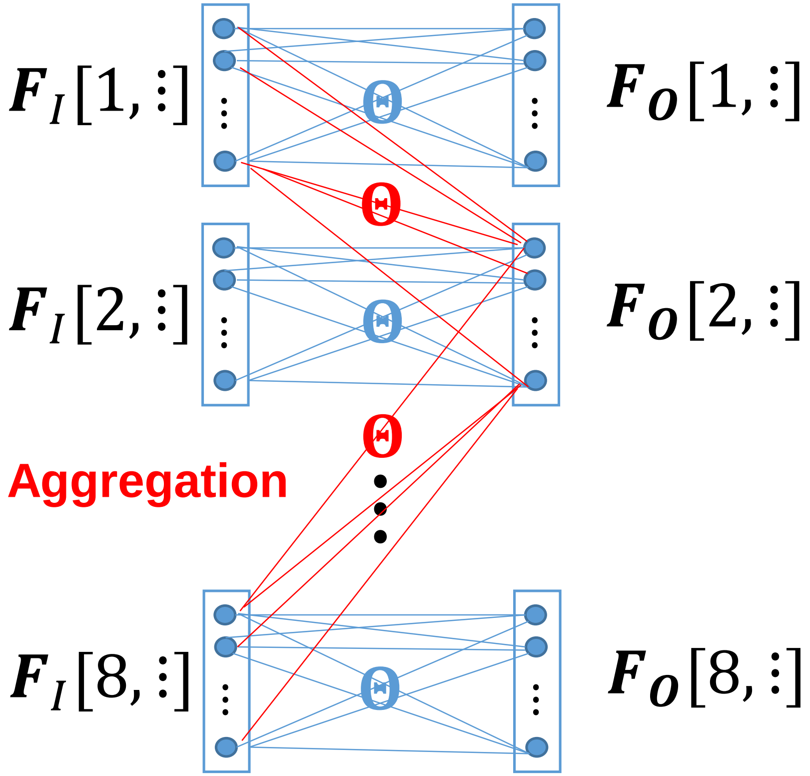
Summary Questions of the lecture

- Present the spatial view of Simplified ShevNet.

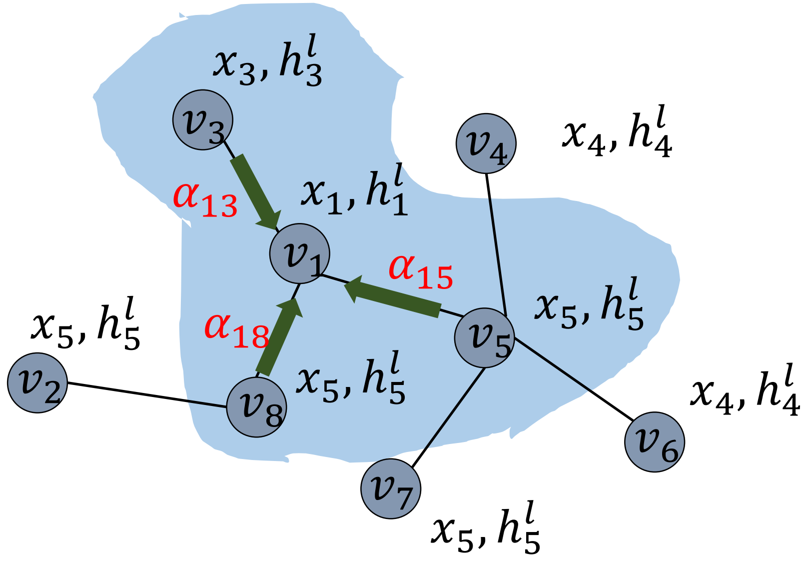
→ From the simplified ChebNet's formulation $F_O = CF_I\Theta$, we observe the output graph signal of the i'th node: $F_O[i,:] = \sum_j C[i,j]F_I[j,:]\Theta$. From the fact that $C[i,j]$ is zero between nodes that are not neighbors, we find that $F_O[i,:]$ is an aggregation from the graph signals of neighbor nodes, weighted by the learnable parameters Θ, which is a spatial smoothing operation and corresponds to a spectral smoothing.



Summary Questions of the lecture

- What is the difference of Simplified ShevNet from a non-graph neural network ?

→ Non-graph neural networks cannot leverage the connectivity information among graph nodes. Thus, the feature of each node (or 'data point' in non-graph networks) is transformed independently. On the other hand, graph neural networks transform each node by aggregating information from connected nodes.



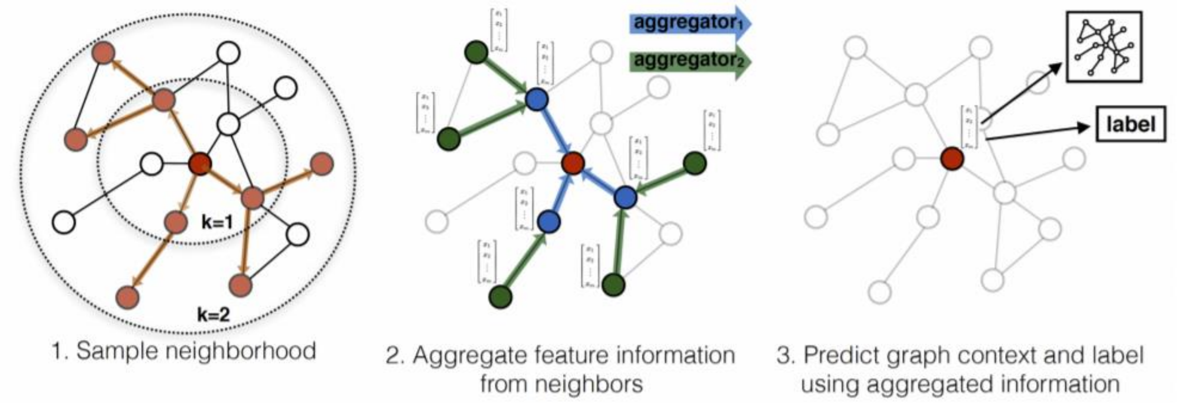
Summary Questions of the lecture

- Explain the key aspects of GraphSAGE (SAmple and aggreGatE).

→ In the context of semi-supervised learning, only aggregating information from nodes of distance 1 has a danger of only encountering unlabeled nodes. Thus, GraphSage tries to sample and aggregate nodes in multi-hop distances. Thus, the resulting message-passing signal is concatenated with the current node's signal and transformed by a learned parameter matrix.

$$\boldsymbol{h}_{N_s(v_i)}^{(l+1)} = \boldsymbol{AGG}\left(\left\{\boldsymbol{h}_j^{(l)} | v_j \in N_s(v_i)\right\}\right)$$

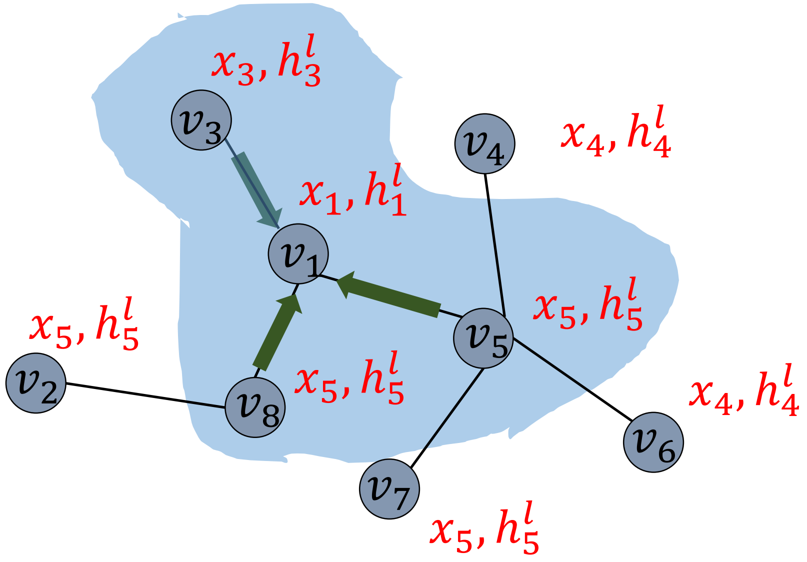
$$\boldsymbol{h}_i^{(l+1)} = \boldsymbol{\sigma}\left(\boldsymbol{\Theta} \cdot \left[\boldsymbol{h}_i^{(l)} \parallel \boldsymbol{h}_{N_s(v_i)}^{(l+1)}\right]\right)$$



$GCN(\mathcal{G})$

Geometric deep learning via graph convolution … (continue)

$$\boldsymbol{h}_i^{(l+1)} = \sum_{v_j \in N(v_i)} \boldsymbol{f}\left(\boldsymbol{x}_i, \boldsymbol{w}_{ij}, \boldsymbol{h}_j^{(l)}, \boldsymbol{x}_j\right)$$



Outline of Lecture (4)

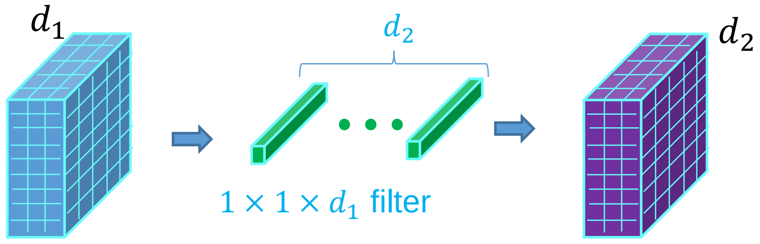
- Spatial GCN
  - Spatial View of Simplified ChebNet
  - GraphSage (Hamilton et al. NIPS 2017)
  - GAT : Graph Attention (Veličković et al. ICLR 2018)
  - MPNN: Message Passing (Glimer et al. ICML 2017)
  - **gPool:** Graph U-Nets (Gao et al. ICML 2019)
  - **DiffPool:** Differentiable Pooling (Ying et al. NeurIPS 2018)
  - **EigenPooling:** EigenPooling (Ma et al. KDD 2019)
- Link Analysis
  - PageRank
  - Diffusion
- Propagation using graph diffusion
  - Predict Then Propagate [ICLR'19]
  - Graph Diffusion-Embedding Networks [CVPR'19]
- Making a new graph
  - Diffusion Improves Graph Learning [NIPS'19]
  - Graph Learning-Convolutional Nets. [CVPR'19]

R: GCN: Multilayer Structure

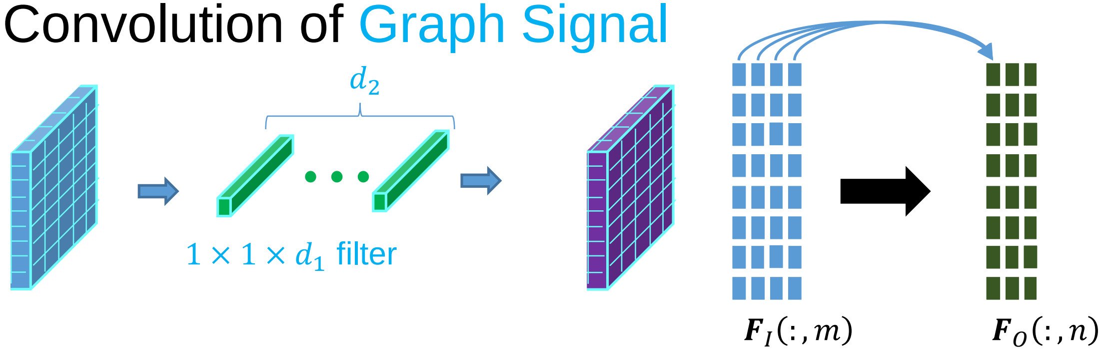
$$y_i = \Theta x_i$$

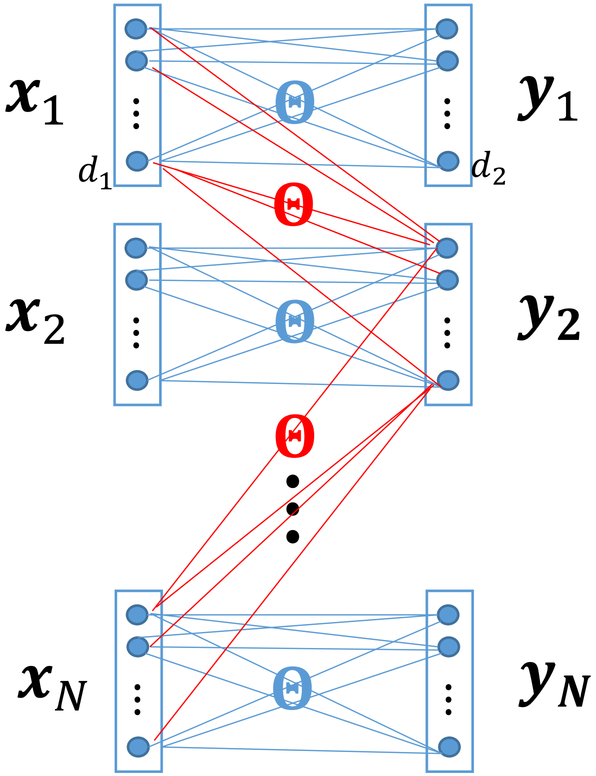
$$\begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{id_2} \end{bmatrix} = \begin{bmatrix} \theta^{(11)} & \cdots & \theta^{(1d_1)} \\ \vdots & \ddots & \vdots \\ \theta^{(d_2 1)} & \cdots & \theta^{(d_2 d_1)} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id_2} \end{bmatrix}$$

1×1 Convolution of Image Signal



1×1 Convolution of Graph Signal





$$y_i^T = x_i^T \Theta^T$$

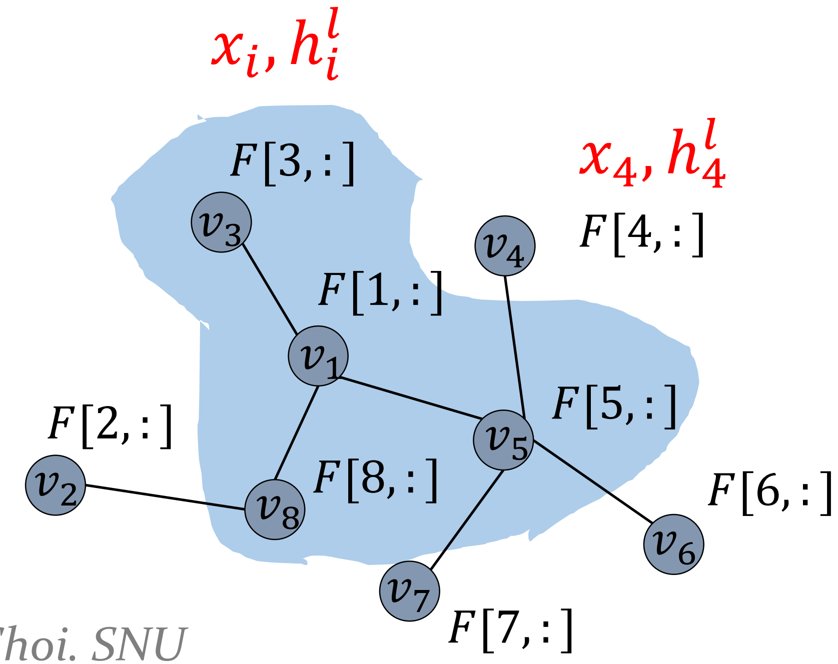
$$[y_{i1}, y_{i2}, \ldots, y_{id_2}] = [x_{i1}, x_{i2}, \ldots, x_{id_1}] \begin{bmatrix} \theta^{(11)} & \cdots & \theta^{(1d_2)} \\ \vdots & \ddots & \vdots \\ \theta^{(d_1 1)} & \cdots & \theta^{(d_1 d_2)} \end{bmatrix}$$

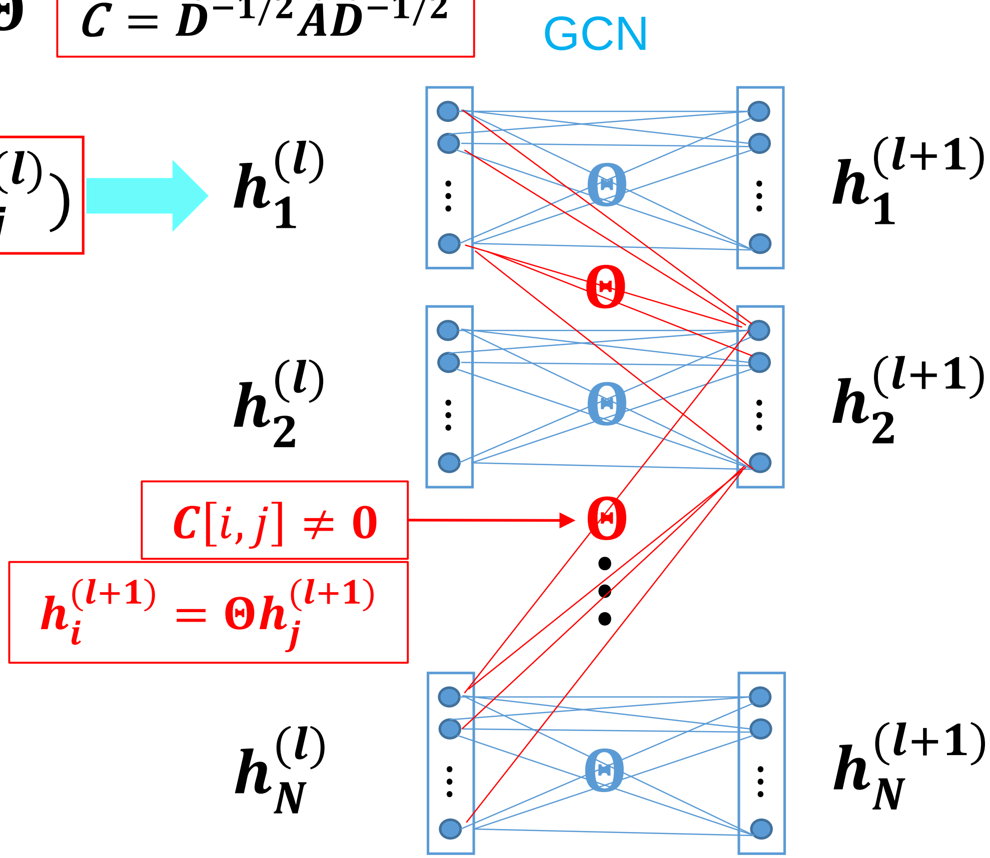
R: GCN: Multilayer Structure

Multilayer Structure **(Simplified ChebNet)**

$$F_O[i,:] = \sum_{v_j \in N(v_i) \cup \{v_i\}} C[i,j] F_I[j,:] \Theta \qquad \boxed{C = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}}$$

$$\boxed{h_i^{(l+1)} = \sigma\left(\sum_{v_j \in N(v_i) \cup \{v_i\}} C[i,j] \Theta^{(l)} h_j^{(l)}\right)}$$





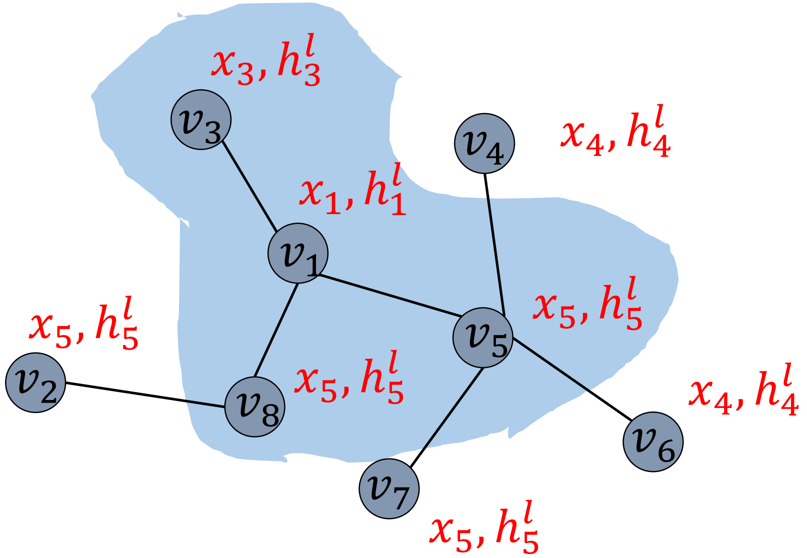
R: GCN: Multilayer Structure

GCN: $\boldsymbol{l}$ –th Layer

$$\boldsymbol{h}_{\boldsymbol{i}}^{(\boldsymbol{l}+\mathbf{1})} = \boldsymbol{\sigma}(\textstyle\sum_{v_j \in N(v_i) \cup \{v_i\}} \underline{C[i,j]\boldsymbol{\Theta}^{(l)}\boldsymbol{h}_{\boldsymbol{j}}^{(l)}})$$

$\boldsymbol{\sigma}$: Relu or SoftMax

$$\boxed{\boldsymbol{C} = \widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}}$$



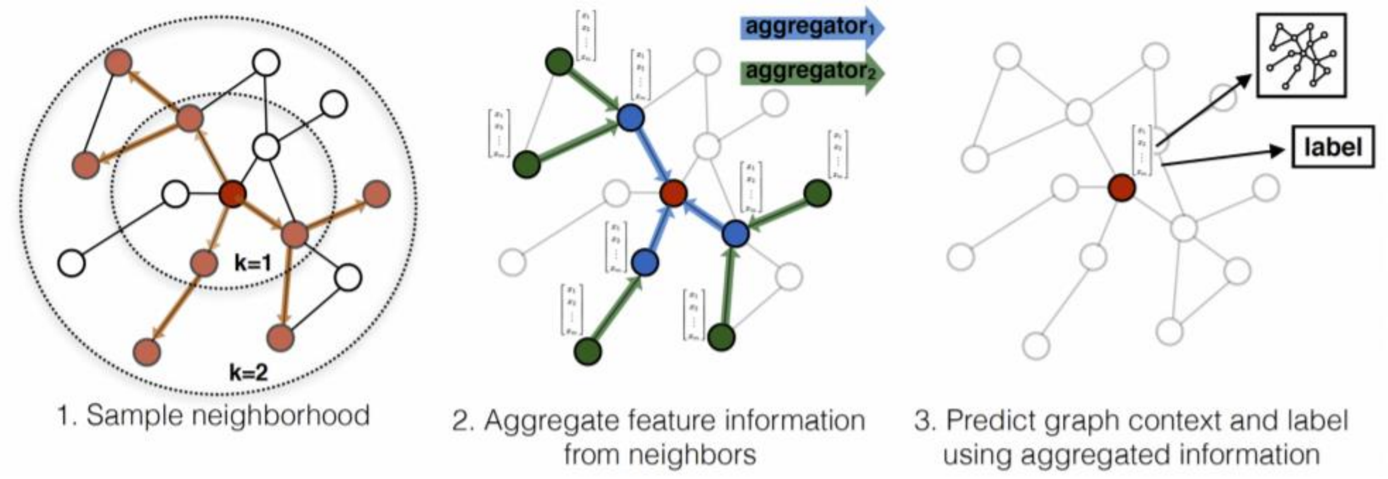
1-st **GNN**: $\boldsymbol{l}$ –th Layer

$$\boldsymbol{h}_{\boldsymbol{i}}^{(\boldsymbol{l}+\mathbf{1})} = \textstyle\sum_{v_j \in N(v_i)} \underline{\boldsymbol{f}(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{h}_{\boldsymbol{j}}^{(\boldsymbol{l})}, \boldsymbol{x}_{\boldsymbol{j}})}$$

R: GCN: Filter in GraphSAGE (SAmple and aggreGatE)

Neighbor Sampling:

$$N(v_i) \Longrightarrow N_s(v_i)$$



Aggregation:

$$\boldsymbol{h}_{N_s(v_i)}^{(l+1)} = \boldsymbol{AGG}\left(\left\{\boldsymbol{h}_j^{(l)} | v_j \in N_s(v_i)\right\}\right)$$

$$\boldsymbol{h}_i^{(l+1)} = \boldsymbol{\sigma}\left(\boldsymbol{\Theta} \cdot \left[\boldsymbol{h}_i^{(l)} \parallel \boldsymbol{h}_{N_s(v_i)}^{(l+1)}\right]\right)$$, ‖: concatination

Mean aggregator

LSTM aggregator

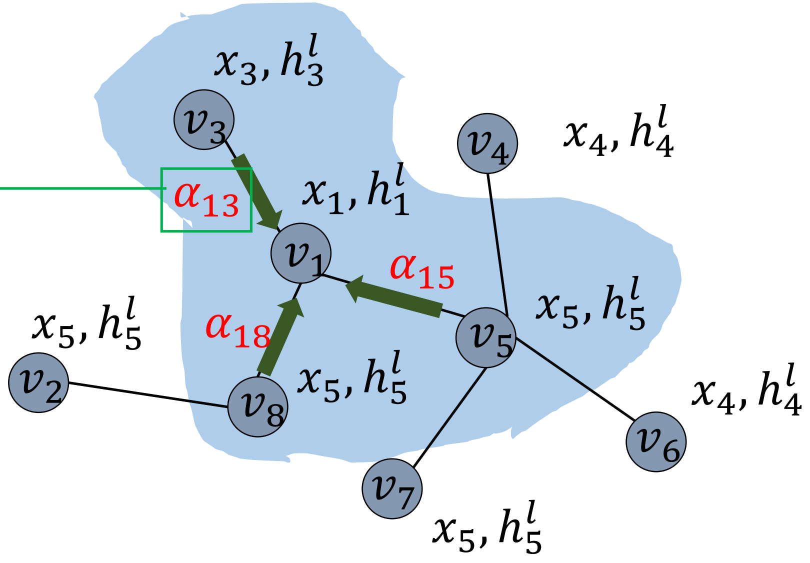
Pooling aggregator

GraphSAGE: Inductive Representation Learning on Large Graphs
(Hamilton et al. NIPS 2017)

GCN: Filter in GAT (Graph ATtention Networks)

Aggregation:

$$\boldsymbol{h}_i^{(l+1)} = \boldsymbol{\sigma}\left(\boldsymbol{\Theta} \sum_{v_j \in N(v_i) \cup \{v_i\}} \boxed{\alpha_{ij}}\, \boldsymbol{h}_j^{(l)}\right)$$

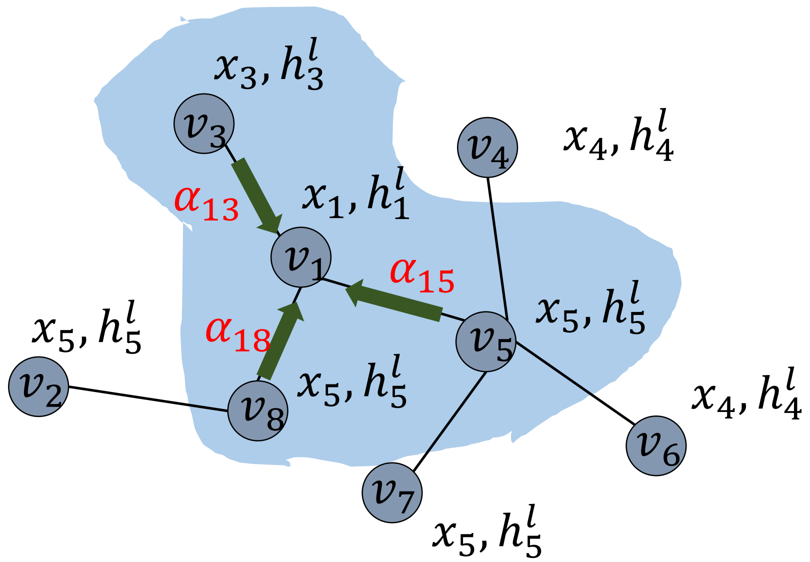


GAT: Graph Attention Networks (https://arxiv.org/pdf/1710.10903.pdf)
(Petar Velickovic et al. ICLR 2018)

GCN: Filter in GAT (Graph ATtention Networks)

Aggregation:

$$\boldsymbol{h}_i^{(l+1)} = \boldsymbol{\sigma}\left(\boldsymbol{\Theta} \sum_{v_j \in N(v_i)} \boxed{\alpha_{ij}}\, \boldsymbol{h}_j^{(l)}\right)$$



$$\boxed{\alpha_{ij}} = \frac{exp\left(LeakyReLU\left(\boldsymbol{a}^T\left[\boldsymbol{\Theta}\cdot\boldsymbol{h}_i^{(l)} \parallel \boldsymbol{\Theta}\cdot\boldsymbol{h}_j^{(l)}\right]\right)\right)}{\sum_{v_k \in N(v_i)} \quad exp\left(LeakyReLU\left(\boldsymbol{a}^T\left[\boldsymbol{\Theta}\cdot\boldsymbol{h}_i^{(l)} \parallel \boldsymbol{\Theta}\cdot\boldsymbol{h}_k^{(l)}\right]\right)\right)}$$

$\boldsymbol{a}, \boldsymbol{\Theta}$: parameters of a single layer neural network

GAT: Graph Attention Networks (https://arxiv.org/pdf/1710.10903.pdf)
(Petar Velickovic et al. ICLR 2018)

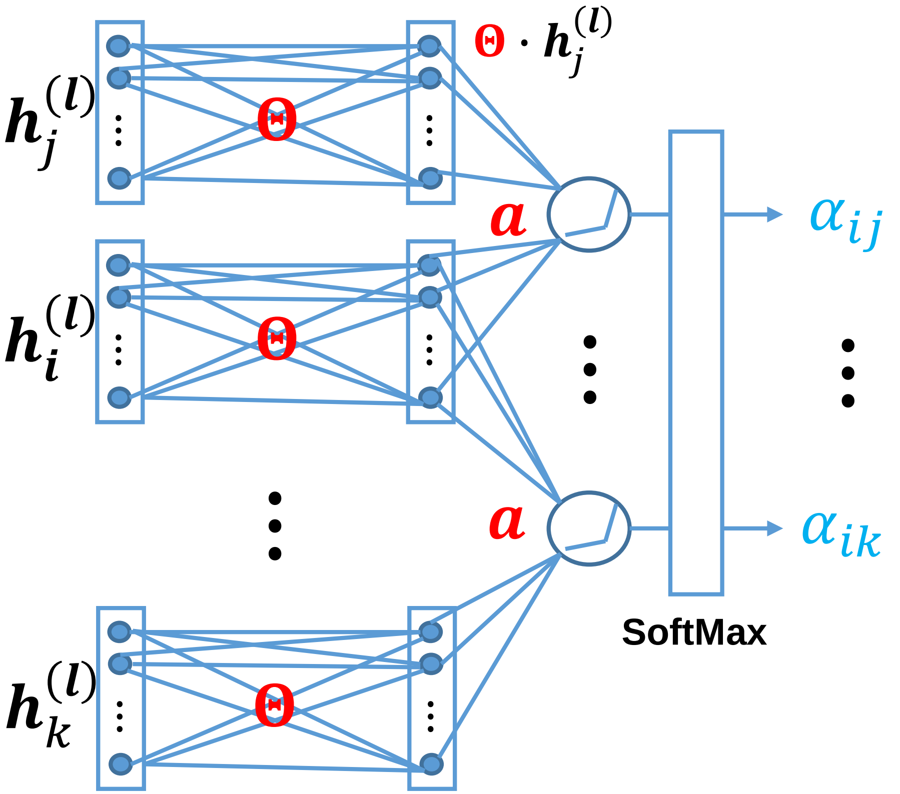
GCN: Filter in GAT (Graph ATtention Networks)

Aggregation:

$$h_i^{(l+1)} = \sigma\left(\sum_{v_j \in N(v_i)} \alpha_{ij}\, \Theta \cdot h_j^{(l)}\right)$$

$$\alpha_{ij} = \frac{exp\left(LeakyReLU\left(a^T\left[\Theta \cdot h_i^{(l)} \parallel \Theta \cdot h_j^{(l)}\right]\right)\right)}{\sum_{v_k \in N(v_i)} exp\left(LeakyReLU\left(a^T\left[\Theta \cdot h_i^{(l)} \parallel \Theta \cdot h_k^{(l)}\right]\right)\right)}$$

a, Θ: parameters of a single layer network

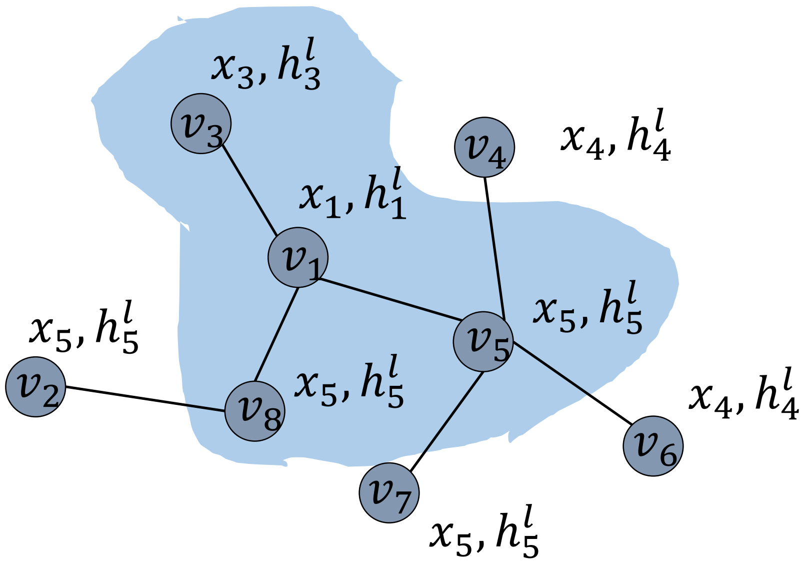


GAT: Graph Attention Networks (https://arxiv.org/pdf/1710.10903.pdf)
(Petar Velickovic et al. ICLR 2018)

GCN: Filter in MPNN (Message Passing Neural Networks)

Message Passing (Aggregation):

$$\boldsymbol{m}_i^{(l+1)} = \sum_{v_j \in N(v_i)} M_l\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}, e_{ij}\right)$$



Feature Updating:

$$\boldsymbol{h}_i^{(l+1)} = U_l\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{m}_i^{(l+1)}\right)$$

M_l, U_l are functions to be designed

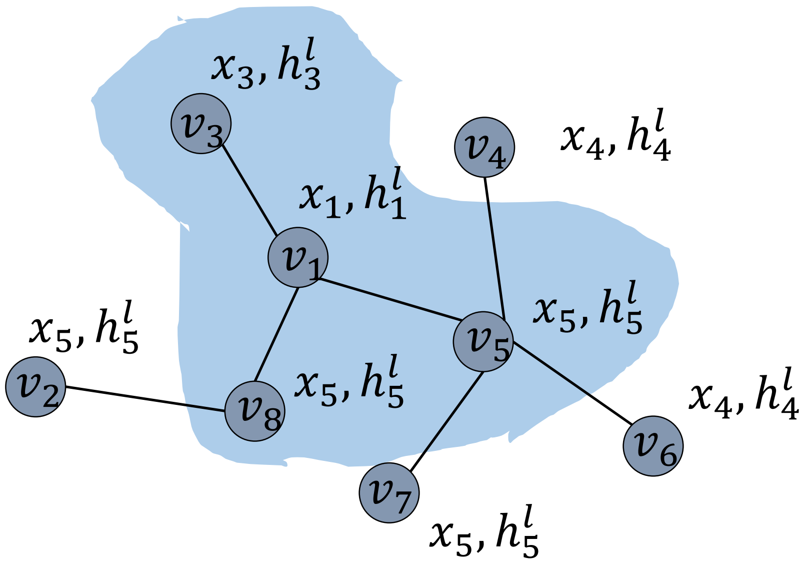
MPNN : **Neural Message Passing for Quantum Chemistry,** (Justin Gilmer et al. ICLR 2018)

GCN: Filter in MPNN (Message Passing Neural Networks)

NPNN (ICLR 2018):

$$\boldsymbol{m}_i^{(l+1)} = \sum_{v_j \in N(v_i)} M_l\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}, e_{ij}\right)$$

$$\boldsymbol{h}_i^{(l+1)} = U_l\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{m}_i^{(l+1)}\right)$$



R: GraphSAGE (NIPS 2017):

$$\boldsymbol{h}_{N_s(v_i)}^{(l+1)} = \boldsymbol{AGG}\left(\left\{\boldsymbol{h}_j^{(l)}, v_j \in N_s(v_i)\right\}\right)$$

$$\boldsymbol{h}_i^{(l+1)} = \boldsymbol{\sigma}\left(\boldsymbol{\Theta} \cdot \left[\boldsymbol{h}_i^{(l)} \parallel \boldsymbol{h}_{N_s(v_i)}^{(l+1)}\right]\right)$$

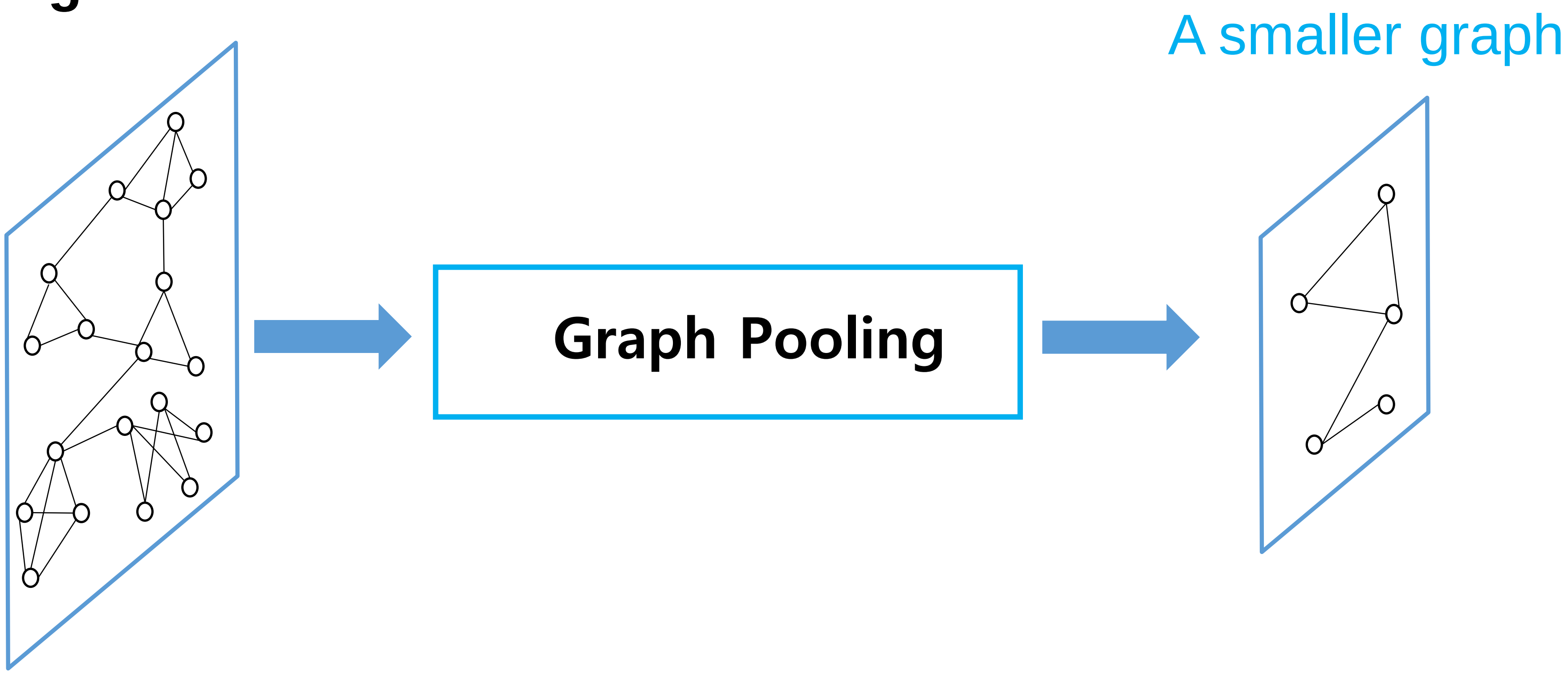
Mean aggregator

LSTM aggregator

Pooling aggregator

GCN: Graph Pooling Operation

Graph pooling

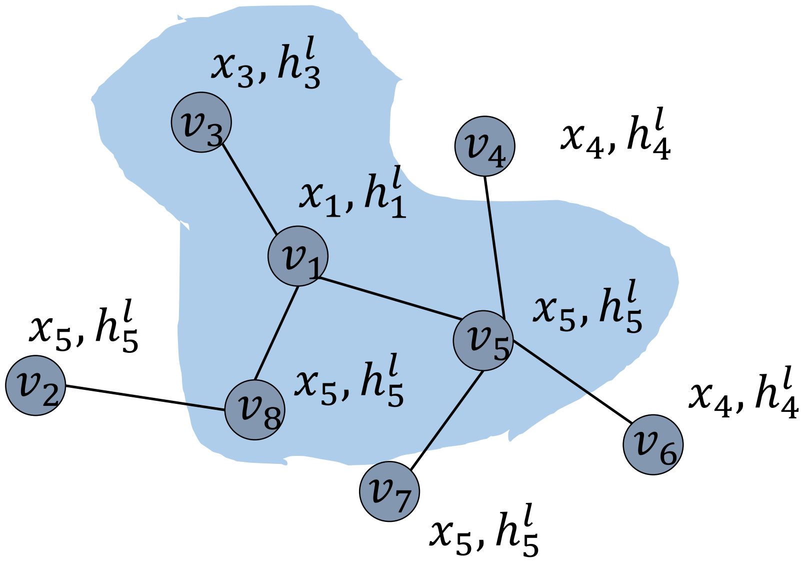


$\boldsymbol{A} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}\times\boldsymbol{n}}, \boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{d}}$

$\boldsymbol{A_p} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n_p}\times\boldsymbol{n_p}}, \boldsymbol{H_p} \in \mathbb{R}^{\boldsymbol{n_p}\times\boldsymbol{d_p}}$

GCN: gPool

Downsample by selecting the most important nodes



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

$$\Downarrow$$

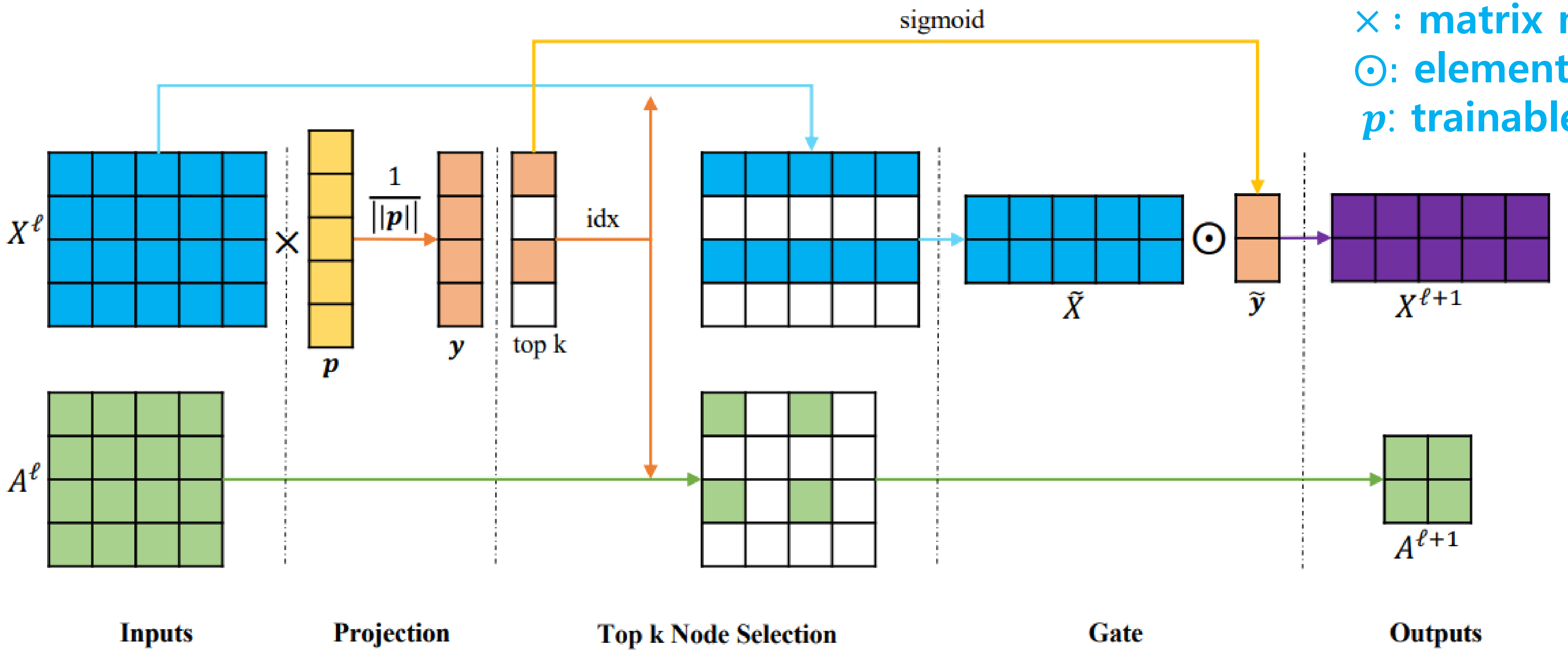
$$\boldsymbol{A}^{l+1} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}^{l+1} \times \boldsymbol{n}^{l+1}}, \boldsymbol{X}^{l+1} \in \mathbb{R}^{\boldsymbol{n}^{l+1} \times \boldsymbol{d}^{l+1}}$$

Usually $\boldsymbol{d}^l = \boldsymbol{d}^{l+1}$

gPool: Graph U-Nets (Gao et al. ICML 2019)
https://arxiv.org/pdf/1905.05178.pdf

GCN: gPool

Downsample by selecting the most important nodes

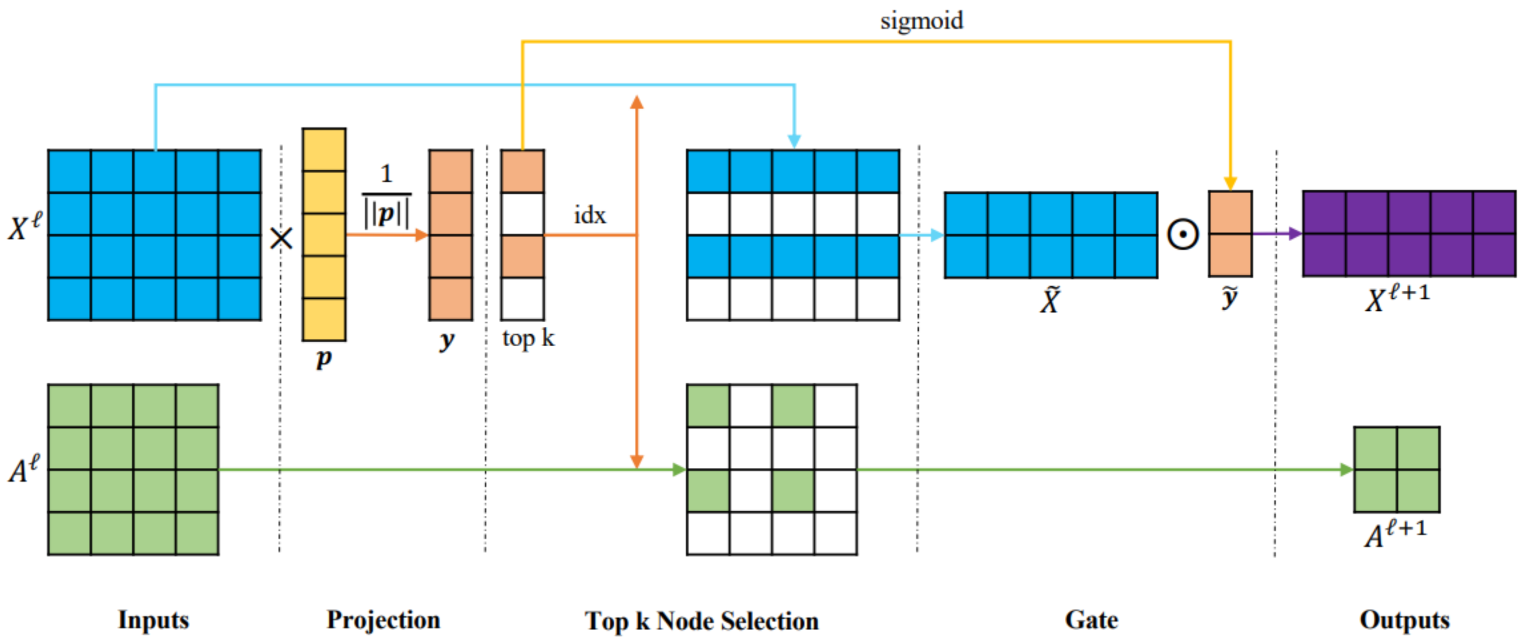


$\times$: matrix multiplication
$\odot$: element-wise product
p: trainable vector

gPool: **Graph U-Nets** (Gao et al. ICML 2019)
https://arxiv.org/pdf/1905.05178.pdf

GCN: gPool

Downsample by selecting the most important nodes



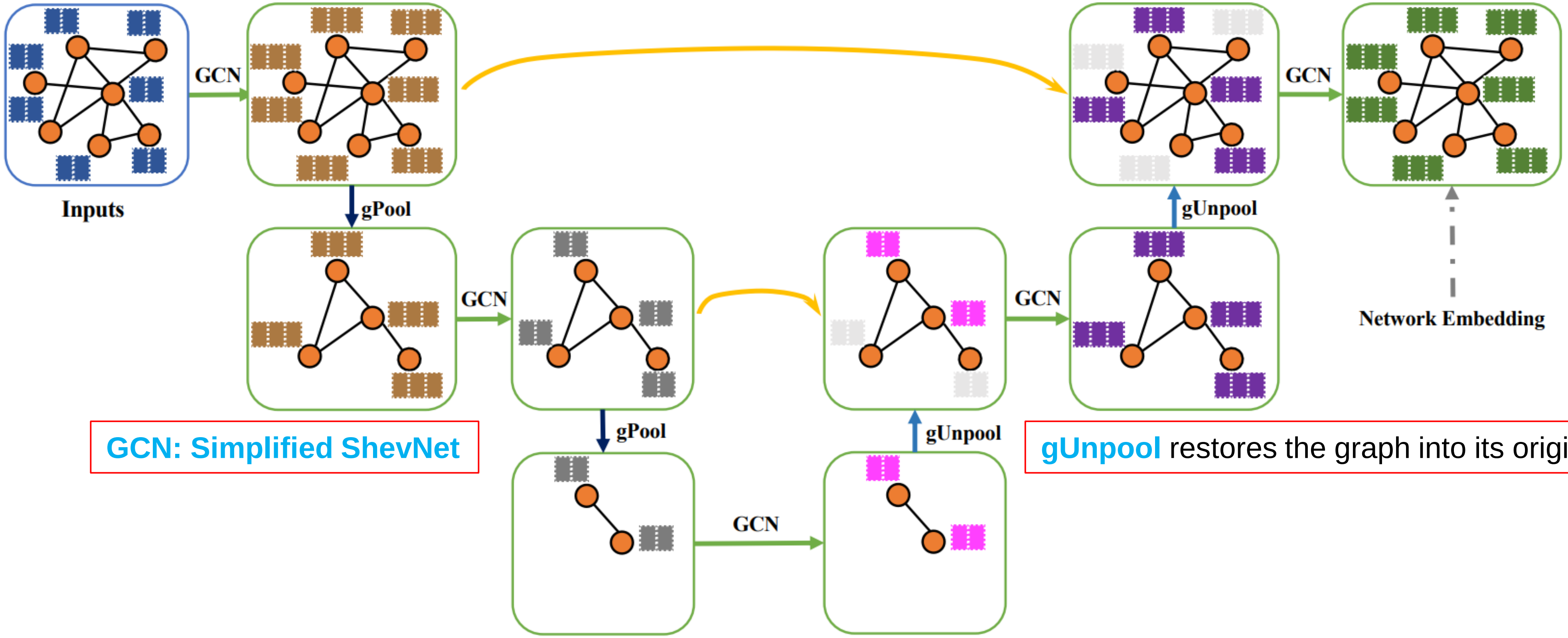
$$\begin{aligned}
\mathbf{y} &= X^\ell \mathbf{p}^\ell / \|\mathbf{p}^\ell\|, \\
\text{idx} &= \text{rank}(\mathbf{y}, k), \\
\tilde{\mathbf{y}} &= \text{sigmoid}(\mathbf{y}(\text{idx})), \\
\tilde{X}^\ell &= X^\ell(\text{idx}, :), \\
A^{\ell+1} &= A^\ell(\text{idx}, \text{idx}), \\
X^{\ell+1} &= \tilde{X}^\ell \odot \left(\tilde{\mathbf{y}} \mathbf{1}_C^T\right),
\end{aligned}$$

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

gPool: Graph U-Nets (Gao et al. ICML 2019)
https://arxiv.org/pdf/1905.05178.pdf

GCN: gPool

Graph U-Nets



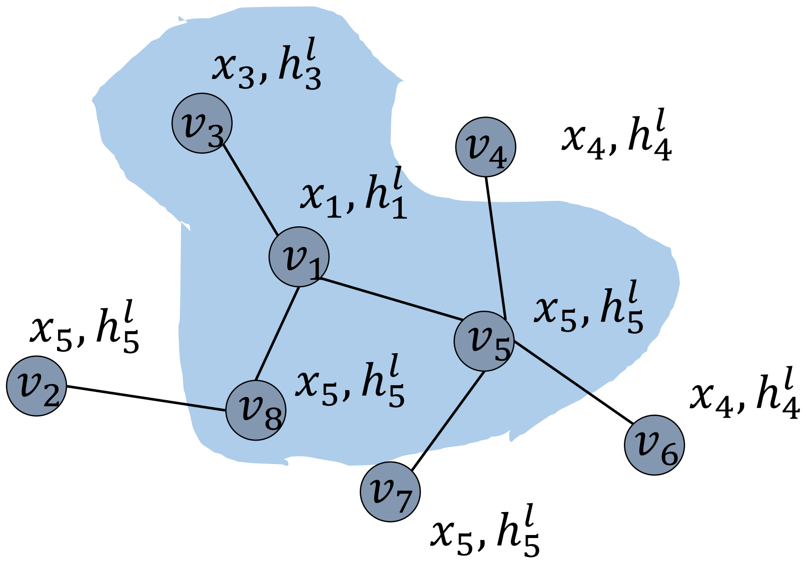
GCN: Simplified ShevNet

gUnpool restores the graph into its original structure

gPool: **Graph U-Nets** (Gao et al. ICML 2019)
https://arxiv.org/pdf/1905.05178.pdf

GCN: DiffPool

Downsample by clustering the nodes using GNN



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

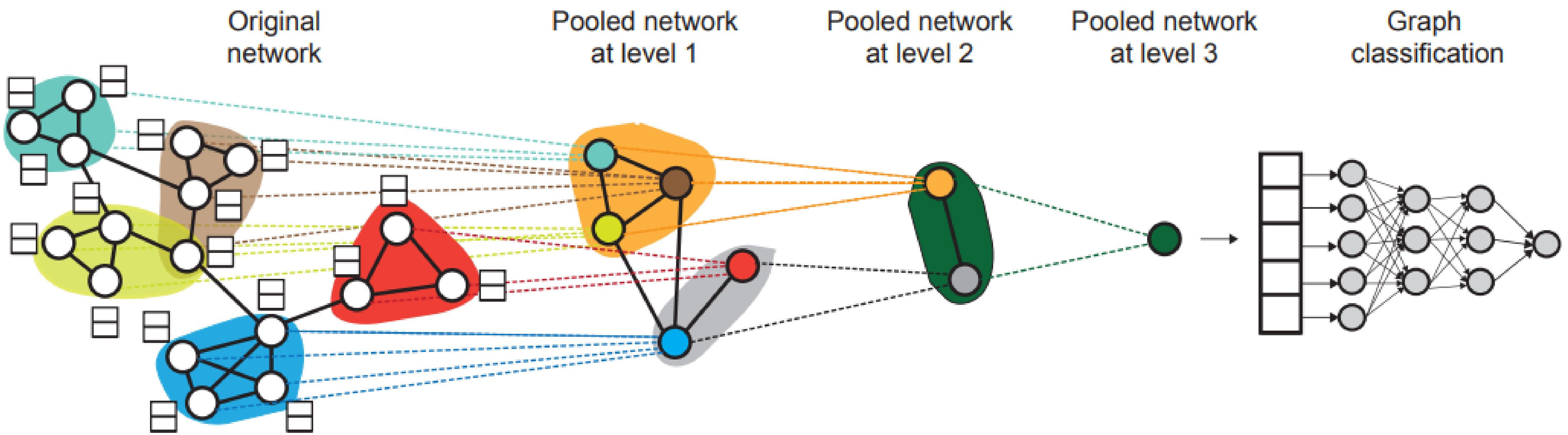
$$\downarrow$$

$$\boldsymbol{A_p} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n_p}\times\boldsymbol{n_p}}, \boldsymbol{H_p} \in \mathbb{R}^{\boldsymbol{n_p}\times\boldsymbol{d_p}}$$

DiffPool: Hierarchical Graph Representation Learning with Differentiable Pooling
(Ying et al. NeurIPS 2018) https://arxiv.org/pdf/1806.08804.pdf

GCN: DiffPool

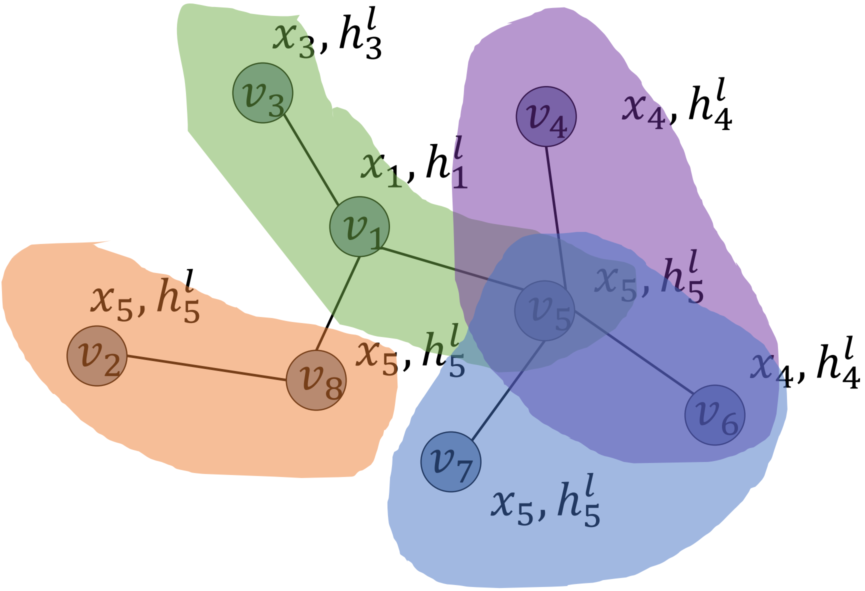
Downsample by clustering the nodes using GNN



DiffPool: Hierarchical Graph Representation Learning with Differentiable Pooling (Ying et al. NeurIPS 2018) https://arxiv.org/pdf/1806.08804.pdf

GCN: DiffPool

Downsample by clustering the nodes using GCN



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

$$\boldsymbol{A_p} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n_p}\times\boldsymbol{n_p}}, \boldsymbol{H_p} \in \mathbb{R}^{\boldsymbol{n_p}\times\boldsymbol{d_p}}$$

Assignment Matrix for pooling: $\boldsymbol{S} \in \mathbb{R}^{n\times n_p}$

$$\boldsymbol{S} = \boldsymbol{SoftMax}\left(\widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta_s}\right) \leftarrow \boldsymbol{GCN}$$

GCN Filtering(node embedding): $\boldsymbol{H} \in \mathbb{R}^{n\times d_p}$

$$\boldsymbol{H} = \boldsymbol{ReLU}\left(\widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta_h}\right) \leftarrow \boldsymbol{GCN}$$

DiffPool layer:

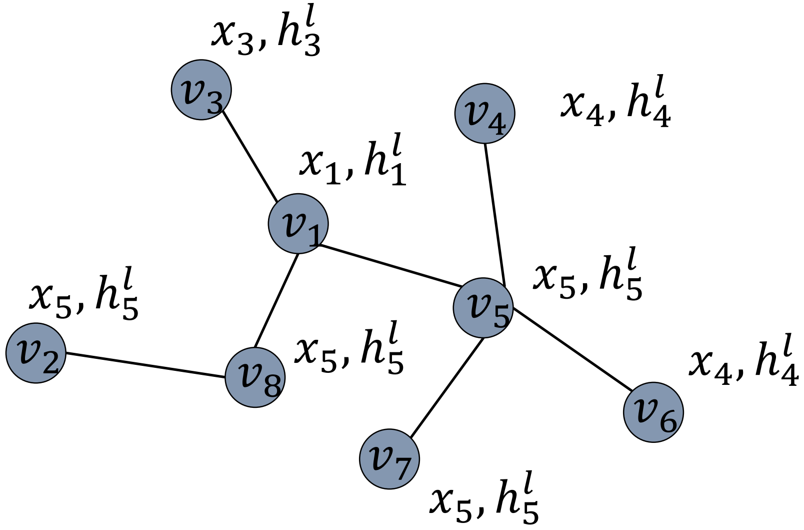
$$\boldsymbol{H_p} = \boldsymbol{S^T H} \in \mathbb{R}^{n_p\times d_p}$$

$$\boldsymbol{A_p} = \boldsymbol{S^T A S} \in \{\mathbf{0}, 1\}^{n_p\times n_p}$$

DiffPool: Hierarchical Graph Representation Learning with Differentiable Pooling
(Ying et al. NeurIPS 2018) https://arxiv.org/pdf/1806.08804.pdf

GCN: EigenPooling

Using Laplacian clustering



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

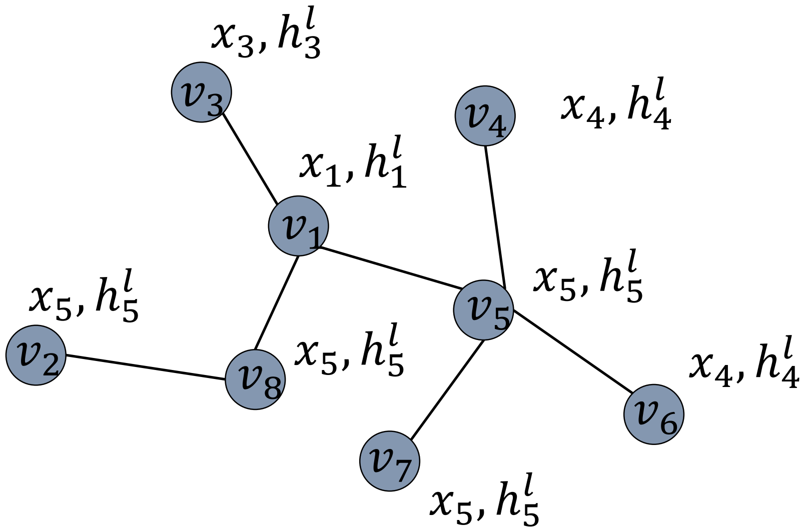
$$\Downarrow$$

$$\boldsymbol{A_p} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n_p} \times \boldsymbol{n_p}}, \boldsymbol{H_p} \in \mathbb{R}^{\boldsymbol{n_p} \times \boldsymbol{d_p}}$$

EigenPooling: Graph Convolutional Networks with EigenPooling
(Ma et al. KDD 2019) https://arxiv.org/pdf/1904.13107.pdf

GCN: EigenPooling

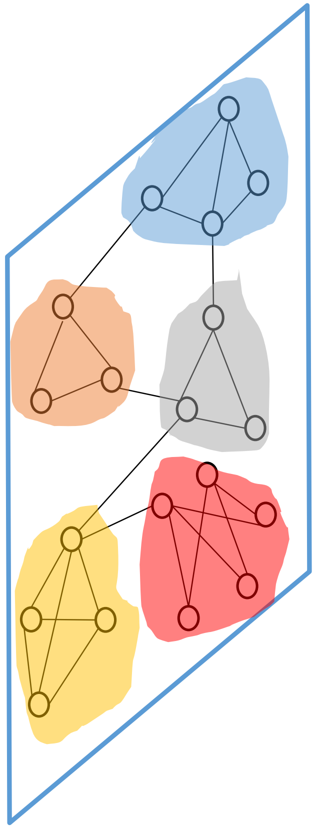
Using Laplacian clustering



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

$$\Downarrow$$

$$\boldsymbol{A_p} \in \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{n_p}\times\boldsymbol{n_p}}, \boldsymbol{H_p} \in \mathbb{R}^{\boldsymbol{n_p}\times\boldsymbol{d_p}}$$



Learn $\boldsymbol{A_p}$ using Laplacian clustering methods

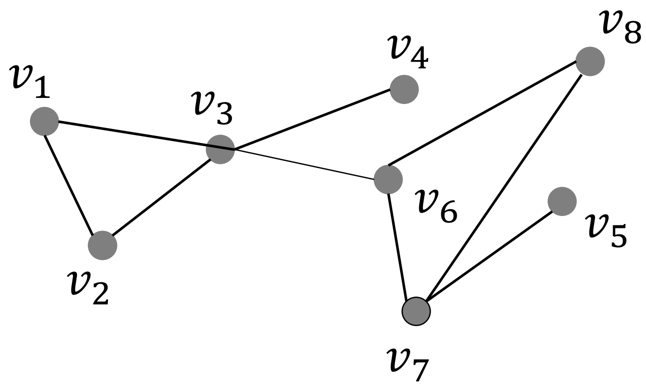
Focus on learning Better $\boldsymbol{H_p}$

Capture both feature and graph structure

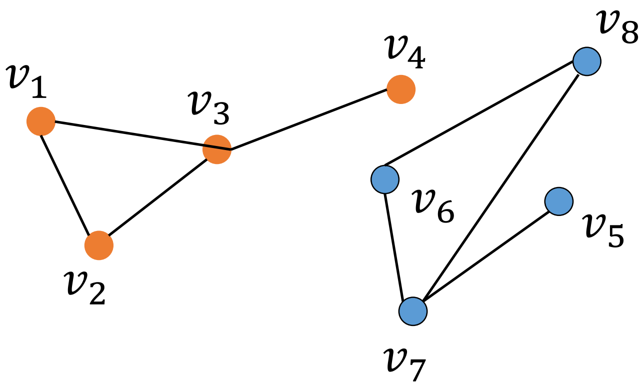
EigenPooling: Graph Convolutional Networks with EigenPooling
(Ma et al. KDD 2019) https://arxiv.org/pdf/1904.13107.pdf

R: Graph Spectral Theory

Laplacian Clustering



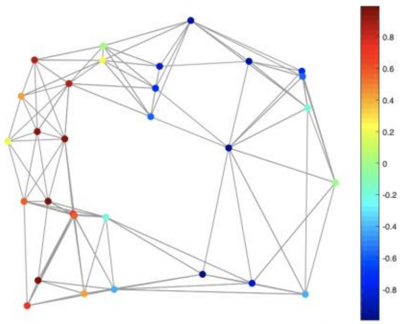
$$\boxed{\min_{\boldsymbol{f}} cut(\boldsymbol{A}, \boldsymbol{B}) = \frac{1}{2} \boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f}}$$



$$\boldsymbol{f} = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

R: Graph Spectral Theory

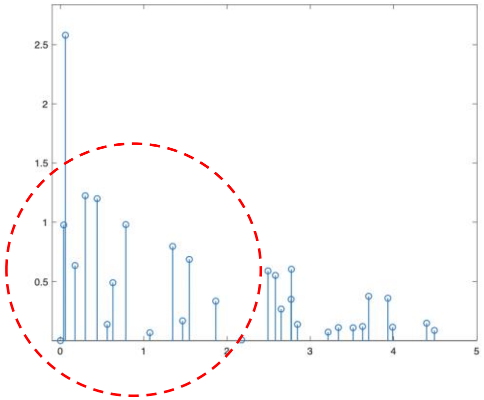
Laplacian Fourier Transform



Spatial domain: $\boldsymbol{f}$

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

Decompose signal f



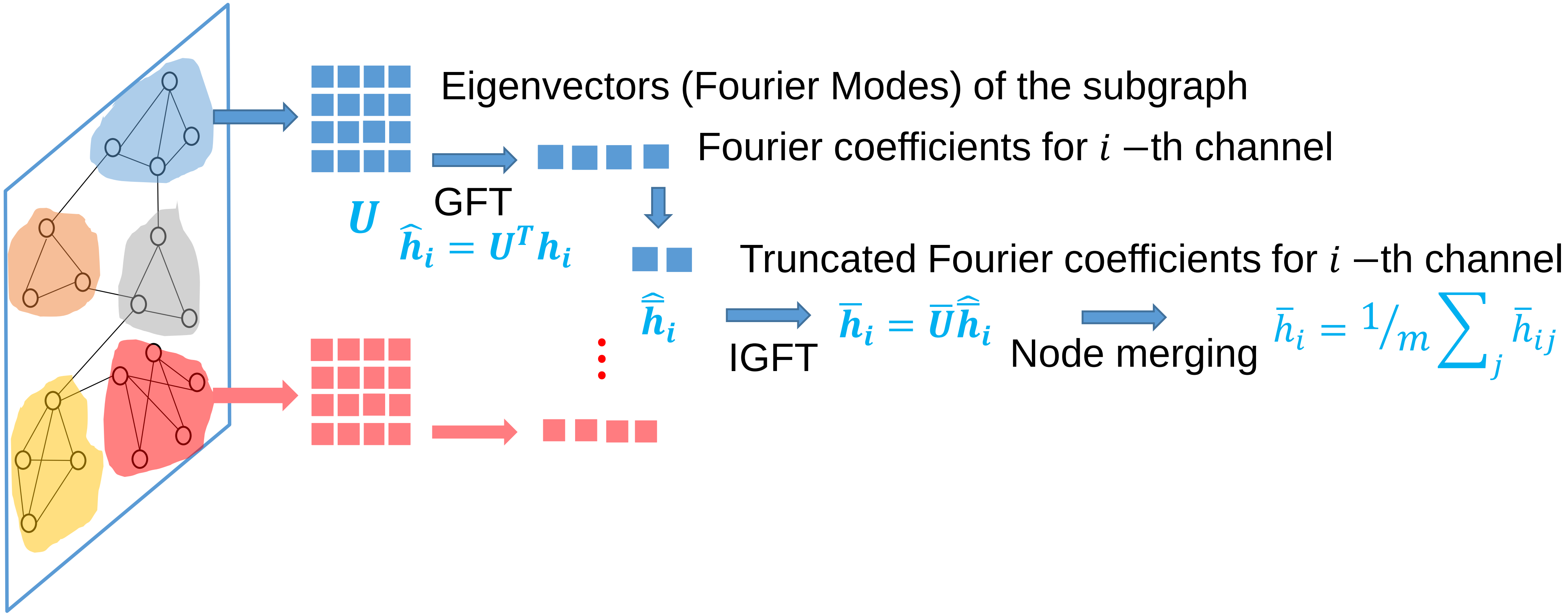
Spectral domain: $\hat{\boldsymbol{f}}$

$$\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}} = \hat{f}_0\boldsymbol{u}_0 + \hat{f}_1\boldsymbol{u}_1 + \cdots + \hat{f}_{N-1}\boldsymbol{u}_{N-1}$$

$$\bar{\boldsymbol{f}} = \bar{\boldsymbol{U}}\hat{\boldsymbol{f}} = \hat{f}_0\boldsymbol{u}_0 + \hat{f}_1\boldsymbol{u}_1 + \cdots + \hat{f}_{K-1}\boldsymbol{u}_{K-1}, \qquad K < N$$

GCN: EigenPooling

Truncated Fourier Coefficients



EigenPooling: Graph Convolutional Networks with EigenPooling
(Ma et al. KDD 2019) https://arxiv.org/pdf/1904.13107.pdf

Summary Questions of the lecture

- Explain the key aspects of GAT: Graph Attention Networks.
- Explain the key aspects of gPool: Graph U-Nets.
- Explain the key aspects of DiffPool: Hierarchical Graph Representation Learning with Differentiable Pooling
- Explain the key aspects of EigenPooling: Graph Convolutional Networks with EigenPooling