CS480/680: Introduction to Machine Learning

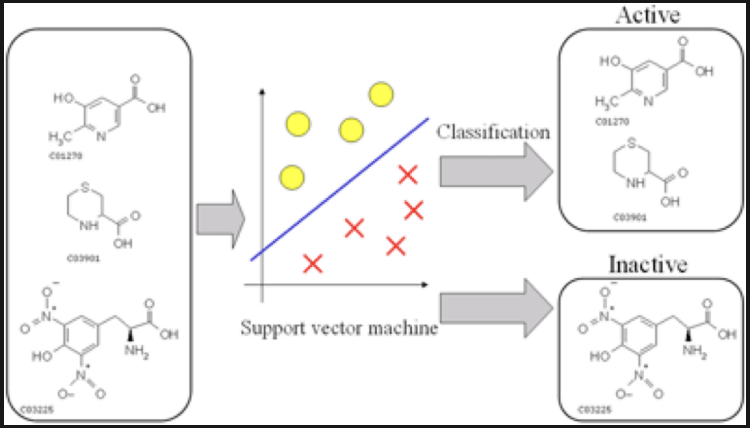
Lec 10: Graph Neural Network

Yaoliang Yu

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

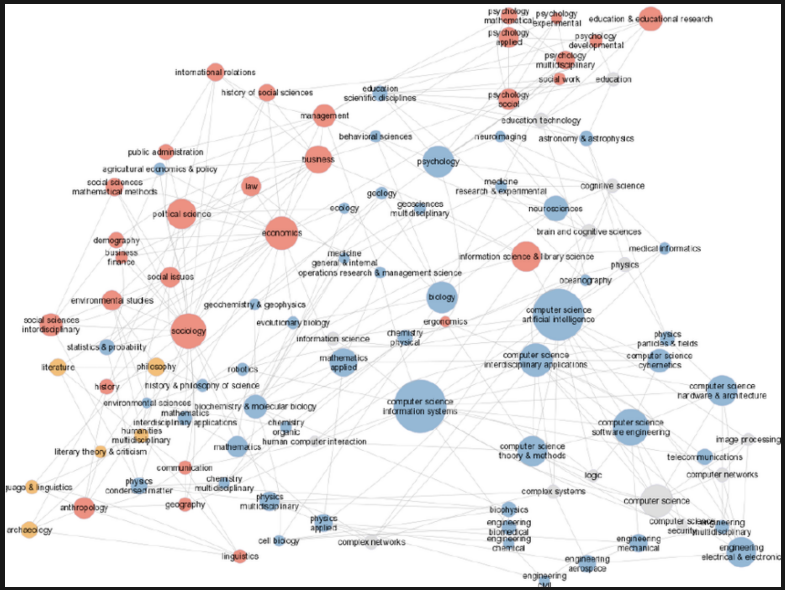
Feb 11, 2025

Chemical Compound



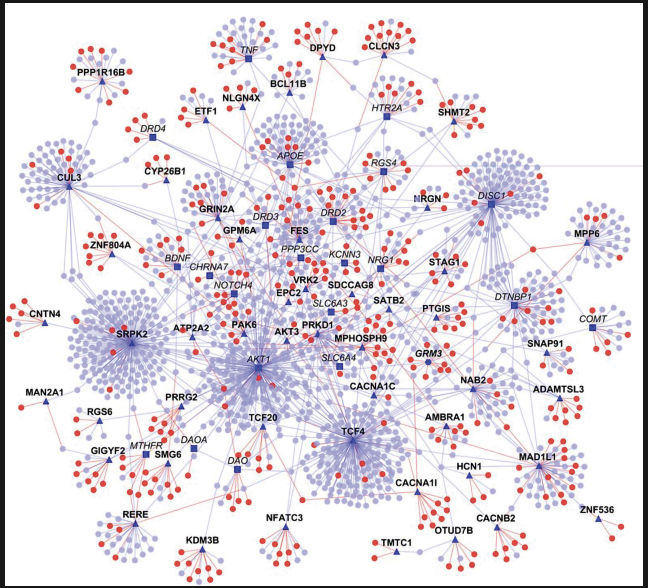
- Nodes are not necessarily in correspondence
- Output: e.g., function mapping unseen compound to level of activity against cancel cells

Collaboration Network



Social Network

Biological Network

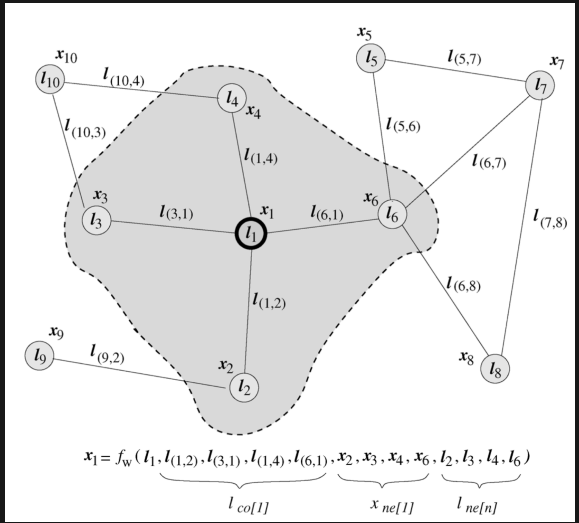


Communication Network

Traffic Network

Graph Neural Networks (GNN)

- Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathfrak{l})$
- State (embedding):
  $h_u = f_{\mathbf{w}}(h_{ne[u]}, l_{ne[u]}, l_{co[u]})$
- Output: $o_u = g_{\mathbf{w}}(h_u, l_u)$
- f, g: neural nets
- Parameters: $\mathbf{w}$, shared among nodes



F. Scarselli et al. "The Graph Neural Network Model". *IEEE Transactions on Neural Networks*, vol. 20, no. 1 (2009), pp. 61–80.

State Update

$$\left.\begin{aligned} h_1 &= f_{\mathbf{w}}(h_{ne[1]}, l_{ne[1]}, l_{co[1]}) \\ &\vdots \\ h_n &= f_{\mathbf{w}}(h_{ne[n]}, l_{ne[n]}, l_{co[n]}) \end{aligned}\right\} \implies \mathbf{h} = F_{\mathbf{w}}(\mathbf{h}, \mathfrak{l})$$

- If $F_{\mathbf{w}}$ is a contraction, then exists a unique state $\mathbf{h}$:

$$\|F_{\mathbf{w}}(\mathbf{h}, \mathfrak{l}) - F_{\mathbf{w}}(\mathbf{z}, \mathfrak{l})\| \leq \gamma \|\mathbf{h} - \mathbf{z}\|, \text{ for some } \gamma \in [0, 1)$$

- $\mathbf{h}^{t+1} \leftarrow F_{\mathbf{w}}(\mathbf{h}^t, \mathfrak{l})$ converges linearly to the fixed point $\mathbf{h}$
- Upon convergence, output $\mathbf{o} = G_{\mathbf{w}}(\mathbf{h}, \mathfrak{l}) =: \hat{\mathbf{y}}(\mathfrak{l}; \mathbf{w})$

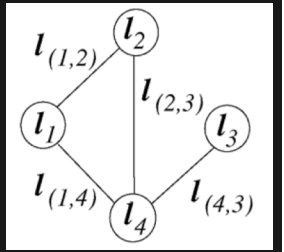
Example

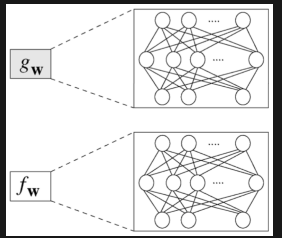
$$h_v = \frac{1}{|ne[v]|} \sum_{u \in ne[v]} \varphi_{\mathbf{w}}(h_u, l_v, l_{(v,u)}, l_u)$$

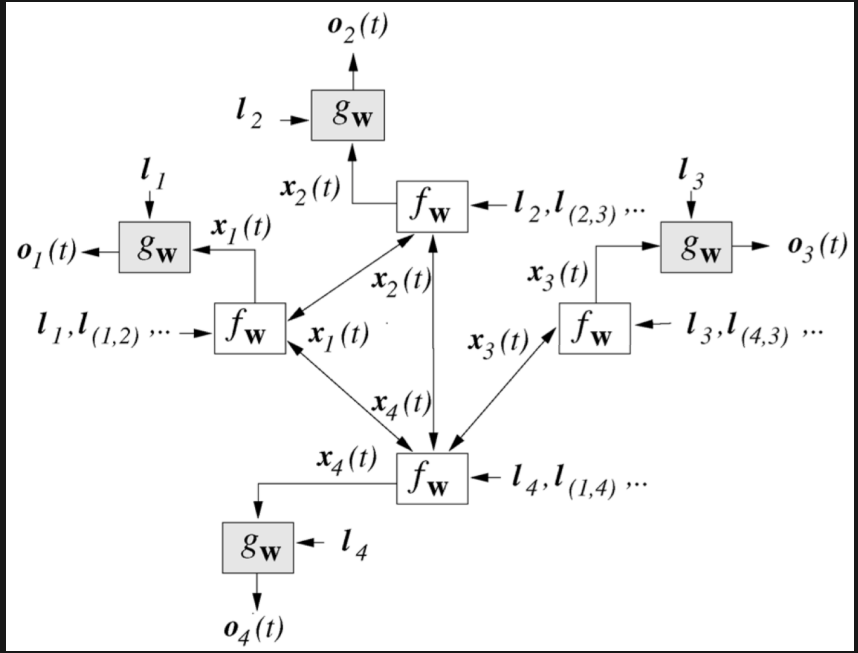
- State h_u, node feature l_u and the edge feature $l_{(v,u)}$ can all be vectors
- Graph structure is used in the sum: only neighbors contribute
- PageRank:

$$h_v = \sum_{v \in ne[u]} a_{vu} h_u, \quad e.g. \;\; a_{vu} := \frac{1}{|ne[u]|} [\![v \in ne[u]]\!]$$

- In GNN, the aggregation function $\varphi_{\mathbf{w}}$ is learnable through $\mathbf{w}$


l_2
$l_{(1,2)}$
$l_{(2,3)}$
l_1
l_3
$l_{(1,4)}$
$l_{(4,3)}$
l_4


$g_{\mathbf{w}}$
$f_{\mathbf{w}}$


$o_2(t)$
l_2
$g_{\mathbf{w}}$
l_1
$x_2(t)$
$f_{\mathbf{w}}$
$l_2, l_{(2,3)}, \ldots$
l_3
$o_1(t)$
$g_{\mathbf{w}}$
$x_1(t)$
$g_{\mathbf{w}}$
$o_3(t)$
$x_2(t)$
$x_3(t)$
$l_1, l_{(1,2)}, \ldots$
$f_{\mathbf{w}}$
$x_1(t)$
$x_3(t)$
$f_{\mathbf{w}}$
$l_3, l_{(4,3)}, \ldots$
$x_4(t)$
$x_4(t)$
$f_{\mathbf{w}}$
$l_4, l_{(1,4)}, \ldots$
$g_{\mathbf{w}}$
l_4
$o_4(t)$

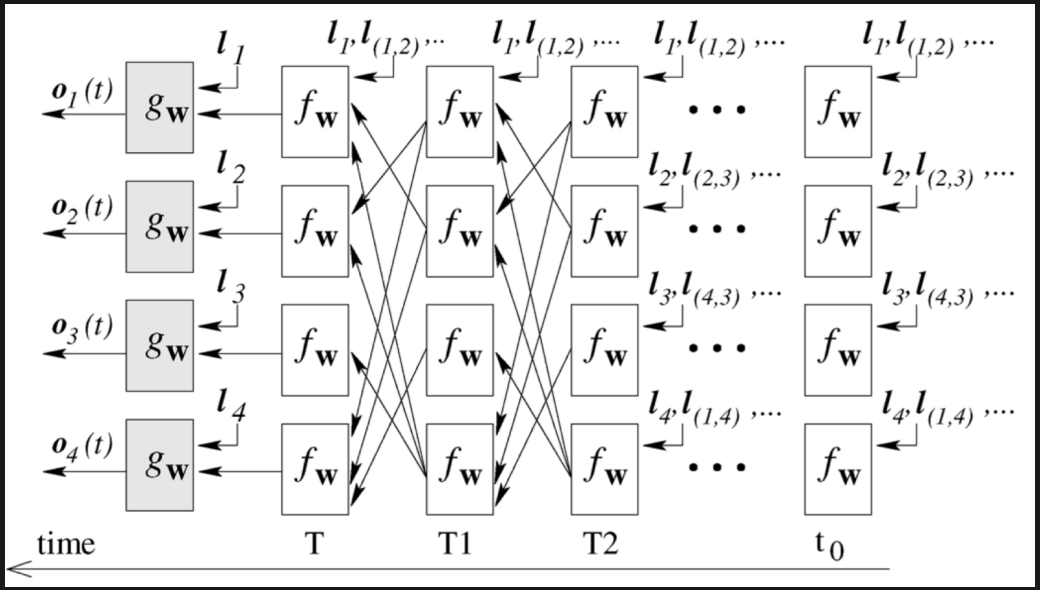
Learning GNN

- Given a supervised set of graphs and labels $(\mathcal{G}_i, \mathbf{y}_i),\ i = 1, \ldots, n$
- Learn a predictor $\hat{\mathbf{y}}$ that maps a new test graph $\mathcal{G}$ to its label: $\hat{\mathbf{y}}(\mathcal{G}) \approx \mathbf{y}$
  - labels could be at the node, edge or graph level
  - do not confuse the label $\mathbf{y}$ with the feature $\mathbf{l}$
- Choose a loss function L to solve:

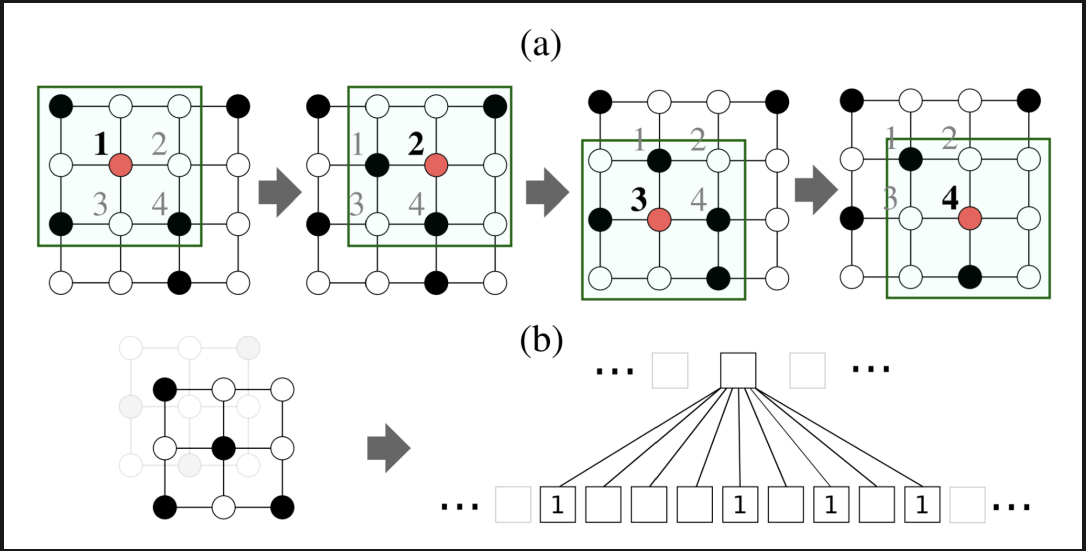
$$\min_{\mathbf{w}} \quad L(\hat{\mathbf{y}}(\mathbf{l}; \mathbf{w}), \mathbf{y})$$

  - unroll k steps: $\hat{\mathbf{y}}(\mathbf{l}; \mathbf{w}) \approx G_{\mathbf{w}}(F_{\mathbf{w}}^{[k]}(\mathbf{h}, \mathbf{l}), \mathbf{l})$
  - or apply implicit function theorem to differentiate $\mathbf{w}$

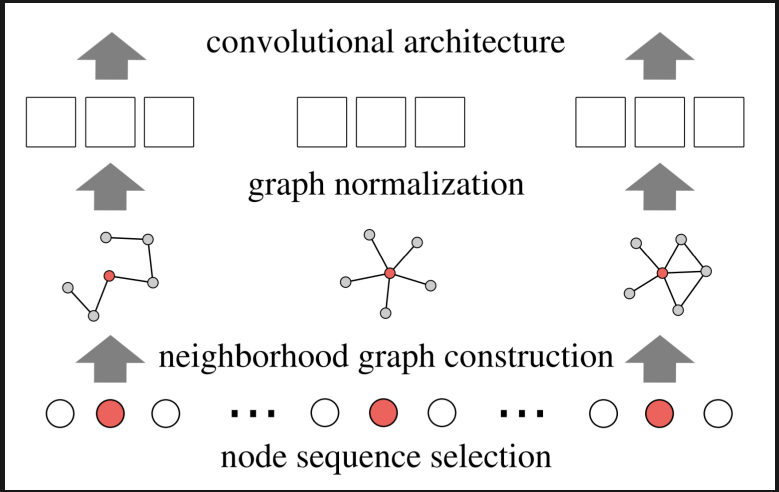
Training by Backpropagation Through Time (BPTT)



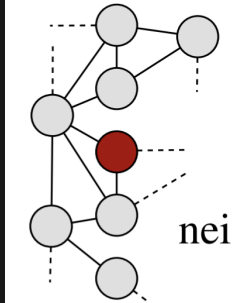
CNN Recalled



Spatial Convolution

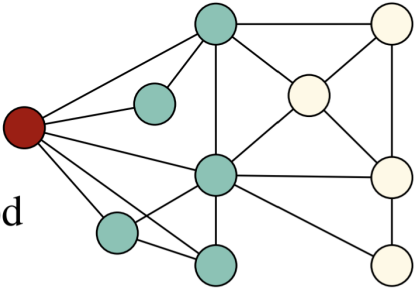


M. Niepert, M. Ahmed, and K. Kutzkov. "Learning Convolutional Neural Networks for Graphs". In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016, pp. 2014–2023.



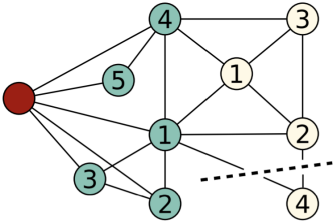
select

neighborhood



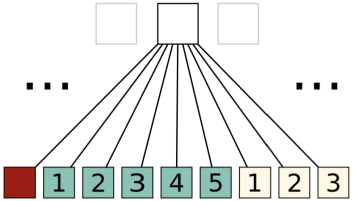
normalize

subgraph



receptive field

reads vertex and edge attributes = channels



Fourier Transform

$$f * g = \mathscr{F}^{-1}\left(\mathscr{F}[f] \cdot \mathscr{F}[g]\right)$$

- Convolution in time domain = multiplication in frequency domain
- Invertible, in fact, orthogonal transform
- Can we do something similar for graphs?

Graph Laplacian

$$A_{uv} = [\![(u,v) \in \mathcal{E}]\!], \qquad D_{uv} = \sum_n A_{un} \cdot [\![u = v]\!]$$

- Laplacian $L = D - A$
  - $L\mathbf{1} = 0 \cdot \mathbf{1}$: # connected components of $\mathcal{G}$ = # multiplicity of $\lambda = 0$
  - symmetric and PSD for undirected graph: $\langle \mathbf{x}, L\mathbf{x} \rangle = \frac{1}{2} \sum_{u,v} A_{uv}(x_u - x_v)^2$
- Normalized Laplacian $\bar{L} = I - D^{-1/2}AD^{-1/2} = D^{-1/2}LD^{-1/2}$

Example

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \qquad D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \mathrm{diag}([2, 2, 2])$$

$$L = D - A = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}, \qquad \bar{L} = D^{-1/2} L D^{-1/2} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}$$

$$\mathring{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad \mathring{D} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Spectral Convolution

- Given two graph signals $\mathbf{x}, \mathbf{g} \in \mathbb{R}^{|\mathcal{V}|}$:

$$\mathbf{x} * \mathbf{g} := U[(U^\top \mathbf{x}) \odot (U^\top \mathbf{g})], \quad \text{where} \quad L = U\Lambda U^\top$$

- One layer of spectral convolution, with filter weights W_r^k to be learned:

$$\mathbf{x}_r^{k+1} := \sigma\Big(\overbrace{\underbrace{U[W_r^k \odot (U^\top X^k)]\,\mathbf{1}}_{\text{aggregation along depth}}}^{\text{conv by } r\text{-th filter}} \Big), \quad r = 1, \ldots, d_{k+1}, \quad X^k = [\mathbf{x}_1^k, \ldots, \mathbf{x}_{d_k}^k]$$

- Can stack to go deep

J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. "Spectral Networks and Locally Connected Networks on Graphs". In: *International Conference on Learning Representations*. 2014.

Chebyshev Net

- Spectral conv requires eigen-decomposition and is not localized
- Rewrite the convolution:

$$\begin{aligned}\mathbf{x} * \mathbf{g} &:= U[(U^\top \mathbf{x}) \odot (U^\top \mathbf{g})] = U[\mathrm{diag}(f(\lambda; \mathbf{w}))(U^\top \mathbf{x})] \\ &= [U\,\mathrm{diag}(f(\lambda; \mathbf{w}))U^\top]\mathbf{x} \\ &:= f(L; \mathbf{w})\mathbf{x}\end{aligned}$$

- Choosing f to be a polynomial dispenses eigen-decomposition
- Resulting conv is localized: degree k polynomial only reqiures k-hop neighbors

M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29*. 2016, pp. 3844–3852.

Graph Convolutional Net (GCN)

- A layer of GCN is defined simply as:

$$X^{k+1} = \sigma\big(\mathring{D}^{-1/2}\mathring{A}\mathring{D}^{-1/2}X^kW^k\big), \quad X^k = [\mathbf{x}_1^k, \ldots, \mathbf{x}_s^k] \in \mathbb{R}^{|\mathcal{V}|\times s}, \quad W^k \in \mathbb{R}^{s\times t}$$

  - $\mathring{A} = A + I$ (i.e. adding self-cycle)
  - $\mathring{D}$ is the usual diagonal degree matrix of $\mathring{A}$
  - s and t are the number of input and output channels, resp.

- One layer of GCN only aggregates info from 1-hop neighbors
- Can stack to get deep and aggregate info from k-hop neighbors

T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations*. 2017.

Connections

- Rewriting GCN in vector form and identify $X_{v:} = \mathfrak{l}_v$:

$$\mathfrak{l}_v^{k+1} = \sigma\left(\left[\frac{1}{d_v+1}\mathfrak{l}_v^k + \sum_{u\in\mathcal{N}_v}\frac{a_{vu}}{\sqrt{(d_v+1)(d_u+1)}}\mathfrak{l}_u^k\right]W^k\right)$$

- This is GNN!

- It also resembles the Weisfeiler-Lehman (WL) algorithm!

WL and Graph Isomorphism

Algorithm 1: Weisfeiler-Lehman iterative color refinement

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathfrak{l}^0)$

Output: $\mathfrak{l}^{|\mathcal{V}|-1}$

1 **for** $t = 0, 1, \ldots, |\mathcal{V}| - 1$ **do**

2 $\quad \mathfrak{l}^{t+1} \leftarrow \texttt{hash}\big([\mathfrak{l}_v^t, \mathfrak{l}_{u\in\mathcal{N}_v}^t] : v \in \mathcal{V}\big)$ `// [·] is a multiset, allowing repetitions`

3 **Function** $\texttt{hash}\big([\mathfrak{l}_v, \mathfrak{l}_{u\in\mathcal{N}_v}] : v \in \mathcal{V}\big)$:

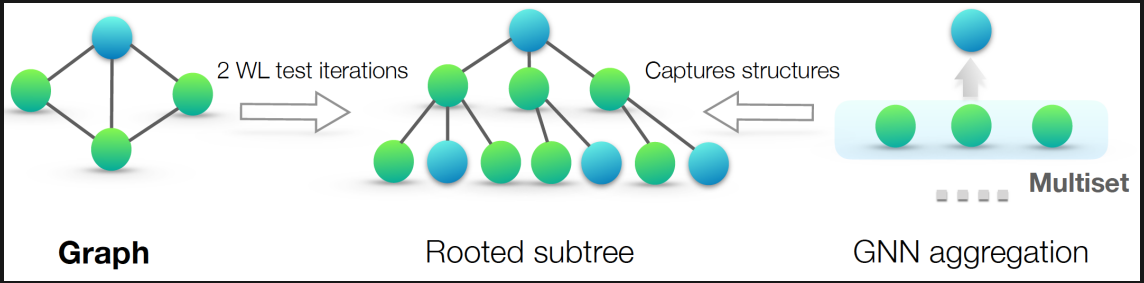
4 $\quad$ **for** $v \in \mathcal{V}$ **do**

5 $\quad\quad \texttt{sort}\big(\mathfrak{l}_{u\in\mathcal{N}_v}\big)$ `// sort the neighbors`

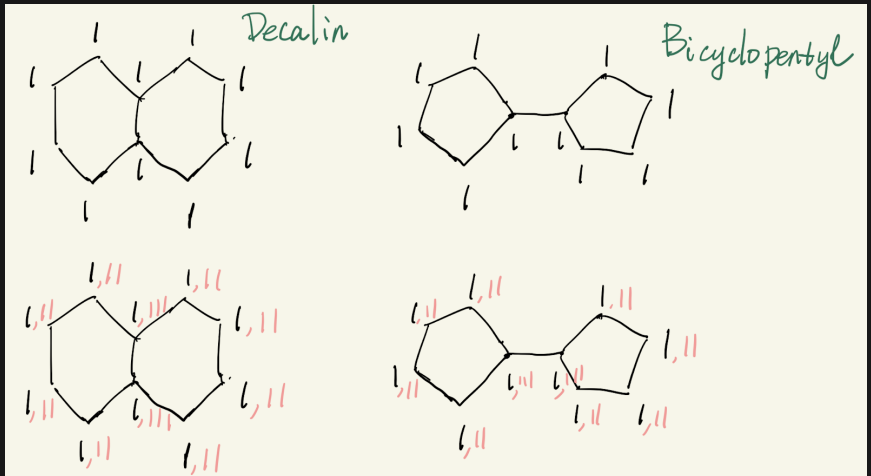
6 $\quad\quad$ prefix $\mathfrak{l}_v$ to sorted list $[\mathfrak{l}_v, \mathfrak{l}_{u\in\mathcal{N}_v}]$ `// ` $\mathfrak{l}_v$ `does not participate in sorting!`

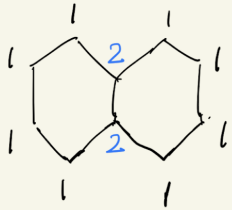
7 $\quad\quad \mathfrak{l}_v^+ \leftarrow f([\mathfrak{l}_v, \mathfrak{l}_{u\in\mathcal{N}_v}])$ `//` $f : \mathsf{L}^* \to \mathsf{L}$ `lexicographic strictly increasing`

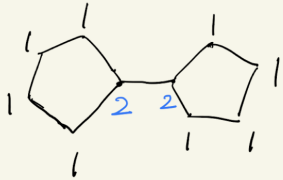
B. Weisfeiler and A. Lehman. "The reduction of a graph to canonical form and the algebra which appears therein". *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9 (1968), pp. 12–16.

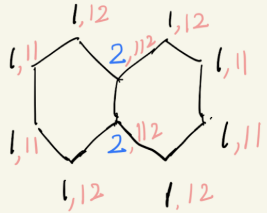

2 WL test iterations
Captures structures
Multiset
Graph
Rooted subtree
GNN aggregation

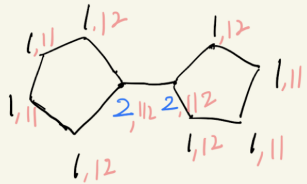
Example


Decalin
Bicyclopentyl
$f:(1,11)\longrightarrow 1$
$(1,111)\longrightarrow 2$





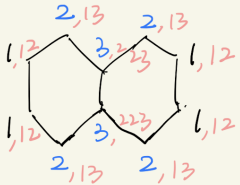


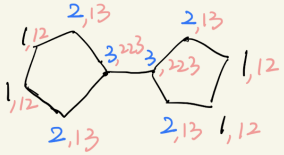


$f: (1,11) \to 1$

$(1,12) \to 2$

$(2,112) \to 3$

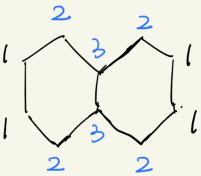




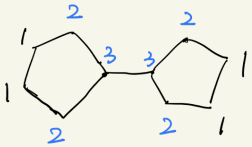
$f: (1, 12) \to 1$

$(2, 13) \to 2$

$(3, 223) \to 3$



$\neq$



converged !

WL cannot distinguish the two graphs.

From 1 to 2

- GCN: $\mathfrak{l}_v \leftarrow \sigma\Big(\overbrace{\Big[\underbrace{\frac{1}{d_v+1}\mathfrak{l}_v + \sum_{u\in\mathcal{N}_v} \frac{a_{vu}}{\sqrt{(d_v+1)(d_u+1)}}\mathfrak{l}_u}_{\text{averaged input}}\Big]W}^{\text{linear layer}}\Big)$

- GraphSAGE: $\mathfrak{l}_v \leftarrow \sigma\Big(\overbrace{\Big[\underbrace{\mathfrak{l}_v \vee \max_{u\in\mathcal{N}_v}\{\mathfrak{l}_u\}}_{\text{max-pooled input}}\Big]W}^{\text{linear layer}}\Big)$

W. Hamilton, Z. Ying, and J. Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 1024–1034.

Graph Isomorphism Network (GIN)

$$\mathfrak{l}_v \leftarrow \texttt{MLP}\Big(\underbrace{(1+\epsilon)\mathfrak{l}_v + \sum_{u \in \mathcal{N}(v)} \mathfrak{l}_u}_{\text{summed input}}\Big)$$

Theorem: Representation power of GNNs

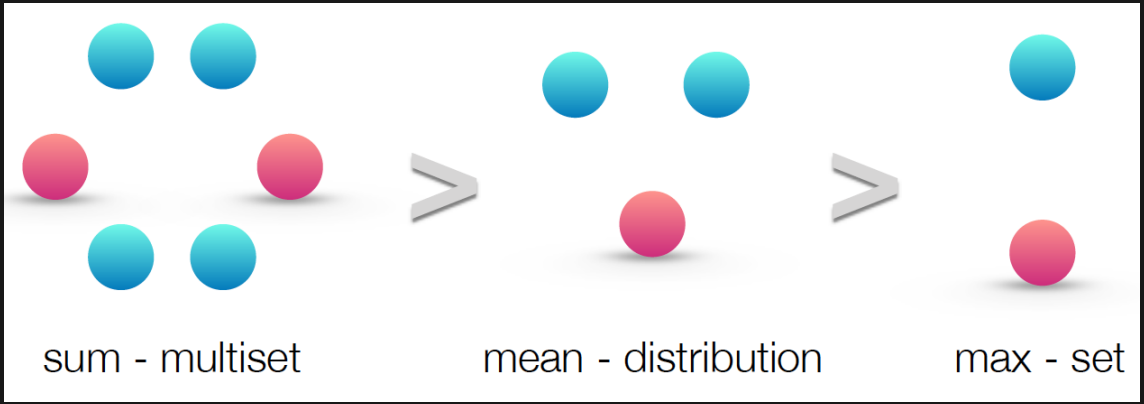
GNNs with aggregation function

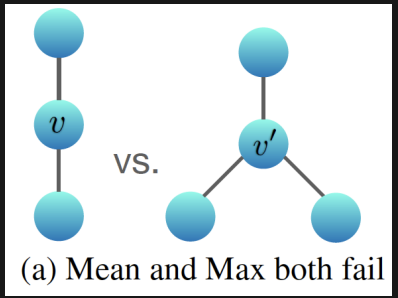
$$\mathfrak{l}_v \leftarrow \sigma\Big(\mathfrak{l}_v, \varphi\big(\mathfrak{l}_u : u \in \mathcal{N}(v)\big)\Big).$$

are no more discriminative than WL, with equality if φ and σ are injective.

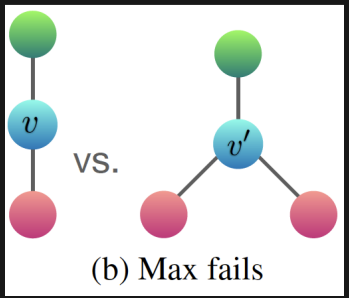
- Compared to WL, GNNs also tend to map similar nodes into similar embeddings

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019.

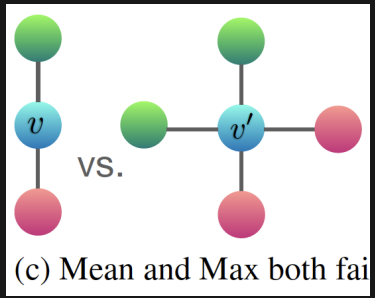

sum - multiset
mean - distribution
max - set



(a) Mean and Max both fail



(b) Max fails



(c) Mean and Max both fai