CS480/680: Introduction to Machine Learning

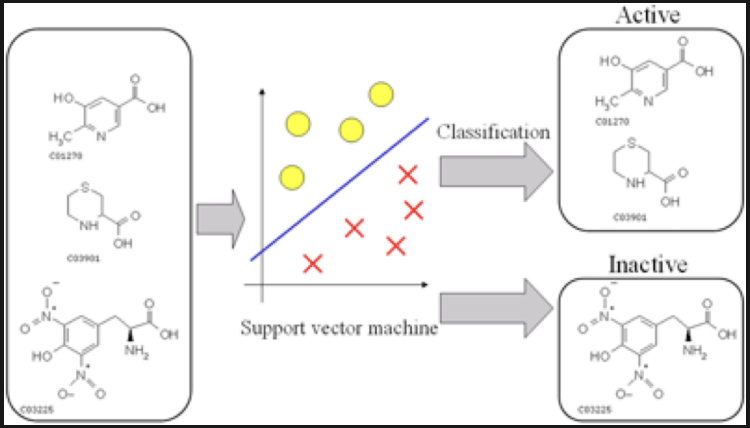
Lec 12: Graph Neural Network

Yaoliang Yu

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

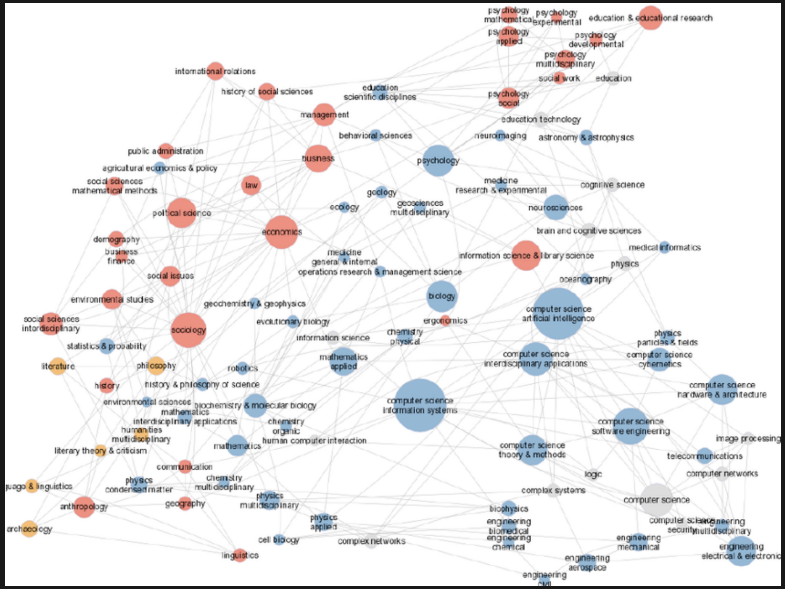
June 17, 2024

Chemical Compound



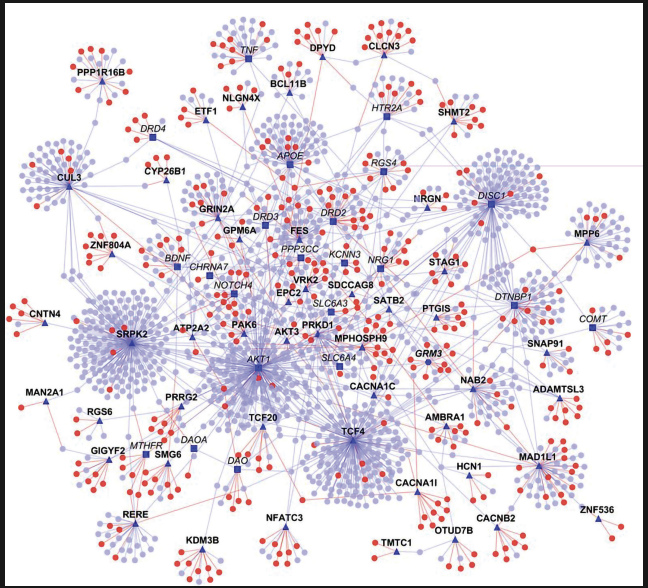
- Nodes are not necessarily in correspondence
- Output: e.g., function mapping unseen compound to level of activity against cancel cells

Collaboration Network



Social Network

Biological Network

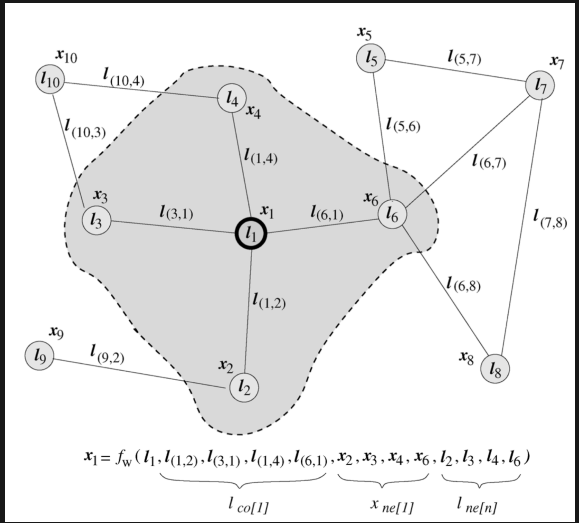


Communication Network

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

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

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

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

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

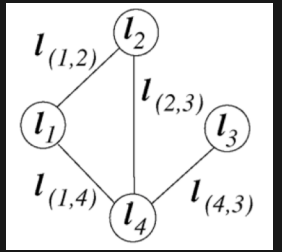
Example

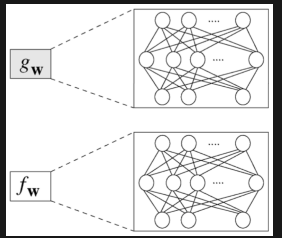
$$h_v = \frac{1}{|ne[v]|} \sum_{u \in ne[v]} \varphi_{\mathbf{w}}(h_u, l_v, l_{(v,u)}, l_u)$$

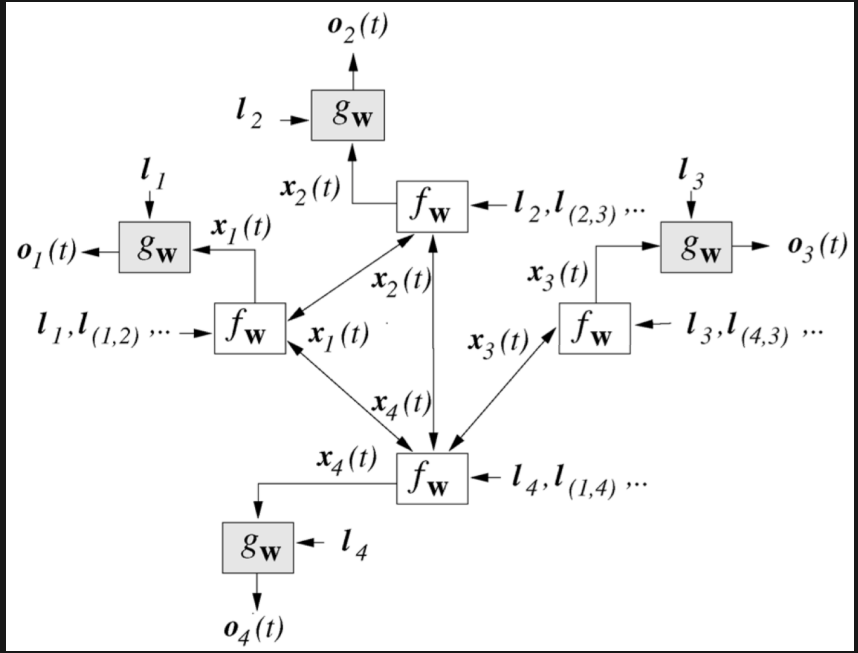
- State h_u, node feature l_u and the edge feature $l_{(v,u)}$ can all be vectors
- Graph structure is used in the sum: only neighbors contribute
- PageRank:

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

- 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_w
f_w


$o_2(t)$
l_2
g_w
l_1
$x_2(t)$
f_w
$l_2, l_{(2,3)} \cdots$
l_3
$o_1(t)$
g_w
$x_1(t)$
$x_2(t)$
$x_3(t)$
g_w
$o_3(t)$
$l_1, l_{(1,2)} \cdots$
f_w
$x_1(t)$
$x_3(t)$
f_w
$l_3, l_{(4,3)} \cdots$
$x_4(t)$
$x_4(t)$
f_w
$l_4, l_{(1,4)} \cdots$
g_w
l_4
$o_4(t)$

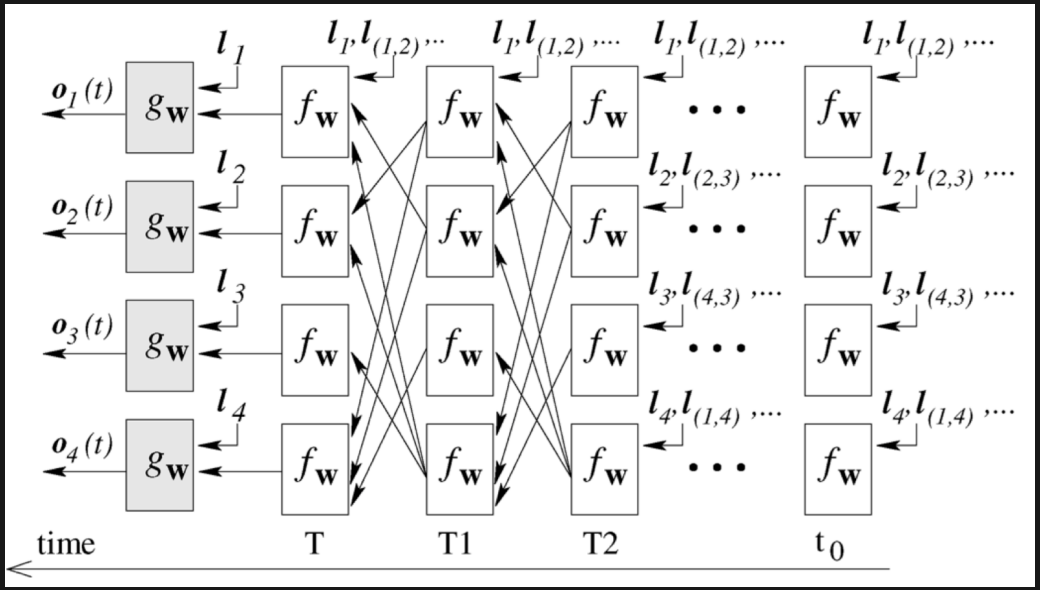
Learning GNN

- 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 $\mathfrak{l}$
- Choose a loss function L to solve:

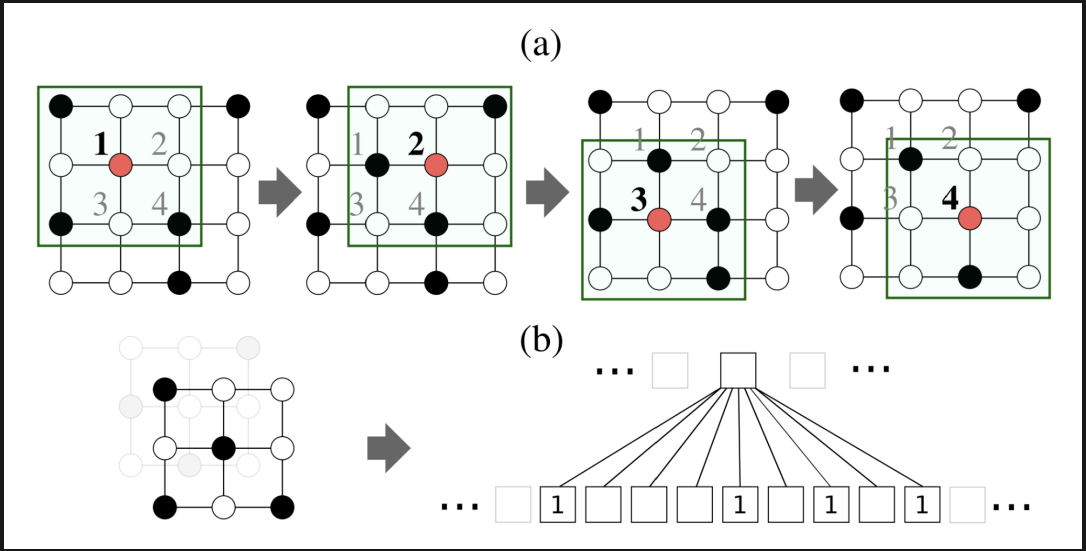
$$\min_{\mathbf{w}} \quad L(\hat{\mathbf{y}}(\mathfrak{l}; \mathbf{w}), \mathbf{y})$$

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

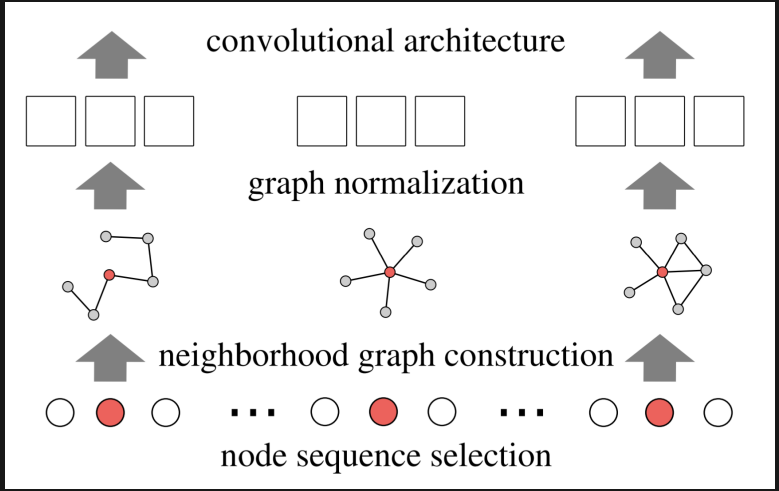
Training by Backpropagation Through Time (BPTT)



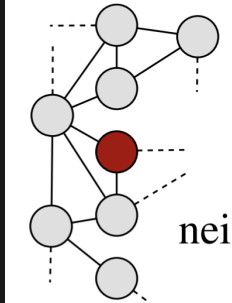
CNN Recalled



Spatial Convolution

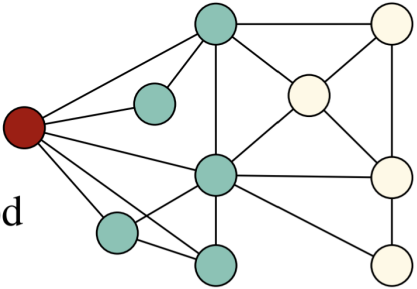


M. Niepert, M. Ahmed, and K. Kutzkov. "Learning Convolutional Neural Networks for Graphs". In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016, pp. 2014–2023.



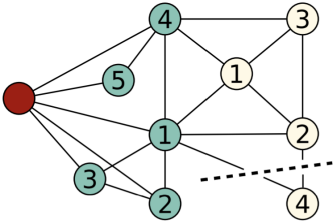
select

neighborhood



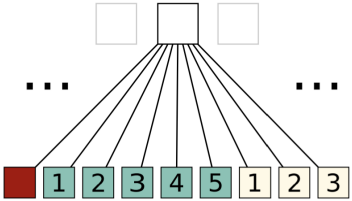
normalize

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 $\bar{L} = I - D^{-1/2} A D^{-1/2} = D^{-1/2} L D^{-1/2}$

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:

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

- 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:

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

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

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

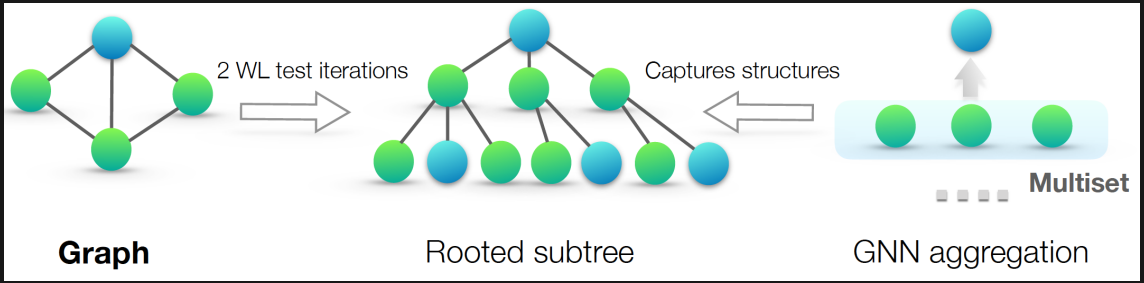
4 $\quad$ **for** $v \in \mathcal{V}$ **do**

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

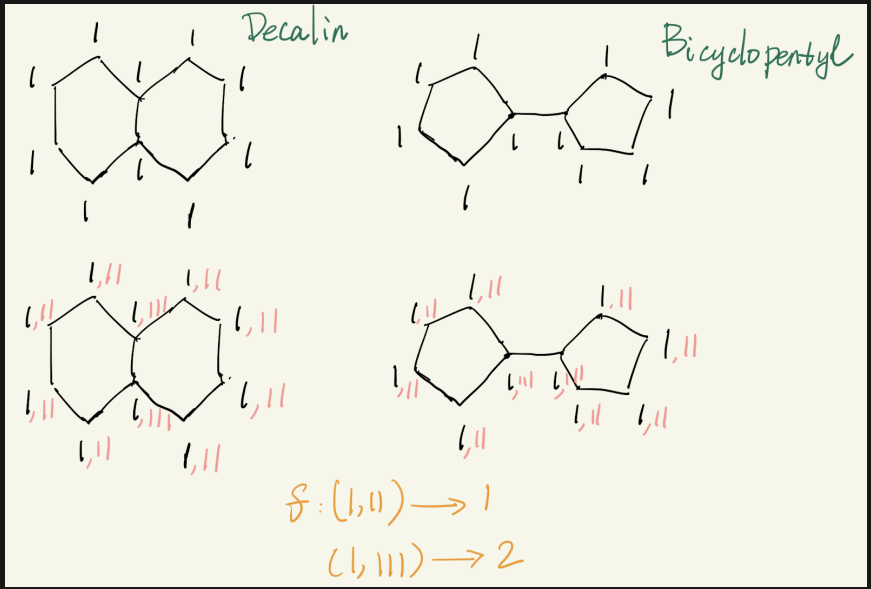
6 $\quad\quad$ prefix $\mathfrak{l}_v$ to sorted list $[\mathfrak{l}_v, \mathfrak{l}_{u\in\mathcal{N}_v}]$ `// ` $\mathfrak{l}_v$ `does not participate in sorting!`

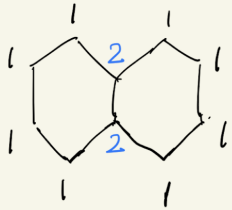
7 $\quad\quad \mathfrak{l}_v^+ \leftarrow f([\mathfrak{l}_v, \mathfrak{l}_{u\in\mathcal{N}_v}])$ `//` $f : \mathsf{L}^* \to \mathsf{L}$ `lexicographic strictly increasing`

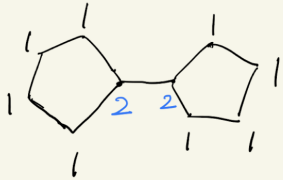
B. Weisfeiler and A. Lehman. "The reduction of a graph to canonical form and the algebra which appears therein". *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9 (1968), pp. 12–16.

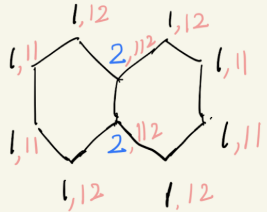

2 WL test iterations
Captures structures
Multiset
Graph
Rooted subtree
GNN aggregation

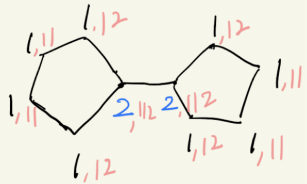
Example


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





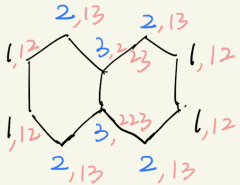


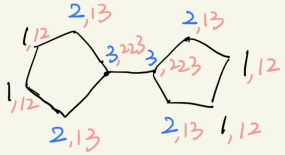


$f: (1,11) \to 1$

$(1,12) \to 2$

$(2,112) \to 3$

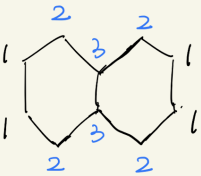




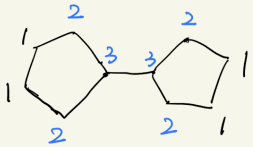
$f: (1, 12) \to 1$

$(2, 13) \to 2$

$(3, 223) \to 3$



$\neq$



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

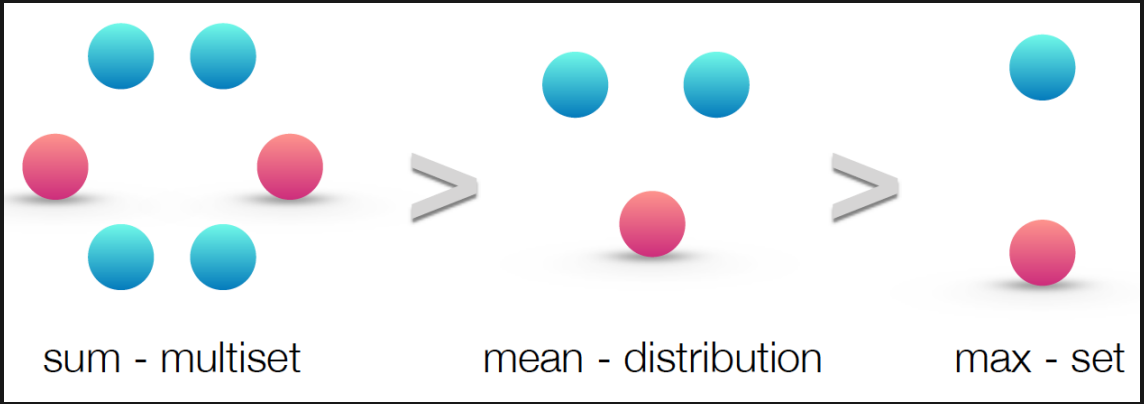
GNNs with aggregation function

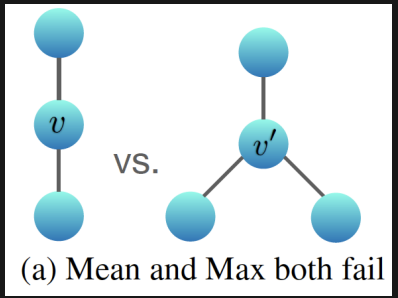
$$\mathfrak{l}_v \leftarrow \sigma\Big(\mathfrak{l}_v, \varphi\big(\mathfrak{l}_u : u \in \mathcal{N}(v)\big)\Big).$$

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

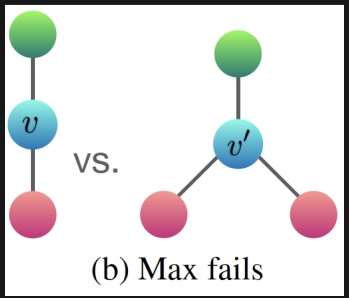
- Compared to WL, GNNs also tend to map similar nodes into similar embeddings

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

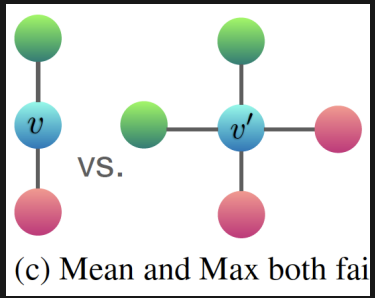

sum - multiset
mean - distribution
max - set



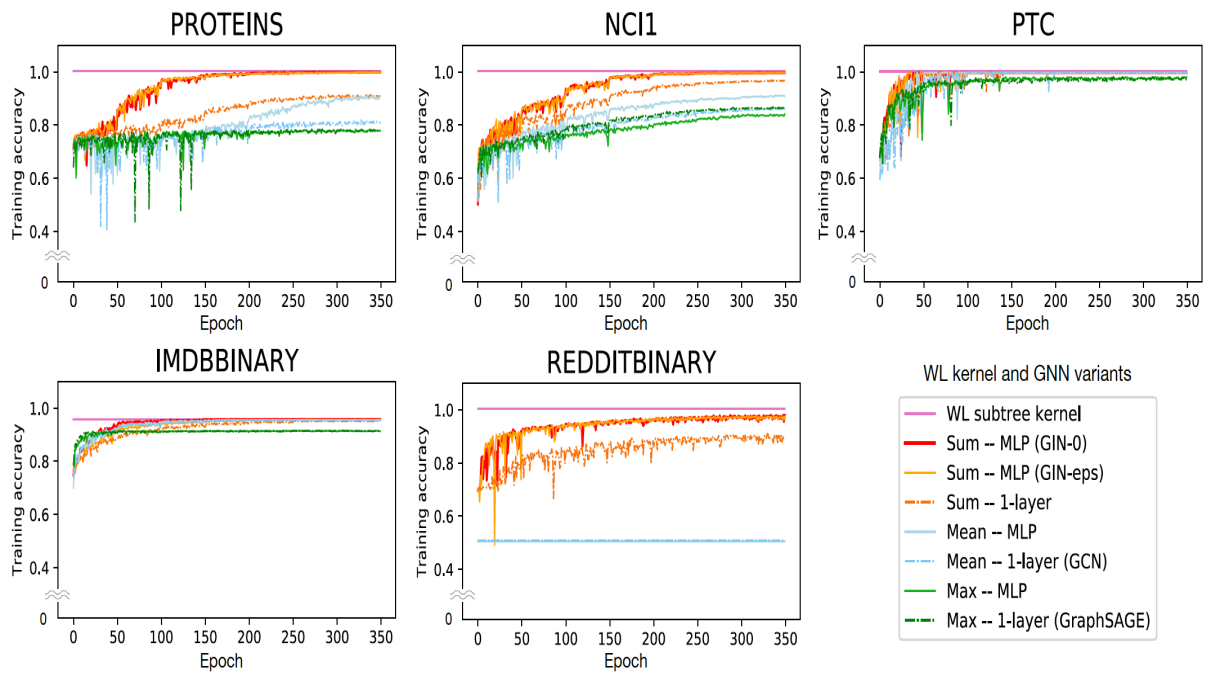
(a) Mean and Max both fail



(b) Max fails



(c) Mean and Max both fai


PROTEINS
NCI1
PTC
IMDBBINARY
REDDITBINARY
Training accuracy
Epoch
WL kernel and GNN variants
WL subtree kernel
Sum -- MLP (GIN-0)
Sum -- MLP (GIN-eps)
Sum -- 1-layer
Mean -- MLP
Mean -- 1-layer (GCN)
Max -- MLP
Max -- 1-layer (GraphSAGE)

	Datasets	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB	MUTAG	PROTEINS	PTC	NCI1
Datasets	# graphs	1000	1500	2000	5000	5000	188	1113	344	4110
	# classes	2	3	2	5	3	2	2	2	2
	Avg # nodes	19.8	13.0	429.6	508.5	74.5	17.9	39.1	25.5	29.8
Baselines	WL subtree	73.8 ± 3.9	50.9 ± 3.8	81.0 ± 3.1	52.5 ± 2.1	78.9 ± 1.9	90.4 ± 5.7	75.0 ± 3.1	59.9 ± 4.3	**86.0 ± 1.8** *
	DCNN	49.1	33.5	–	–	52.1	67.0	61.3	56.6	62.6
	PATCHYSAN	71.0 ± 2.2	45.2 ± 2.8	86.3 ± 1.6	49.1 ± 0.7	72.6 ± 2.2	**92.6 ± 4.2** *	75.9 ± 2.8	60.0 ± 4.8	78.6 ± 1.9
	DGCNN	70.0	47.8	–	–	73.7	85.8	75.5	58.6	74.4
	AWL	74.5 ± 5.9	51.5 ± 3.6	87.9 ± 2.5	54.7 ± 2.9	73.9 ± 1.9	87.9 ± 9.8	–	–	–
GNN variants	SUM-MLP (**GIN-0**)	**75.1 ± 5.1**	**52.3 ± 2.8**	**92.4 ± 2.5**	**57.5 ± 1.5**	**80.2 ± 1.9**	**89.4 ± 5.6**	**76.2 ± 2.8**	**64.6 ± 7.0**	**82.7 ± 1.7**
	SUM-MLP (**GIN-ϵ**)	**74.3 ± 5.1**	**52.1 ± 3.6**	**92.2 ± 2.3**	**57.0 ± 1.7**	**80.1 ± 1.9**	**89.0 ± 6.0**	**75.9 ± 3.8**	63.7 ± 8.2	**82.7 ± 1.6**
	SUM-1-LAYER	74.1 ± 5.0	**52.2 ± 2.4**	90.0 ± 2.7	55.1 ± 1.6	**80.6 ± 1.9**	**90.0 ± 8.8**	**76.2 ± 2.6**	63.1 ± 5.7	82.0 ± 1.5
	MEAN-MLP	73.7 ± 3.7	**52.3 ± 3.1**	50.0 ± 0.0	20.0 ± 0.0	79.2 ± 2.3	83.5 ± 6.3	75.5 ± 3.4	**66.6 ± 6.9**	80.9 ± 1.8
	MEAN-1-LAYER (GCN)	74.0 ± 3.4	51.9 ± 3.8	50.0 ± 0.0	20.0 ± 0.0	79.0 ± 1.8	85.6 ± 5.8	76.0 ± 3.2	64.2 ± 4.3	80.2 ± 2.0
	MAX-MLP	73.2 ± 5.8	51.1 ± 3.6	–	–	–	84.0 ± 6.1	76.0 ± 3.2	64.6 ± 10.2	77.8 ± 1.3
	MAX-1-LAYER (GraphSAGE)	72.3 ± 5.3	50.9 ± 2.2	–	–	–	85.1 ± 7.6	75.9 ± 3.2	63.9 ± 7.7	77.7 ± 1.5