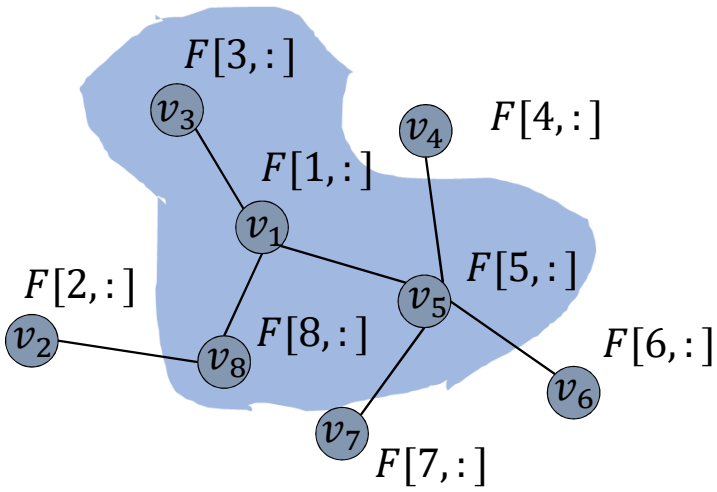
Denote $C = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$



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



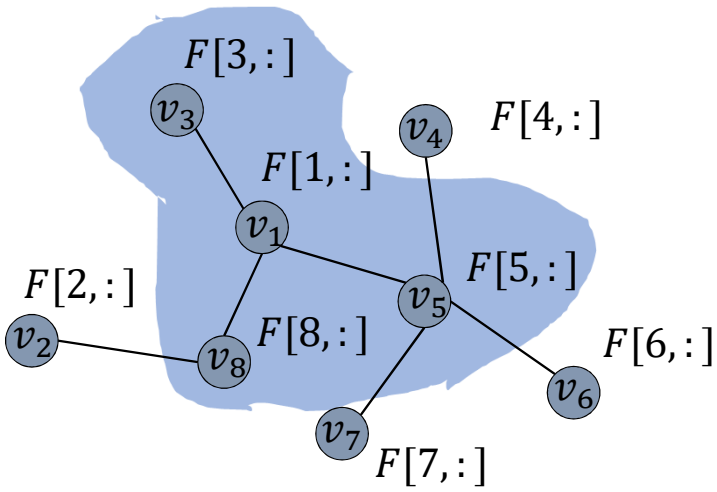
A Spatial View of GCN Filter

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

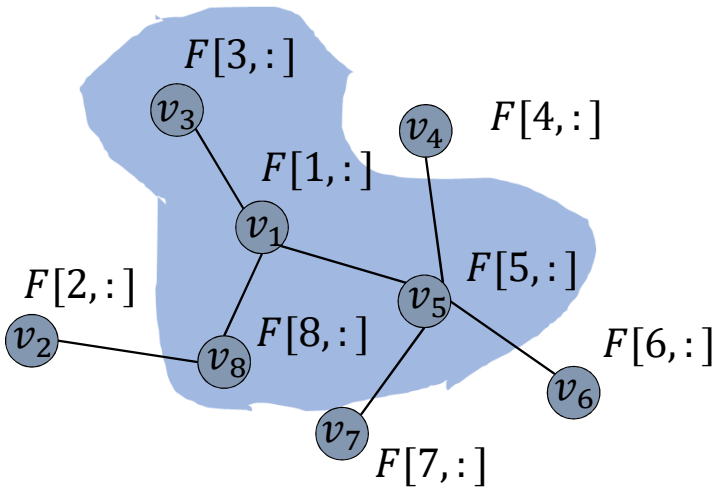
- Then $F_{out} = CF_{in}\Theta$
- For node v_i, $F_{out}[i,:] = \sum_j C[i,j]F_{in}[j,:]\Theta$

Observe that:

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

Hence,

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



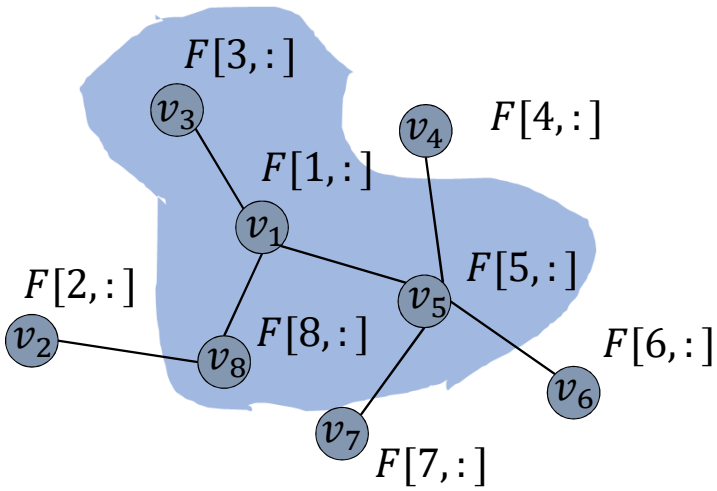
A Spatial View of GCN Filter

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

- 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,

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

Feature transformation

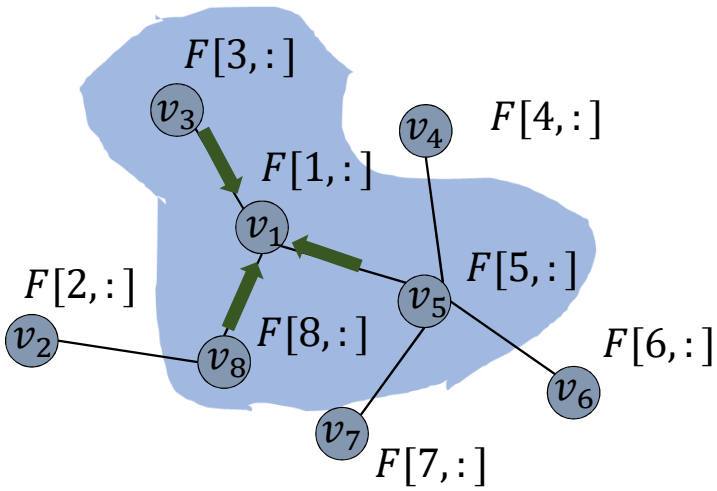
A Spatial View of GCN Filter

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

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

Observe that:

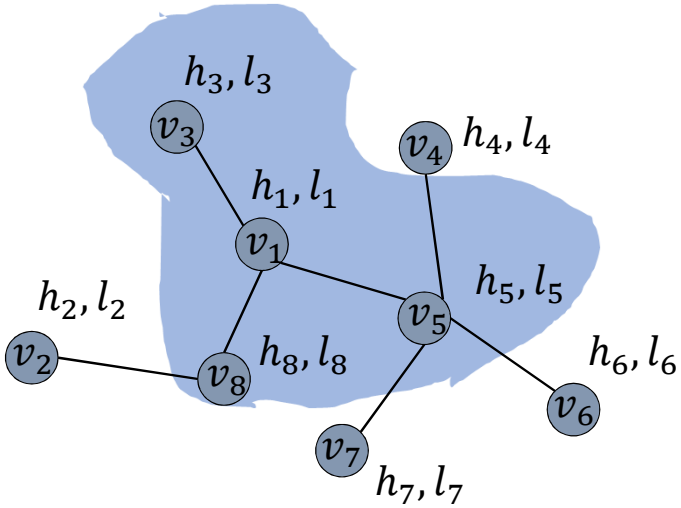
- $C[i,j] = 0$ for $v_j \notin N(v_i) \cup \{v_i\}$



Hence,

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

Filter in GCN VS Filter in the First GNN



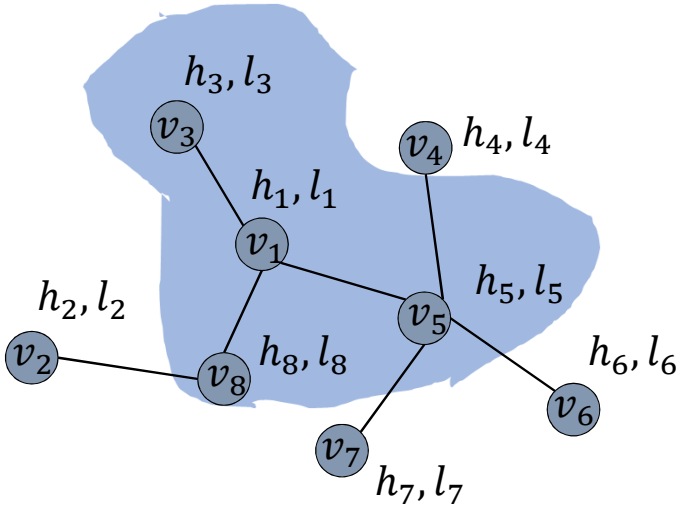
GCN: k-th layer

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

The first GNN: k-th layer

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

Filter in GCN VS Filter in the First GNN



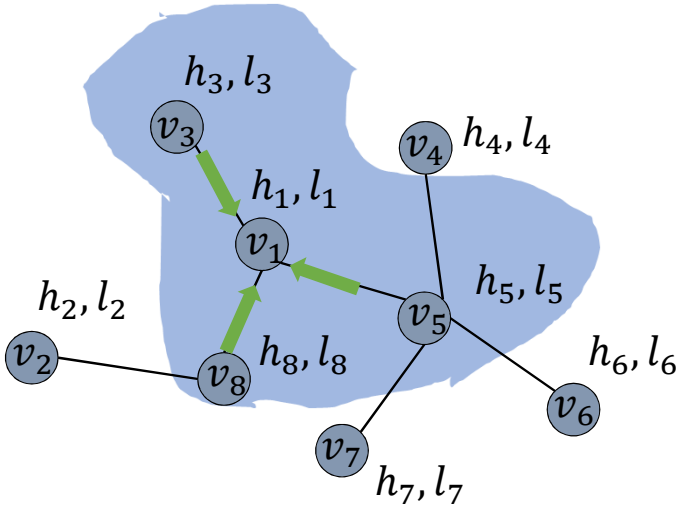
GCN: k-th layer

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

The first GNN: k-th layer

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

Filter in GCN VS Filter in the First GNN



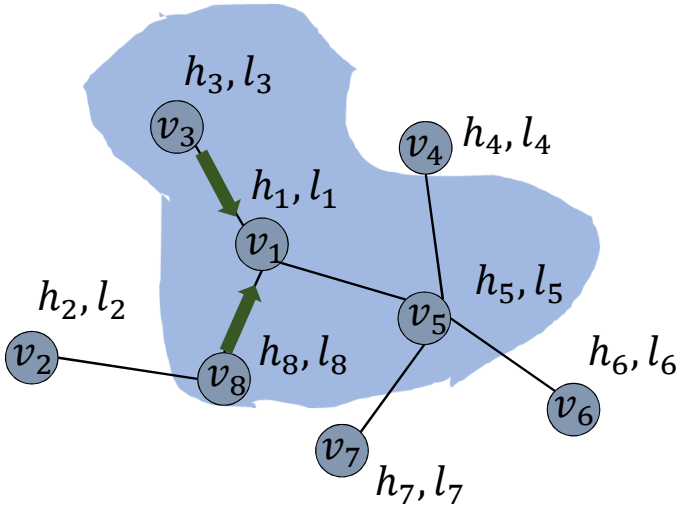
GCN: k-th layer

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

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

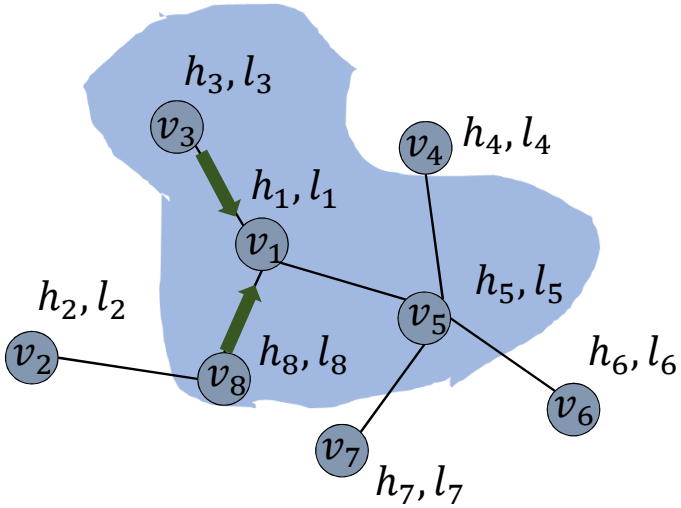


Neighbor Sampling

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

Inductive Representation Learning on Large Graphs. NIPS 2017.

Filter in GraphSage



Neighbor Sampling

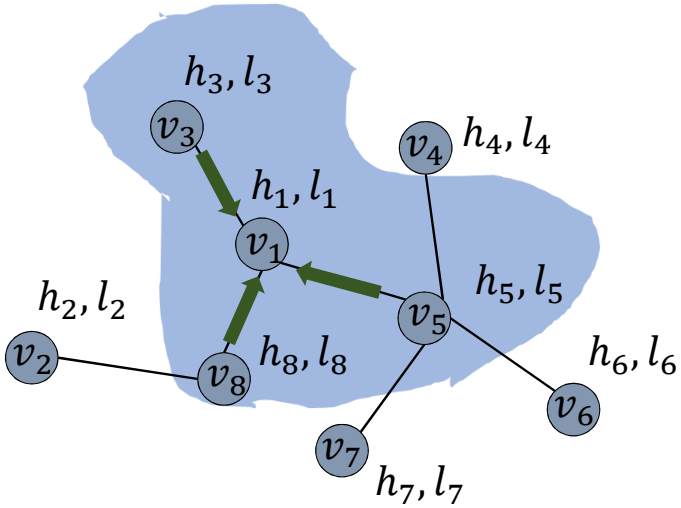
$$\mathcal{N}(v_i) \rightarrow \mathcal{N}_s(v_i)$$

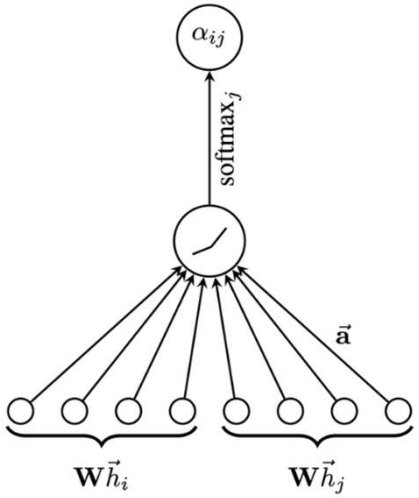
Aggregation

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

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

Filter in GAT

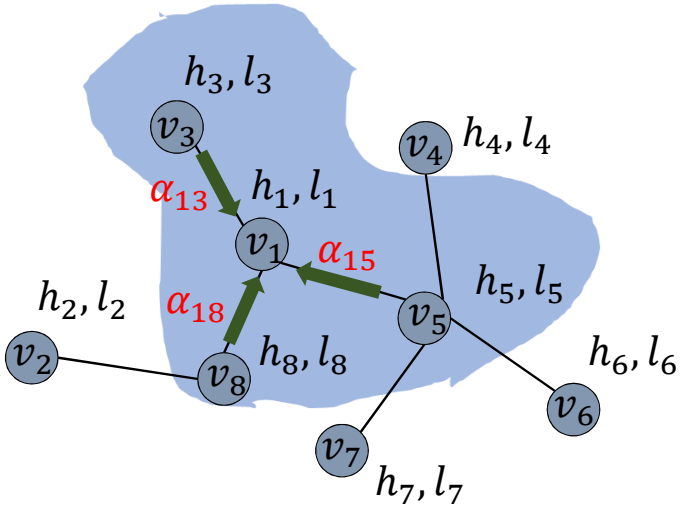


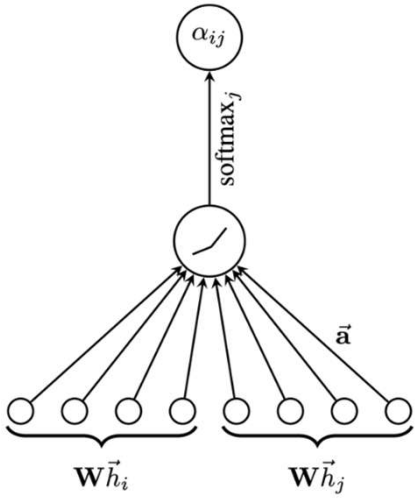


$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Graph Attention Networks. ICLR 2018.

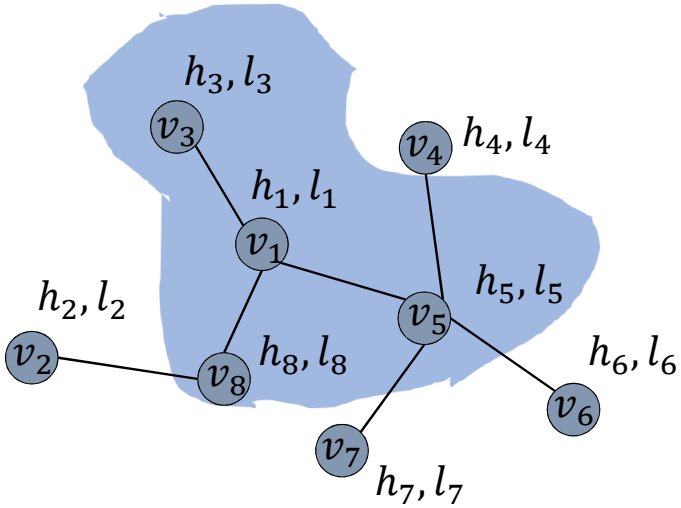
Filter in GAT





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

Filter in MPNN



Message Passing

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

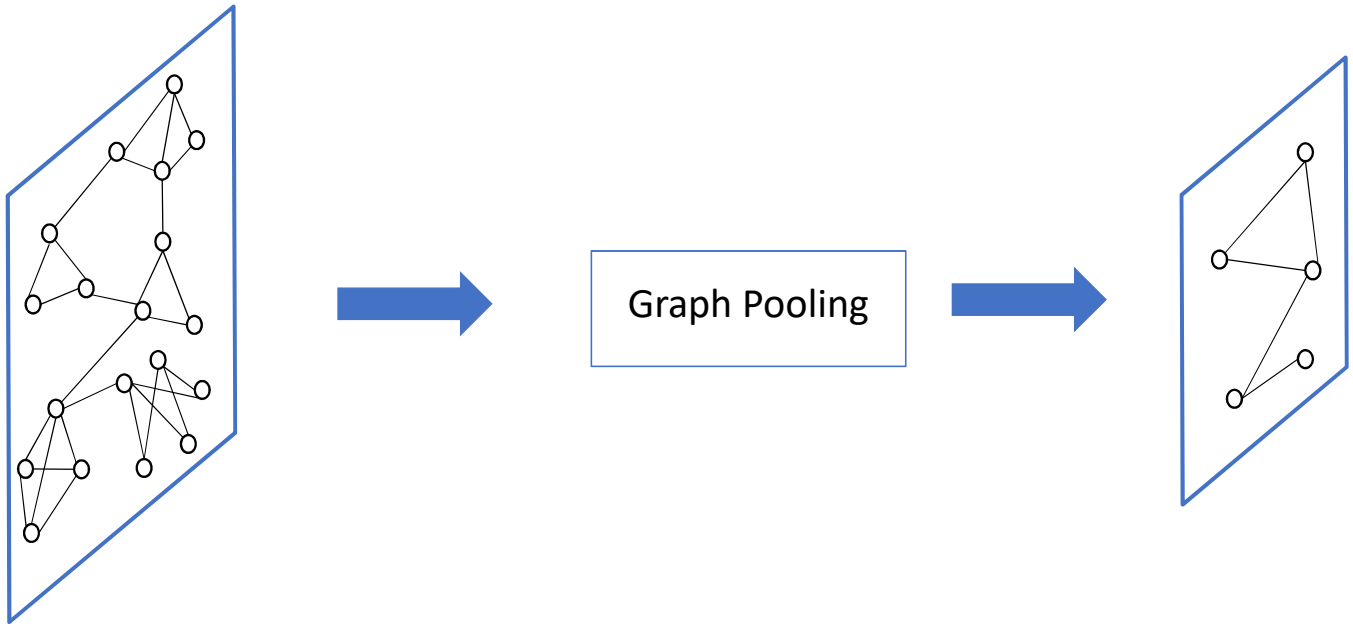
Feature Updating

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

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

Neural Message Passing for Quantum Chemistry. ICML 2017.

Graph Pooling Operation

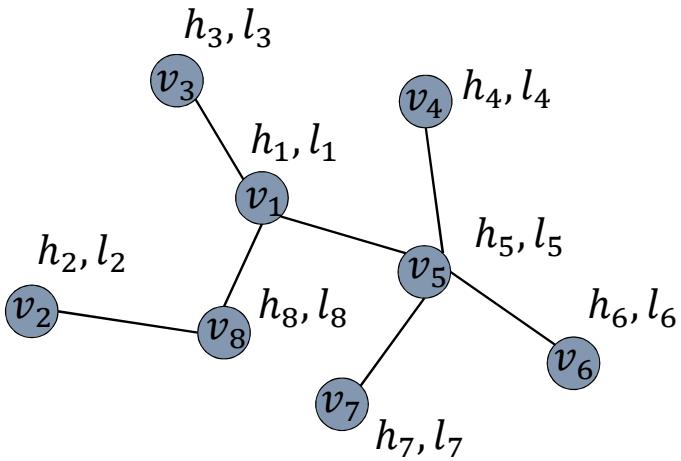


$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{X} \in \mathbb{R}^{n \times d}$

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

gPool

Downsample by selecting the most importance nodes



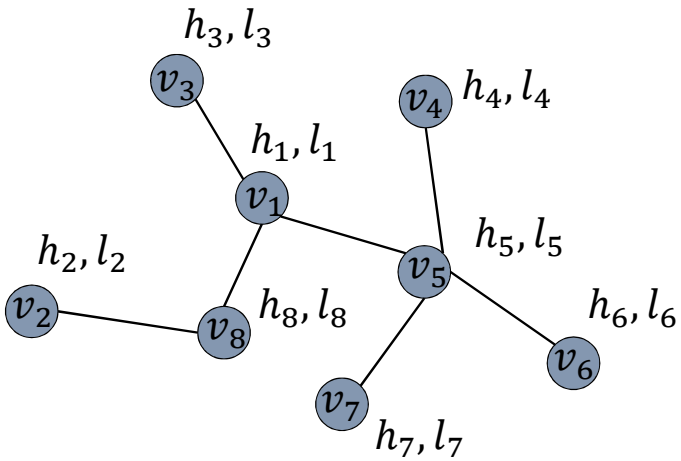
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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

Graph U-Nets. ICML 2019.

gPool

Downsample by selecting the most importance nodes



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

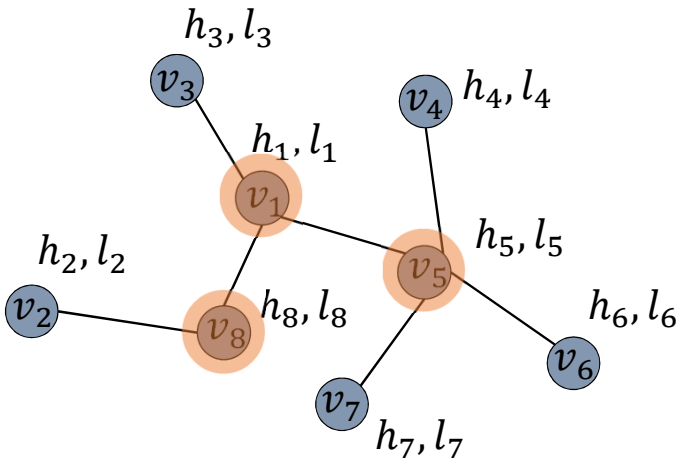
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

Importance Measure

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

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

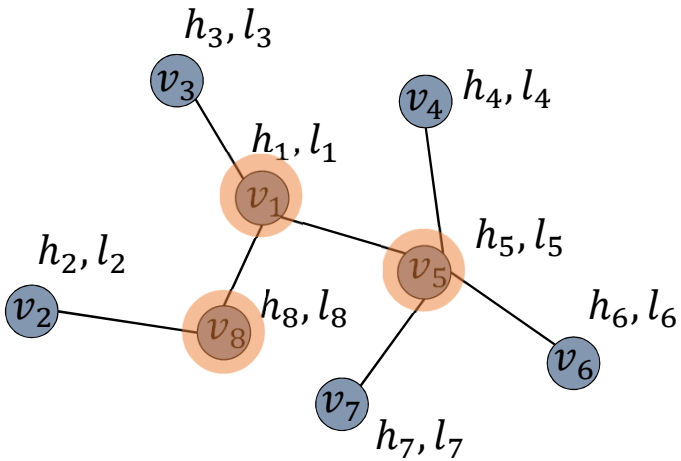
$$v_i \rightarrow y_i \qquad y_i = \frac{h_i^T p}{||p||}$$

Select top the n_p nodes

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

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

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

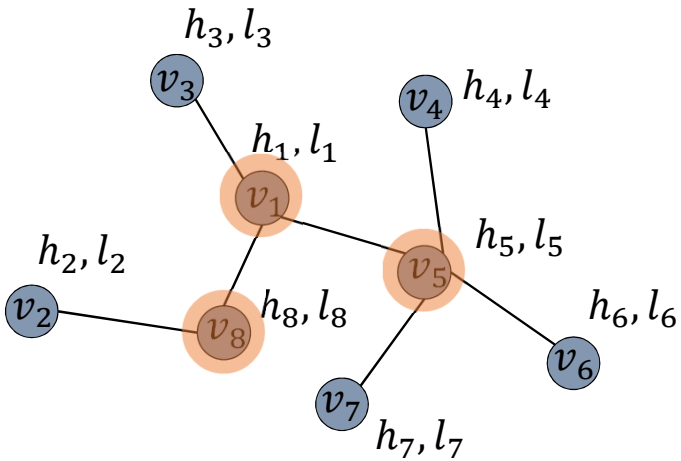
Generate A_p and intermediate H_{inter}

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

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

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}

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

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

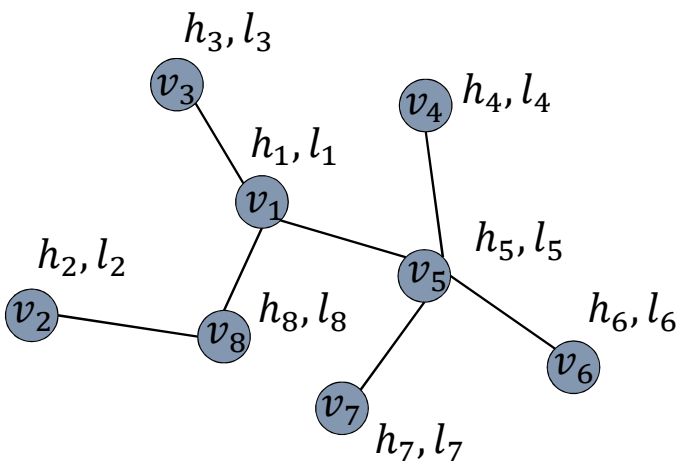
Generate H_p

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

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

DiffPool

Downsample by clustering the nodes using GNN



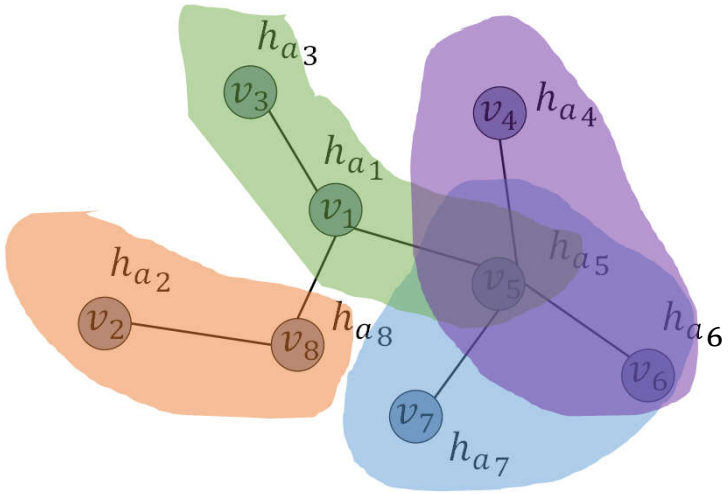
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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

Hierarchical Graph Representation Learning with Differentiable Pooling. NeurIPS 2018.

DiffPool

Downsample by clustering the nodes using GNN



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

$$\Downarrow$$

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

Filter1:

Generate a soft-assign matrix

2 filters

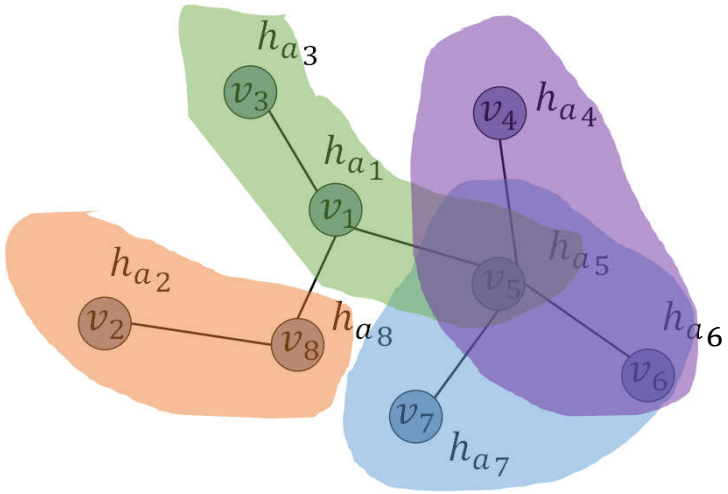
$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

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

DiffPool

Downsample by clustering the nodes using GNN



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

$$\Downarrow$$

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

2 filters

Filter1:

Generate a soft-assign matrix

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

$$\Downarrow$$

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

Filter2:

Generate new features

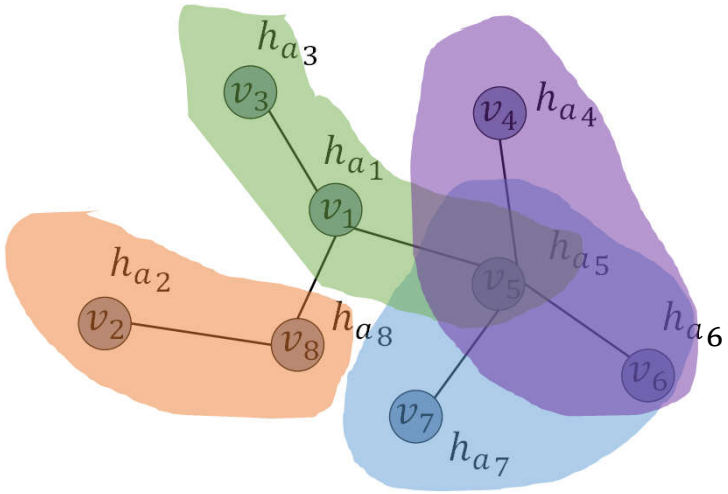
$$\mathbf{A} \in \{0,1\}^{n\times n}, \mathbf{H} \in \mathbb{R}^{n\times d}$$

$$\Downarrow$$

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

DiffPool

Downsample by clustering the nodes using GNN



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

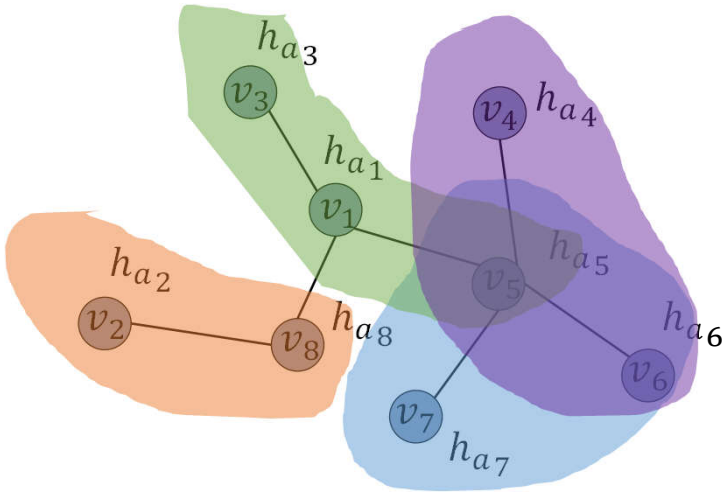
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$

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

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

DiffPool

Downsample by clustering the nodes using GNN



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

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

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

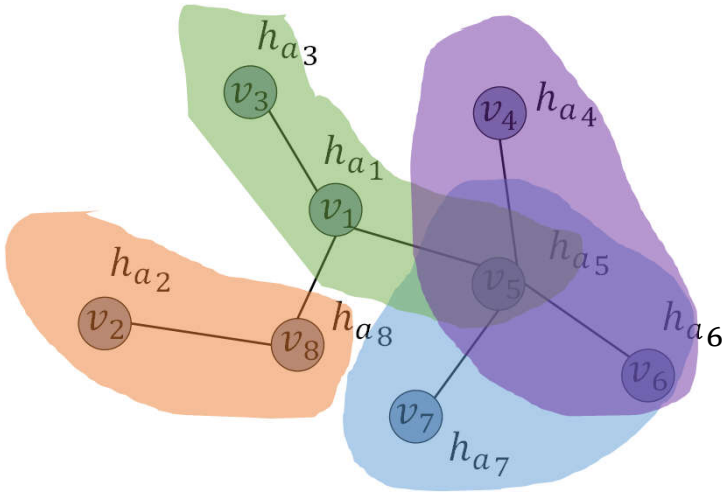
Generated new features $\mathbf{H}_f \in \mathbb{R}^{n\times d_{new}}$

Generate A_p

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

DiffPool

Downsample by clustering the nodes using GNN



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

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

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

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

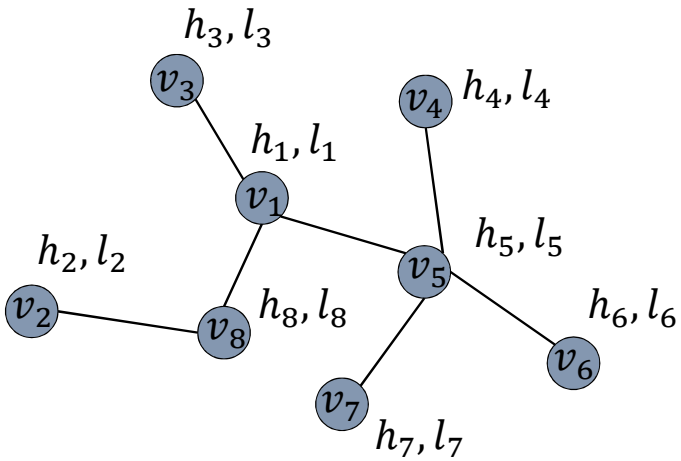
Generate A_p

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

Generate H_p

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

Eigenpooling

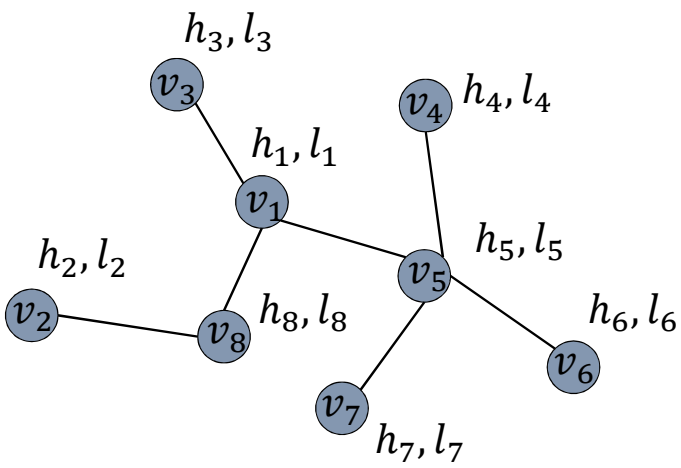


$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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

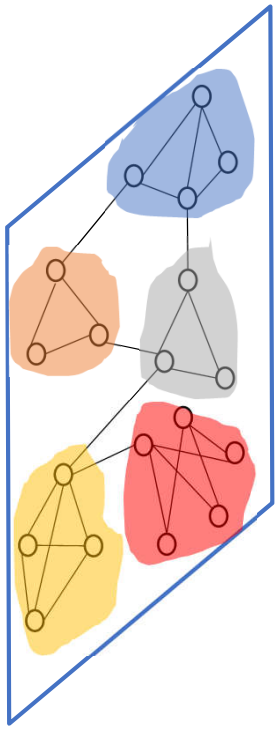
Graph Convolutional Networks with EigenPooling. KDD 2019.

Eigenpooling



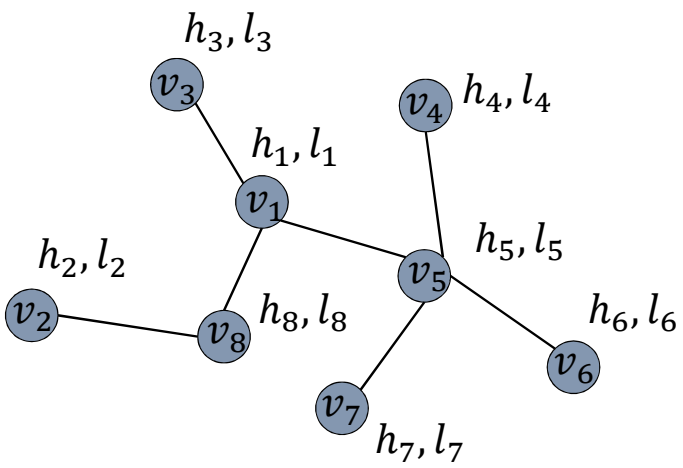
$$\mathbf{A} \in \{0,1\}^{n \times n}, \mathbf{H} \in \mathbb{R}^{n \times d}$$

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



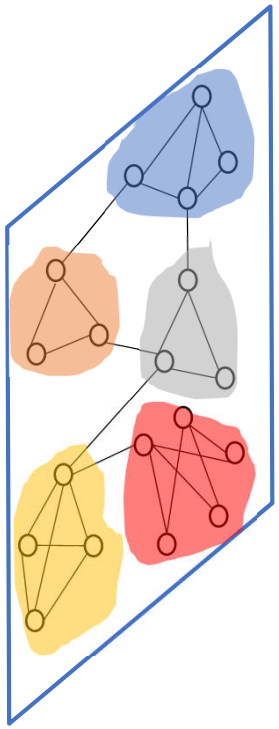
Learn A_p using clustering methods

Eigenpooling



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

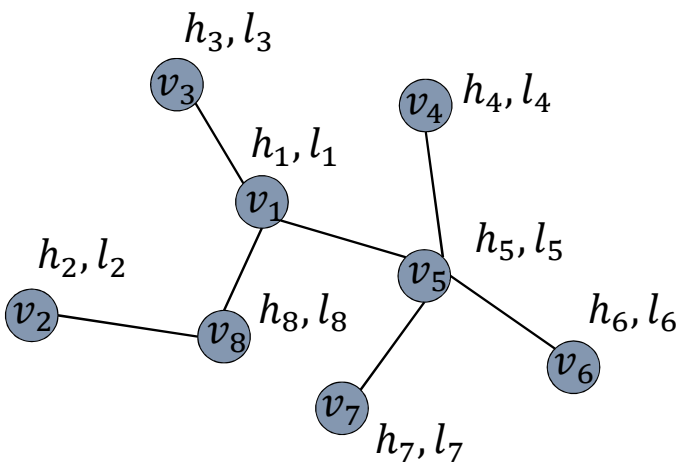
$$\mathbf{A}_p \in \{0,1\}^{n_p \times n_p}, \mathbf{H}_p \in \mathbb{R}^{n_p \times d_{new}}, n_p < n$$



Learn A_p using clustering methods

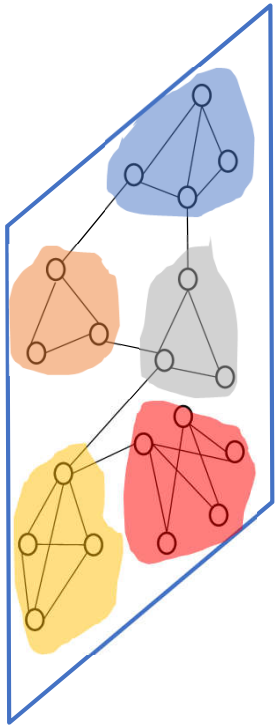
Focus on learning better H_p

Eigenpooling



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

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



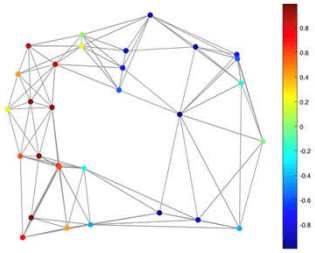
Learn A_p using clustering methods

Focus on learning better H_p

Capture both feature and graph structure

Going Back to Graph Spectral Theory

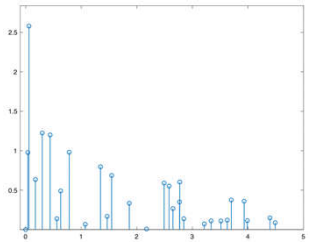
Recall:



Spatial domain: $\boldsymbol{f}$

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

Decompose signal f

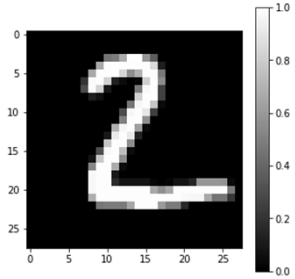


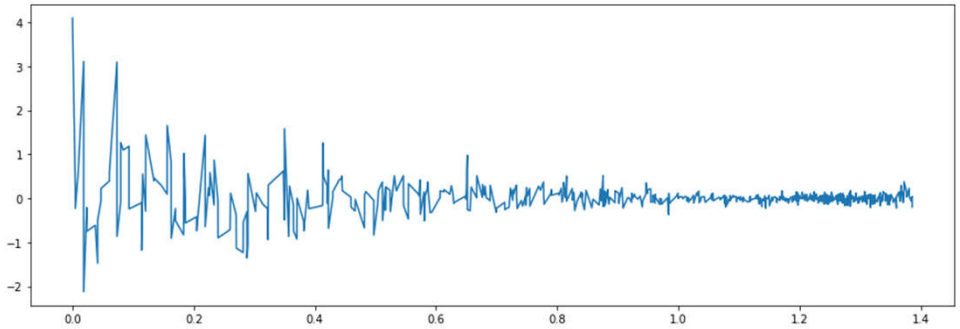
Spectral domain: $\hat{\boldsymbol{f}}$

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

Going Back to Graph Spectral Theory

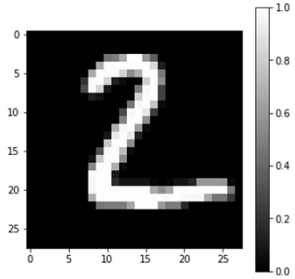
Do we need all the coefficients to reconstruct a “good” signal?

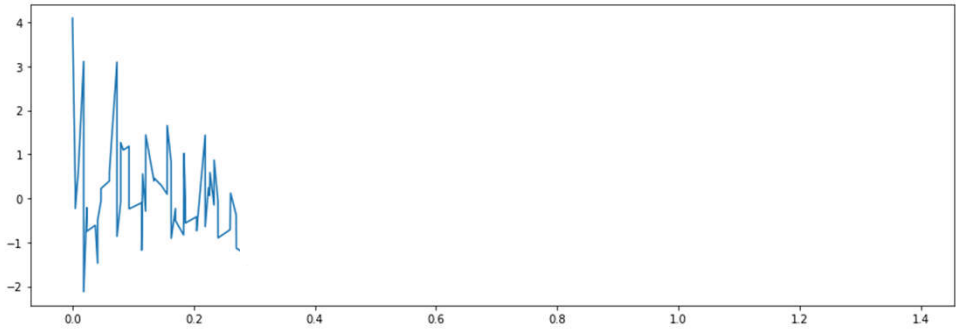




Going Back to Graph Spectral Theory

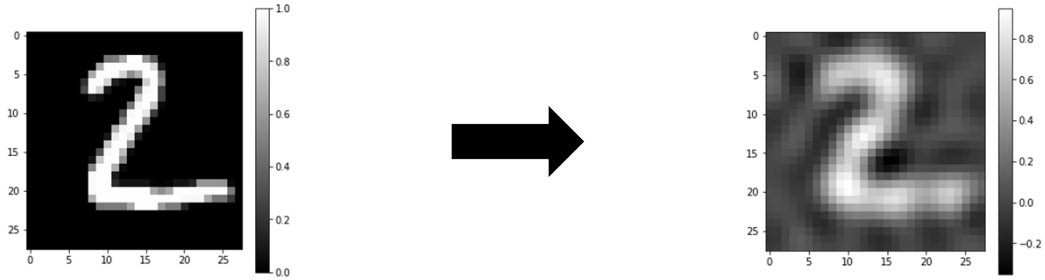
Do we need all the coefficients to reconstruct a “good” signal?

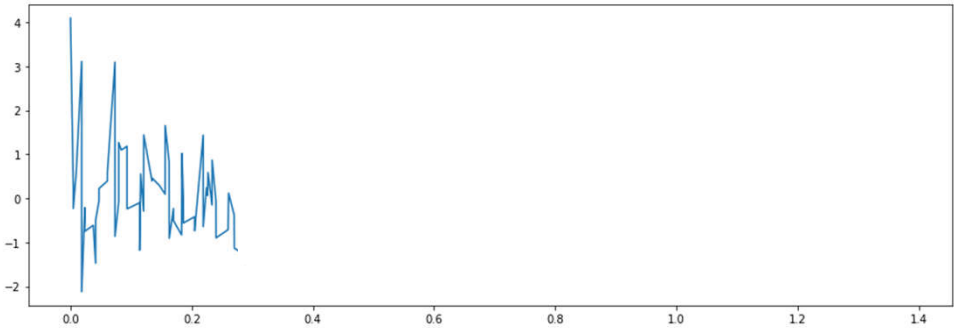




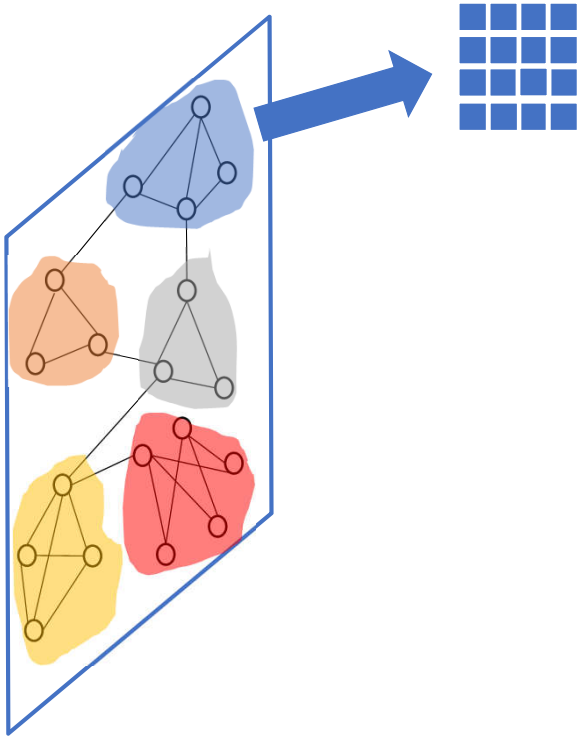
Going Back to Graph Spectral Theory

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



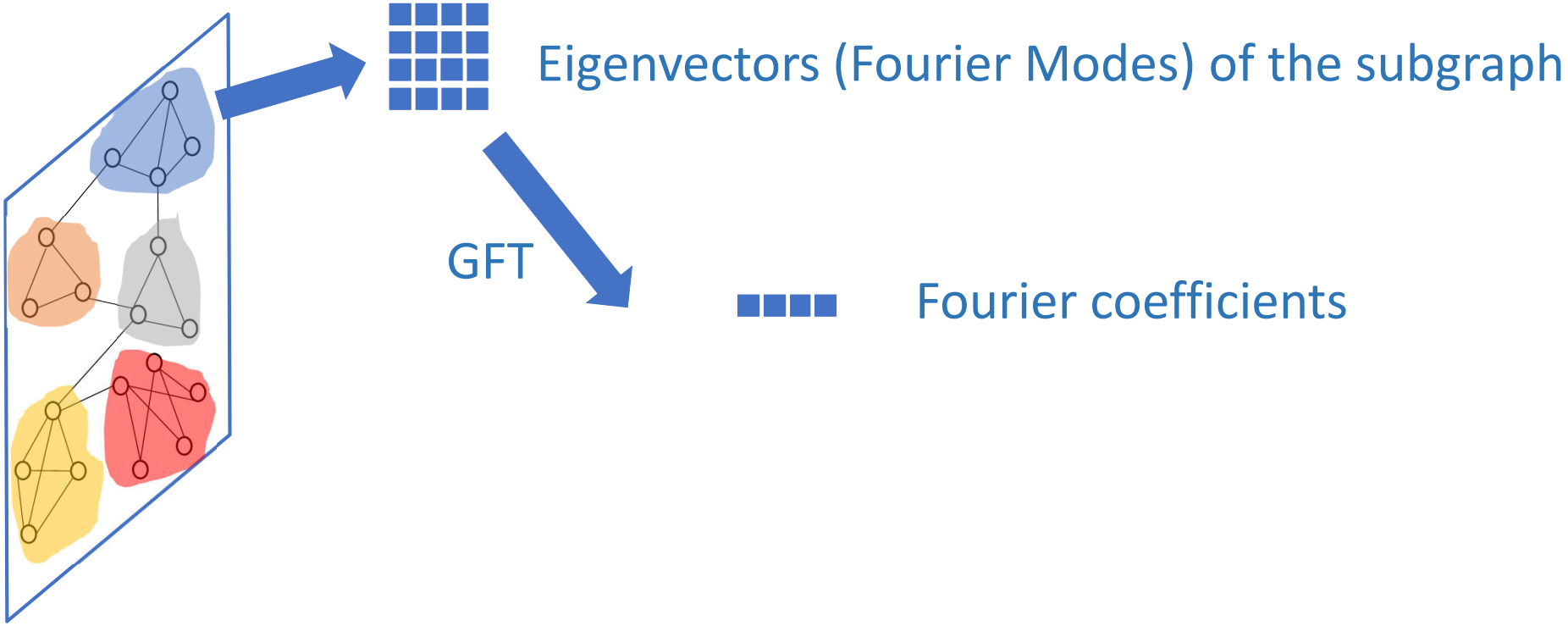


Eigenpooling: Truncated Fourier Coefficients

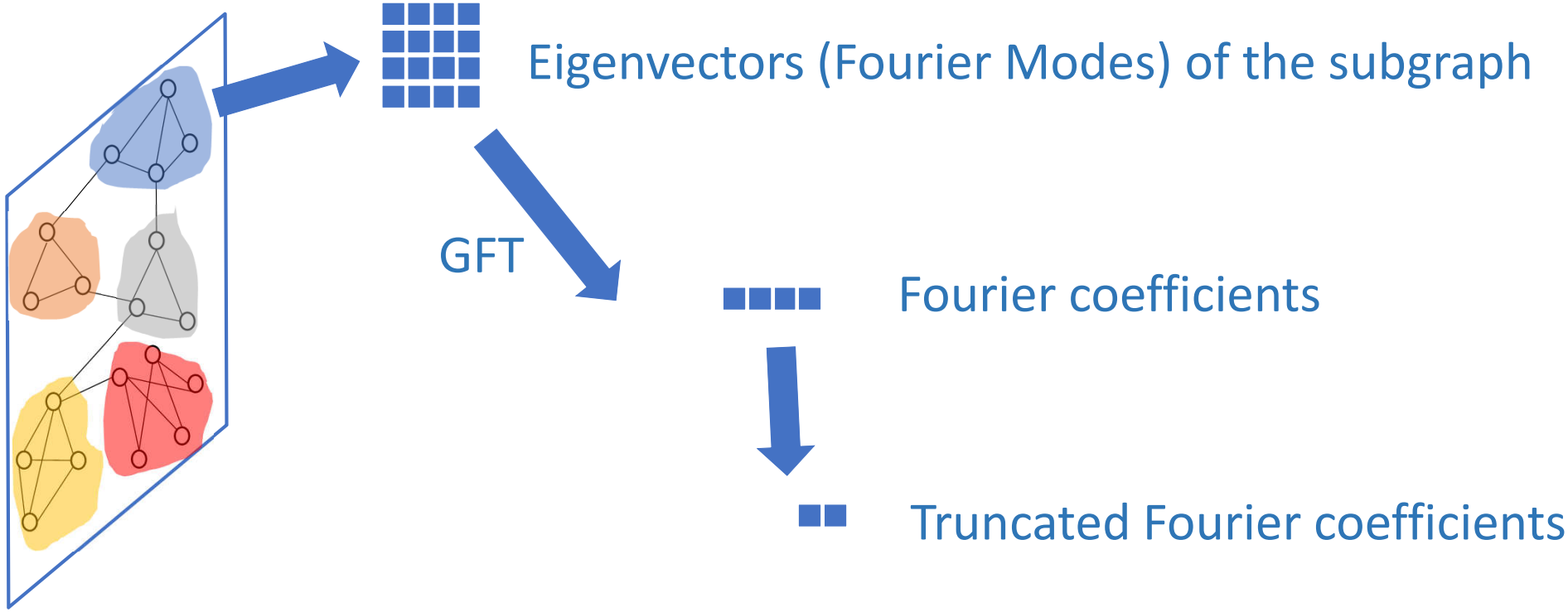


Eigenvectors (Fourier Modes) of the subgraph

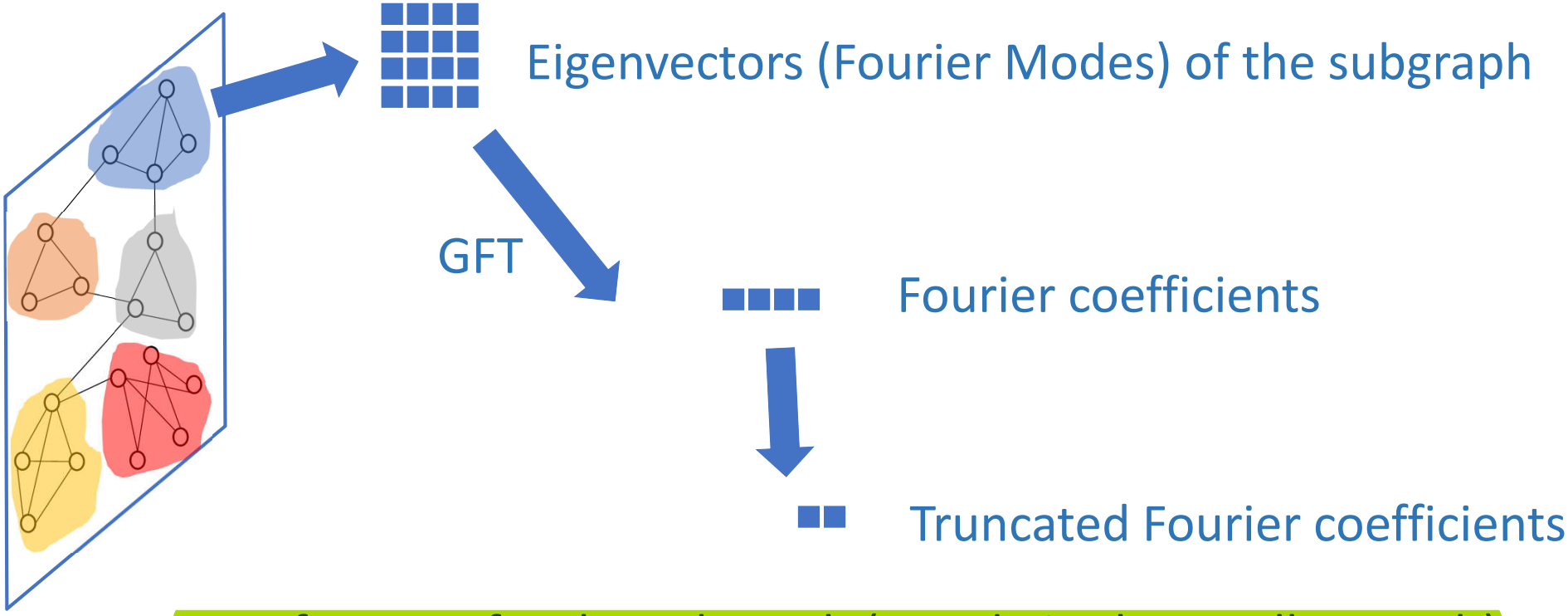
Eigenpooling: Truncated Fourier Coefficients



Eigenpooling: Truncated Fourier Coefficients



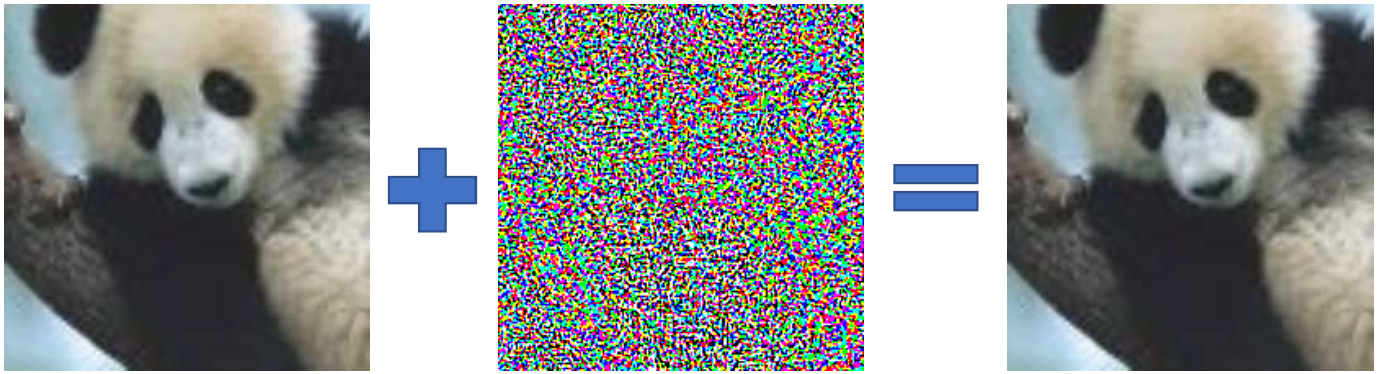
Eigenpooling: Truncated Fourier Coefficients



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

Robustness of GNN

Adversarial Attacks on Deep Learning



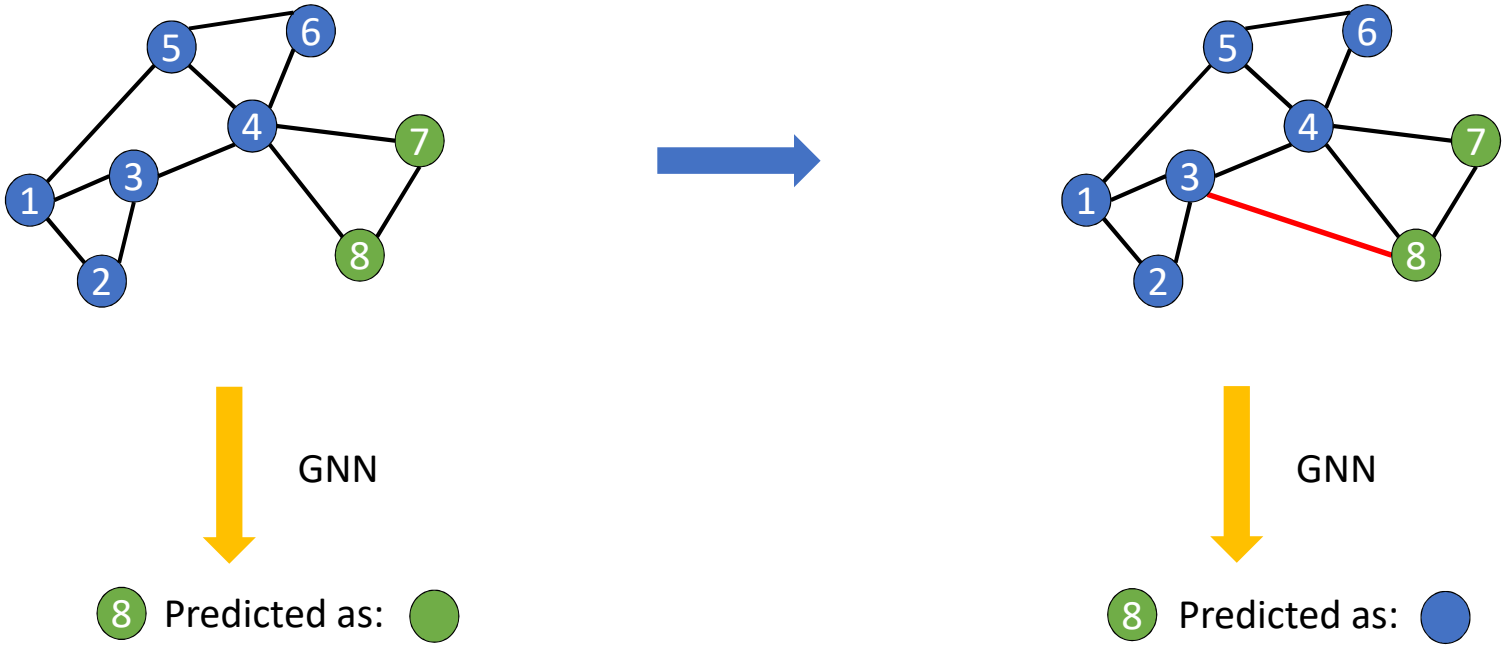
Classified as panda | Small adversarial noise | Classified as gibbon

x | ϵ | x'

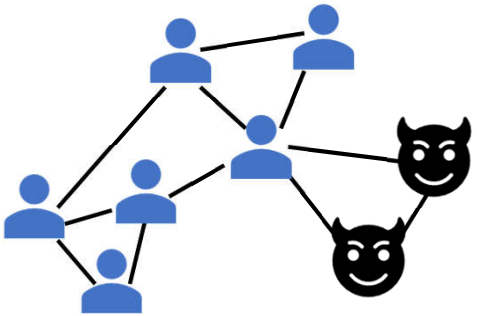
Find x' satisfying $||x' - x|| \leq \Delta$
such that $C(x') \neq y$

Do Graph Neural Networks Suffer the Same Problem?

Adversarial Attacks on GNN



Consequences



Financial Systems

- Credit Card Fraud Detection

Recommender Systems

- Social Recommendation
- Product Recommendation

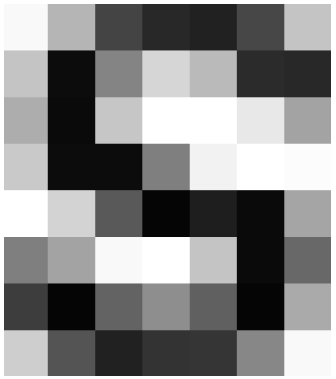
....

Image vs Graph

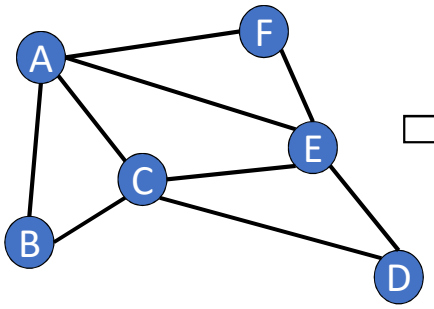
Discreteness

Perturbation Measure

Perturbation Type

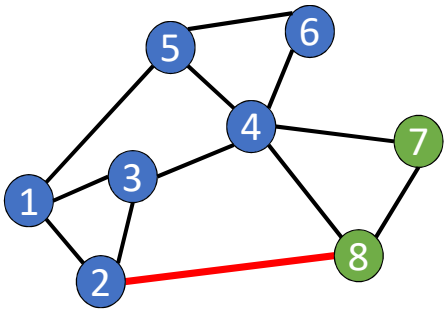


$$\begin{bmatrix} 0.98, & 0.71, & 0.27, & 0.16, & 0.13, & 0.28, & 0.77 \\ 0.77, & 0.05, & 0.52, & 0.84, & 0.73, & 0.17, & 0.16 \\ 0.68, & 0.04, & 0.78, & 1.00, & 1.00, & 0.91, & 0.64 \\ 0.79, & 0.05, & 0.05, & 0.50, & 0.95, & 1.00, & 0.99 \\ 1.00, & 0.83, & 0.35, & 0.02, & 0.12, & 0.04, & 0.65 \\ 0.50, & 0.64, & 0.98, & 1.00\ , & 0.77, & 0.04, & 0.41 \\ 0.24, & 0.02, & 0.39, & 0.56, & 0.38, & 0.02, & 0.67 \\ 0.81, & 0.33, & 0.13, & 0.20, & 0.21, & 0.53, & 0.98 \end{bmatrix}$$

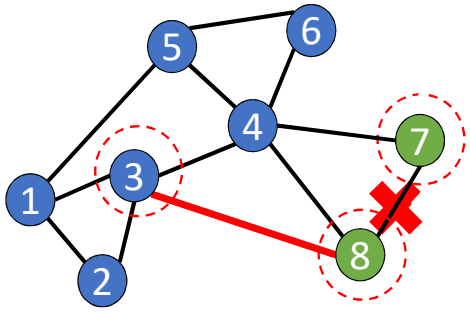


	A	B	C	D	E	F
A	0	1	1	0	1	1
B	1	0	1	0	0	0
C	1	1	0	1	1	0
D	0	0	1	0	1	0
E	1	0	1	1	0	0
F	1	0	0	0	1	0

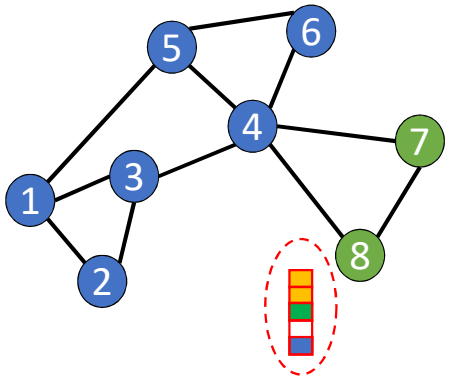
Perturbation Type



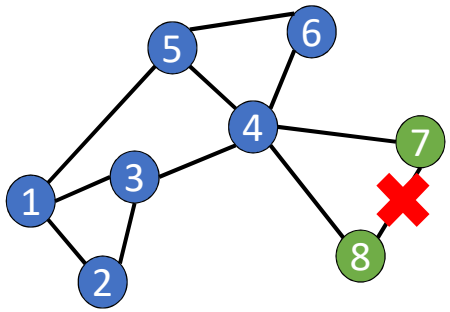
Adding an edge



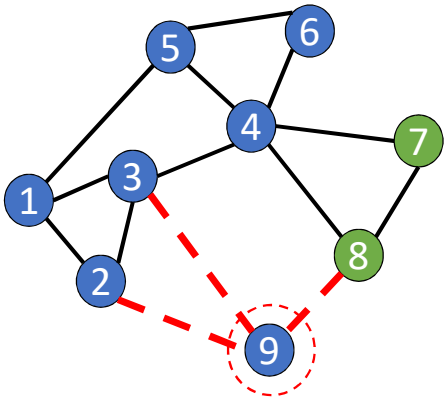
Rewiring



Modifying Features



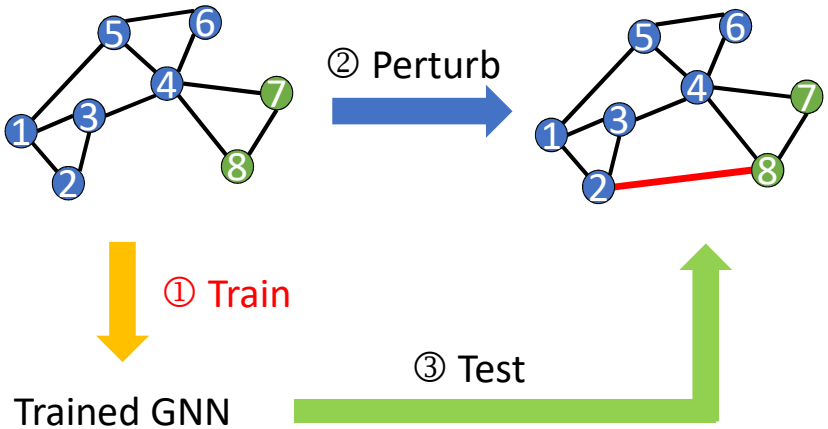
Deleting an edge



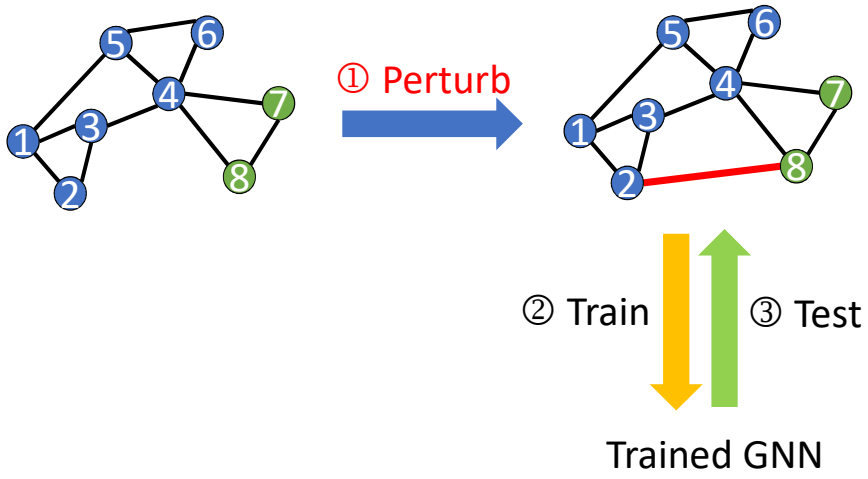
Node Injection

Evasion & Poisoning Attack

Evasion Attack

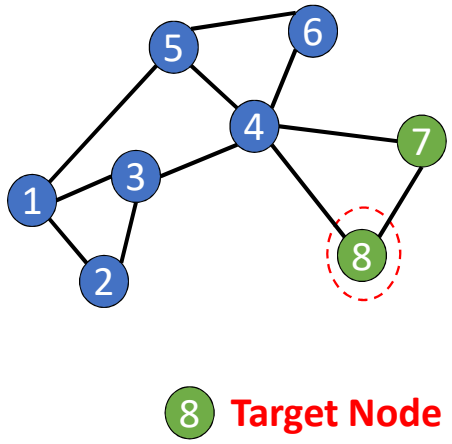


Poisoning Attack

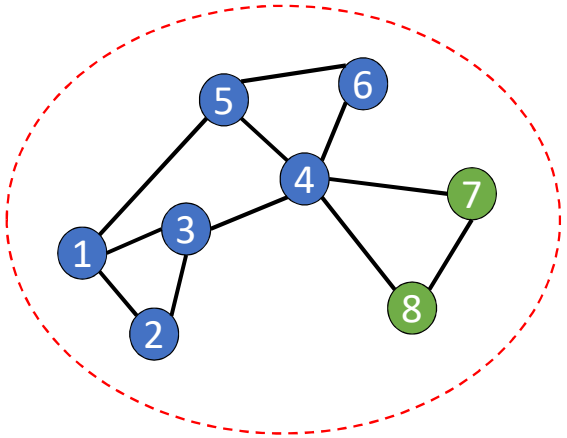


Targeted & Non-Targeted

Targeted Attack



Non-Targeted Attack



Attack Methods

Attack Methods	Injecting Node	Adding /Deleting Edge	Rewiring	Modifying Features	Evasion	Poisoning	Targeted	Non-Targeted
Grad-Argmax	✓	✓		✓	✓	✓	✓	✓
Nettack		✓		✓	✓	✓	✓	
ReWatt			✓		✓			✓
RL-S2V		✓			✓		✓	
Meta-Attack		✓				✓		✓
NIPA	✓					✓		✓

Attack Methods

Attack Methods	Injecting Node	Adding /Deleting Edge	Rewiring	Modifying Features	Evasion	Poisoning	Targeted	Non-Targeted
Grad-Argmax	✓	✓		✓	✓	✓	✓	✓

GradArgmax

Our Goal:

$$\arg\max_{\hat{A},\hat{X}} \sum_{u \in V_t} \ell\left(f_{\theta^*}(\hat{A},\hat{X})_u, y_u\right)$$

$$s.t.\ \theta^* = arg\min_{\theta} \mathcal{L}(f_\theta(A', X'))$$

$$\left|\hat{A} - A\right| + \left|\hat{X} - X\right| < \Delta$$

Adversarial Attack on Graph Structured Data. ICML 2018.
Adversarial Attacks on Neural Networks for Graph Data. KDD 2018

GradArgmax

Our Goal:

$$\arg\max_{\hat{A},\hat{X}} \sum_{u \in V_t} \ell\left(f_{\theta^*}(\hat{A},\hat{X})_u, y_u\right)$$

$$s.t.\ \theta^* = arg\min_{\theta} \mathcal{L}(f_\theta(A', X'))$$

$$\left|\hat{A} - A\right| + \left|\hat{X} - X\right| < \Delta$$

Perturbations

Modifications on A, X

Evasion: $A' = A,\ X' = X$

Poisoning: $A' = \hat{A},\ X' = \hat{X}$

Non-Targeted: $V_t = V_{all}$

Targeted: V_t is a small subset

GradArgmax

Our Goal:

$$\arg\max_{\hat{A},\hat{X}} \sum_{u \in V_t} \ell\left(f_{\theta^*}(\hat{A},\hat{X})_u, y_u\right)$$

$$s.t.\ \theta^* = arg\min_{\theta} \mathcal{L}(f_\theta(A',X'))$$

$$\left|\hat{A} - A\right| + \left|\hat{X} - X\right| < \Delta$$

Perturbations

Modifications on A, X

Evasion: $A' = A,\ X' = X$

Poisoning: $A' = \hat{A},\ X' = \hat{X}$

Non-Targeted: $V_t = V_{all}$

Targeted: V_t is small subset

GradArgmax

Our Goal:

$$\arg\max_{\hat{A},\hat{X}} \sum_{u \in V_t} \ell\left(f_{\theta^*}(\hat{A},\hat{X})_u, y_u\right)$$

$$s.t.\ \theta^* = arg\min_{\theta} \mathcal{L}(f_\theta(A', X'))$$

$$\left|\hat{A} - A\right| + \left|\hat{X} - X\right| < \Delta$$

Perturbations

Modifications on A, X

Evasion: $A' = A,\ X' = X$

Poisoning: $A' = \hat{A},\ X' = \hat{X}$

Non-Targeted: $V_t = V_{all}$

Targeted: V_t is small subset

GradArgmax

Gradient Ascent:

$$\hat{A} = \hat{A} + \gamma_1 \nabla_A \mathcal{L}(A, X)$$
$$\hat{X} = \hat{X} + \gamma_2 \nabla_X \mathcal{L}(A, X)$$

Greedy Method

In each step, choose the perturbation with maximum gradient

Attack Methods

Attack Methods	Injecting Node	Adding /Deleting Edge	Rewiring	Modifying Features	Evasion	Poisoning	Targeted	Non-Targeted
Grad-Argmax	✓	✓		✓	✓	✓	✓	✓
Nettack		✓		✓	✓	✓	✓	

Nettack

Shortcomings of GradArgmax

- Need to access the model parameters

  Gradient Information

- Perturbation constraint is not enough

$$\left|\hat{A} - A\right| + \left|\hat{X} - X\right| \leq \Delta$$

Adversarial Attacks on Neural Networks for Graph Data. KDD 2018.

Nettack

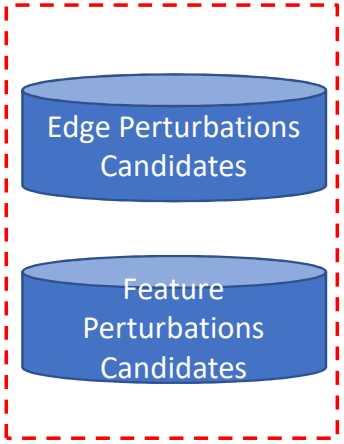
Idea 1: Train a surrogate model

A two-layer linearized GCN trained on original graph

Idea 2: Perturbation Measure

- $|A - A'| + |X - X'| < \Delta$
- Preserving Degree Distribution
- Preserving Feature Co-occurrence

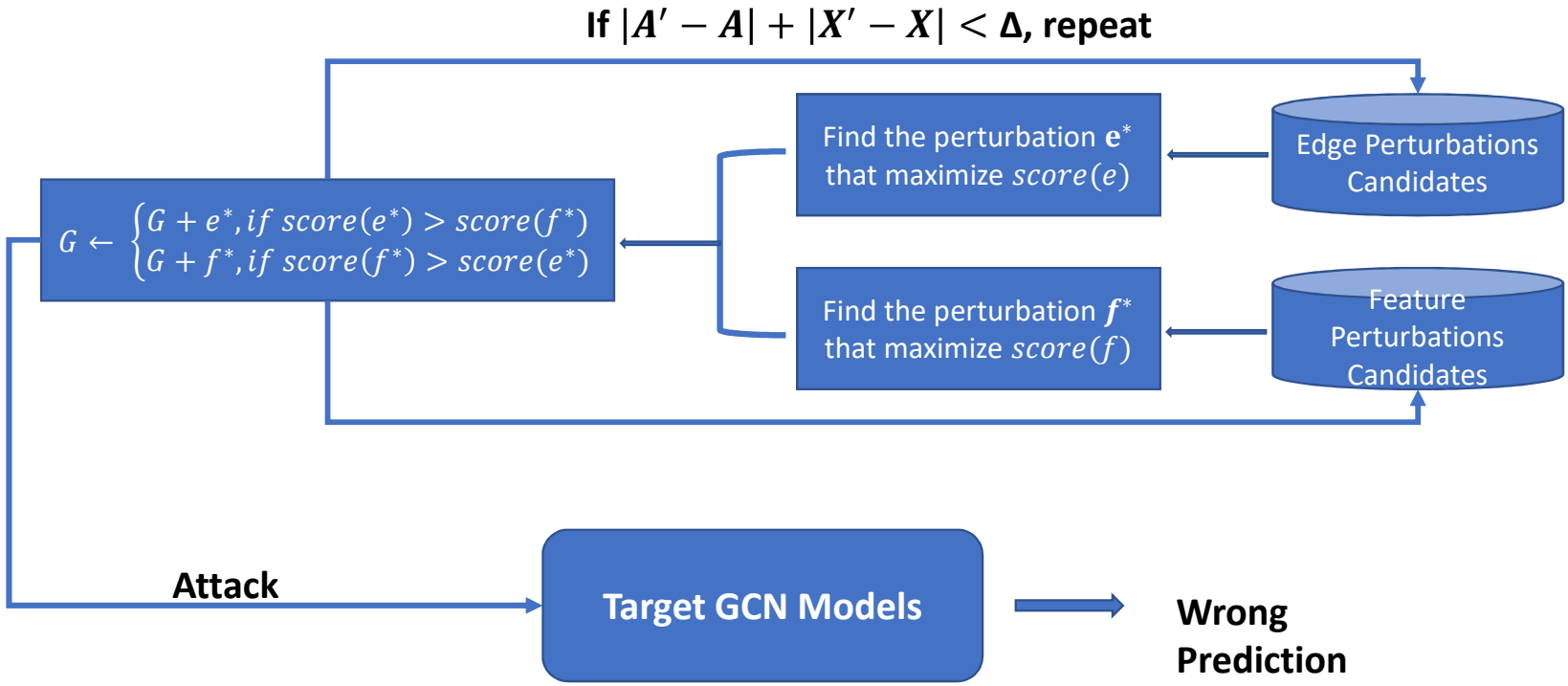
Nettack



Degree Distribution

Feature Co-occurrence

Nettack


If $|A' - A| + |X' - X| < \Delta$, repeat
Find the perturbation $\mathbf{e}^*$ that maximize $score(e)$
Edge Perturbations Candidates
$G \leftarrow \begin{cases} G + e^*, if\ score(e^*) > score(f^*) \\ G + f^*, if\ score(f^*) > score(e^*) \end{cases}$
Find the perturbation $\boldsymbol{f}^*$ that maximize $score(f)$
Feature Perturbations Candidates
Attack
Target GCN Models
Wrong Prediction

Attack Methods

Attack Methods	Injecting Node	Adding /Deleting Edge	Rewiring	Modifying Features	Evasion	Poisoning	Targeted	Non-Targeted
Grad-Argmax	✓	✓		✓	✓	✓	✓	✓
Nettack		✓		✓	✓	✓	✓	
ReWatt			✓		✓			✓

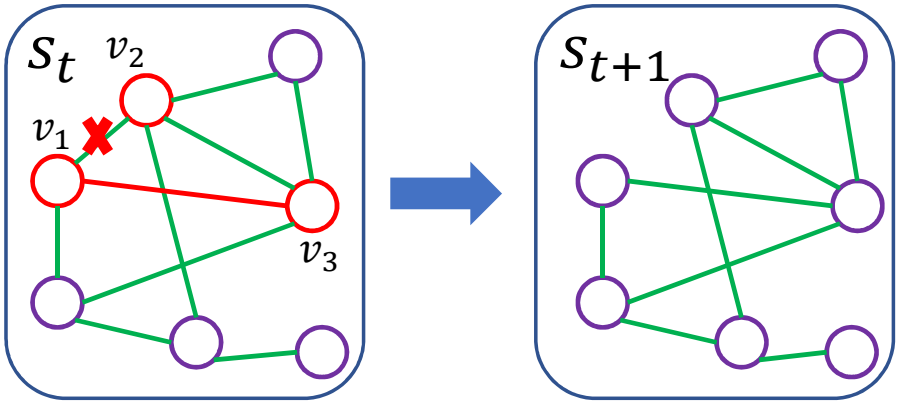
ReWatt

Motivation

Degree distribution may not be an ideal measure for perturbations

How to make perturbation more unnoticeable?

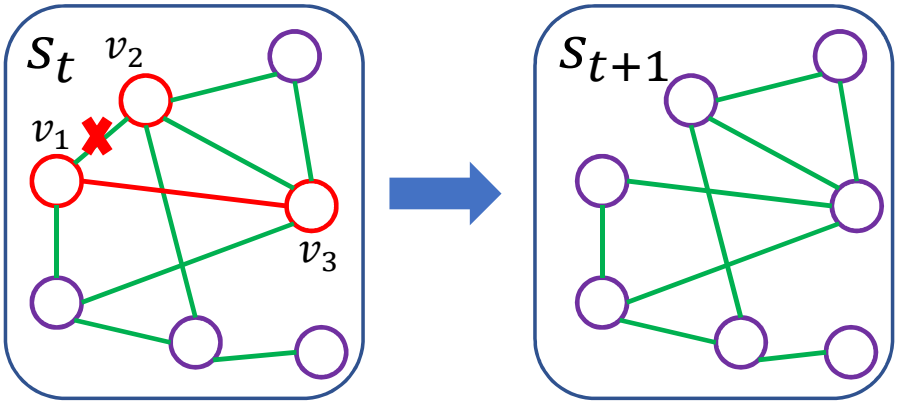
Rewiring



Attacking Graph Convolutional Networks via Rewiring. Arxiv 2019.

ReWatt

Rewiring

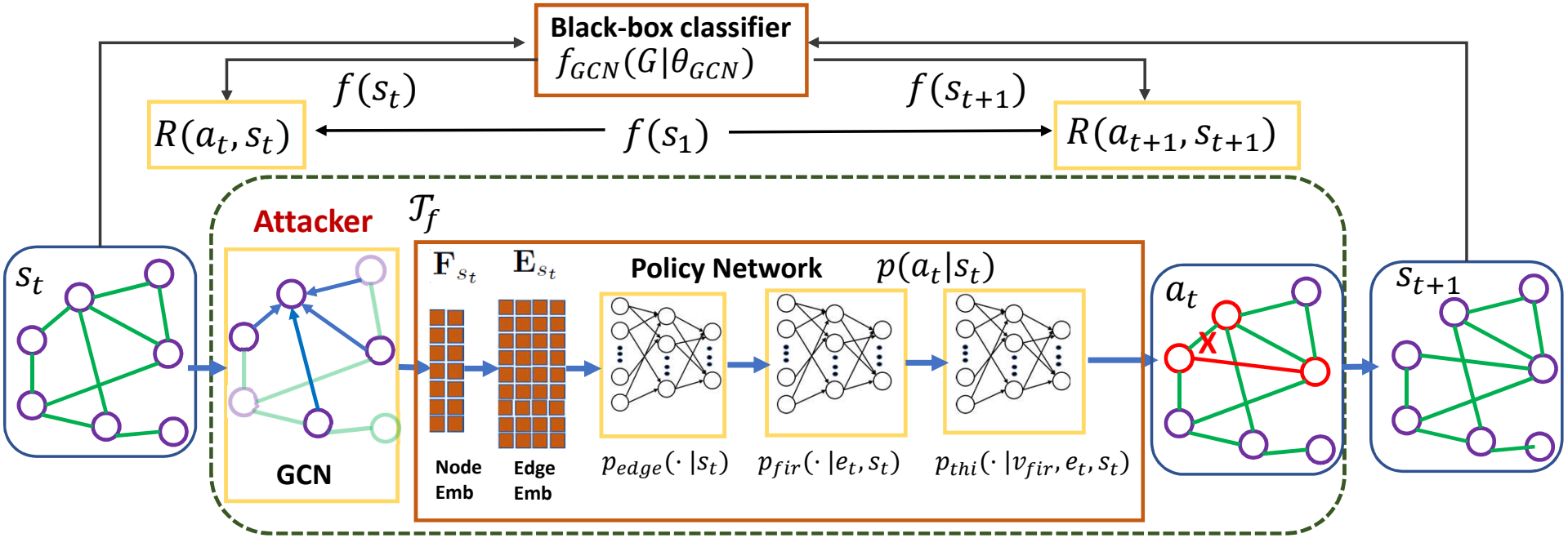


Advantages

- Number of nodes and edges remain the same
- Affects algebraic connectivity in a smaller way
- Affects effective graph resistance in a smaller way

ReWatt

Reinforcement Learning



Defending Against Attacks

Adversarial Training

Preprocessing

Attention Mechanism

Adversarial Training

Motivation

Augment the training set with adversarial data

Main Idea

$$\min_{\theta} \max_{\substack{\delta_A \in \mathcal{P}_A \\ \delta_X \in \mathcal{P}_X}} f_\theta(A + \delta_A, X + \delta_X)$$

Latent Adversarial Training of Graph Convolution Networks. ICML 2019 workshop.

Adversarial Training

Obstacles

- A is discrete
- X is often discrete

Hidden Adversarial Training

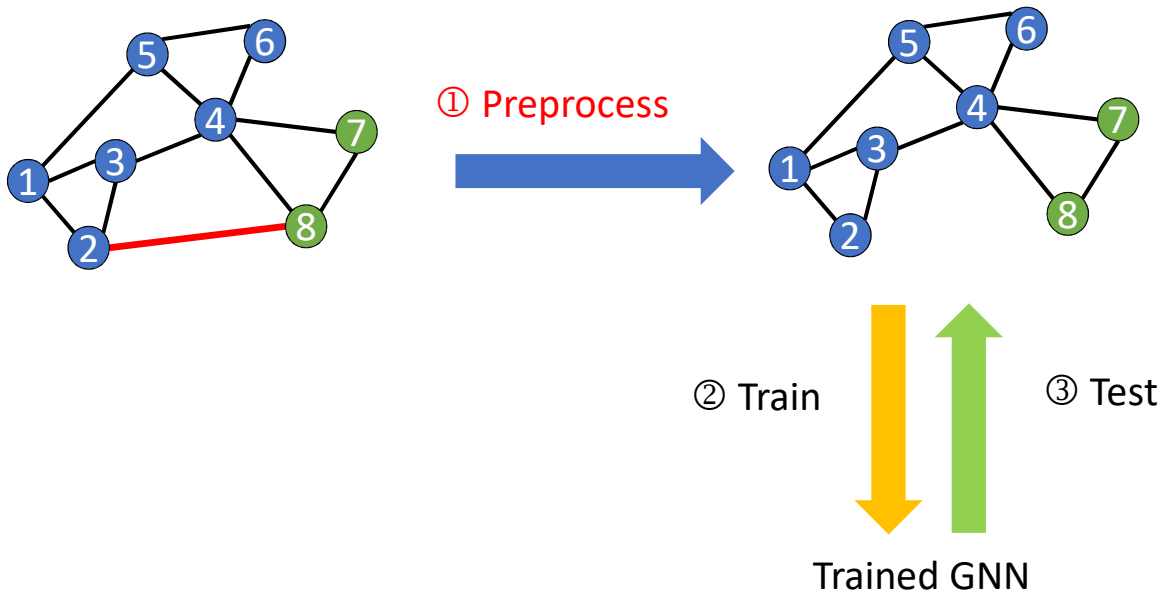
Apply it on the hidden layer !

$$\min_{\theta} \max_{\delta \in \mathcal{P}} f_{\theta}(H^{(1)} + \delta)$$

Preprocessing

Main Idea

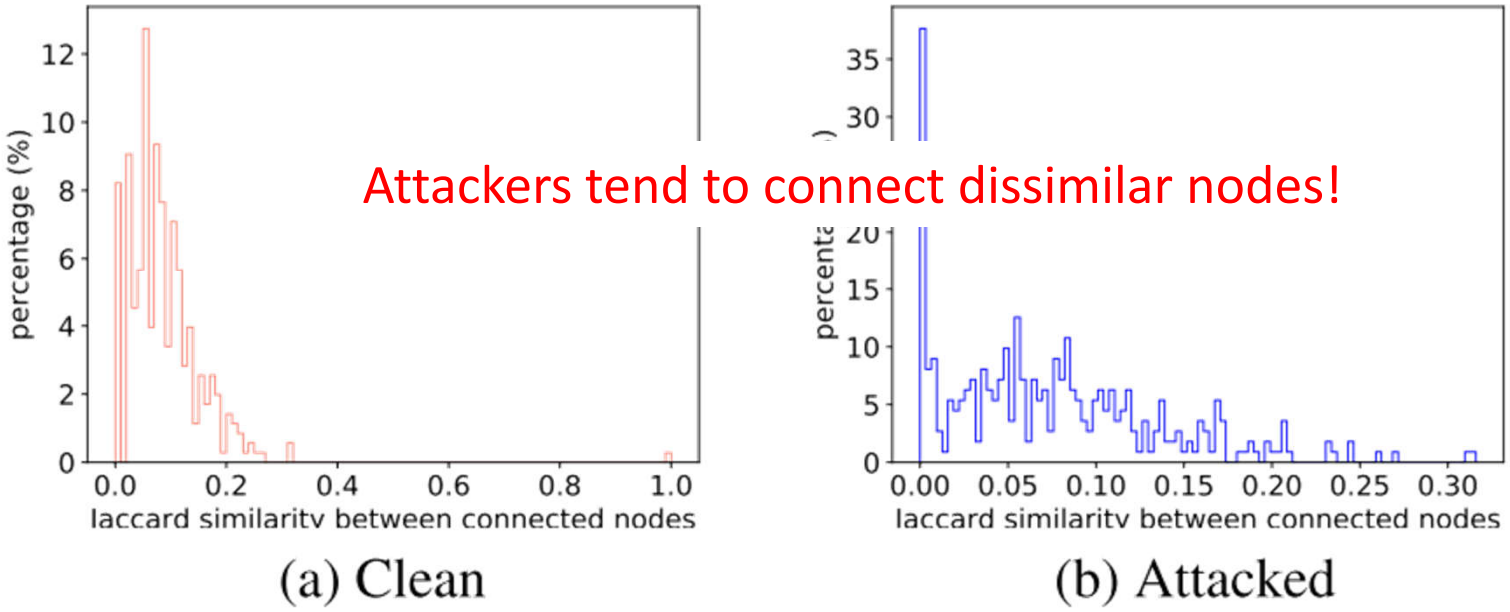
- Purify the poisoned graph
- Train GNN on the purified graph



Preprocessing

Observations

- Attackers favor adding edges than removing edges
-



Attackers tend to connect dissimilar nodes!

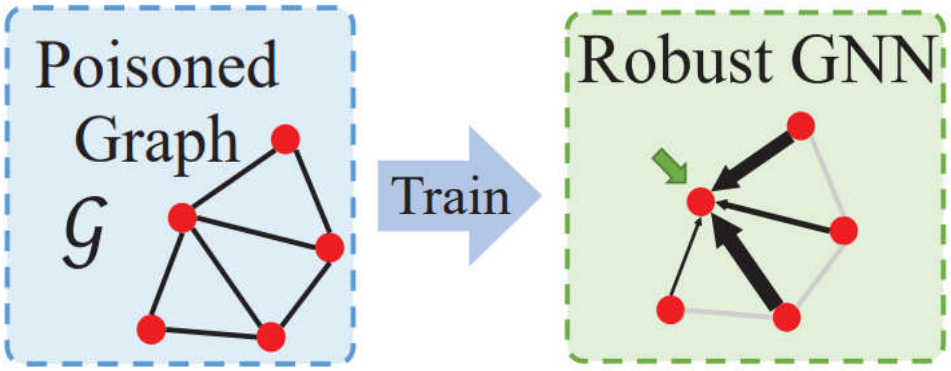
Adversarial Examples on Graph Data: Deep Insights into Attack and Defense. IJCAI 2019.

Attention Mechanism

Motivation

Reduce impact of adversarial edges

-- give lower attention score to adversarial edges



Thicker arrows indicate higher attention coefficients

RGCN

Motivation

Attacked nodes may have high uncertainty

Give lower attention score to reduce their impact

Robust Graph Convolutional Networks Against Adversarial Attacks. KDD 2019.

RGCN

Embed nodes as Gaussian distributions to capture uncertainty

$$\boldsymbol{h}_i^{(l)} \sim N(\boldsymbol{\mu}_{\mathrm{i}}^{(l)}, diag\left(\boldsymbol{\sigma}_i^{(l)}\right))$$

Aggregating

Assuming all h_i^l are independent

$$\boldsymbol{h}_{N(i)}^{(l)} = \sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}}} \boldsymbol{h}_{\mathrm{j}}^{(l)}$$

$$\sim N\left(\sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}}} \boldsymbol{\mu}_{\mathrm{j}}^{(l)}, diag\left(\sum_{j \in N(i)} \frac{1}{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}} \boldsymbol{\sigma}_{\mathrm{j}}^{(l)}\right)\right)$$

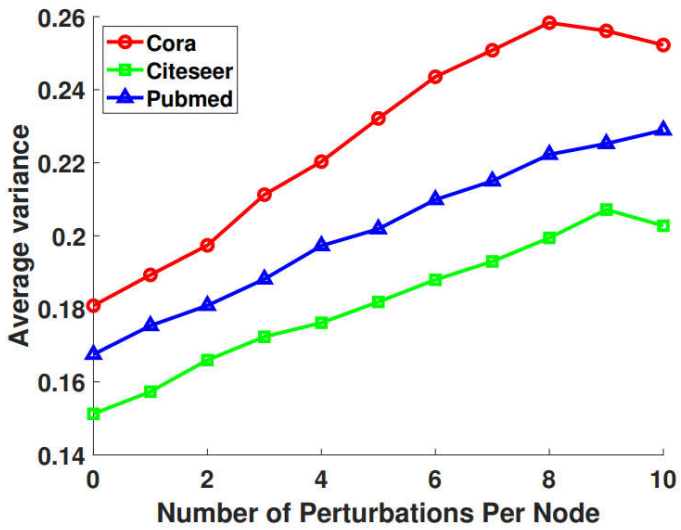
RGCN

Attention Mechanism

$$\boldsymbol{h}_{N(i)}^{(l)} = \sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}}} (\boldsymbol{h}_{\mathrm{j}}^{(l)} \odot \boldsymbol{\alpha}_j^l)$$

$$\boldsymbol{\alpha}_j^{(l)} = \exp(-\gamma \boldsymbol{\sigma}_j^{(l)})$$

Attacked nodes do have higher variance!



PA-GNN

Motivation

- Only relying on perturbed graph to learn attention coefficients is not enough.
- We should exploit information from clean graphs.

Clean graphs from similar domain

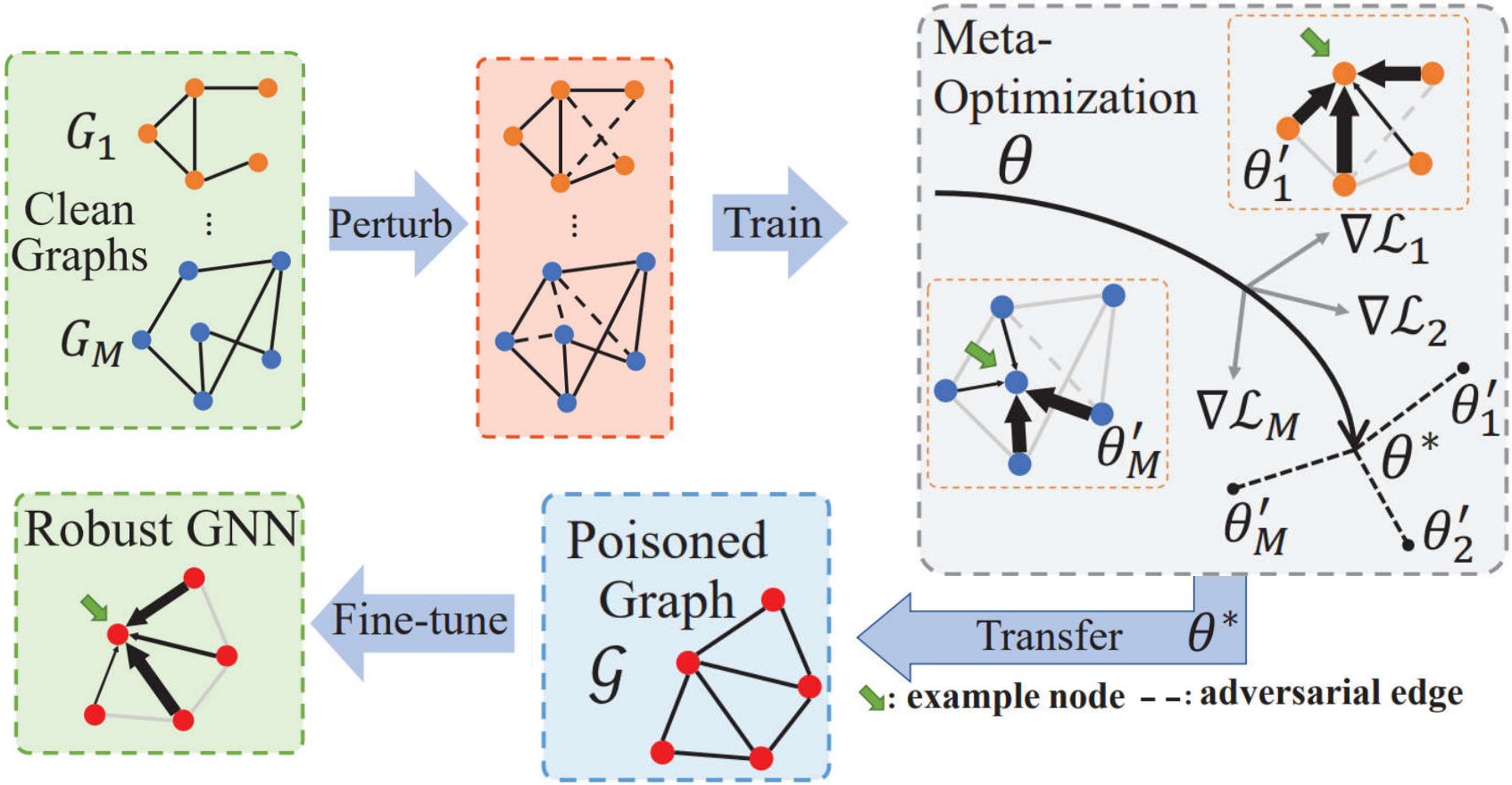
Facebook & Twitter

Yelp & Foursquare

Then Use Transfer Learning/Meta Learning!

Robust Graph Neural Network Against Poisoning Attacks via Transfer Learning. WSDM 2020.

PA-GNN


G_1
Clean Graphs
G_M
Perturb
Train
Meta-Optimization
θ
θ'_1
$\nabla\mathcal{L}_1$
$\nabla\mathcal{L}_2$
$\nabla\mathcal{L}_M$
θ'_M
θ^*
θ'_1
θ'_M
θ'_2
Transfer θ^*
Poisoned Graph $\mathcal{G}$
Fine-tune
Robust GNN
: example node
: adversarial edge

Scalable Learning of Graph Neural Networks

Tengfei Ma

IBM Research AI
IBM T. J. Watson Research Center

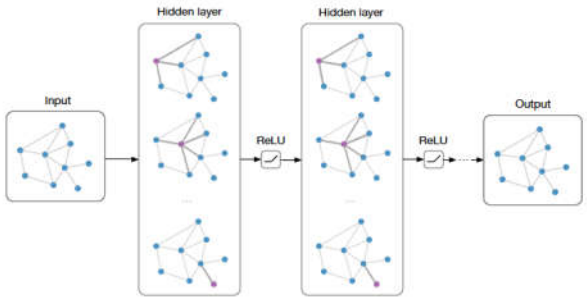
@AAAI 2020 Tutorial

Graph Convolutional Networks (GCN) - (Kipf and Welling 2017)

Objective: graph node embedding for arbitrary graphs

Basic Notations

- Node features $\boldsymbol{X}$
- Adjacency matrix $\boldsymbol{A}$ that represents graph structure



Output: Node embedding $\boldsymbol{Z}$

Stack multiple convolution layers

$$\boldsymbol{H}^{(l+1)} = f(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$, $\boldsymbol{D}$ is a diagonal matrix such that $D_{ii} = \sum_j \tilde{A}_{ij}$, $\boldsymbol{W}^{(l)}$ is layer-specific parameter matrix, $\boldsymbol{H}^{(l)}$ is node representation of lth layer. $\boldsymbol{H}^{(0)} = \boldsymbol{X}$

Motivation

Matrix form of a GCN layer: $\mathbf{H}^k = \sigma(\widehat{\mathbf{A}}\mathbf{H}^{k-1}\mathbf{W}^k)$

Problems of GCN:

- Time and memory cost for large graphs: $A \in \mathbb{R}^{n*n}$

Per-node form (embedding vectors are oriented as column vectors)

$$\mathbf{h}_v^k = \sigma\left(\widehat{a}_{vv}(\mathbf{W}^k)^\top \mathbf{h}_v^{k-1} + \sum_{u \in \boxed{\mathcal{N}(v)}} \widehat{a}_{vu}(\mathbf{W}^k)^\top \mathbf{h}_u^{k-1}\right)$$

Full neighborhood of v

From a single node as the start, after a few layers, almost the whole graph will be touched. It means, even batch training is also expensive

Node Sampling
-GraphSAGE (Hamilton et al. 2017)

Node sampling: for each node just sample a fixed number of neighbors

- Matrix form $\mathbf{H}^k(\mathrm{idx}_k, :) = \sigma(\widehat{\mathbf{A}}(\mathrm{idx}_k, \mathrm{idx}_{k-1})\mathbf{H}^{k-1}(\mathrm{idx}_{k-1}, :)\mathbf{W}^k)$
  - Where idx_{k-1} is a uniformly randomly sampled subset of nodes. (For nonuniform random, need proper scaling.)
- Per-node form
  - For all v in idx_k only

$$\mathbf{h}_v^k = \sigma\left(\widehat{a}_{vv}(\mathbf{W}^k)^\top \mathbf{h}_v^{k-1} + \sum_{\boxed{u \in \mathrm{idx}_{k-1}}} \widehat{a}_{vu}(\mathbf{W}^k)^\top \mathbf{h}_u^{k-1}\right)$$

- Problems of node sampling
  - Still power law
  - No formal analysis to justify

Sampled neighborhood
(same for all v in idx_k)

Layer Sampling
- FastGCN (ICLR 2018)

Generalization of a GCN layer (assume each layer independent)

$$\tilde{h}^{(l+1)}(v) = \int \hat{A}(v,u)h^{(l)}(u)W^{(l)}\,dP(u), \quad h^{(l+1)}(v) = \sigma(\tilde{h}^{(l+1)}(v))$$

Monte Carlo sampling

- For **each layer/batch** sample nodes

$$u_1^{(l)}, \ldots, u_{t_l}^{(l)} \sim P$$

$$h_{t_{l+1}}^{(l+1)}(v) = \sigma\left(\frac{1}{t_l}\sum_{j=1}^{t_l}\hat{A}(v,u_j^{(l)})h_{t_l}^{(l)}(u_j^{(l)})W^{(l)}\right)$$

- matrix form (in a batch, all “v” has the same sampled neighbors)

$$H_{t_{l+1}}^{(l+1)}(v,:) = \sigma\left(\frac{n}{t_l}\sum_{j=1}^{t_l}\hat{A}(v,u_j^{(l)})H_{t_l}^{(l)}(u_j^{(l)},:)W^{(l)}\right)$$

Comparison

GCN

FastGCN

Importance Sampling

- Uniform sampling

$$H_{t_{l+1}}^{(l+1)}(v,:) = \sigma\left(\frac{n}{t_l}\sum_{j=1}^{t_l}\hat{A}(v,u_j^{(l)})H_{t_l}^{(l)}(u_j^{(l)},:)W^{(l)}\right) \quad u_1^{(l)},\ldots,u_{t_l}^{(l)} \sim P$$

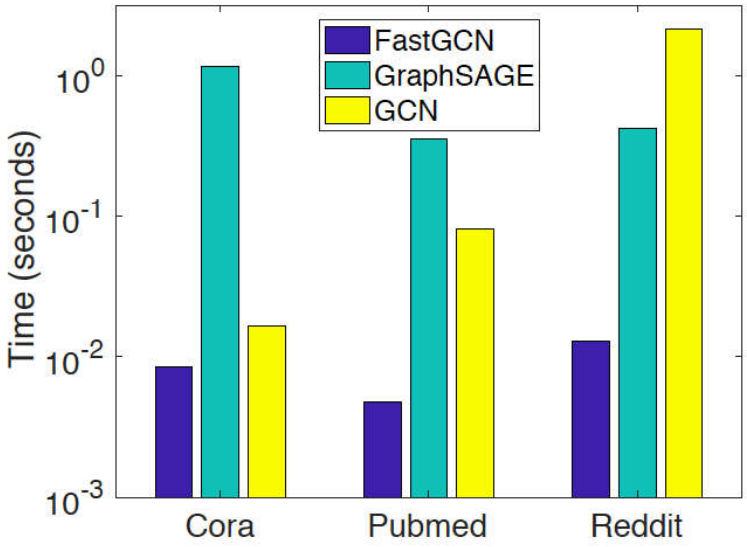
- Variance reduction
  - Importance sampling, sampling from Q instead of a uniform P

$$H^{(l+1)}(v,:) = \sigma\left(\frac{1}{t_l}\sum_{j=1}^{t_l}\frac{\hat{A}(v,u_j^{(l)})H^{(l)}(u_j^{(l)},:)W^{(l)}}{q(u_j^{(l)})}\right), \quad u_j^{(l)} \sim q$$

$$q(u) = \|\hat{A}(:,u)\|^2 / \sum_{u'\in V}\|\hat{A}(:,u')\|^2, \quad u \in V$$

Results

Per-batch training time



Prediction accuracy

Micro F1 Score

	Cora	Pubmed	Reddit
FastGCN	0.850	0.880	0.937
GraphSAGE-GCN	0.829	0.849	0.923
GraphSAGE-mean	0.822	0.888	0.946
GCN (batched)	0.851	0.867	0.930
GCN (original)	0.865	0.875	NA

Adaptive Sampling (Huang et al. 2018)

Problem of FastGCN:

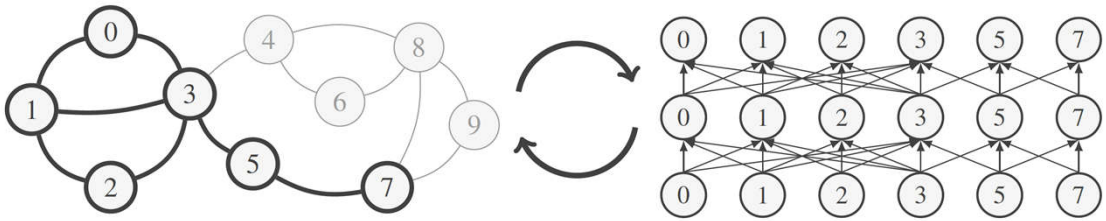
- Layer-independent assumption
- Too sparse sampling ->lower accuracy

Extension

- Layer-dependent: sample the lower layer ***conditioned*** on the top one
- Based on importance sampling schema, they ***learn a self-dependent function*** of each node to determine its importance for the sampling
- To explicitly reduce sampling variance, they ***add the variance to the loss*** function and explicitly minimize the variance by model training

Graph Sampling
- GraphSAINT (Zeng et al. 2020)

Instead of node sampling or layer sampling, do graph sampling



$\mathcal{G}_s = \texttt{SAMPLE}(\mathcal{G})$ Full GCN on $\mathcal{G}_s$

$$\zeta_v^{(\ell+1)} = \sum_{u\in\mathcal{V}} \frac{\widetilde{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} \left(\boldsymbol{W}^{(\ell)}\right)^{\mathsf{T}} \boldsymbol{x}_u^{(\ell)} \mathbb{1}_{u|v} = \sum_{u\in\mathcal{V}} \frac{\widetilde{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} \tilde{\boldsymbol{x}}_u^{(\ell)} \mathbb{1}_{u|v},$$

If $\alpha_{u,v} = \frac{p_{u,v}}{p_v}$ it is an unbiased estimator of the aggregator, i.e. $\mathbb{E}\left(\zeta_v^{(\ell+1)}\right) = \sum_{u\in\mathcal{V}} \widetilde{\boldsymbol{A}}_{v,u} \tilde{\boldsymbol{x}}_u^{(\ell)}$

- Pre-processing: repeatedly sample n subgraphs, and set

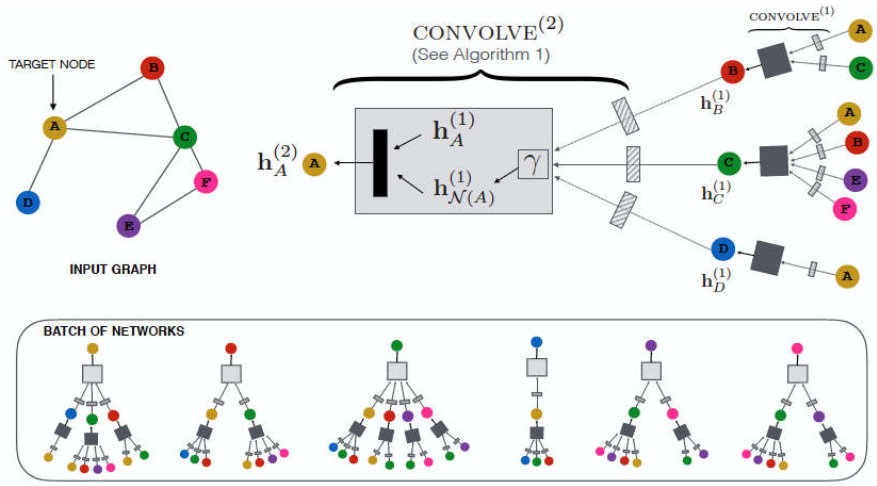
$$\alpha_{u,v} = \frac{C_{u,v}}{C_v} = \frac{C_{v,u}}{C_v}$$

- Run a full GCN on each subgraph

Application in Real-World - PinSAGE (Ying et al. 2018)

An early industry-level GNN-based recommendation system

- The core of PinSAGE is a neighborhood aggregation algorithm similar to GraphSAGE
- Novelty: how to define the neighborhood?
  - Importance-based: the neighborhood of a node *u* is defined as the *T* nodes that exert the most influence on node *u*.
  - Random walks from node *u*, top T visited nodes.
- Efficient Training:
  - Does not train on the whole graph, but only on targeted node set and their neighborhood
- MapReduce for model inference



Application in Real-World -Anti-Money Laundering

Application of FastGCN on a synthetic AML dataset:

- Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics (NeurIPS 2018 WS)
- Entities as nodes and transactions as edges

A Real AML dataset: Elliptic Dataset

- Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics (Weber et al. 2019)
- Transactions (in Bitcoins) as nodes, money flow as edges
- https://www.kaggle.com/ellipticco/elliptic-data-set

Advertisement☺ we have a paper regarding dynamic graph learning on that dataset in AAAI20 -- EvolveGCN (Feb 10 evening poster)

Graph Neural Networks for Healthcare Applications

--Drug Discovery and Medical Recommendation

Tengfei Ma

IBM Research AI
IBM T. J. Watson Research Center

Drug Discovery

Drug Discovery is a long tedious costly process

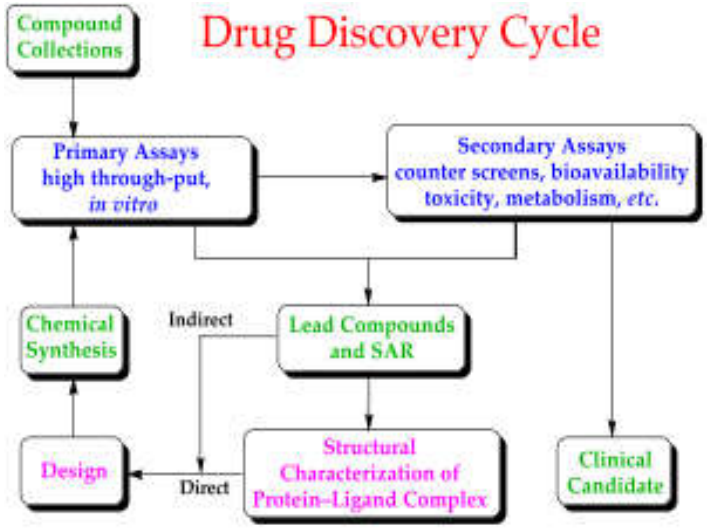
- Machine learning can help
  - *de novo* drug design
    - Generating new molecules for desired target.
  - drug safety checking
    - Toxicity
    - Adverse reaction/drug-drug interaction

Interestingly, they are all related to graphs

- Molecule -- graph
- DDI – graph

It is natural to develop GNN-based methods

- Molecule generation
- DDI prediction

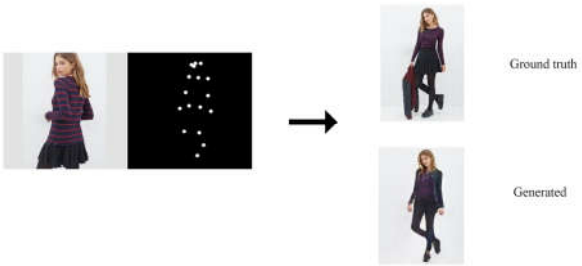


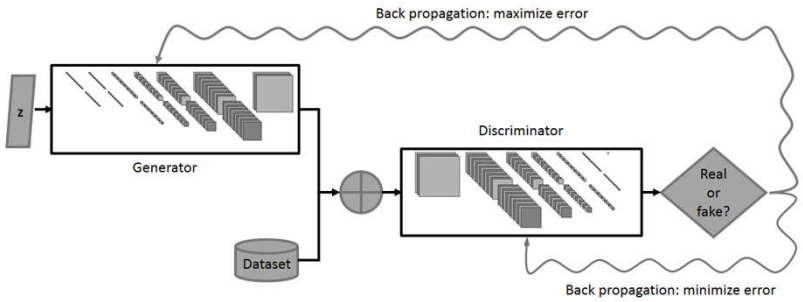
Constrained Generation of Semantically Valid Graphs via Regularizing Variational Autoencoders

NeurIPS 2018.

Molecule Graph Generation

Generative Models for Images/Sequences





But generation of graphs?

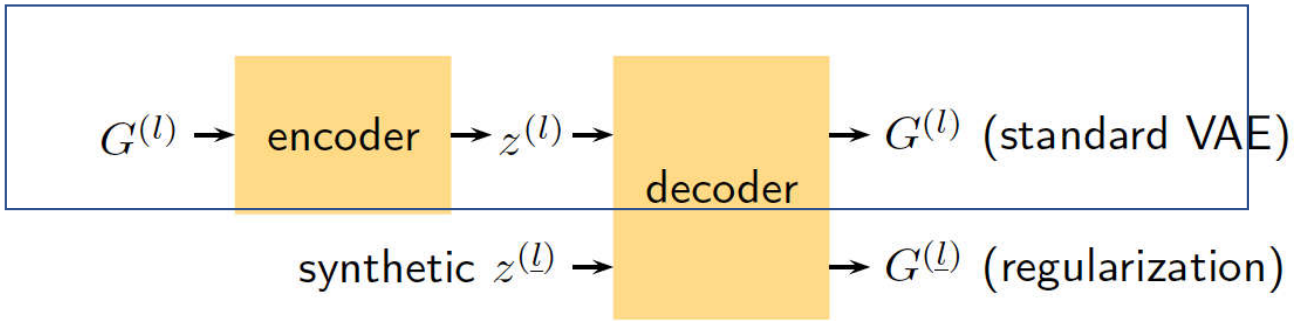
- Graph neural networks need to know the predefined graph structure
  - GNNs can be used as encoders, but how to design an decoder/generator?
- How to guarantee the generated sample is a valid graph?

Ideas:

- Represent graphs as concatenation of its node matrix and edge matrix and treat it as an image –> so we can use the same decoder as image
- Validness? add constraints for the generator

Constrained Graph Variational Auto-Encoder

Overview of the framework



- A graph auto-encoder used to generate the graph
- In addition to a standard VAE (within the rectangle), we add a regularization term.
- f(x) is the original VAE loss
- h and g are regularization terms

$$
\begin{aligned}
\min_{x} \quad & f(x) \\
\text{subject to} \quad & \text{for almost all } z \sim p_x(z), \\
& h_1(x,z) = 0, \ldots, h_m(x,z) = 0, \\
& g_1(x,z) \leq 0, \ldots, g_r(x,z) \leq 0.
\end{aligned}
$$

Approximate Training

A Lagrangian relaxation

$$-L_{\text{ELBO}}(\theta,\phi) + \mu \sum_i \left[\int g_i(\theta,z)_+^2 p_\theta(z)\, dz \right]^{\frac{1}{2}}$$

Training in Standard VAE

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

- Monte Carlo sampling $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$

$$\frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)})$$

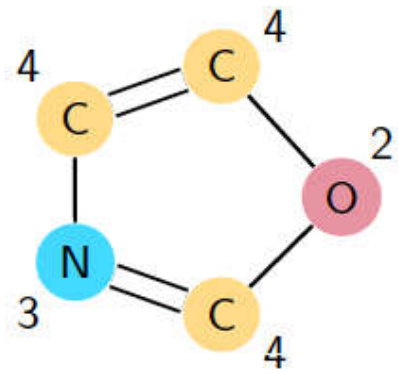
Similarly for the regularization term

$$-L_{\text{ELBO}}(\theta,\phi) + \mu \sum_i g_i(\theta,z)_+, \quad \text{where} \quad z \sim p_\theta(z)$$

Constraints

Molecules

- Valence
  - Expected node capacity (sum of edges) <= valence
- Connectivity
  - Every node pair much be connected by a path
  - B = A + A^2 + … + A^{n-1}
  - If node i and j are connected, B_{ij} != 0



Results

Compared to VAE with no regularization

QM9		
Method	% Valid	ELBO
Standard	83.2	-17.3
Regul.	96.6	-18.5

ZINC		
Method	% Valid	ELBO
Standard	29.6	-46.5
Regul.	34.9	-47.0

Compared to previous works

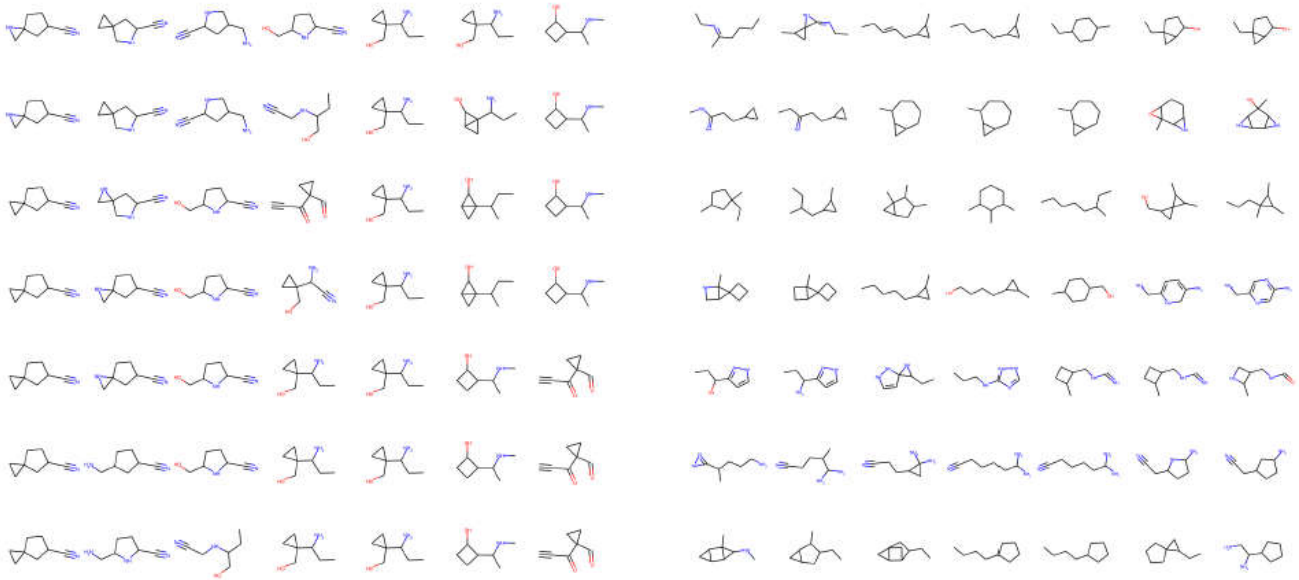
QM9			
Method	% Valid	% Novel	% Recon.
Proposed	96.6	97.5	61.8
GVAE	60.2	80.9	96.0
CVAE	10.3	90.0	3.61

ZINC			
Method	% Valid	% Novel	% Recon.
Proposed	34.9	100	54.7
GVAE	7.2	100	53.7
CVAE	0.7	100	44.6

Visualization of Generated Molecules

Left: two-dimensional interpolation

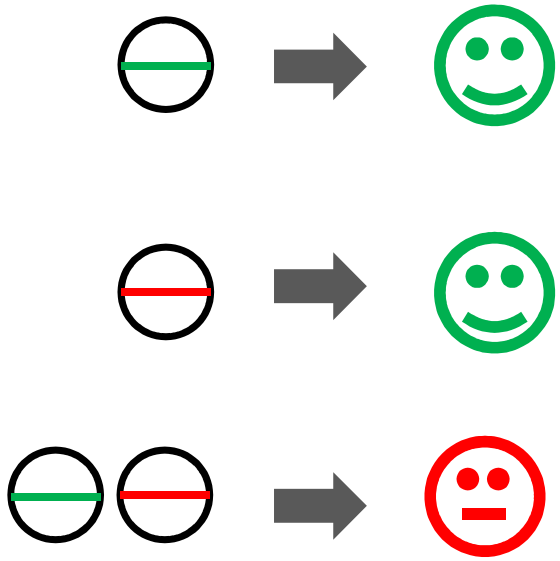
Right: one-dimensional (column-wise) interpolation



Drug Similarity Integration Through Multi-view Graph Auto-Encoders

IJCAI 2018

Adverse Drug-Drug Interaction (DDI)



Drugs may interact to cause adverse DDIs.

Common among patients with complex diseases or comorbidities.

Hard to observe in clinical testing.

Affects 15% U.S. population. Cost more than $177 billion per year in disease management.

DDI Prediction

Drug Features (database)

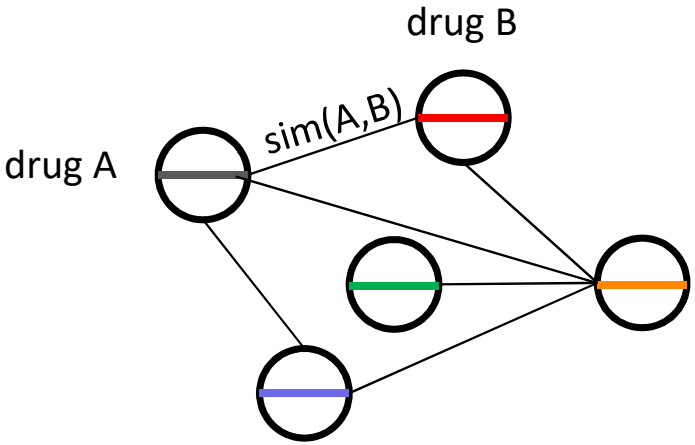
- Label Side Effect (SIDER)
- Off-Label Side Effect (OFFSIDES)
- Molecular substructure
- Drug Indication (MedDRA)
- ……

Assumption: similar drugs may have similar interaction to another drug.

Related to Graphs?

- Constructing a DDI graph, we can predict unknown DDIs as a link prediction problem.
- Constructing a similarity graph, where DDIs are regarded as node labels, and DDI prediction is a node classification problem.

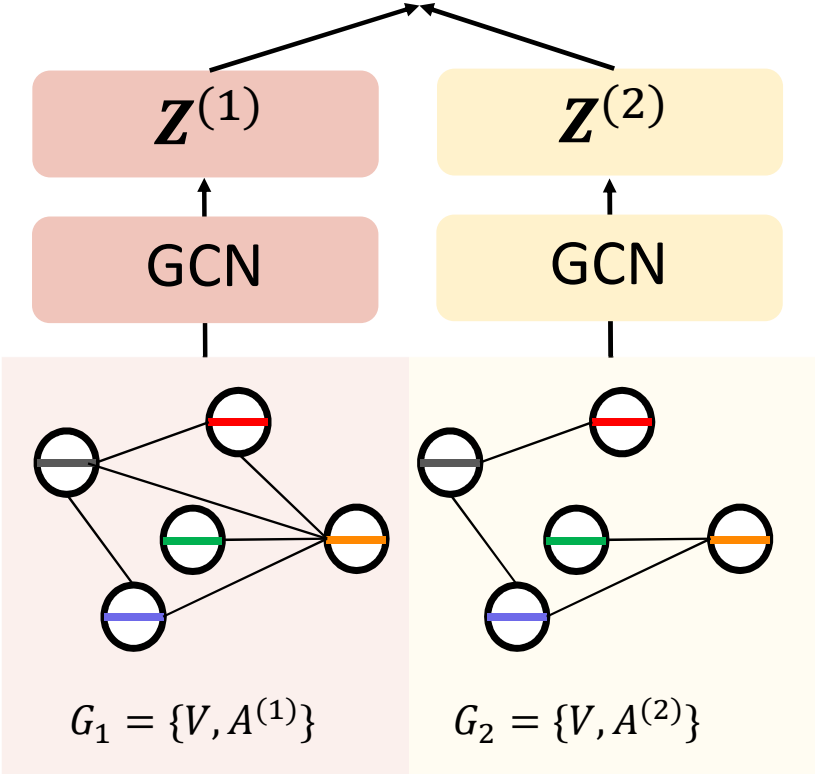
MV-GAE: Drug Similarity Integration Through Multi-view Graph Auto-Encoders (IJCAI 2018)



- Challenges of multi-view learning
  - the underlying relations of biomedical events are often nonlinear and complex over all types of features
  - features have different importance toward different target outcomes
- Our solution:
  - Construct drug similarity graph for each view, where DDIs are node labels
  - Graph convolutional network (GCN) based model for node embedding and prediction
  - Attention mechanism to integrate different views

A Simple Multi-View GCN

$$Y = softmax(W[\boldsymbol{Z}^{(1)}, \dots, \boldsymbol{Z}^{(T)}] + \boldsymbol{b})$$



Embedding

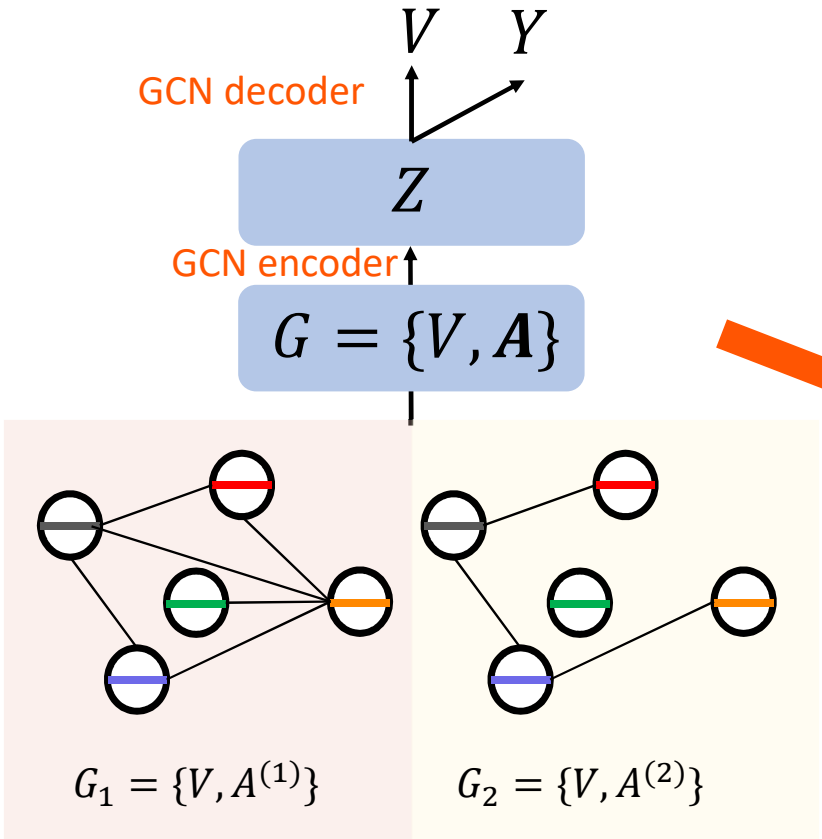
For view $u \in \{1,\dots,T\}$, we encode the nodes as

$$\boldsymbol{Z}^{(u)} = f\left(\boldsymbol{X}^{(u)}, \boldsymbol{A}^{(u)}; \boldsymbol{W}_1^{(u)}\right)$$
$$= softmax(\widehat{\boldsymbol{A}}^u ReLU(\widehat{\boldsymbol{A}}^u \boldsymbol{X}^{(u)} \boldsymbol{W}_0^{(u)}) \boldsymbol{W}_1^{(u)})$$

where $\widehat{\boldsymbol{A}}^u = \widetilde{\boldsymbol{D}}^{(u)-\frac{1}{2}} \widetilde{\boldsymbol{A}}^{(u)} \widetilde{\boldsymbol{D}}^{(u)-\frac{1}{2}}$, and $\boldsymbol{W}_0^{(u)}$ and $\boldsymbol{W}_1^{(u)}$ are weight matrices.

Question: what if we do not have labels?

Attentive Multiview Similarity Fusion with Graph Auto-Encoders (GAE)



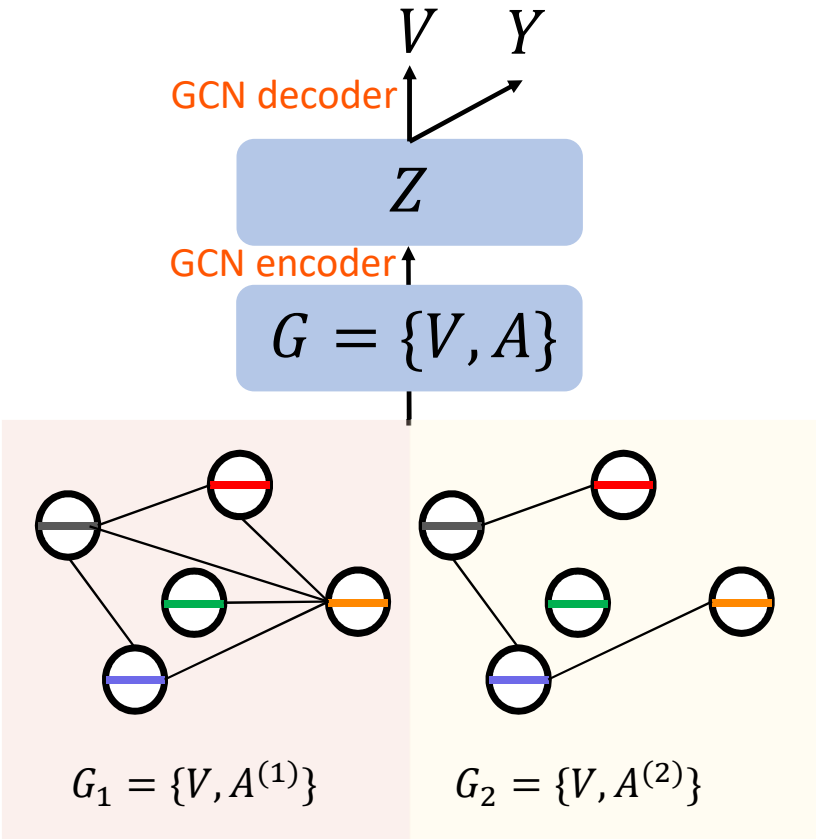
When we do not have labels, we reconstruct $\boldsymbol{V}$ from $\boldsymbol{Z}$ to minimize autoencoder loss, $\mathrm{L}_{ed} = \sum|\boldsymbol{X} - \boldsymbol{X}'|^2$.

$$A = \sum_u diag(g^u) * A^u$$

$$g'^u = W^u A^u + b^u \xrightarrow{\text{Normalize}} g^u$$

attention weights decided by data and target

Semi-Supervised Extensions (SemiGAE)



SemiGAE (for partial labels)

Objective: $\min L = L_{train} + L_{ed}$

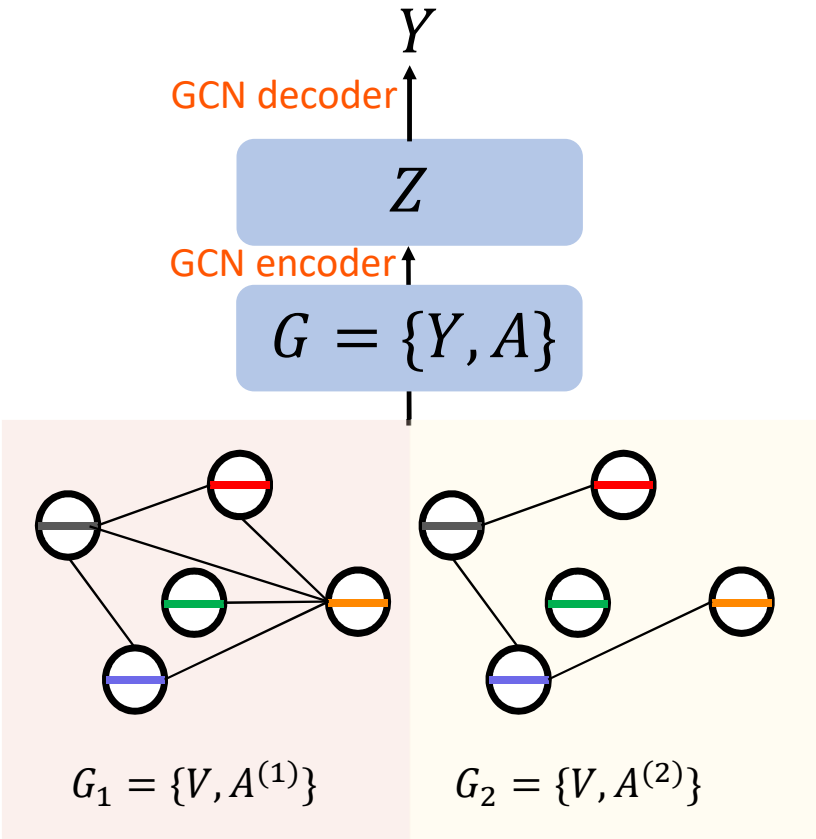
Training loss for labeled data

$$L_{train} = \sum_{y \in Y_{train}} \sum_{y' \in h(Z_{train})} -y \ln y'$$

Auto-encoder loss for all data

$$L_{ed} = \sum |X - X'|^2$$

What if we do not have node features?



TransGAE (for lack of node features)

For the case when we do not have node feature

$$\min L = |Y'_{train} - Y_{train}|^2 + |Y'_{test} - Y_{test}|^2 + \mu |Y_{test}|^2$$

where we also treat DDI label as input variable

$$\{Y'_{train}, Y'_{test}\} = f'(f(\{Y_{train}, Y_{test}\}, \hat{A}), \hat{A})$$

Results

Table 2: Predicting Specific DDI Types (Multiple Outcomes) on Dataset 2.

	Methods	Test Split (25%)		Test Split (50%)	
		ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
Using Single View					
Baselines	NN	0.627 ± 0.043	0.594 ± 0.078	0.594 ± 0.033	0.554 ± 0.061
	LP	0.773 ± 0.025	$\mathbf{0.670 \pm 0.052}$	0.747 ± 0.028	$\mathbf{0.650 \pm 0.053}$
	GraphCNN	0.738 ± 0.047	0.594 ± 0.080	0.698 ± 0.090	0.583 ± 0.102
Proposed	SemiGAE	$\mathbf{0.798 \pm 0.029}$	0.661 ± 0.059	$\mathbf{0.784 \pm 0.028}$	0.649 ± 0.059
	TransGAE	0.790 ± 0.028	0.661 ± 0.068	0.770 ± 0.031	0.633 ± 0.080
Using Multiple Views					
Baselines	LP	0.774 ± 0.025	0.672 ± 0.052	0.748 ± 0.028	0.653 ± 0.055
	GraphCNN	0.601 ± 0.067	0.526 ± 0.120	0.578 ± 0.067	0.526 ± 0.108
	MKL	0.766 ± 0.030	0.650 ± 0.061	0.724 ± 0.026	0.586 ± 0.066
Proposed	AttSemiGAE	$\mathbf{0.802 \pm 0.029}$	$\mathbf{0.678 \pm 0.060}$	$\mathbf{0.786 \pm 0.030}$	$\mathbf{0.662 \pm 0.064}$
	AttTransGAE	0.782 ± 0.026	0.670 ± 0.058	0.764 ± 0.025	0.652 ± 0.061

Analysis of Attention

		Attention Weights			
DDI Type	AUC	Chem.	indi.	TTDS	CPI
Chest Pain	0.772	0.151	0.303	0.144	0.402
Insomnia	0.755	0.380	0.261	0.078	0.291
indication	0.774	0.117	0.301	0.283	0.299

Results

Views “indication” and “CPI” receive high weights for the ADR “Chest pain”.

Graph-Enhanced Medication Recommendation: GAMENet and G-BERT

AAAI 2019, IJCAI 2019.

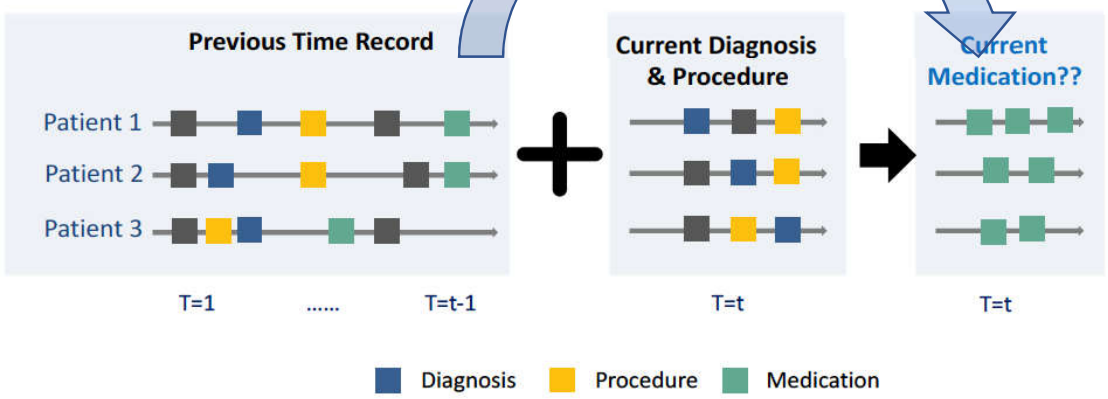
EHR Phenotyping and Medication Recommendation

EHR (electronic healthcare record):

- Representation $\{\mathcal{X}_1^{(n)}, \mathcal{X}_2^{(n)}, \cdots, \mathcal{X}_{T^{(n)}}^{(n)}\}$ $\mathcal{X}_t = \mathcal{C}_d^t \cup \mathcal{C}_m^t$
- Phenotyping is important for
  - Disease prediction
  - **Medication recommendation**
  - Readmission prediction

What if there is a ***DDI***?

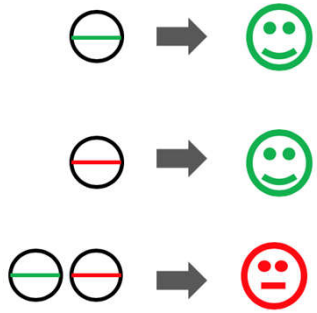
Mec



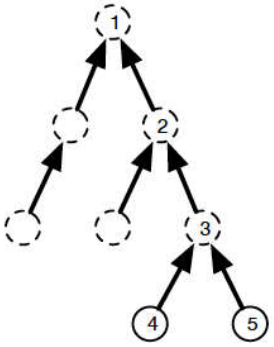
Challenges

Problems of previous approaches

- Without consideration of DDIs
  - It is possible to recommend some un-safe drugs
- Lack of structure information for medical codes
- Throwing out a lot of resources
  - The single-visit EHR sequences
    - -> we cannot use it in RNN
    - ->we cannot use it to predict the next visit



Drugs may interact to cause adverse DDIs.



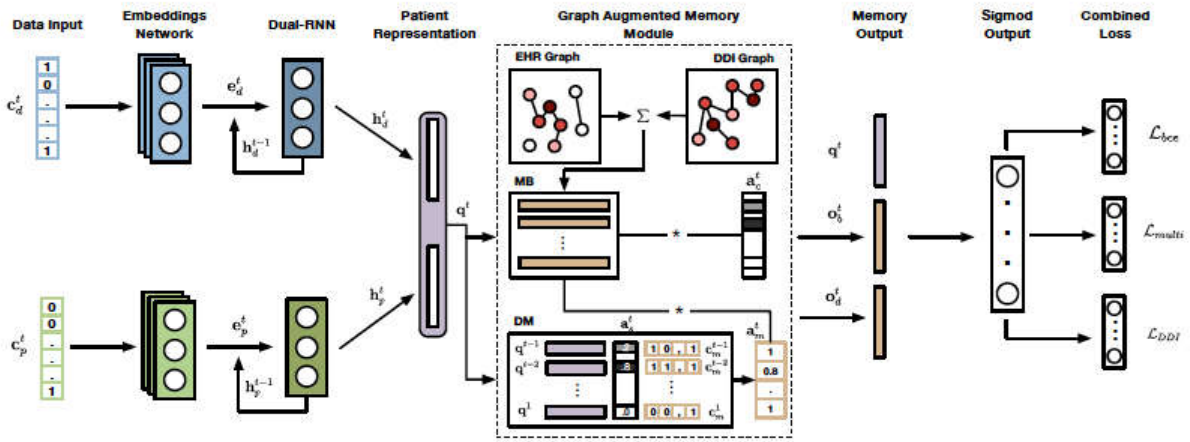
index	icd-9	description
1	root	root
2	420-429	Other forms of heart disease
3	428	Heart failure
4	428.0	LEFT HEART FAILURE
5	428.1	CONGESTIVE HEART FAILURE

Figure 1: Graphical illustration of ICD-9 Ontology.

Medical Recommendation I: GAMENet

GAMENet: Graph Augmented MEmory Networks for Recommending Medication Combination (Shang et al. 2019a)

- Key ideas: integrate the DDI graphs to provide safer medical recommendation
- Method: encode both EHRs and DDI graphs in the memory and impact on the memory output



Graph Augmented Memory Module

I: Input memory representation converts inputs into query for memory reading.

- RNN

G: Generalization is the process of generating and updating the memory representation.

- Static memory bank M_b: GCN
- Dynamic memory M_d : adding key-value pair

O: Output memory representation produces outputs given the patient representation (the query) and the current memory state M_b and M_d.

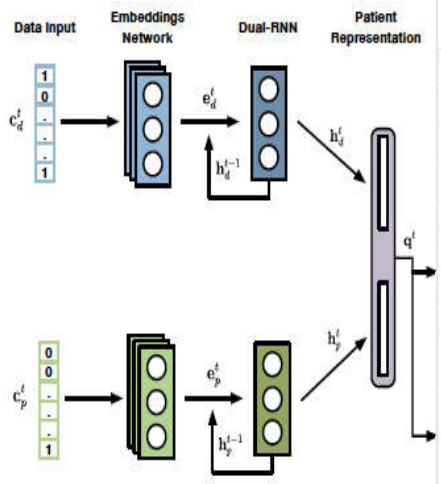
$$a_c^t = \text{Softmax}(M_b q^t) \qquad a_s^t = \text{Softmax}(M_{d,k}^t q^t)$$

$$o_b^t = M_b a_c^t \qquad a_m^t = M_{d,v}^{t\,\mathrm{T}} a_s^t$$

$$o_d^t = M_b^{\mathrm{T}} a_m^t$$

R: Response is the final step to utilize patient representation and memory output to predict the multilabel medication.

$$\hat{y}_t = \sigma([q^t, o_b^t, o_d^t])$$



Graph Augmented Memory Module

I: Input memory representation converts inputs into query for memory reading.

- RNN

G: Generalization is the process of generating and updating the memory representation.

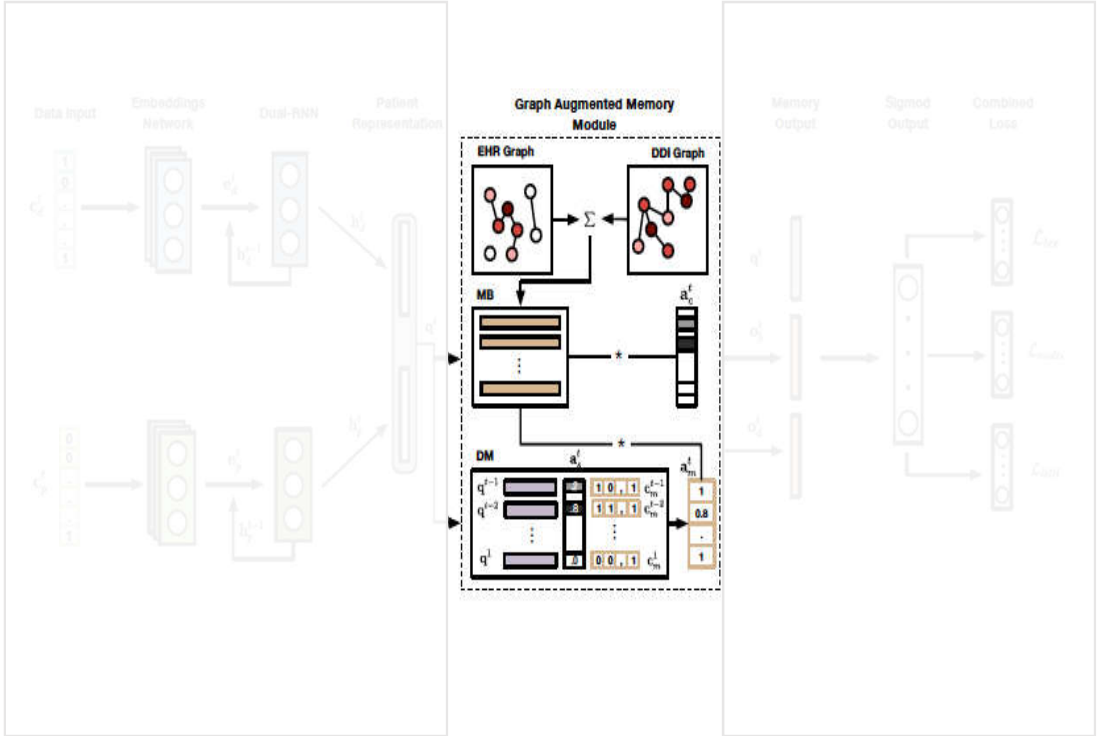
- Static memory bank M_b: GCN
- Dynamic memory M_d : adding key-value pair

O: Output memory representation produces outputs given the patient representation (the query) and the current memory state M_b and M_d.

$$a_c^t = \mathrm{Softmax}(M_b q^t)$$
$$o_b^t = M_b a_c^t$$
$$a_s^t = \mathrm{Softmax}(M_{d,k}^t q^t)$$
$$a_m^t = M_{d,v}^{t\,\mathrm{T}} a_s^t$$
$$o_d^t = M_b^{\mathrm{T}} a_m^t$$

R: Response is the final step to utilize patient representation and memory output to predict the multilabel medication.

$$\hat{y}_t = \sigma([q^t, o_b^t, o_d^t])$$



Graph Augmented Memory Module

I: Input memory representation converts inputs into query for memory reading.

- RNN

G: Generalization is the process of generating and updating the memory representation.

- Static memory bank M_b: GCN
- Dynamic memory M_d : adding key-value pair

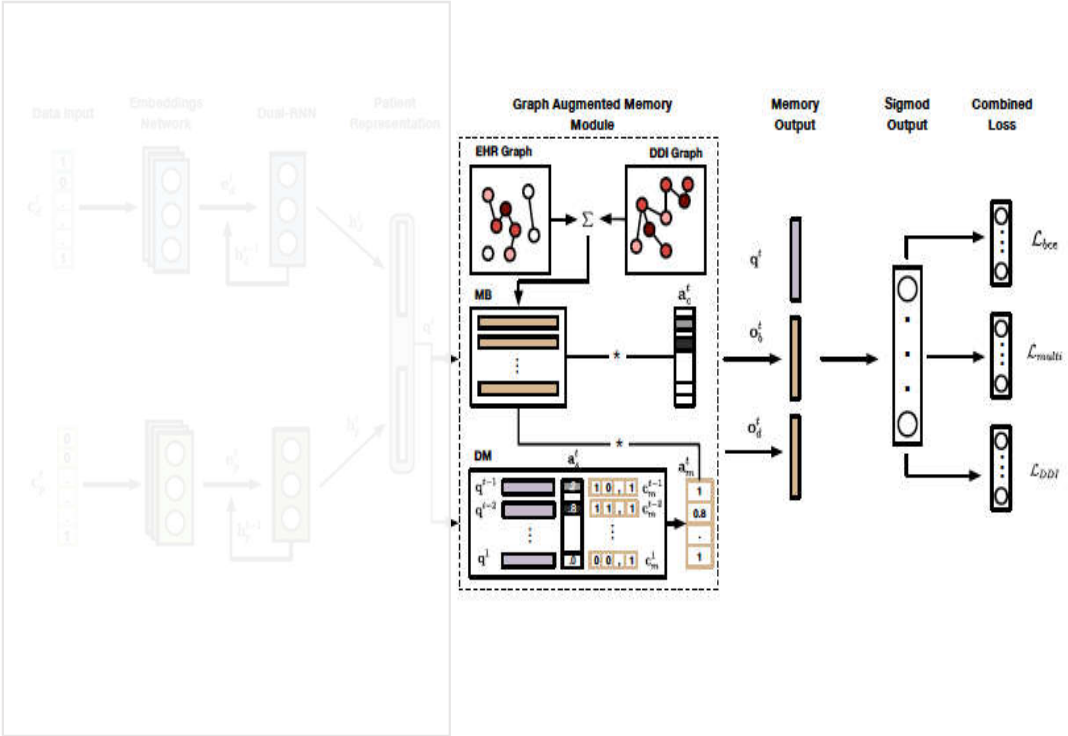
O: Output memory representation produces outputs given the patient representation (the query) and the current memory state M_b and M_d.

$$\begin{aligned} \boldsymbol{a}_c^t &= \mathrm{Softmax}(\boldsymbol{M}_b \boldsymbol{q}^t) \\ \boldsymbol{o}_b^t &= \boldsymbol{M}_b \boldsymbol{a}_c^t \end{aligned}$$

$$\begin{aligned} \boldsymbol{a}_s^t &= \mathrm{Softmax}(\boldsymbol{M}_{d,k}^t \boldsymbol{q}^t) \\ \boldsymbol{a}_m^t &= \boldsymbol{M}_{d,v}^t{}^{\mathrm{T}} \boldsymbol{a}_s^t \\ \boldsymbol{o}_d^t &= \boldsymbol{M}_b^{\mathrm{T}} \boldsymbol{a}_m^t \end{aligned}$$

R: Response is the final step to utilize patient representation and memory output to predict the multilabel medication.

$$\hat{\boldsymbol{y}}_t = \sigma([\boldsymbol{q}^t, \boldsymbol{o}_b^t, \boldsymbol{o}_d^t])$$



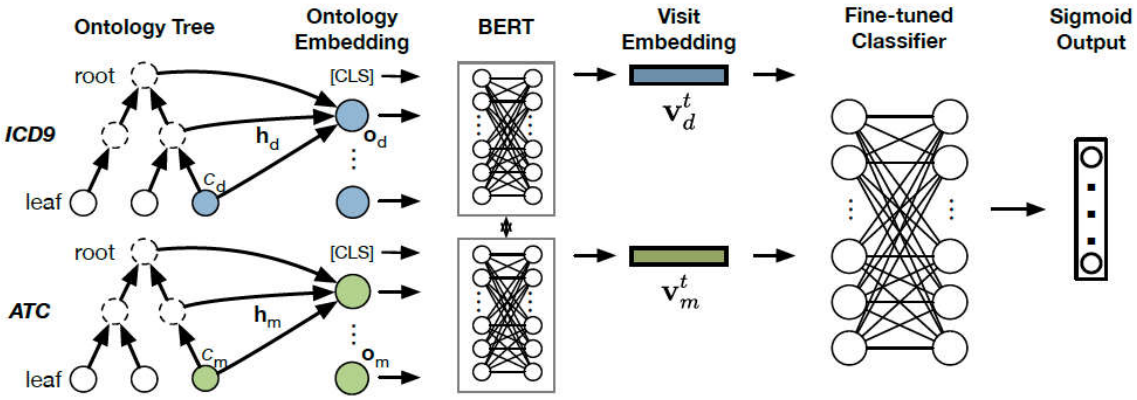
Medical Recommendation II: GBERT (Graph + Pre-training)

Motivation

- To utilize the hierarchy information of medical codes
- To pretrain on single-visit data (which was generally discarded in previous systems)

Framework:

- ontology embedding -> visit embedding -> pre-training



Ontology Embedding

A modified GNN to model the ontology and get initial embeddings for all medical codes

- Leaf to root

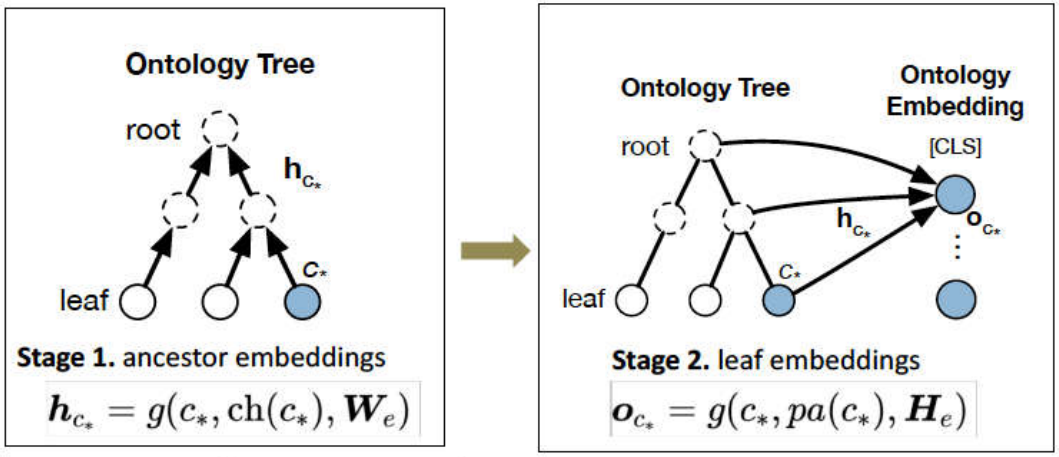
$$\boldsymbol{h}_{c_*} = g(c_*, ch(c_*), \boldsymbol{W}_e)$$

- Root to leaf

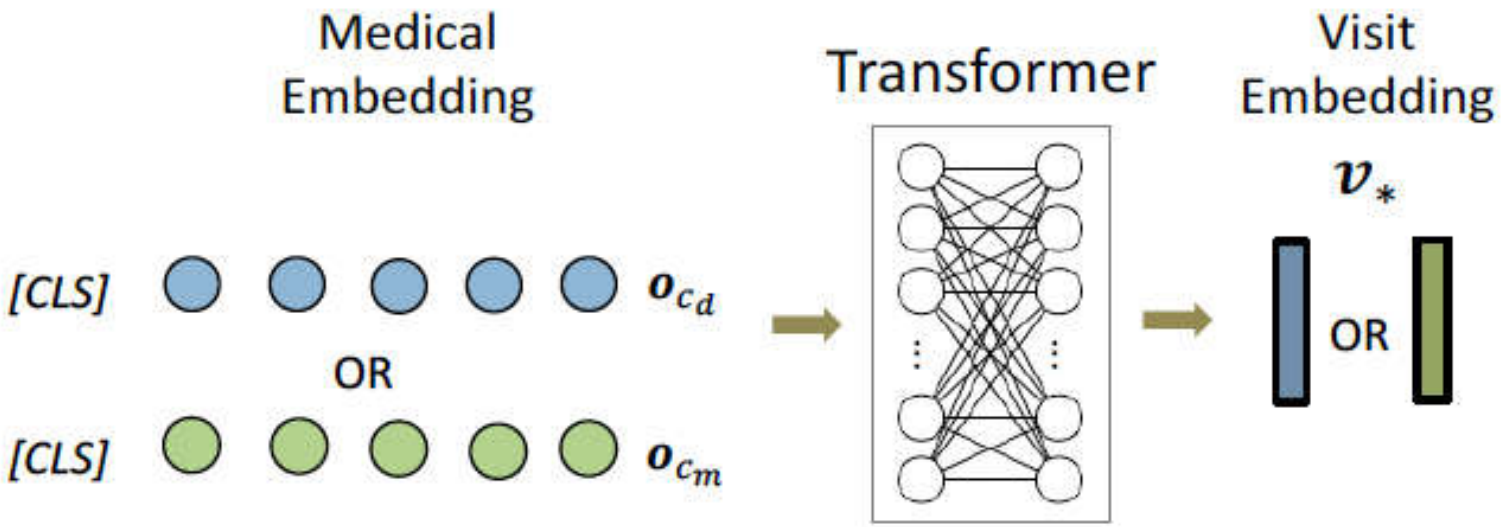
$$\boldsymbol{o}_{c_*} = g(c_*, pa(c_*), \boldsymbol{H}_e)$$

- GAT style message passing

$$g(c_*, p(c_*), \boldsymbol{H}_e) = \mathop{\Big\|}_{k=1}^{K} \sigma\left(\sum_{j \in \{c_*\} \cup pa(c_*)} \alpha_{i,j}^k \boldsymbol{W}^k \boldsymbol{h}_j\right)$$



Visit Embedding



$$\boldsymbol{v}_*^t = \text{Transformer}\left(\{[CLS]\} \cup \{\boldsymbol{o}_{c.}^t | c_* \in \mathcal{C}_*^t\}\right)[0]$$

[CLS]: Special token in BERT

Pre-training

Input example:

- [CLS] d1, d2, mask d4, m1, mask m3
- d: diagnosis; m: medication

pre-training on each visit of EHR sequences

- Self-prediction: same as BERT, use to a mask to mask out some codes and predict them
- Dual-prediction: known all diagnosis, predict medication; known medication, predict diagnosis
- Note: No position embedding, because there is no order within one visit.

Results

- We used EHR data from MIMIC-III [Johnson et al., 2016] and conducted all our experiments on a cohort where patients have more than one visit.
- For GAMENet, we used DDI knowledge from TWOSIDES dataset.
- For G-BERT, we utilize data from patients with both single visit and multiple visits in the training dataset as pre-training data source.

Methods	Jaccard	PR-AUC	F1	# of parameters
LR	0.4075	0.6716	0.5658	-
GRAM	0.4176	0.6638	0.5788	3,763,668
LEAP	0.3921	0.5855	0.5508	1,488,148
RETAIN	0.4456	0.6838	0.6064	2,054,869
GAMENet^{-}	0.4401	0.6672	0.5996	5,518,646
GAMENet	0.4555	0.6854	0.6126	5,518,646
G-BERT$_{G-,P-}$	0.4186	0.6649	0.5796	2,634,145
G-BERT$_{G-}$	0.4299	0.6771	0.5903	2,634,145
G-BERT$_{P-}$	0.4236	0.6704	0.5844	3,034,045
G-BERT	**0.4565**	**0.6960**	**0.6152**	3,034,045

Table 3: Performance on Medication Recommendation Task.

MICHIGAN STATE UNIVERSITY IBM

Thank you!

Welcome to my other presentations:
Oral
"**Online Planner Selection with Graph Neural Networks and Adaptive Scheduling**"
and poster
"**EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs**"
Both on **Feb 10th**.

Tutorial Overview

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

→

Models

- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

→

Applications

- Healthcare
- Natural Language Processing
- Recommendations

Graph-to-Sequence Learning in Natural Language Processing

Lingfei Wu
IBM Research AI
IBM T. J. Watson Research Center

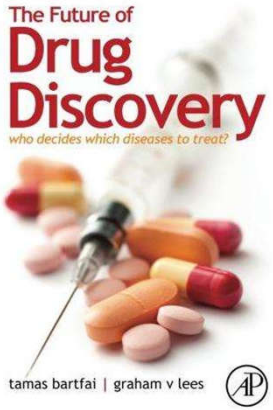
Joint work with Kun Xu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin

@AAAI 2020 Tutorial

Seq2Seq: Applications and Challenges

Applications

- Machine translation
- Natural Language Generation
- Logic form translation
- Drug Discovery



Challenges

- Only applied to problems whose inputs are represented as sequences
- Cannot handle more complex structure such as graphs
- Converting graph inputs into sequences inputs lose information
- Augmenting original sequence inputs with additional structural information enhances word sequence feature

Contributions and Highlighted Research

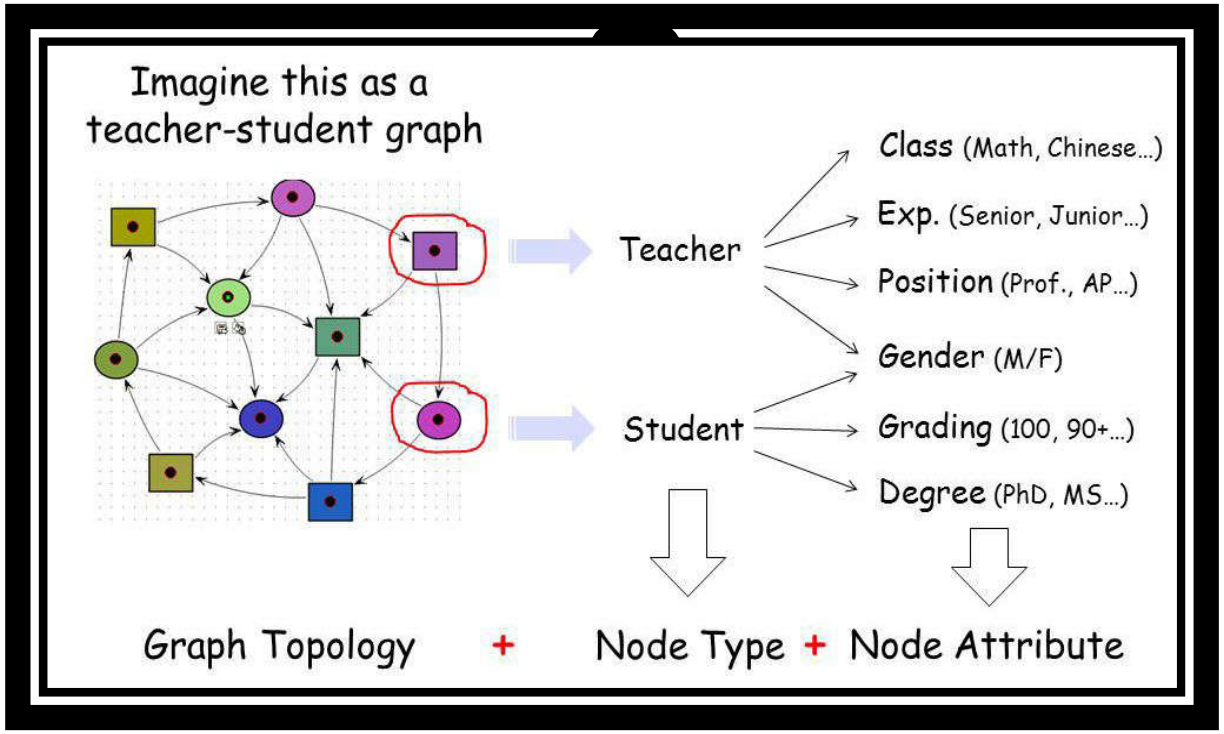
Fundamental contributions in this research:

- Presented Graph2Seq, a generalized seq2seq model for graph inputs
- Attention-based encoder-decoder model for graph-to-sequence learning

- Two highlighted NLP tasks using Graph2Seq model:
  - SQL-to-text Generation with Graph2Seq Model
  - Semantic Parsing by exploiting rich syntactic information with Graph2Seq Model

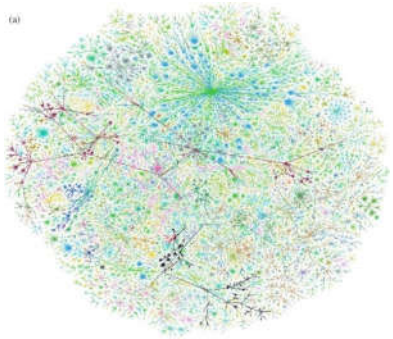
Graphs: A Universal Language

Graphs are a general language for describing and modeling complex systems



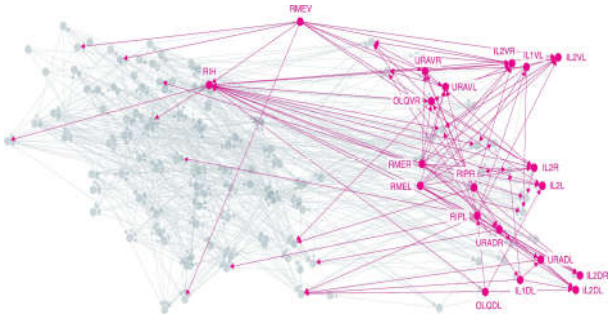
Graph!

Graph-structured Data Are Ubiquitous

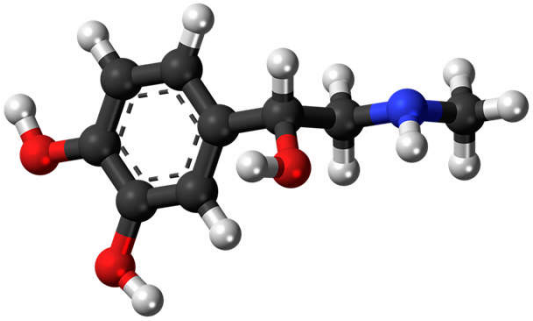


Internet

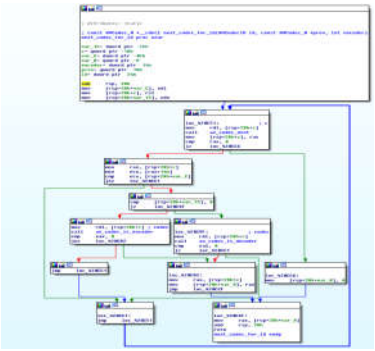
Social networks



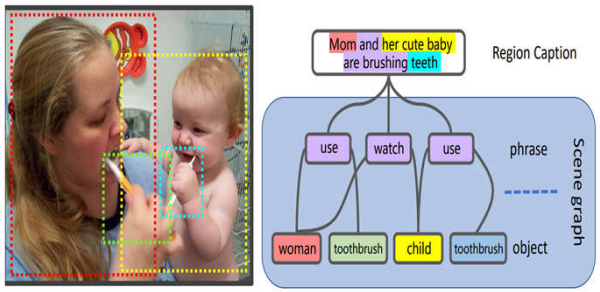
Networks of neurons



Biomedical graphs



Program graphs



Scene graphs

Why Graphs? Why Now?

Universal language for describing complex data

- Networks/graphs from science, nature, and technology are more similar than one would expect

Shared vocabulary between fields

- Computer Science, Social science, Physics, Biology, Economics

Data availability (+ computational challenges)

- Social/internet, text, logic, program, bio, health, and medical

Impact

- Social networking, Social media, Drug design, Event detection, Natural language processing, Computer vision, and Logic reasoning

Machine(Deep) Learning with Graphs

Classical ML tasks in graphs:

Node classification

- Predict a type of a given node

Link prediction

- Predict whether two nodes are linked

Community detection

- Identify densely linked clusters of nodes

Graph similarity

- How similar are two (sub)graphs

Recent ML tasks in graphs:

- Graph classification
  - Predict a type of a given graph
- Graph generation
  - Generate graphs from learned distribution
- Graph structure learning
  - Identify densely linked clusters of nodes
- Graph-to-XXX learning
  - Graph Inputs – XXX outputs

Graph Representation Learning (GNNs)

Graph Neural Networks (GNNs) extends the well known CNN and RNN on graphs, from Euclidean data to Graphs and Manifolds

RNN-based GNNs:

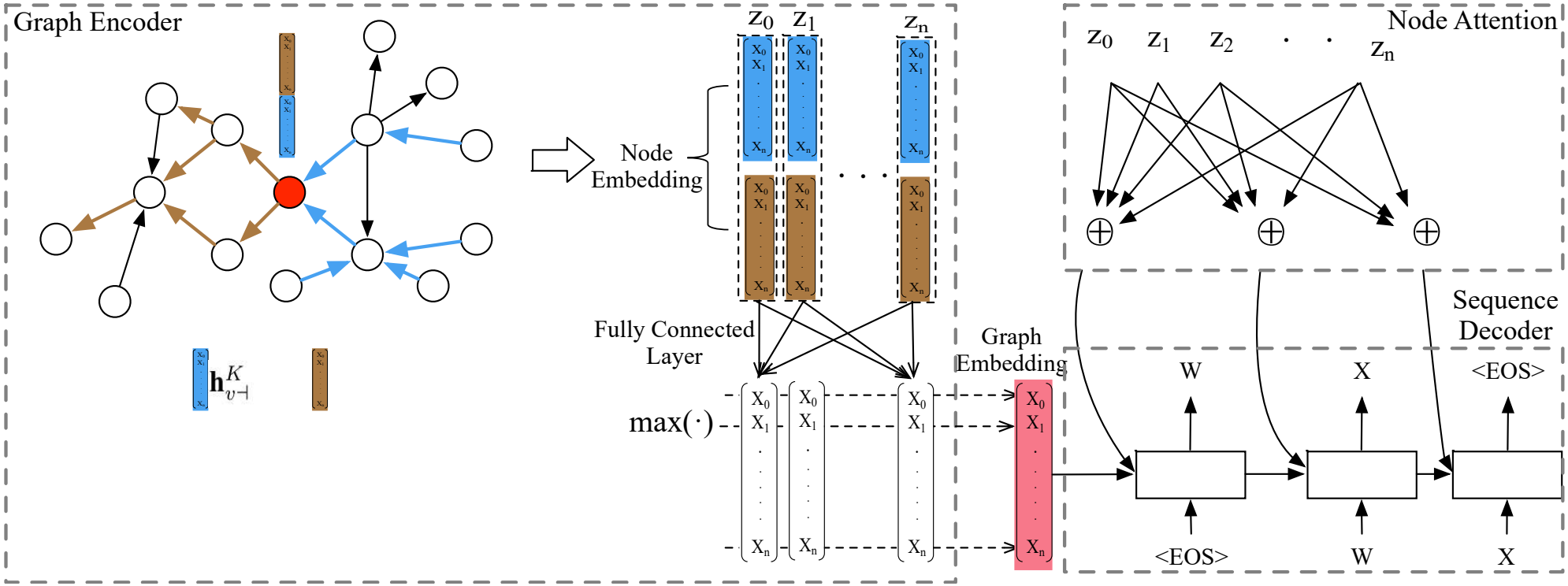
- Graph neural networks (Scarselli et al., 2009)
- Gated graph sequence neural networks (GGS-NNs) (Li et al., ICLR 2016)

CNN-based GNNs:

- Graph Convolutional Networks (GCN) (Kipf & Welling, ICLR 2017)

- Message Passing-based GNNs:
  - GraphSAGE (Hamilton & Ying & Leskovec, NIPS 2017)
  - Graph Attention Networks (GAT) (Velickovic et al., ICLR 2018)
  - MPNN (Gilmer et al., ICML 2017)

Graph-to-Sequence Model [1]



[1] Kun Xu*, Lingfei Wu*, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin (equally contributed), "Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks", arXiv preprint 2018.

Graph Encoding

Bidirectional Node embedding (take node *v* as an example)

① transform each node's text attribute to a feature vector by looking up the embedding matrix

$$\mathbf{h}^0_{v\vdash} = \mathbf{a}_v, \mathbf{h}^0_{\vdash v} = \mathbf{a}_v, \forall v \in \mathcal{V}$$

② classify *v*'s neighbors into ***forward*** and ***backward*** neighbors, aggregate neighbors information using a fully connect network followed by a max pooling operation

$$\mathbf{h}^k_{\mathcal{N}_\vdash(v)} = \mathbf{M}^k_\vdash(\{\mathbf{h}^{k-1}_{u\vdash}, \forall u \in \mathcal{N}_\vdash(v)\})$$
$$\mathbf{h}^k_{v\vdash} = \sigma(\mathbf{W}^k \cdot \mathrm{CONCAT}(\mathbf{h}^{k-1}_{v\vdash}, \mathbf{h}^k_{\mathcal{N}_\vdash(v)}))$$

$$\mathbf{h}^k_{\mathcal{N}_\dashv(v)} = \mathbf{M}^k_\dashv(\{\mathbf{h}^{k-1}_{u\dashv}, \forall u \in \mathcal{N}_\dashv(v)\})$$
$$\mathbf{h}^k_{v\dashv} = \sigma(\mathbf{W}^k \cdot \mathrm{CONCAT}(\mathbf{h}^{k-1}_{v\dashv}, \mathbf{h}^k_{\mathcal{N}_\dashv(v)}))$$

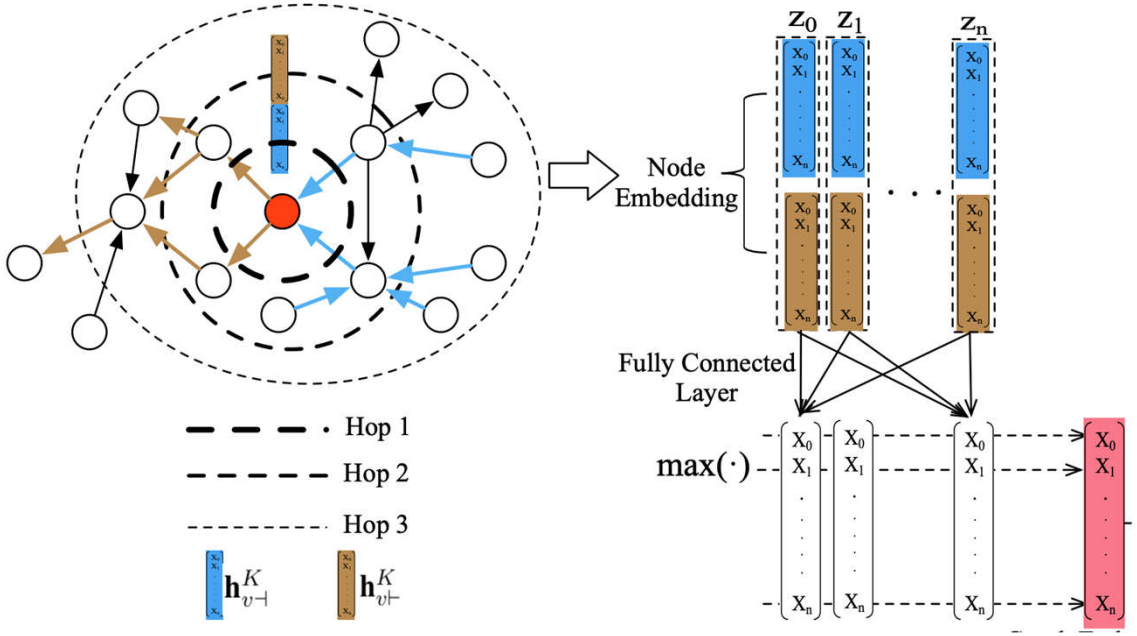
③ repeat steps 2 for *K* times to aggregate neighbors information in *K* hops

④ concatenate final *v*'s ***forward*** and ***backward*** node embeddings as the final bi-directional representation of node *v*

Graph Encoding

Graph embedding

- Pooling based graph embedding (*max, min and average pooling*)
- Node based graph embedding
  - Add one super node which is connected to all other nodes in the graph
  - The embedding of this super node is treated as graph embedding



Attention Based Sequence Decoding

$$c_i = \sum_{j=1}^{\mathcal{V}} \alpha_{ij} h_j, \; where \; \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathcal{V}} \exp(e_{ik})}, \; e_{ij} = a(s_{i-1}, h_j)$$

context vector

node representation

Attention Based Sequence Decoding

$$c_i = \sum_{j=1}^{\mathcal{V}} \alpha_{ij} h_j, \; where \; \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathcal{V}} \exp(e_{ik})}, \; e_{ij} = a(s_{i-1}, h_j)$$

context vector

node representation

attention weights

alignment model

Attention Based Sequence Decoding

$$c_i = \sum_{j=1}^{\mathcal{V}} \alpha_{ij} h_j, \; where \; \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathcal{V}} \exp(e_{ik})}, \; e_{ij} = a(s_{i-1}, h_j)$$

context vector

node representation

attention weights

alignment model

Objective Function

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n | y_{<t}^n, x^n)$$

Experiments: Text Reasoning and Shortest Path

garden (A) bathroom (B) bedroom (C)
hallway (D) office (E) kitchen (F)

1 The **garden** is west of the **bathroom**.
2 The **bedroom** is north of the **hallway**.
3 The **office** is south of the **hallway**.
4 The **bathroom** is north of the **bedroom**.
5 The **kitchen** is east of the **bedroom**.

Transform →

A west B
B north D
E south D
B north C
F east C

Q: How do you go from the **bathroom** to the **hallway**

Transform →

Q:path(B, D)

	bAbI T19	SP-S	SP-L
LSTM	25.2%	8.1%	2.2%
GGS-NN	98.1%	100.0%	95.2%
GCN	97.4%	100.0%	96.5%
Graph2Seq	**99.9%**	100.0%	**99.3%**

Experiments: Model Variations

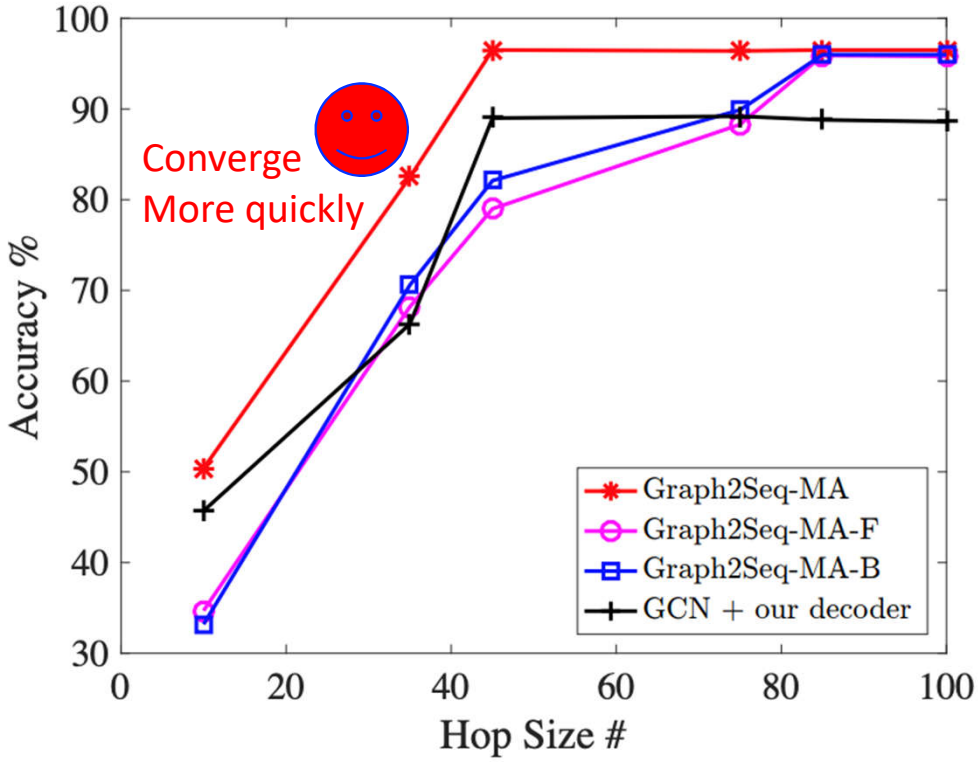
Method	SDP_{DAG}	SDP_{DCG}	SDP_{SEQ}
G2S-MA	**99.8%**	**99.2%**	**100%**
G2S-LA	91.7%	90.9%	99.9%
G2S-PA	96.7%	98.4%	**100%**
G2S-MA-F	78.8%	98.7%	70.2%
G2S-MA-B	80.1%	99.1%	68.6%

Node Aggregation Schemes

Forward or Backward Node Embedding

DAG: directed acyclic graphs; DCG: directed cyclic graphs; SEQ: essentially sequential lines

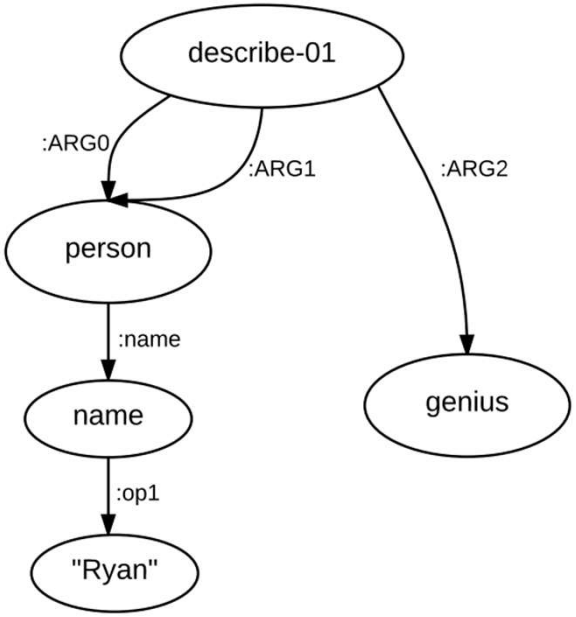
Experiments: Bidirectional Node Embedding



Bidirectional Node Embedding
VS Unidirectional Node Embedding

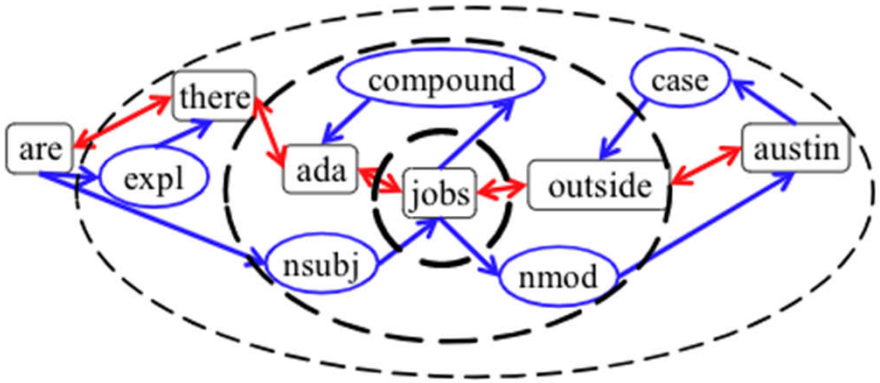
When Shall We Use Graph2Seq?

Case I: the inputs are naturally or best represented in graph



"Ryan's description of himself: a genius."

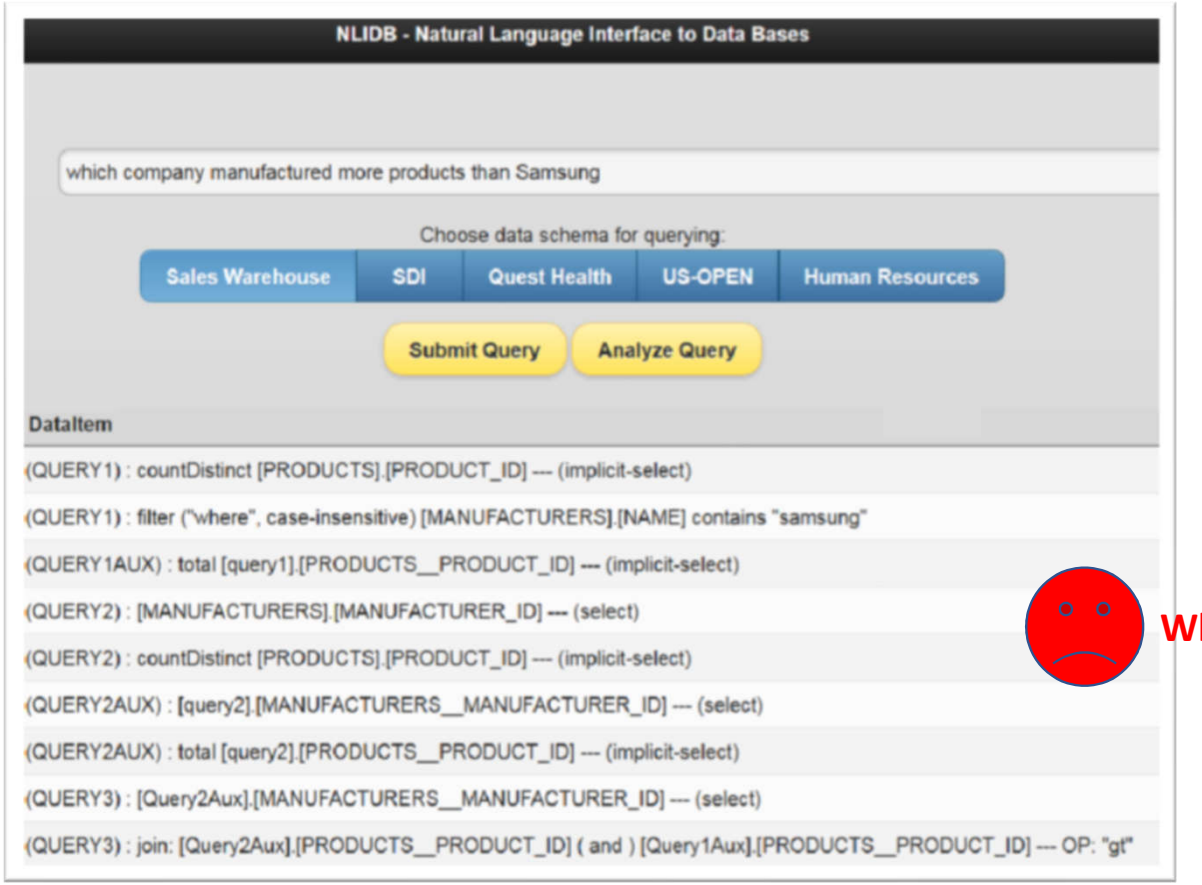
Case II: Hybrid Graph with sequence and its hidden structural information



Augmenting "are there ada jobs outside Austin" with its dependency parsing tree results

SQL-to-text Generation with Graph-to-Sequence Model (EMNLP'18)

Natural Language Interface to Database



Need explanation !

What is the meaning ?

SQL-to-text Generation Task

(SQL): **SELECT** company **WHERE** assets > val_0 **AND** sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

↓

Interpretation:

which company has both the market value and assets higher than val_0, ranking in top val_2 and revenue of val_3

Previous Approaches

Template-based approaches

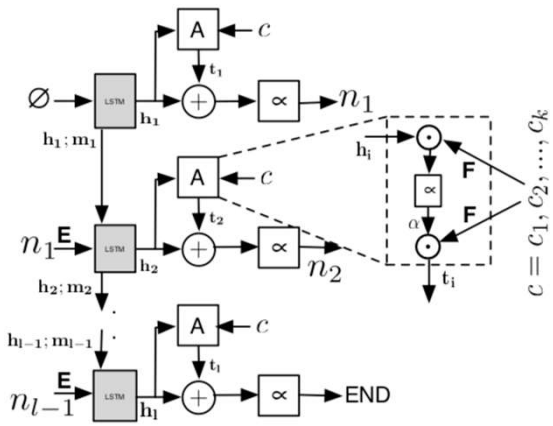
- Koutrikaal et al. 2010
- Ngonga Ngomo et al. 2013

Time consuming and generating rigid and stylized language !

Operators	Translation	Translation variables	Translation	Template labels	Description
$l(=)$	'is'	VAL_SEL	"whose"	$l(e_\sigma)$	$l(R_i)$+VAL_SEL+$l(A^i_j)$
$l(\leq)$	"does not exceed"	$COORD_CONJ$	"and"	$l(e_\theta)$	$l(A^i_j) + l(\theta) + l(\Omega)$
$l(>)$	"greater than"	$CONJ_NOUN$	"that"	$l(e_\mu)$	" the " + $l(A^i_j)$ + " of " + $l(R_i)$
$l(LIKE)$	"looks like'	$CONJ_PROJ, CONJ_SEL$	"and"	$l(R_i \xrightarrow{\sigma} A^i_j \xrightarrow{\theta} A^i_j \xrightarrow{\sigma} R_i)$	$l(R_i)$ + " with the same " + $l(A^i_j)$

Deep learning models

- Iyer et al. 2016 (***Sequence to sequence*** model)



Problem

SQL query is a graph structured query

Naïve sequence encoders may need an elaborate design to fully capture the global structure information

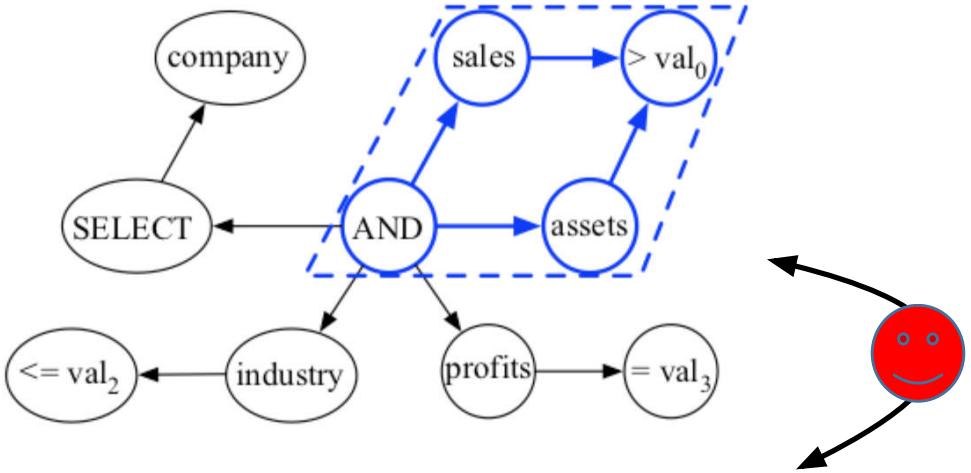
(SQL): **SELECT** company **WHERE** ***assets > val_0 AND sales > val_0*** **AND** industry_rank <= val_2 **AND** revenue = val_3

difficult to learn this mapping using naive sequence encoder

Interpretation:

which company has ***both the market value and assets higher than val_0***, ranking in top val_2 and revenue of val_3

Motivation



It seems easier to learn this mapping using the graph representation

Interpretation:

which company has ***both the market value and assets higher than val$_0$*** ranking in top val$_2$ and revenue of val$_3$

Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause

- Where clause

Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*

SELECT

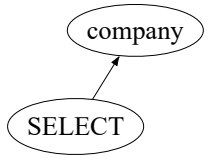
- Where clause

Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes



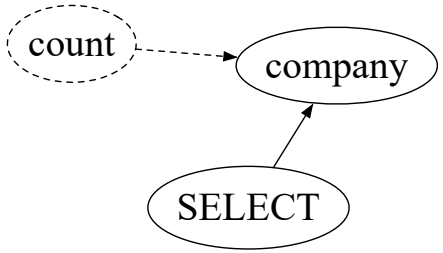
- Where clause

Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes
  c) In some cases, there may exist aggregation functions such as *count* and *max*; add aggregation node and connect it with column node
- Where clause

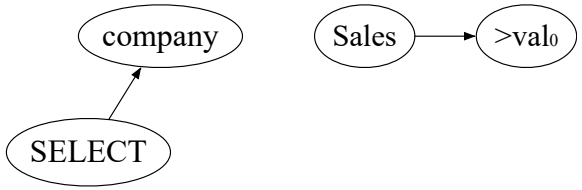


Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes
  c) In some cases, there may exist aggregation functions such as *count* and *max*; add aggregation node and connect it with column node



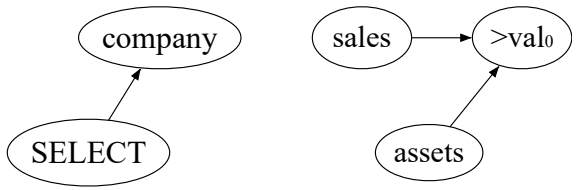
- Where clause
  a) for each condition, we use the same process as for the Select clause to create nodes

Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes
  c) In some cases, there may exist aggregation functions such as *count* and *max*; add aggregation node and connect it with column node
- Where clause
  a) for each condition, we use the same process as for the Select clause to create nodes

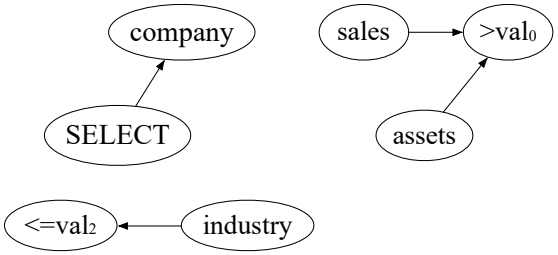


Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes
  c) In some cases, there may exist aggregation functions such as *count* and *max*; add aggregation node and connect it with column node
- Where clause
  a) for each condition, we use the same process as for the Select clause to create nodes

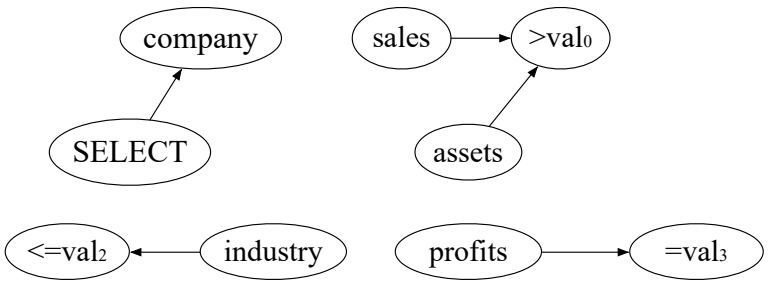


Graph Representation of SQL Query

(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes
  c) In some cases, there may exist aggregation functions such as *count* and *max*; add aggregation node and connect it with column node
- Where clause
  a) for each condition, we use the same process as for the Select clause to create nodes

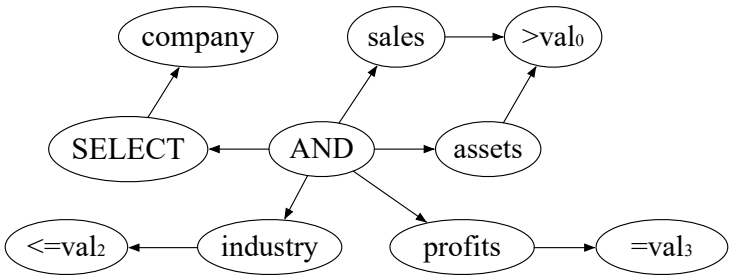


Graph Representation of SQL Query

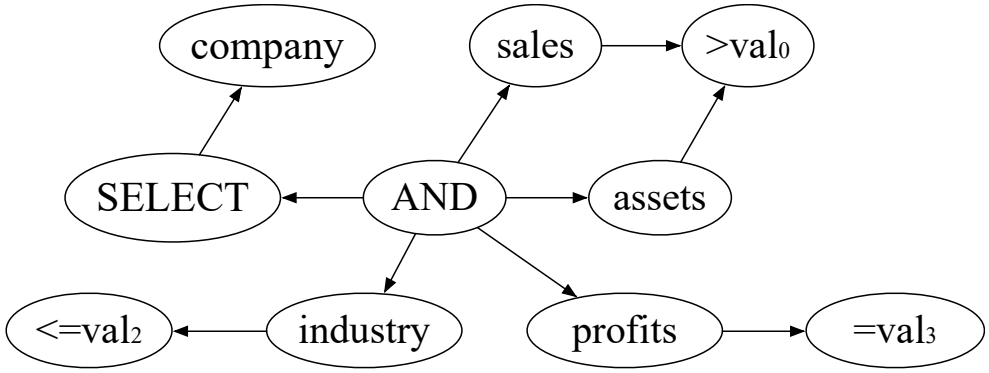
(SQL): **SELECT** company **WHERE** assets > val_0 AND sales > val_0 **AND** industry_rank <= val_2 **AND** revenue = val_3

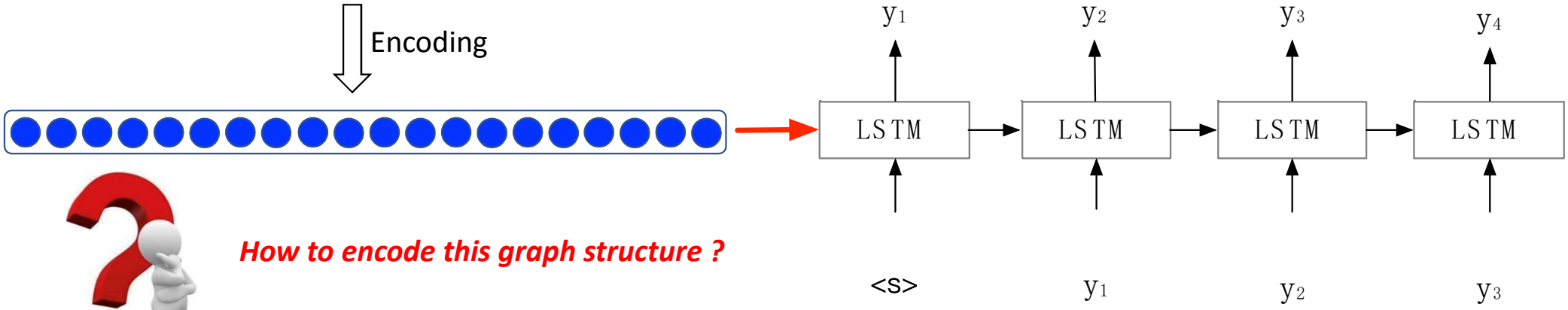
Represent the SQL query as a graph

- Select clause
  a) create a node assigned with text attribute *select*
  b) connect SELECT node with column nodes
  c) In some cases, there may exist aggregation functions such as *count* and *max*; add aggregation node and connect it with column node
- Where clause
  a) for each condition, we use the same process as for the Select clause to create nodes
  b) add logical operators such as AND, OR and NOT



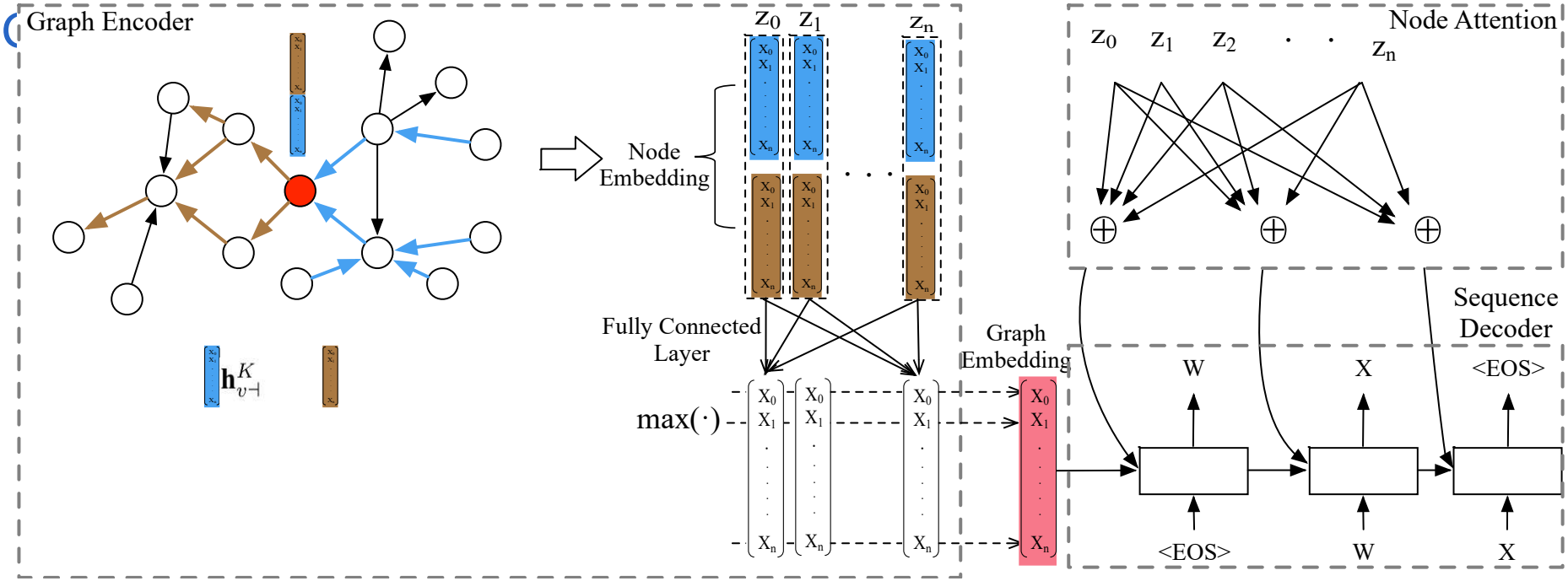
Encoder-Decoder Architecture





How to encode this graph structure ?

Graph-to-Sequence Model [1]



[1] Kun Xu*, Lingfei Wu*, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin (both authors contributed equally), "Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks", arXiv preprint 2018.

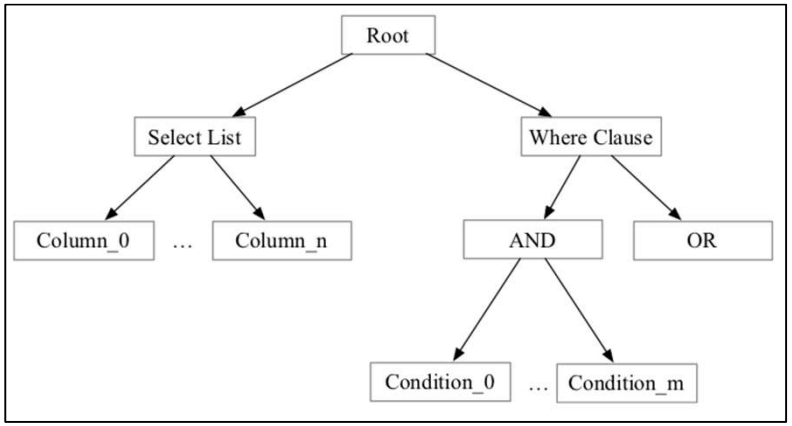
Experiments

Datasets

- WikiSQL (61,297 for training, 9,145 for development and 17,284 for test)
- Stackoverflow (25,869 for training, 3,234 for development and 3,234 for test)

Baselines

- Template
  a) first map each element of a SQL query to an utterance
  b) then use simple rules to assemble these utterances
- Seq2Seq
  We implement the model proposed in Bahdanau et al. 2014
- Seq2Seq + Copy
  We implement the model proposed in Gu et al. 2016
- Tree2Seq
  We implement the model proposed in Eriguchi et al. 2016



Results

Criteria

- BLEU-4
- Grammar (human evaluation)
- Correct (human evaluation)

	BLEU-4	Grammar.	Correct.
Template	15.71	1.50	-
Seq2Seq	20.91	2.54	62.1%
Seq2Seq + Copy	24.12	2.65	64.5%
Tree2Seq	26.67	2.70	66.8%
Graph2Seq-PGE	**38.97**	**3.81**	**79.2%**
Graph2Seq-NGE	34.28	3.26	75.3%
(Iyer et al., 2016)	18.4	3.16	64.2%
Graph2Seq-PGE	**23.3**	**3.23**	**70.2%**
Graph2Seq-NGE	21.9	2.97	65.1%

WikiSQL

Stack overflow

Results

	BLEU-4	Grammar.	Correct.
Template	15.71	1.50	-
Seq2Seq	20.91	2.54	62.1%
Seq2Seq + Copy	24.12	2.65	64.5%
Tree2Seq	26.67	2.70	66.8%
Graph2Seq-PGE	**38.97**	**3.81**	**79.2%**
Graph2Seq-NGE	34.28	3.26	75.3%
(Iyer et al., 2016)	18.4	3.16	64.2%
Graph2Seq-PGE	**23.3**	**3.23**	**70.2%**
Graph2Seq-NGE	21.9	2.97	65.1%

Pooling based graph embedding is better than node based graph embedding

Results

	BLEU-4	Grammar.	Correct.
Template	15.71	1.50	-
Seq2Seq	20.91	2.54	62.1%
Seq2Seq + Copy	24.12	2.65	64.5%
Tree2Seq	26.67	2.70	66.8%
Graph2Seq-PGE	**38.97**	**3.81**	**79.2%**
Graph2Seq-NGE	34.28	3.26	75.3%
(Iyer et al., 2016)	18.4	3.16	64.2%
Graph2Seq-PGE	**23.3**	**3.23**	**70.2%**
Graph2Seq-NGE	21.9	2.97	65.1%

Results

	BLEU-4	Grammar.	Correct.
Template	15.71	1.50	-
Seq2Seq	20.91	2.54	62.1%
Seq2Seq + Copy	24.12	2.65	64.5%
Tree2Seq	26.67	2.70	66.8%
Graph2Seq-PGE	**38.97**	**3.81**	**79.2%**
Graph2Seq-NGE	34.28	3.26	75.3%
(Iyer et al., 2016)	18.4	3.16	64.2%
Graph2Seq-PGE	**23.3**	**3.23**	**70.2%**
Graph2Seq-NGE	21.9	2.97	65.1%

Tree2Seq model performs better than both Seq2Seq model and Seq2Seq + Copy model

Examples

SQL Query & Interpretations	
1. COUNT Player WHERE starter = val_0 AND touchdowns = val_1 AND position = val_2	
S: How many players played in position val_2	Seq2Seq model
G: number of players with starter val_0 and get touchdowns val_1 for val_2	Graph2Seq model
2. SELECT Tires WHERE engine = val_0 AND chassis = val_1 AND team = val_2	
S: which tire has engine val_0 and chassis val_1 and val_2	Seq2Seq model
G: which tire does val_2 run with val_0 engine and val_1 chassis	Graph2Seq model

Exploiting Rich Syntactic Information for Semantic Parsing with Graph-to-Sequence Model (EMNLP'18)

Semantic Parsing Task

What are the jobs for programmer that has salary 50000 that uses c++ and not related with AI

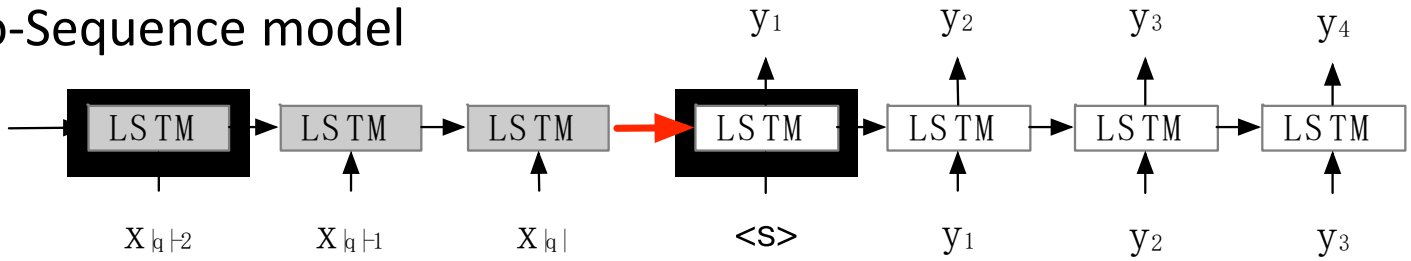
Semantic Parsing

answer(J, (job(J), -((area(J,R), const(R, 'ai'))), language(J, L), const(P, 'Programmer'), title(J, P), const(P, 'Programmer'), salary_greater_than(J, 50000, year))))

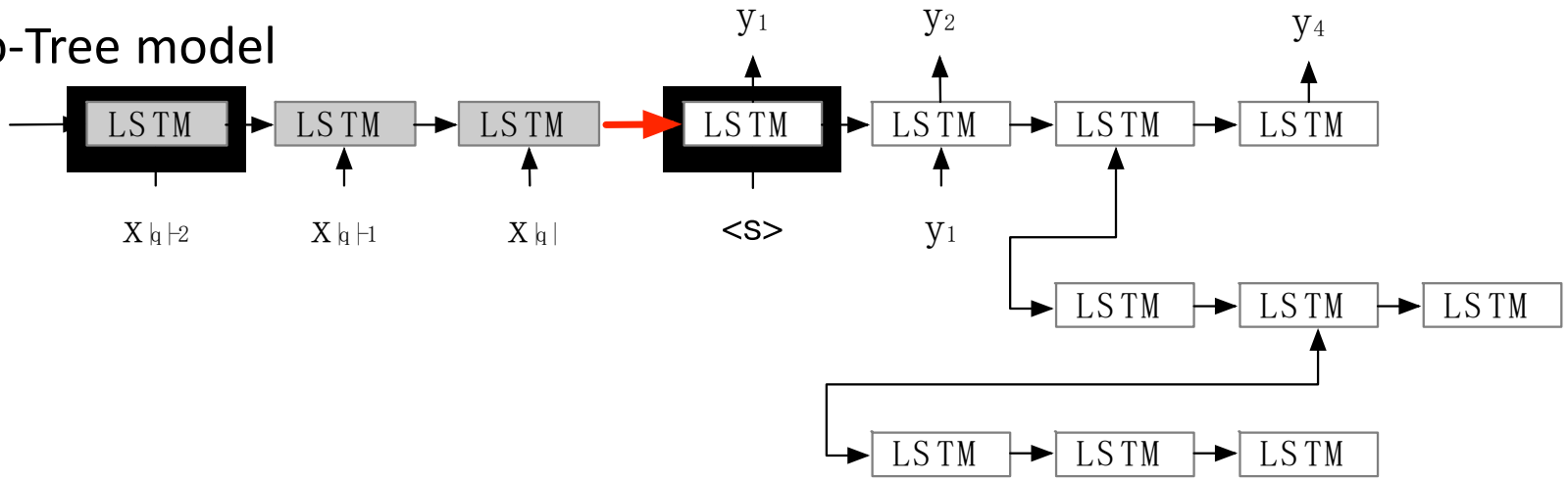
Neural Semantic Parser

Translation problem

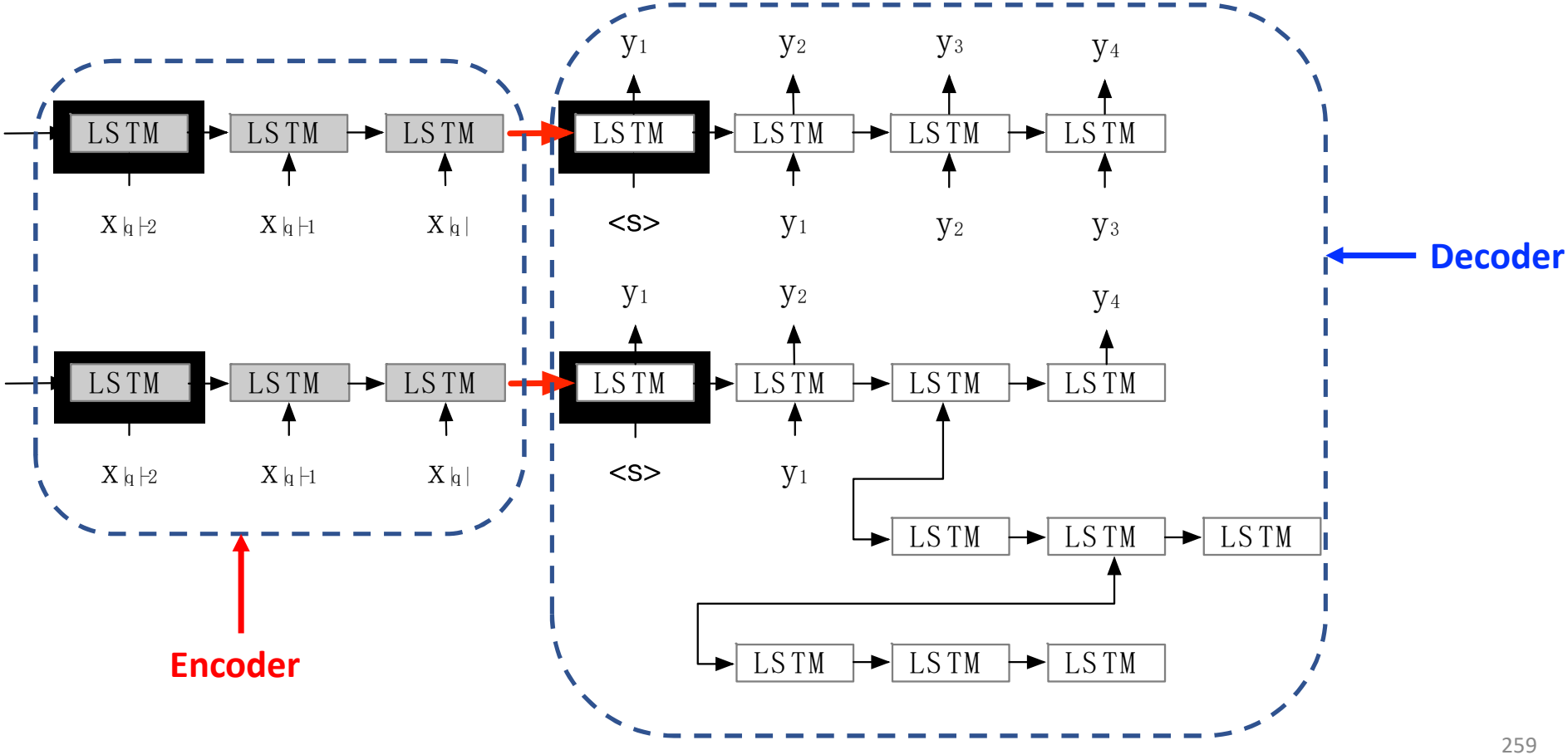
a) Sequence-to-Sequence model



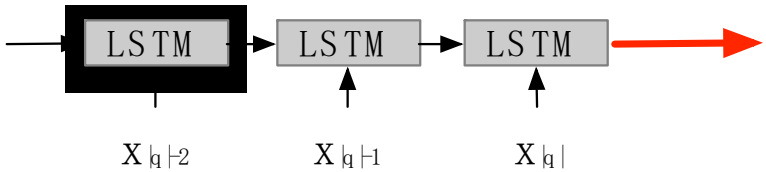
b) Sequence-to-Tree model



Encoder-Decoder Architecture



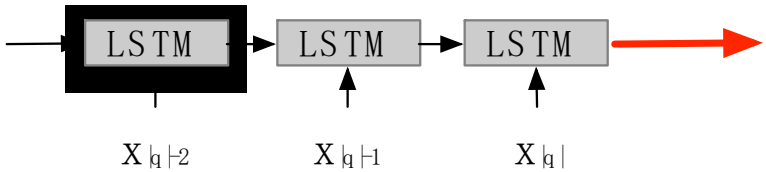
Problem



Sequence LSTM encodes word order features but **loses more complicated syntactic information** such as dependency and constituency trees.

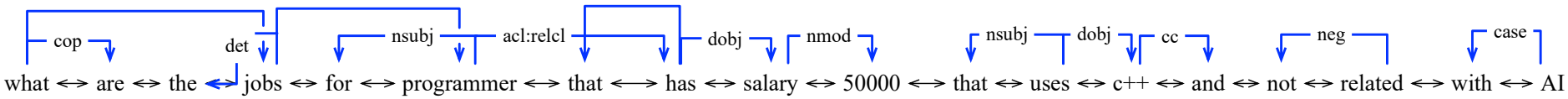
a) Dependency Feature

Problem

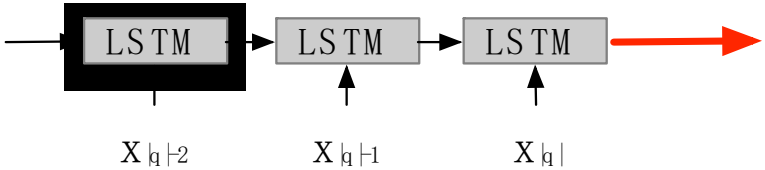


Sequence LSTM encodes word order features but **loses more complicated syntactic information** such as dependency and constituency trees.

a) Dependency Feature

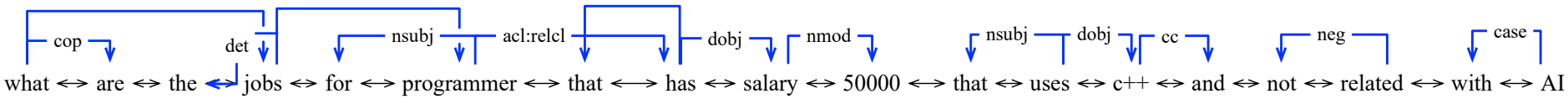


Problem



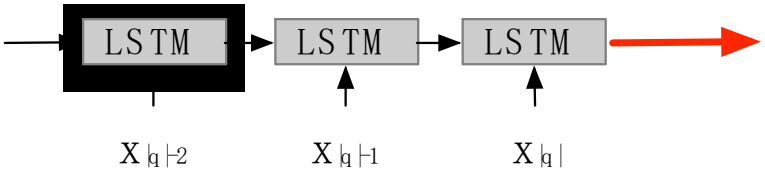
Sequence LSTM encodes word order features but **loses more complicated syntactic information** such as dependency and constituency trees.

a) Dependency Feature



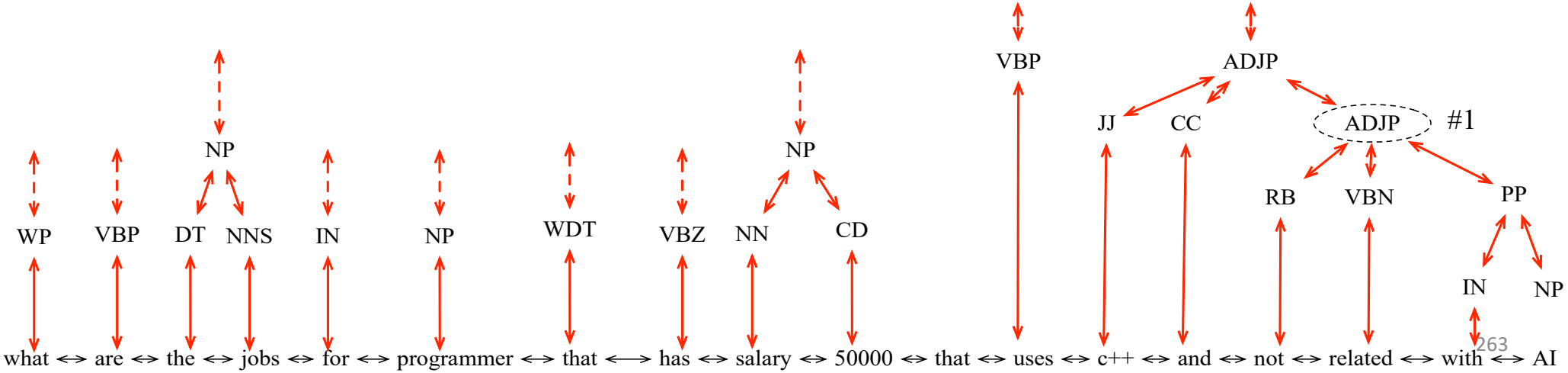
Describes the grammatical relations that hold among a pair of words.

Problem

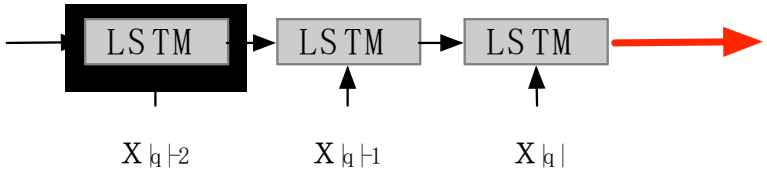


Sequence LSTM encodes word order features but **loses more complicated syntactic information** such as dependency and constituency trees.

(b) Constituency Feature

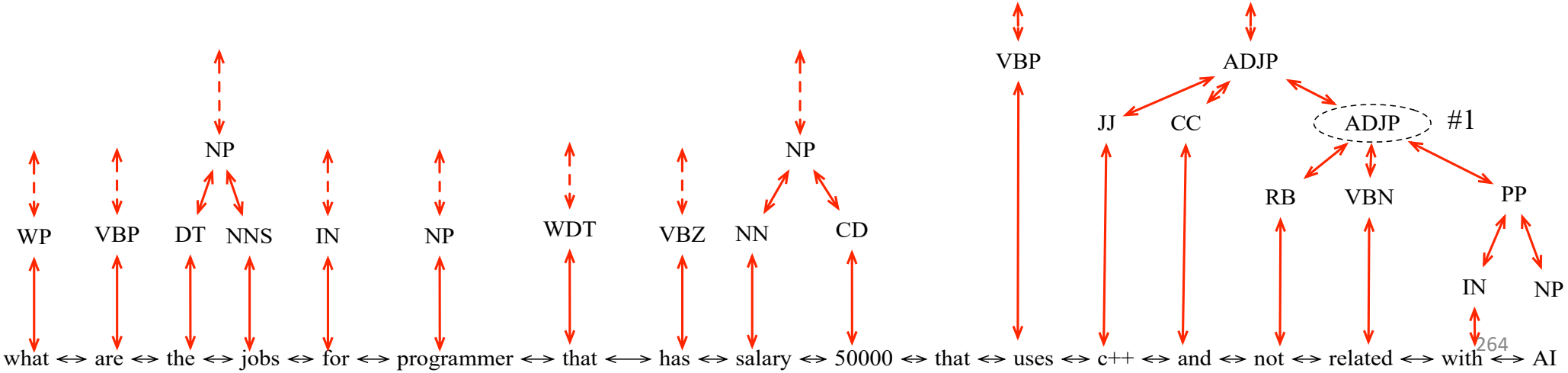


Problem



Sequence LSTM encodes word order features but **loses more complicated syntactic information** such as dependency and constituency trees.

(b) Constituency Feature represents the sentence structure



Our Approach

Syntactic graph

a) word order
b) dependency tree
c) constituency tree

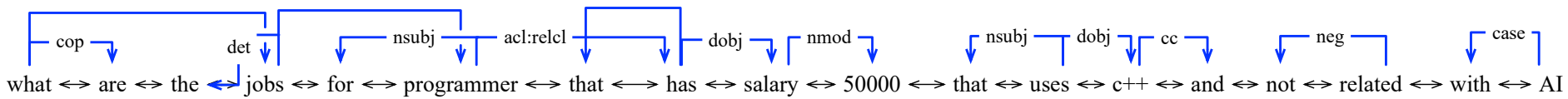
How about combine them together ?

Our Approach

Adding labeled edges between words

Syntactic graph

a) word order
b) dependency tree
c) constituency tree

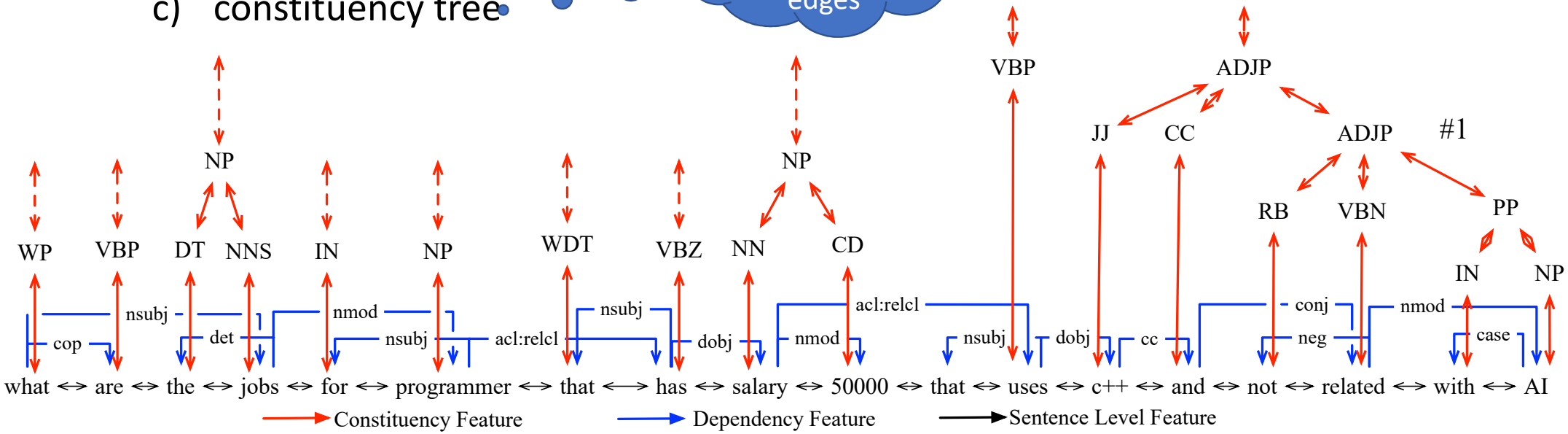


Our Approach

Syntactic graph

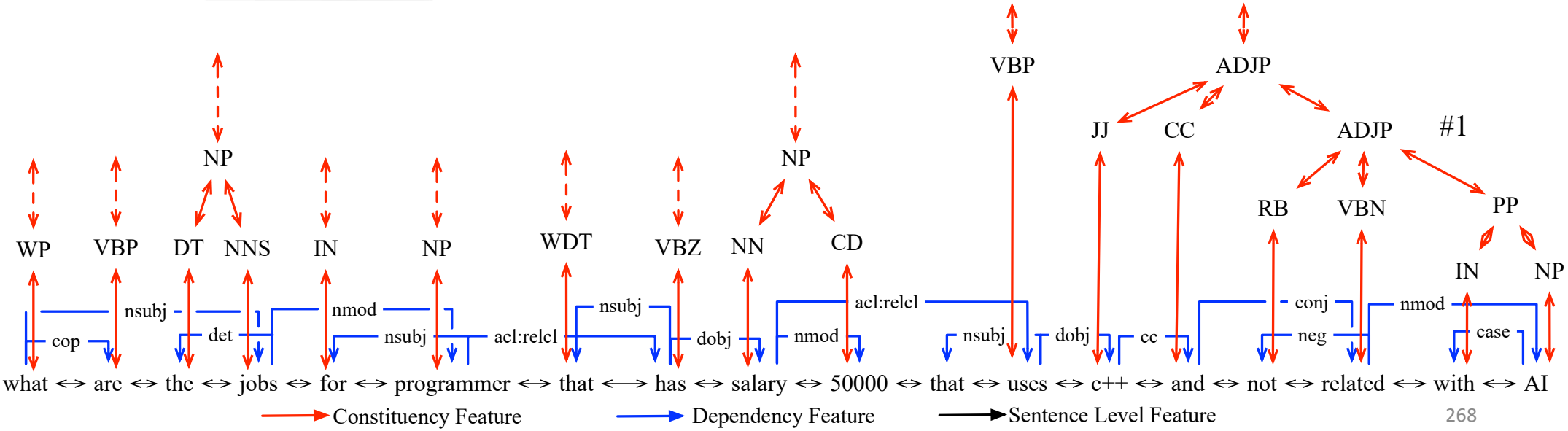
a) word order
b) dependency tree
c) constituency tree

Add non-terminal nodes and edges

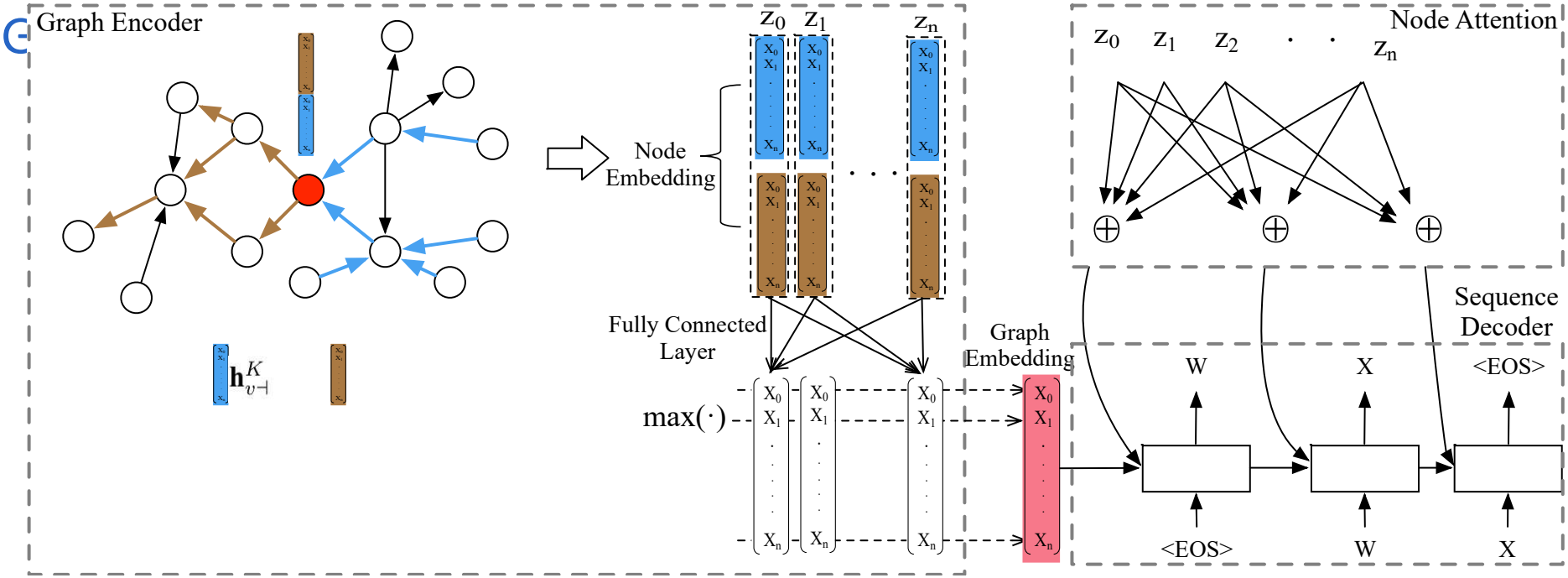


Syntactic Graph Representation

How to model this hybrid graph structure ?



Graph-to-Sequence Model [1]



[1] Kun Xu*, Lingfei Wu*, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin (both authors contributed equally), "Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks", arXiv preprint 2018.

Experiments

Datasets

- Jobs640
- Geo880
- ATIS

Baseline

a) use SeqLSTM to extract word order features, resulting word embeddings W_{seq}
b) take W_{seq} as initial word embeddings and use two treeLSTMs to extract dependency and constituency trees
c) the resulted word embeddings and non-terminal node embeddings are fed into a sequence decoder

Results

Graph2Seq	91.2	88.1	85.9
w/o word order features	86.7	84.4	82.9
w/o dependency features	89.3	85.8	83.8
w/o constituency features	88.9	84.7	84.6
w/ word order features	88.0	84.8	83.1
BASELINE	88.1	84.9	83.0

Graph2Seq significantly outperforms BASELINE

a) Jointly extracting these syntactic features in a unified model may be ***better*** than separately modeling these features

Word order features is the most ***important*** among these three features

Dependency and constituency features may have ***different*** impacts on the performance across datasets

Results

Method	Jobs	Geo	ATIS
Zettlemoyer and Collins (2007)	79.3	86.1	84.6
Kwiatkowski et al. (2011)	-	88.6	82.8
Liang et al. (2011)	90.7	87.9	-
Kwiatkowski et al. (2013)	-	**89.0**	-
Wang et al. (2014)	-	90.4	**91.3**
Zhao and Huang (2015)	85.0	88.9	84.2
Jia and Liang (2016)	-	85.0	76.3
Dong and Lapata (2016)-Seq2Seq	87.1	85.0	84.2
Dong and Lapata (2016)-Seq2Tree	90.0	87.1	84.6
Rabinovich et al. (2017)	**92.9**	85.7	85.3
Graph2Seq	91.2	88.1	85.9
w/o word order features	86.7	84.4	82.9
w/o dependency features	89.3	85.8	83.8
w/o constituency features	88.9	84.7	84.6
w/ word order features	88.0	84.8	83.1
BASELINE	88.1	84.9	83.0

Graph2Seq achieves competitive performance on three datasets

Graph2Seq performs better than Seq2Seq

a) Introducing more syntactic features could help ***better*** model the input text

Results

What type e parse features ?

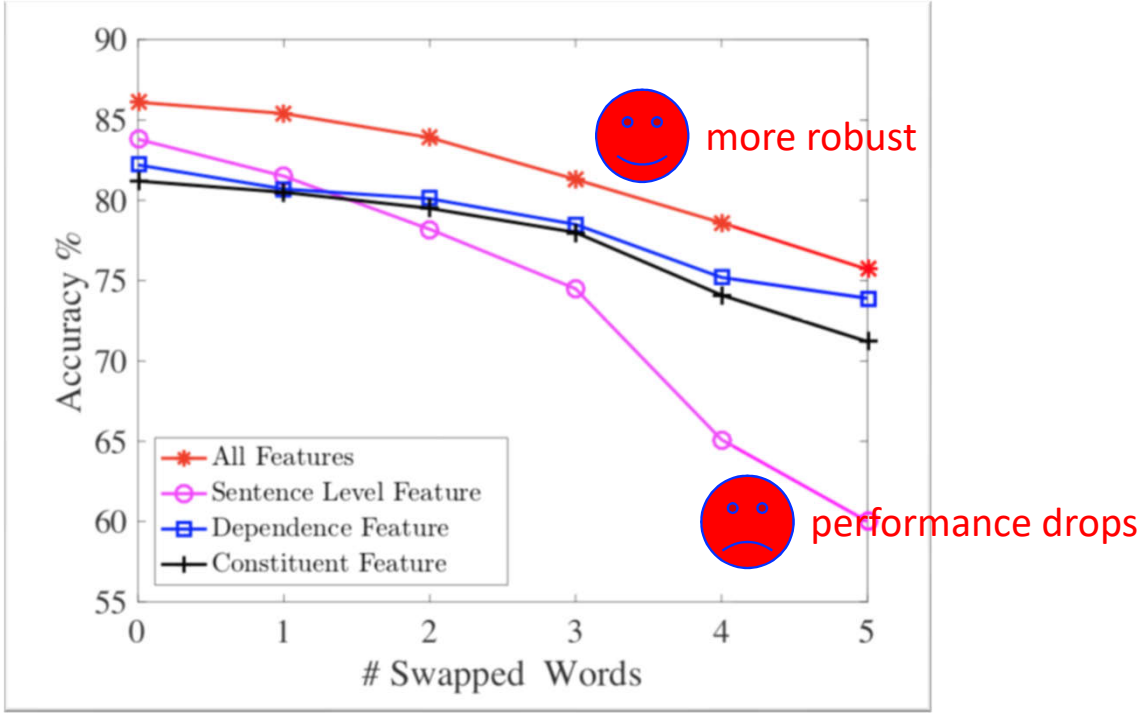
Complicated Query & Predicted Logical Forms
Jobs Q: *what are the jobs for programmer that has salary 50000 that uses c++ and not related with AI*
Pred: answer(J,(job(J),-((area(J,R),const(R,'ai'))), language(J,L),const(L,'c++'), title(J,P), const(P,'Programmer'),salary_greater_than(J, 50000,year)))).
Geo Q: *which is the density of the state that the largest river in the united states run through*
Pred: answer(A,(density(B,A),state(B), longest(C,(river(C),loc(C,D),const(D,id(usa)))), traverse(C,B))))
ATIS Q: *please find a flight round trip from los angeles to tacoma washington with a stopover in san francisco not exceeding the price of 300 dollars for june tenth 1993*
Pred: (lambda \$0 e (and (flight \$0) (round_trip \$0) (from \$0 *los angeles*) (to \$0 *tacoma washington*) (stop \$0 *san francisco*) (< (cost \$0) 300) (day_number \$0 *tenth*) (month \$0 *june*) (year \$0 1993)))

Robustness Study

adversarial examples

- SWAP examples

  We swap two letters of a word, e.g., noise → nosie



Robustness Study

adversarial examples

- paraphrased examples

We translate a text to Chinese and then translate it back to English using Google translate API

Feature	Acc_{ori}	Acc_{para}	Diff.
Word Order	84.8	78.7	-6.1
Dep	83.5	80.1	-3.4
Cons	82.9	77.3	-5.6
Dep + Cons	84.0	80.7	-3.3
Word Order + Dep	85.2	82.3	-2.9
Word Order + Cons	84.9	79.9	-5.0
Word Order + Dep + Cons	**86.0**	**83.5**	**-2.5**

more robust

Conclusion and Future Work

We presented Graph2Seq model, a generalized Seq2Seq model for graph inputs

We demonstrate the advantages of Graph2Seq on two useful scenarios:

- Graph data (natural or best expressed)
- Augmented sequence with hidden structure information

Other interesting works:

- A new task – Reinforcement Learning based Graph2Seq for Question Generation from Text (ICLR'20)
- A new task – Subgraph-guided Graph2Seq for Knowledge Graph Question Generation (ACL'20)
- A new model - Graph2Tree for graph structured inputs and tree structured outputs (ACL'20)
- A new model - Graph2Graph for both graph structured inputs and outputs (Target EMNLP'20)
- A new model – Deep Graph Matching for learning similarity of graph structured objects (ICML'20)
- A new model – Deep Graph Learning for joint learning graph structure and GNN (ICML'20)

W8: Deep Learning on Graphs: Methodologies and Applications (DLGMA'20)
Murray Hill East, 2nd floor, Hilton Middle Town,
New York, NY, USA, February 8, 2020
https://dlg2019.bitbucket.io/aaai20/

- **Topic of interest (including but not limited to):**
- Graph neural networks on node-level, graph-level embedding
- Dynamic/incremental graph-embedding
- Deep graph structured learning and graph matching
- Deep generative models for graph generation/semantic-preserving transformation
- Graph2seq, graph2tree, and graph2graph models
- Deep reinforcement learning and Adversarial machine learning on graphs

- **And with particular focuses but not limited to these application domains:**
- learning and reasoning (machine reasoning, inductive logic programming, theory proving)
- natural language processing (information extraction, semantic parsing, text generation, machine comprehension)
- reinforcement learning (multi-agent learning, compositional imitation learning)
- program synthesis and analysis
- bioinformatics (drug discovery, protein generation, etc.)

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

Collaborative Filtering

Similar users may share similar preference

Modeling user's preference based on past interactions

items

users

1	0	1	1
0	1	0	0
1	1	0	0
1	0	0	1

0/1 Interaction matrix

Suggest relevant items to users

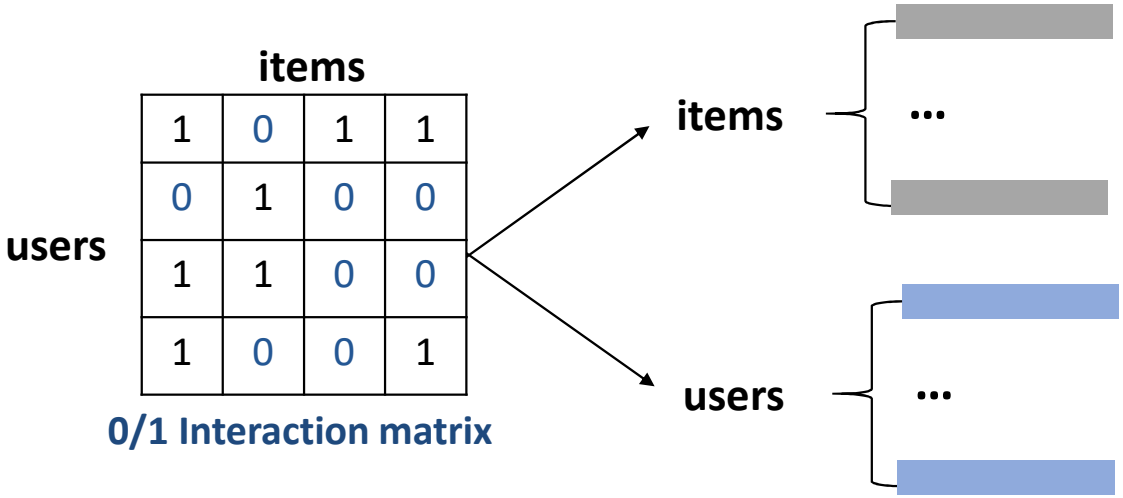
Recommender Systems

Suggest relevant items to users

Collaborative Filtering

Similar users may share similar preference

Modeling user's preference based on past interactions



Interactions as Bipartite Graph

items

users

1	0	1	1
0	1	0	0
1	1	0	0
1	0	0	1

0/1 Interaction matrix

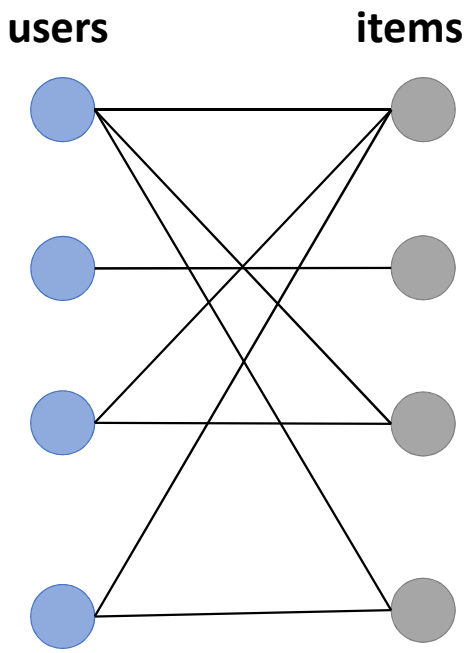
Interactions as Bipartite Graph

items

users

1	0	1	1
0	1	0	0
1	1	0	0
1	0	0	1

0/1 Interaction matrix

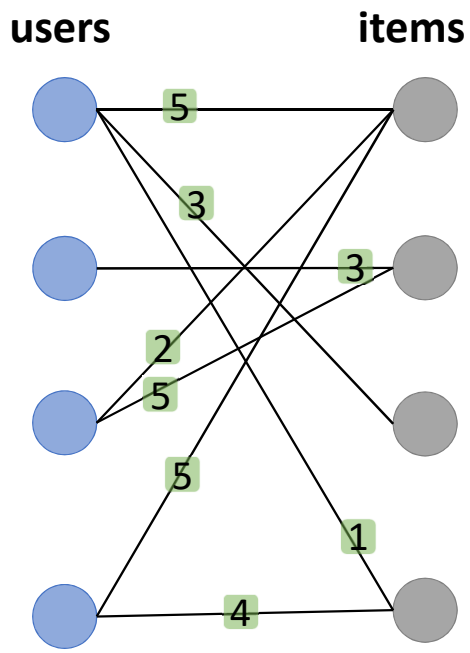


Bipartite Graph

Interactions as Bipartite Graph

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

Weighted interaction matrix

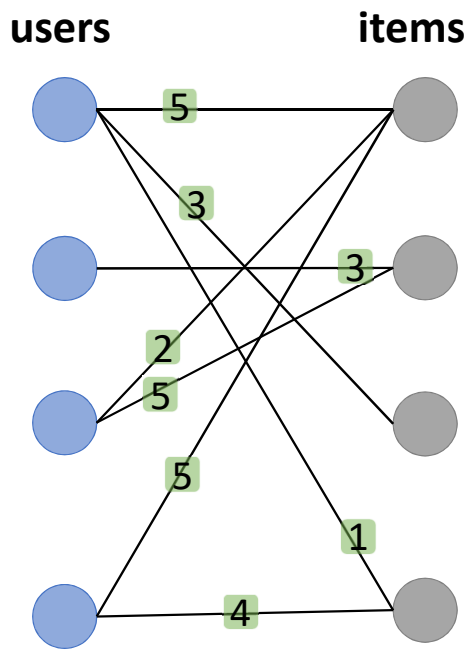


Bipartite Graph

Graph Convolution Matrix Completion

User representation learning

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



Bipartite Graph

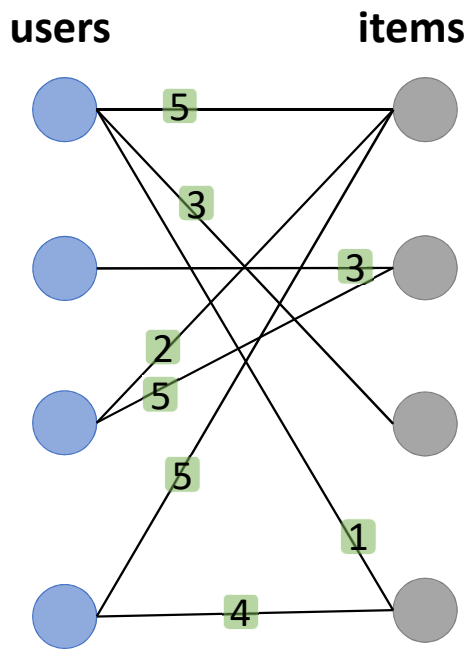
Graph Convolutional Matrix Completion. 2017.

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

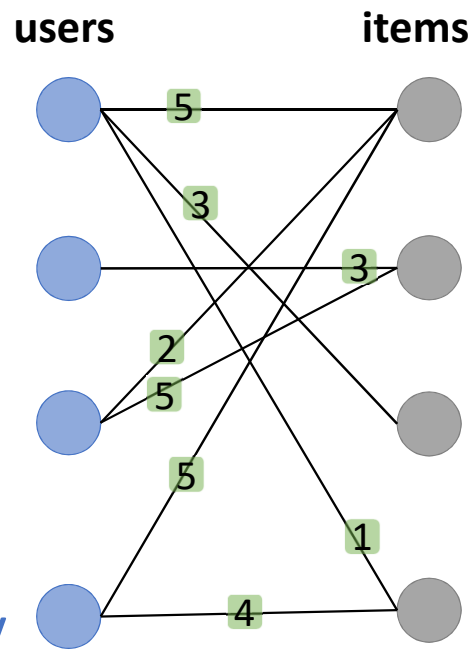
Graph Convolution Matrix Completion

User representation learning

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

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

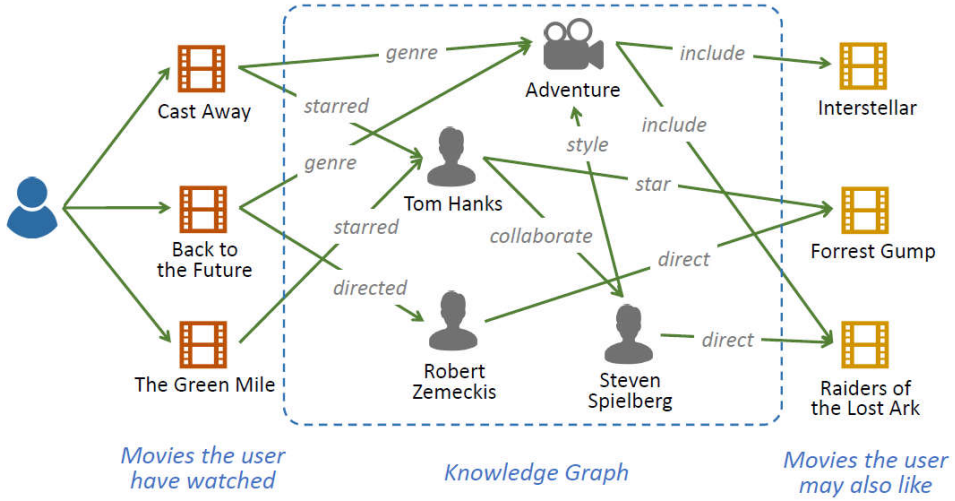
Item representation learning in a similar way



Bipartite Graph

Knowledge-graph-aware Recommendation

Side information about items: Knowledge graph (KG)

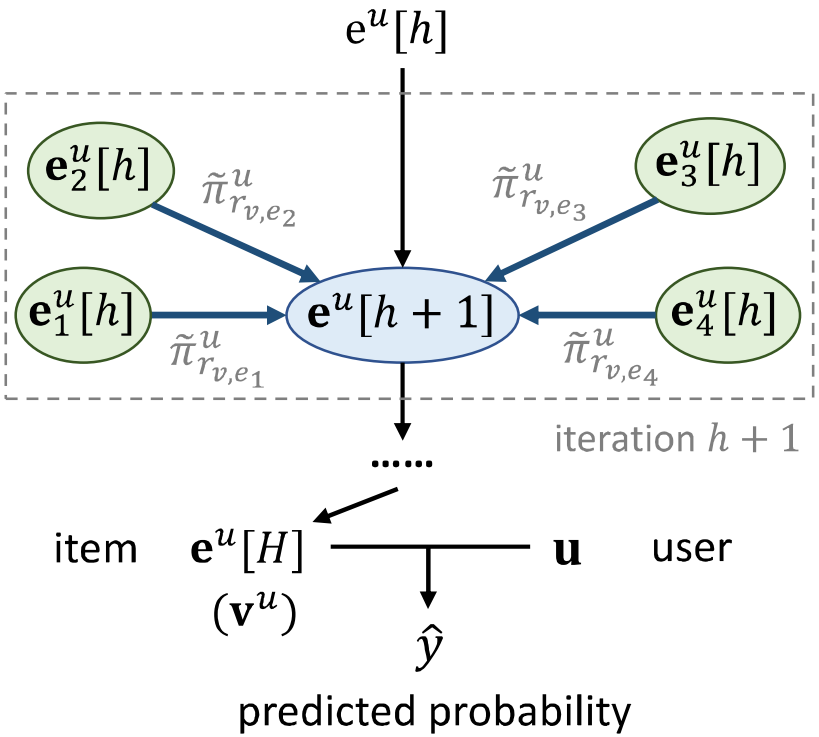


KG can help in learning item representations.

Triples:(head, relation, tail)

KGCN

Learning item representations by aggregation over KG

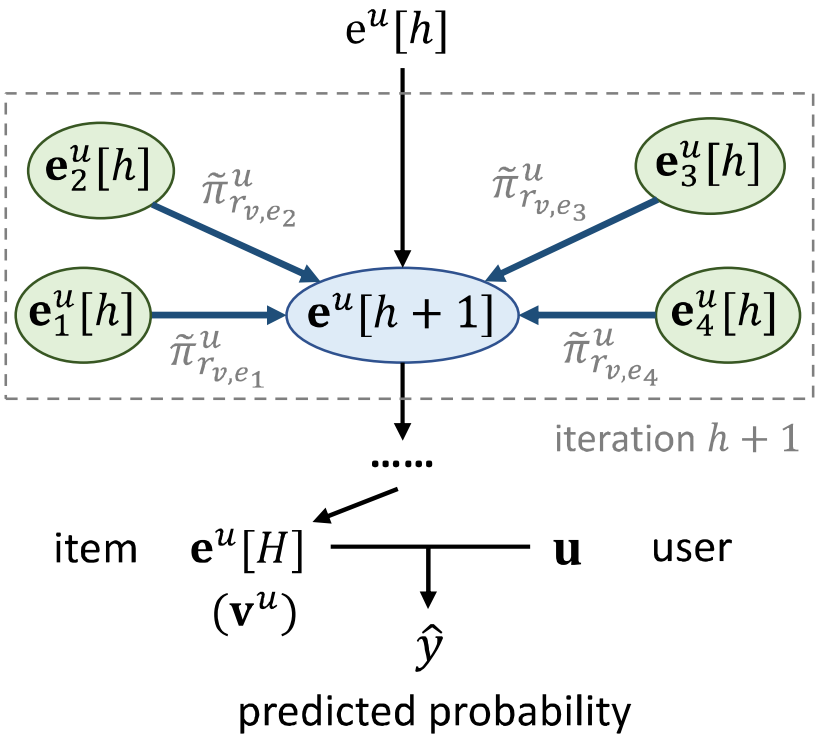


Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. KDD 2019.

KGCN

Learning item representations by aggregation over KG

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

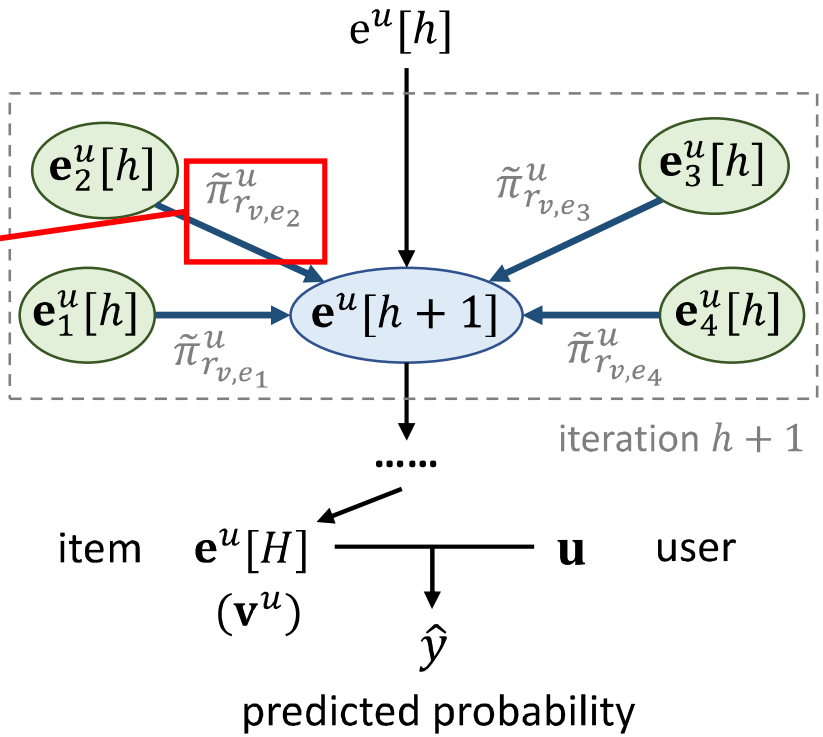


KGCN

Learning item representations by aggregation over KG

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

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



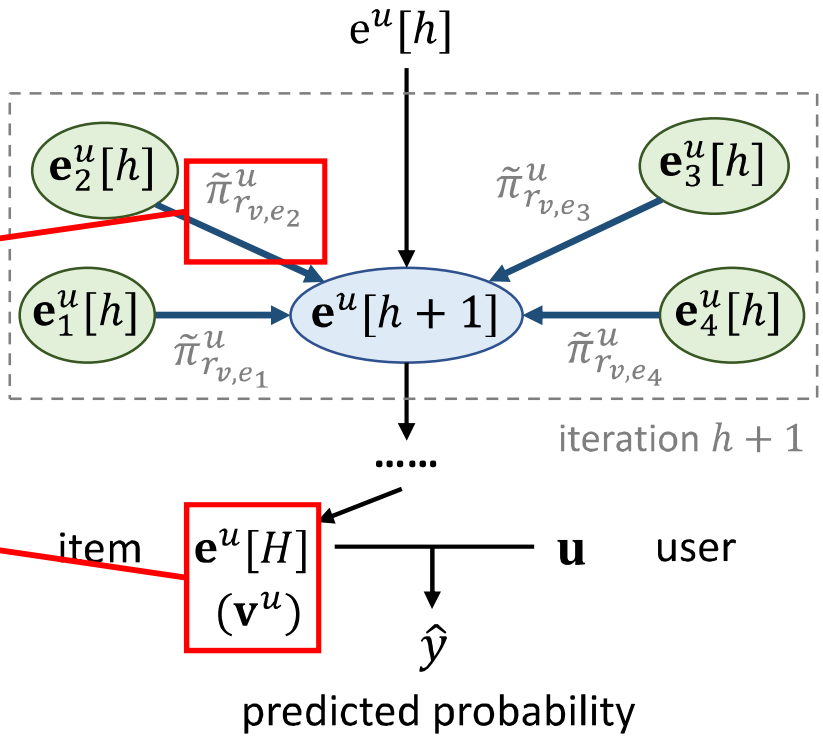
KGCN

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

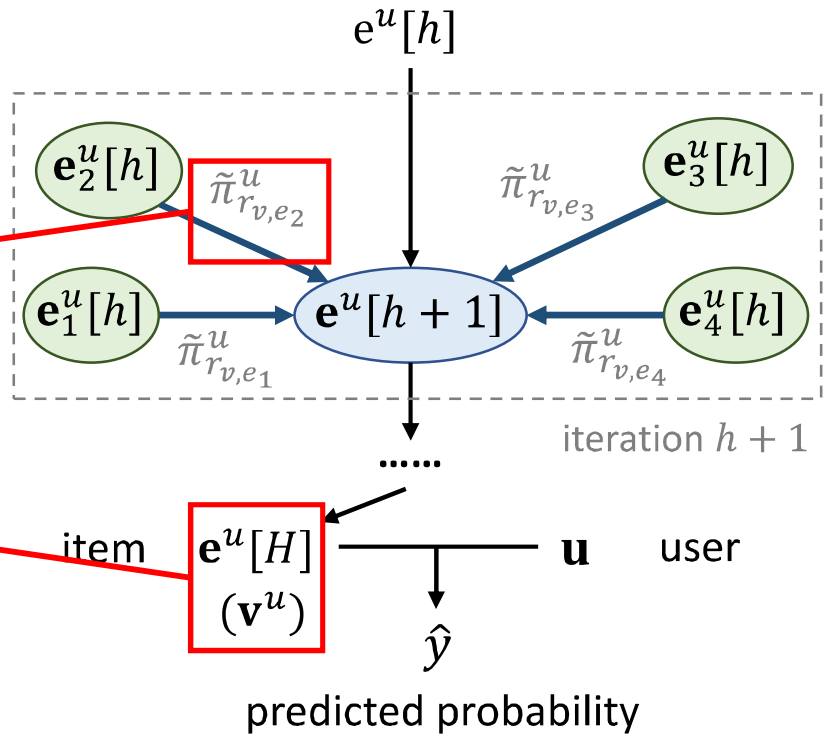
Learning item representations by aggregation over KG

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

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

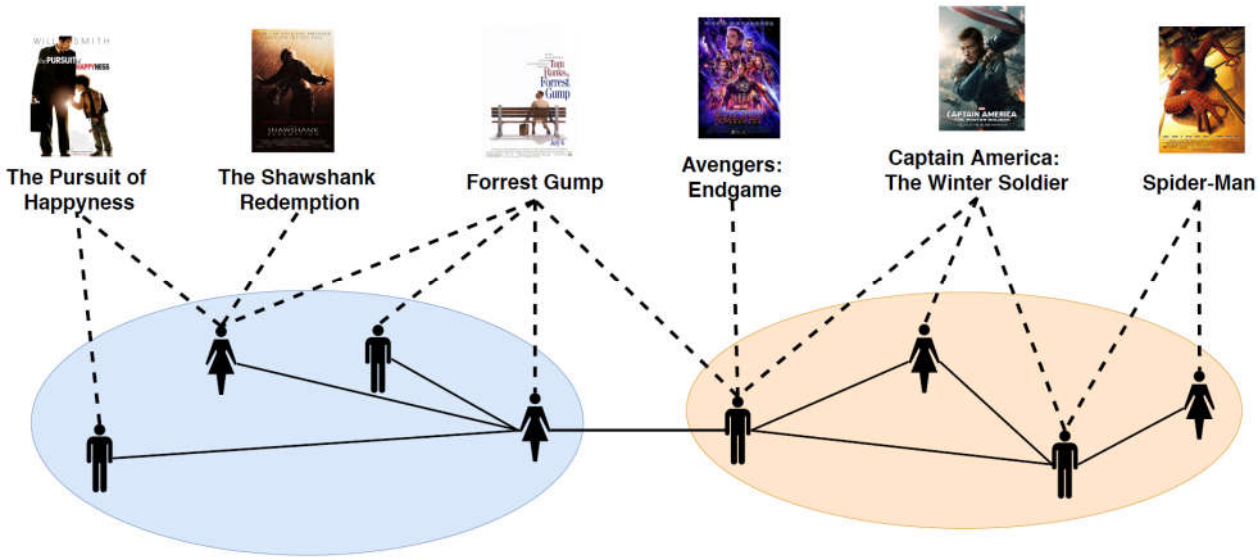
$$\mathbf{v}^u_{\mathcal{N}(v)} = \sum_{e \in \mathcal{N}(v)} \tilde{\pi}^u_{r_{v,e}} \mathbf{e}$$

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



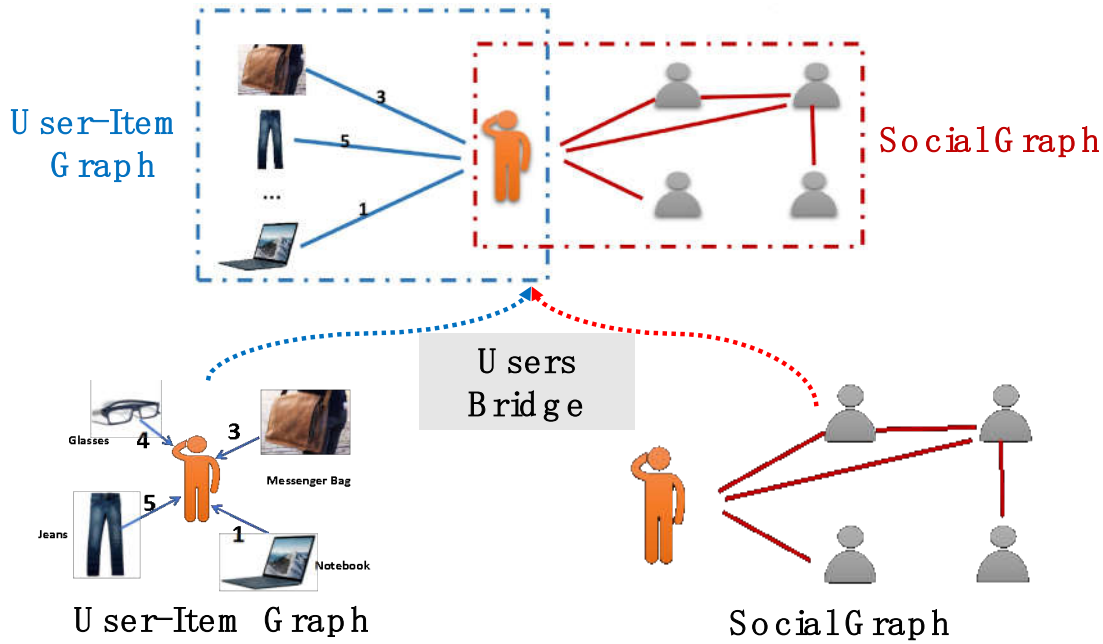
Social Recommendation

Side information about users: social networks



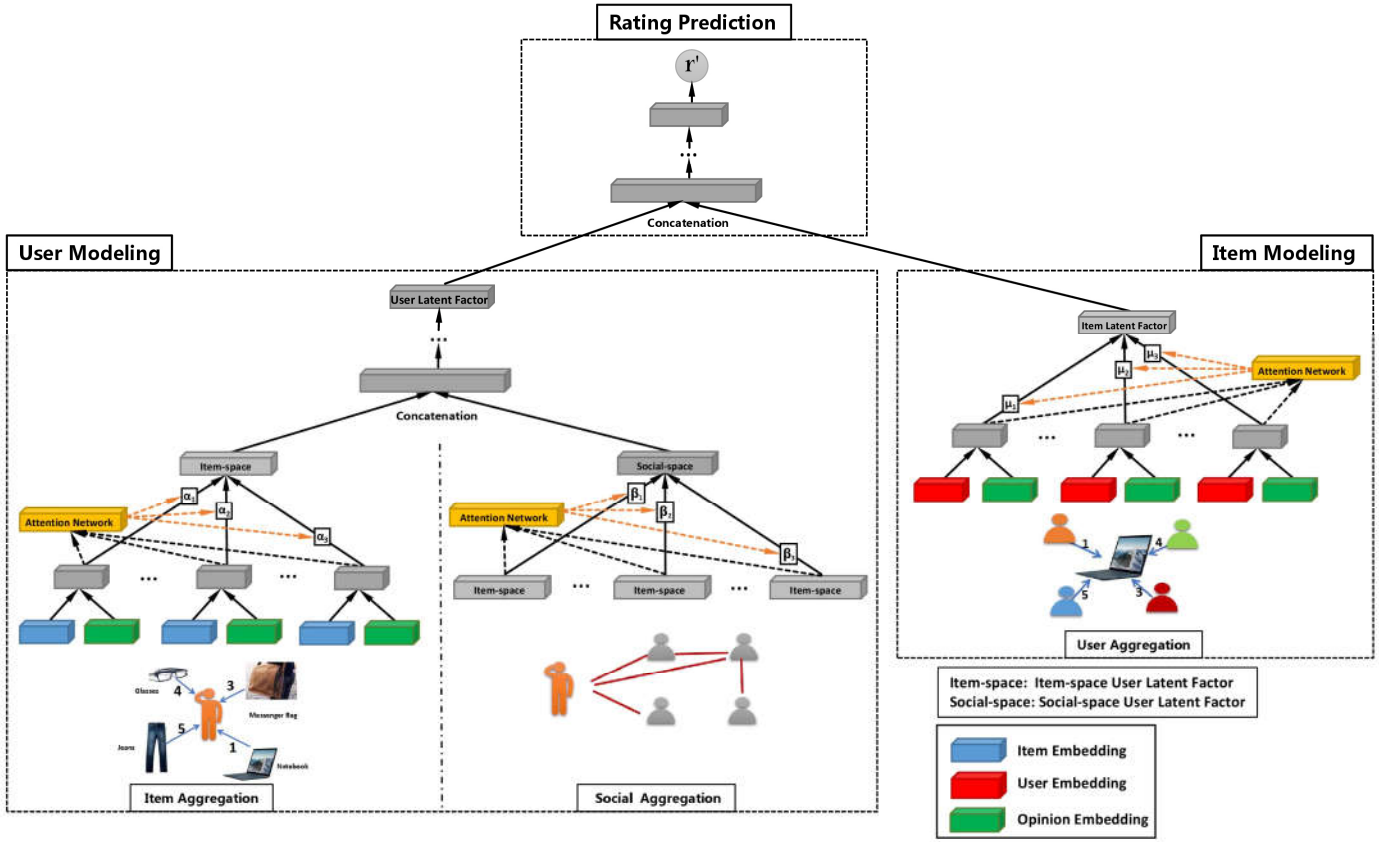
GraphRec

Utilizing both graphs for item and user representation learning

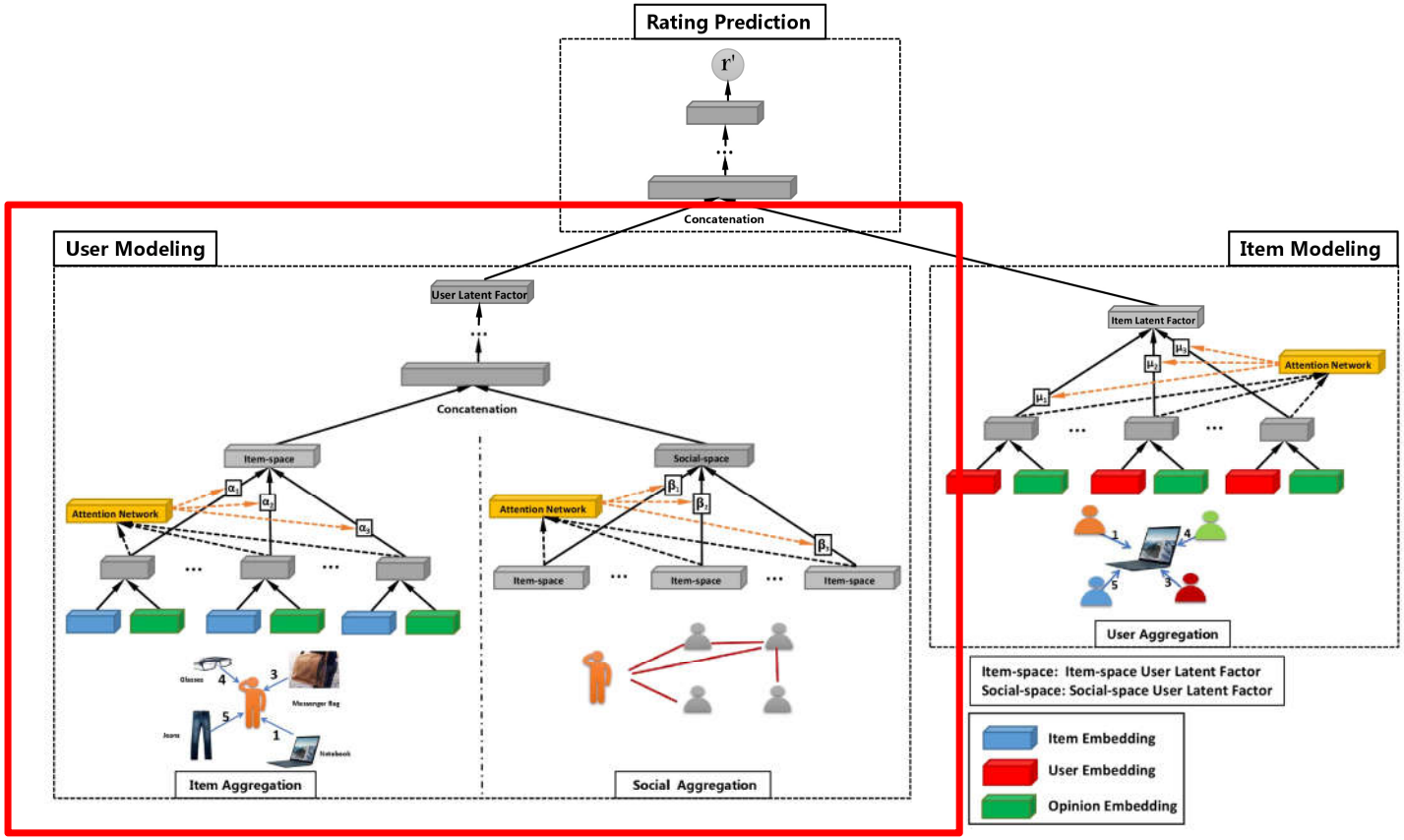


Graph Neural Networks for Social Recommendation. WWW 2019.

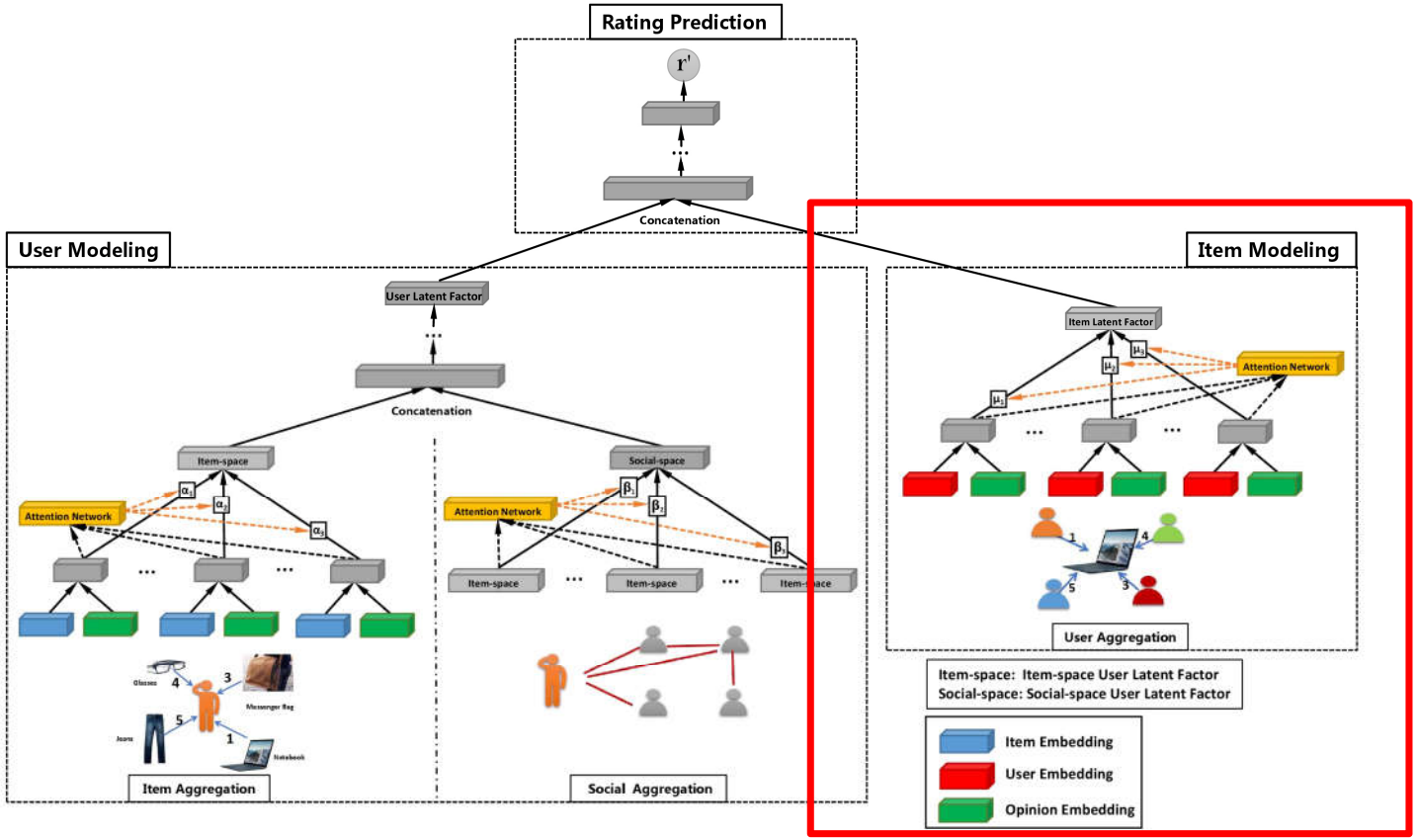
GraphRec


Rating Prediction
r'
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
r'
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
r'
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

Conclusion

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

→

Models

- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

→

Applications

- Healthcare
- Natural Language Processing
- Recommendations

Conclusion: Future Directions

Reasoning and analysis of GNNs

Why and how do GNNs work?

Conclusion: Future Directions

Reasoning and analysis of GNNs

Why and how do GNNs work?

Deeper GNNs

Can we make GNNs as deep as CNNs?

Conclusion: Future Directions

Reasoning and analysis of GNNs

Why and how do GNNs work?

Deeper GNNs

Can we make GNNs as deep as CNNs?

Interpretable GNNs

Can we explain predictions made by GNNS?

A Book

MICHIGAN STATE UNIVERSITY IBM

Foundations

- Basic Graph Theory
- Spectral Graph Theory
- Graph Fourier Analysis

Models

- Spectral-based GNN
- Spatial-based GNN
- Pooling Layers in GNN
- Robustness of GNN
- Scalable Learning for GNN

Applications

- Healthcare
- Natural Language Processing
- Recommendations

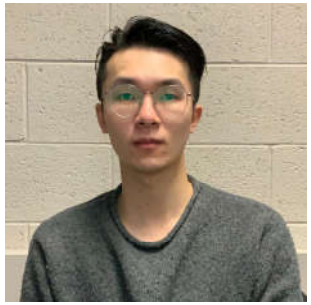
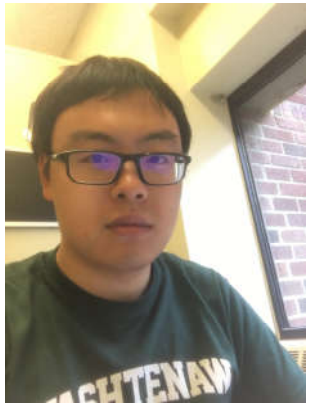
Feature Selection on Graphs
Graph Embedding

More advances in Models
Other Deep models on Graphs
GNNs for Complex Graphs

Data Mining
Computer Vision
Biology, Chemistry and Physics

A Repository

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



DeepRobust: A repository about general adversarial attacks and defenses