Seminar Series on Graph Neural Networks 04

From Label Propagation to Graph Neural Networks

Yong-Min Shin
School of Mathematics and Computing (Computational Science and Engineering)
Yonsei University

2025.04.28

Before going in....

Towards application of graph neural networks

Towards efficient graph learning

Explainable graph neural networks

Fundamental topics on graph neural networks

On the representational power of graph neural networks

A graph signal processing viewpoint of graph neural networks

From label propagation to graph neural networks

On the problem of oversmoothing and oversquashing

Introduction to graph mining and graph neural networks
(Basic overview to kick things off)

* Presentation slides are available at:
(jordan7186.github.io/presentations/)

Objectives

1. Understanding **label propagation**
2. Connecting between label propagation and graph neural networks
3. Understanding how **homophily** interacts with graph neural networks
4. Going beyond homophily: H2GCN

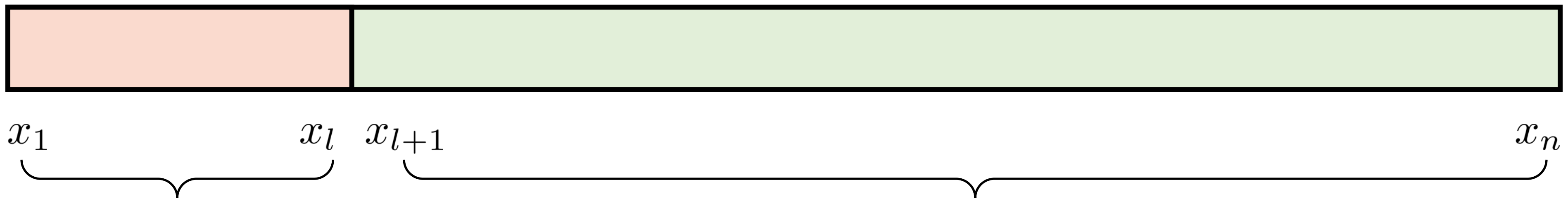
Understanding label propagation

Zhou et al., Learning with Local and Global Consistency, NeurIPS'04

This is a really good and insightful paper that can provide
(1) Underlying assumptions in semi-supervised learning, and
(2) A classical solution to semi-supervised classification that eventually has connections to message-passing

Re-introduction to semi-supervised learning

Given data: $\mathcal{X} = \{x_1, \cdots, x_l, x_{l+1}, \cdots, x_n\}$



Only a portion of data is labeled.

Objective: to classify these data

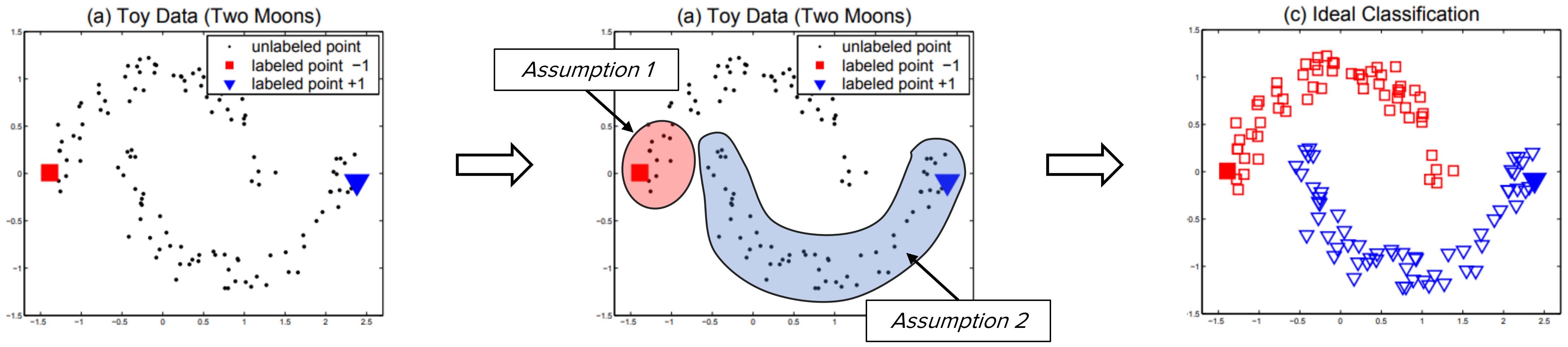
Semi-supervised learning attempts to *predict the labels of unlabeled data*, where the *portion of labeled data is small*.

Semi-supervised problem setting is very *practical* in the sense that labeling usually requires human effort and labeling every data can be very challenging if the size of the data is huge.

"Such a learning problem is often called semi-supervised or transductive."

Zhou et al. *"Learning with Local and Global Consistency"*, NeurIPS'04

Re-introduction to semi-supervised learning



Exploiting the labeled information is based on *two assumptions*.

1. **Local assumption:**
   Nearby points are likely to have the same label.
2. **Global assumption:**
   Points on the same structure/cluster/manifold are likely to have the same label.

The labeled information is <u>spread out through the whole data</u> with these assumptions.

Solving semi-supervised learning via label propagation

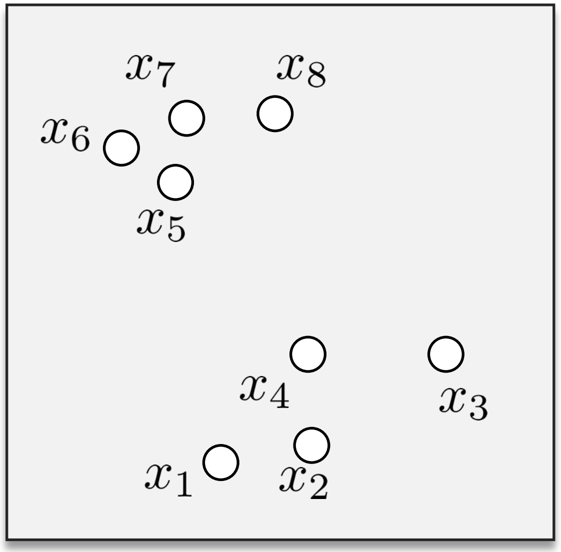
1. Setting and notations

Set of data points: $\mathcal{X} = \{\overbrace{x_1, \cdots, x_l}^{\textit{labeled}}, \overbrace{x_{l+1}, \cdots, x_n}^{\textit{unlabeled}}\}$

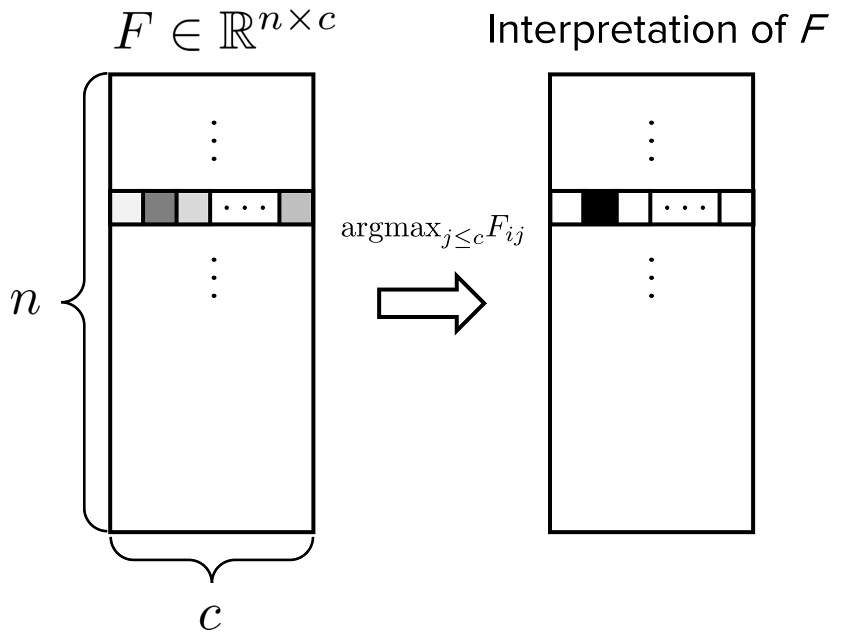
Set of labels: $\mathcal{L} = \{1, \cdots, c\}$

Each data is/can be labeled from 1 to *c*.

$\mathcal{X}_{example} = \{x_1, \cdots, x_8\}$

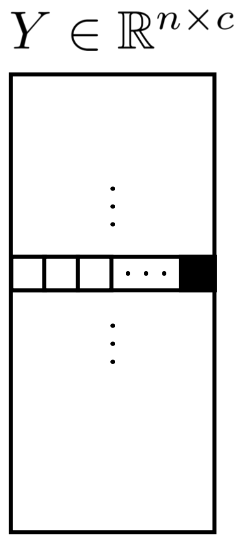


Toy example of 8 data points
(arbitrarily ordered)



$F \in \mathbb{R}^{n \times c}$
Interpretation of *F*
n
c
$\mathrm{argmax}_{j \le c} F_{ij}$

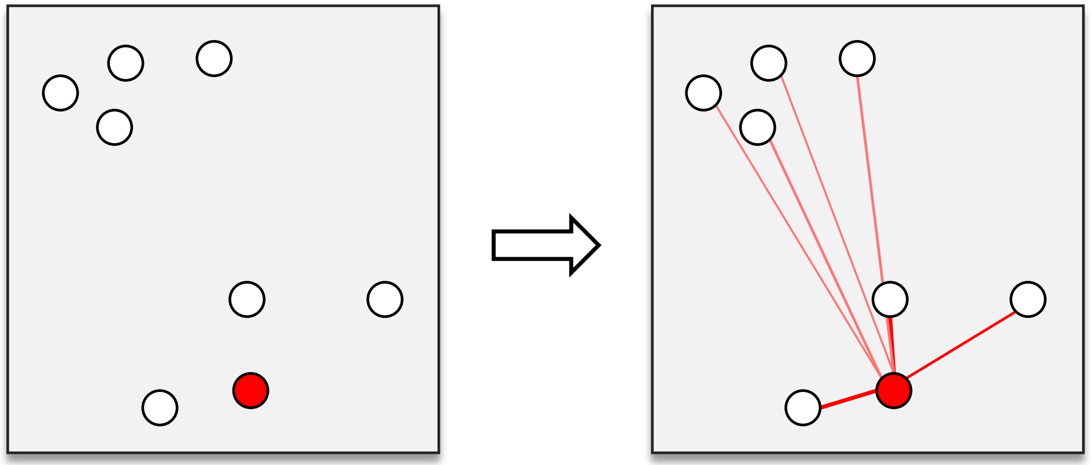
The elements of *F* matrix predicts the labels.



Y stores the labels of $\{x_1, \cdots, x_l\}$

Solving semi-supervised learning via label propagation

2. Calculating the affinity matrix



Toy example of 8 data points, with the perspective of the red data point.

$$x_i\text{: data point}$$

$$W_{ij} = \exp(-||x_i - x_j||^2/2\sigma^2)$$

Calculate the **weights between each data point** with respective to the distances.

This is an interpretation of the data points as an **undirected weighted fully connected graph**

$$G = (V, E)$$

where the vertices are data points, and edge weights are calculated by the formula above.
Therefore, we can also think W_{ij} as the adjacency matrix.

Solving semi-supervised learning via label propagation

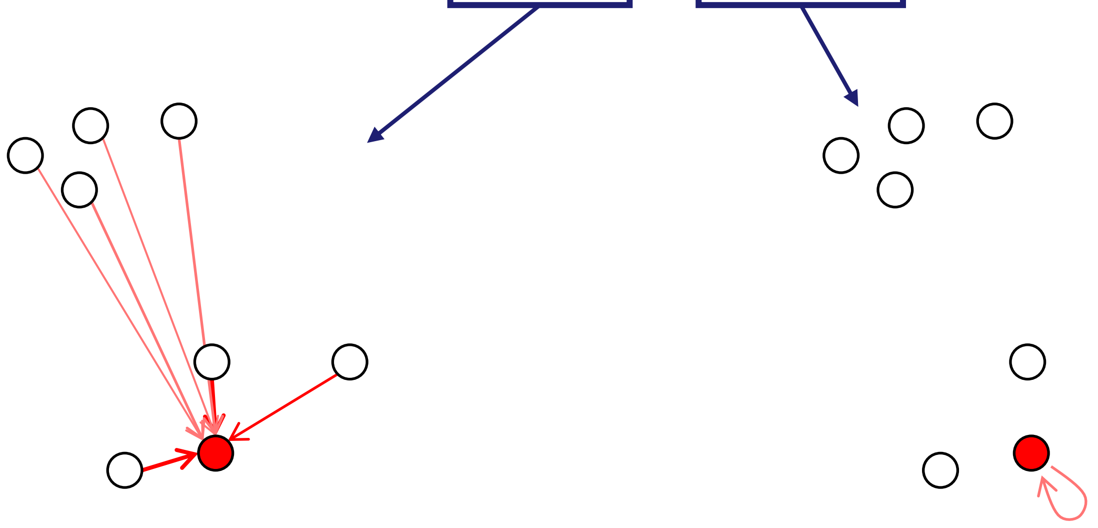
3. Normalization

$$S = D^{-1/2} W D^{-1/2}$$

$$D = \mathrm{diag}(d_1, \cdots, d_{n-1}, d_n), \; d_i = \sum_{j=1}^{n} W_{ij}$$

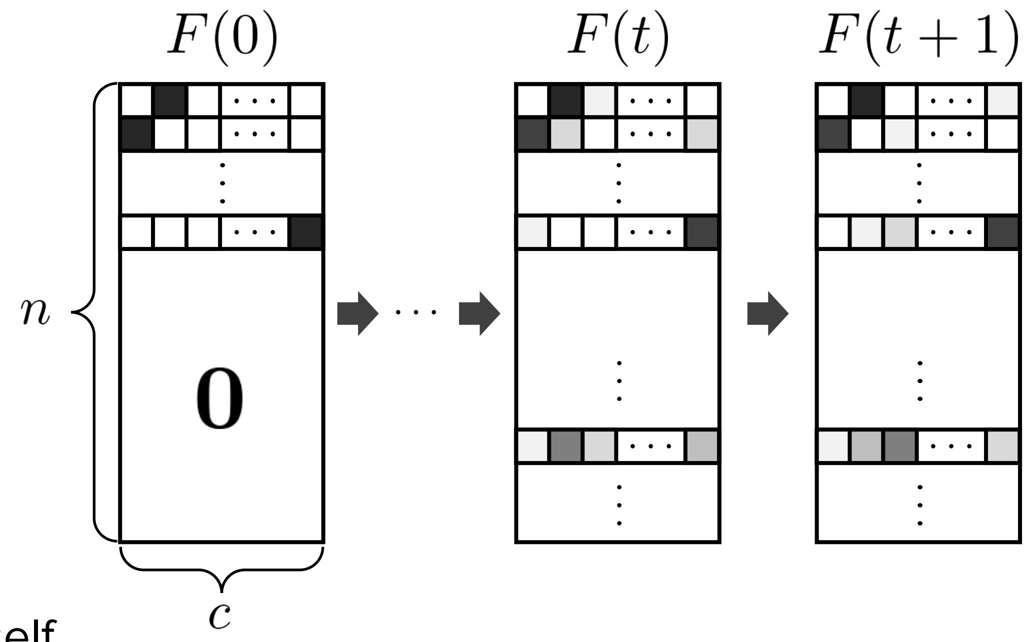
4. Iteration until convergence

$$F(t+1) = \boxed{\alpha S F(t)} + \boxed{(1-\alpha) Y}$$



Receives information from neighbors, scaled down to α.

Retains information of itself, scaled down to $(1-\alpha)$.



Solving semi-supervised learning via label propagation

Proof of convergence

Let $F(0) = Y$.

$$\downarrow$$

$$\boxed{F(t+1) = \alpha S F(t) + (1-\alpha)Y}$$

$$F(1) = \alpha S Y + (1-\alpha)Y$$

$$\begin{aligned} F(2) &= \alpha S(\alpha S Y + (1-\alpha)Y) + (1-\alpha)Y \\ &= (\alpha S)^2 Y + (1-\alpha)(1+\alpha S)Y \end{aligned}$$

$$\begin{aligned} F(3) &= \alpha S(\alpha S(\alpha S Y + (1-\alpha)Y) + (1-\alpha)Y) + (1-\alpha)Y \\ &= (\alpha S)^3 Y + (1-\alpha)(1+\alpha S + (\alpha S)^2)Y \end{aligned}$$

$$\vdots$$

$$\downarrow$$

$$F(t) = (\alpha S)^t Y + (1-\alpha)\sum_{i=0}^{t-1}(\alpha S)^i Y$$

Solving semi-supervised learning via label propagation

Proof of convergence

$$F(t) = \boxed{(\alpha S)^t} Y + (1-\alpha) \boxed{\sum_{i=0}^{t-1} (\alpha S)^i Y}$$

$$\boxed{(\alpha S)^t} \xrightarrow{\lim t \to \infty} O \qquad \boxed{\sum_{i=0}^{t-1} (\alpha S)^i Y} \xrightarrow{\lim t \to \infty} (I - \alpha S)^{-1}$$

Therefore, the limit of the sequence F^* is:

$$F^* = (1-\alpha)(I - \alpha S)^{-1}$$

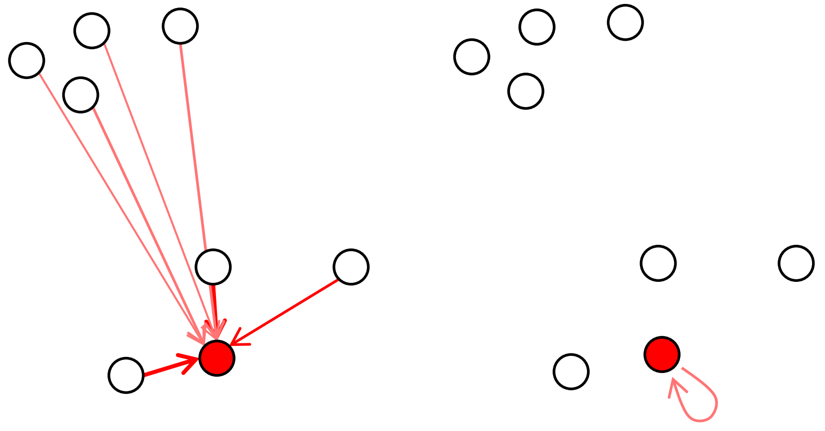
(This is also beneficial in the sense that we know the closed-form solution)

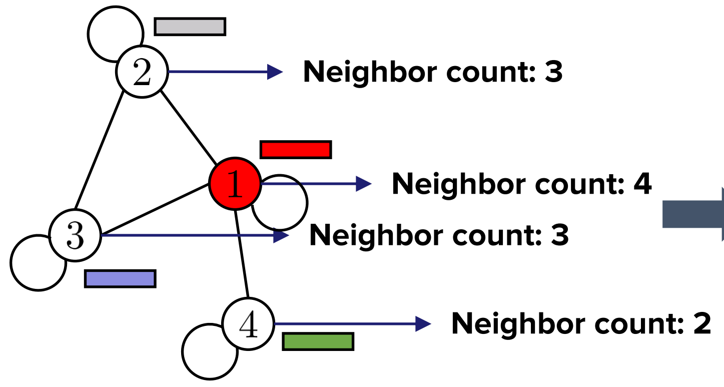
Which is also the minimum to the cost function:

$$Q(F) = \frac{1}{2}\Big(\sum_{i,j=1}^{n} \underbrace{W_{ij}}_{\text{2. If they are strongly connected...}} \underbrace{\Big|\Big|\frac{1}{\sqrt{D_{ii}}}F_i - \frac{1}{\sqrt{D_{jj}}}F_j\Big|\Big|^2}_{\text{1. Enforce two predictions to be the same...}} + \mu \sum_{i=1}^{n} \underbrace{||F_i - Y_i||^2}_{\text{3. Follow the initial label}}\Big)$$

Connecting between label propagation and graph neural networks

We can also view GNNs from the lens of label propagation





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

$$\tilde{A} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2} = \begin{pmatrix} \frac{1}{4} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{12}} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{\sqrt{12}} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{\sqrt{8}} & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

$$F(t) = (\alpha S)^t Y + (1-\alpha)\sum_{i=0}^{t-1}(\alpha S)^i Y$$

$\longleftrightarrow$

Final layer of GCN: $\sigma(\tilde{A}X\Theta)$

1. **Local assumption: Nearby points** are likely to have the same label.
2. **Global assumption: Points on the same structure/cluster/manifold** are likely to have the same label.

$\longrightarrow$

Does this mean GNNs thrive under <u>similar</u> assumptions?

We can also view GNNs from the lens of label propagation

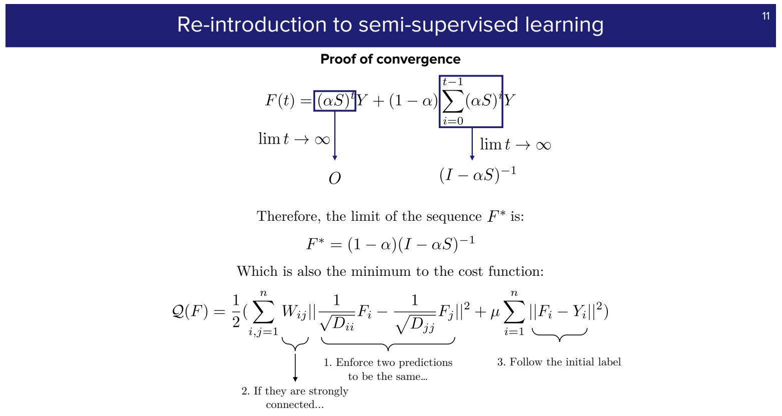
Theoretical connections between LP & GNN

Ma et al,. A Unified View on Graph Neural Networks as Graph Signal Denoising, CIKM'21

PROBLEM 1 (GRAPH SIGNAL DENOISING). *Given a noisy signal* $\mathbf{S} \in \mathbb{R}^{N\times d}$ *on a graph* $\mathcal{G}$, *the goals is to recover a clean signal* $\mathbf{F} \in \mathbb{R}^{N\times d}$, *assumed to be smooth over* $\mathcal{G}$, *by solving the following optimization problem:*

$$\arg\min_{\mathbf{F}} \mathcal{L} = \|\mathbf{F} - \mathbf{S}\|_F^2 + c \cdot tr(\mathbf{F}^\top \mathbf{L}\mathbf{F}). \tag{8}$$

THEOREM 3. *When we adopt the normalized Laplacian matrix* $\mathbf{L} = \mathbf{I} - \tilde{\mathbf{A}}$, *the feature aggregation operation in GCN (Eq. (2)) can be regarded as solving Problem 1 using one-step gradient descent with* $\mathbf{X}'$ *as the input noisy signal and stepsize* $b = \frac{1}{2c}$.



Re-introduction to semi-supervised learning

11

Proof of convergence

$$F(t) = \boxed{(\alpha S)^t} Y + (1-\alpha)\boxed{\sum_{i=0}^{t-1}(\alpha S)^i} Y$$

$$\lim t \to \infty: \quad (\alpha S)^t \to O, \qquad \sum_{i=0}^{t-1}(\alpha S)^i \to (I - \alpha S)^{-1}$$

Therefore, the limit of the sequence F^* is:

$$F^* = (1-\alpha)(I - \alpha S)^{-1}$$

Which is also the minimum to the cost function:

$$\mathcal{Q}(F) = \frac{1}{2}\Big(\sum_{i,j=1}^{n} \underbrace{W_{ij}}_{\text{2. If they are strongly connected...}} \underbrace{\Big\|\frac{1}{\sqrt{D_{ii}}}F_i - \frac{1}{\sqrt{D_{jj}}}F_j\Big\|^2}_{\text{1. Enforce two predictions to be the same...}} + \mu \sum_{i=1}^{n} \underbrace{\|F_i - Y_i\|^2}_{\text{3. Follow the initial label}}\Big)$$

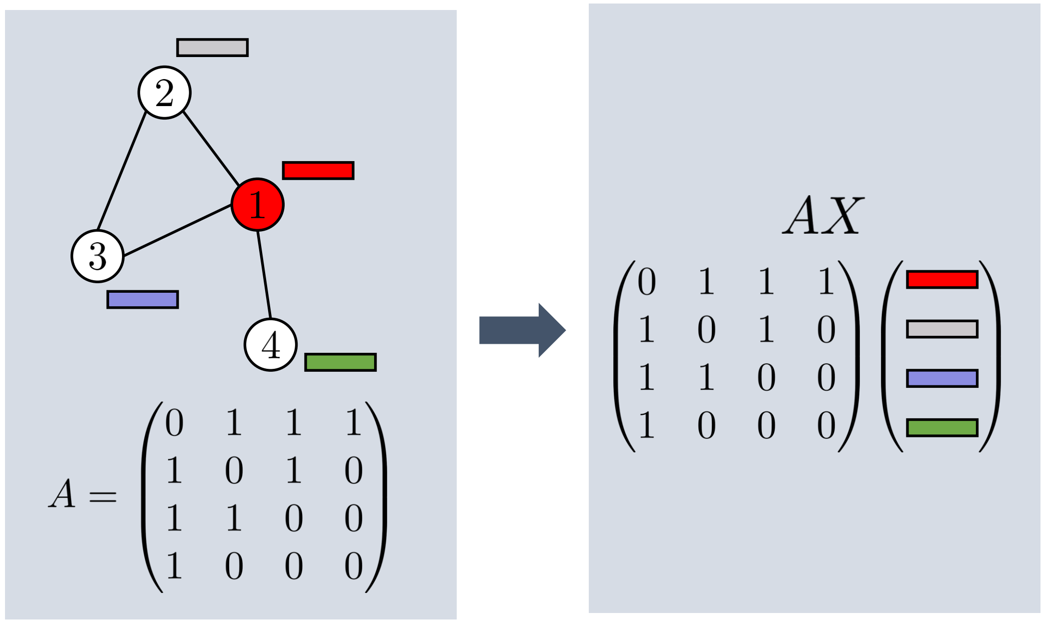
The theorem states that **GCN and LP essentially solves the same problem**, i.e., signal denoising in graphs

GNNs and LPs are natural low-pass filters

Theoretical connections between LP & GNN

NT & Maehara, Revisiting graph neural networks: All we have is low-pass filters. arXiv 2019 (citation > 500)

Assumption 1. *Input features consist of low-frequency true features and noise. The true features have sufficient information for the machine learning task.*



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

- "...Multiplying the (normalized) adjacency matrix corresponds to applying graph filter h(λ) = 1 – λ."
- [Theorem 3] (Informal) Multiplying the adjacency matrix with self-loops shifts the frequency of the graph signal towards zero, effectively applying a low-pass filter.
- The graph filter 1 – λ is also the first-order Taylor approximation of the optimal solution to the problem of graph signal denoising:

$$F(t) = \boxed{(\alpha S)^t} Y + (1-\alpha) \boxed{\sum_{i=0}^{t-1} (\alpha S)^i} Y$$

$$\begin{tikzcd} \boxed{(\alpha S)^t} \arrow[d, "\lim t \to \infty"'] & \boxed{\sum_{i=0}^{t-1} (\alpha S)^i} \arrow[d, "\lim t \to \infty"] \\ O & (I - \alpha S)^{-1} \end{tikzcd}$$

Therefore, the limit of the sequence F^* is:

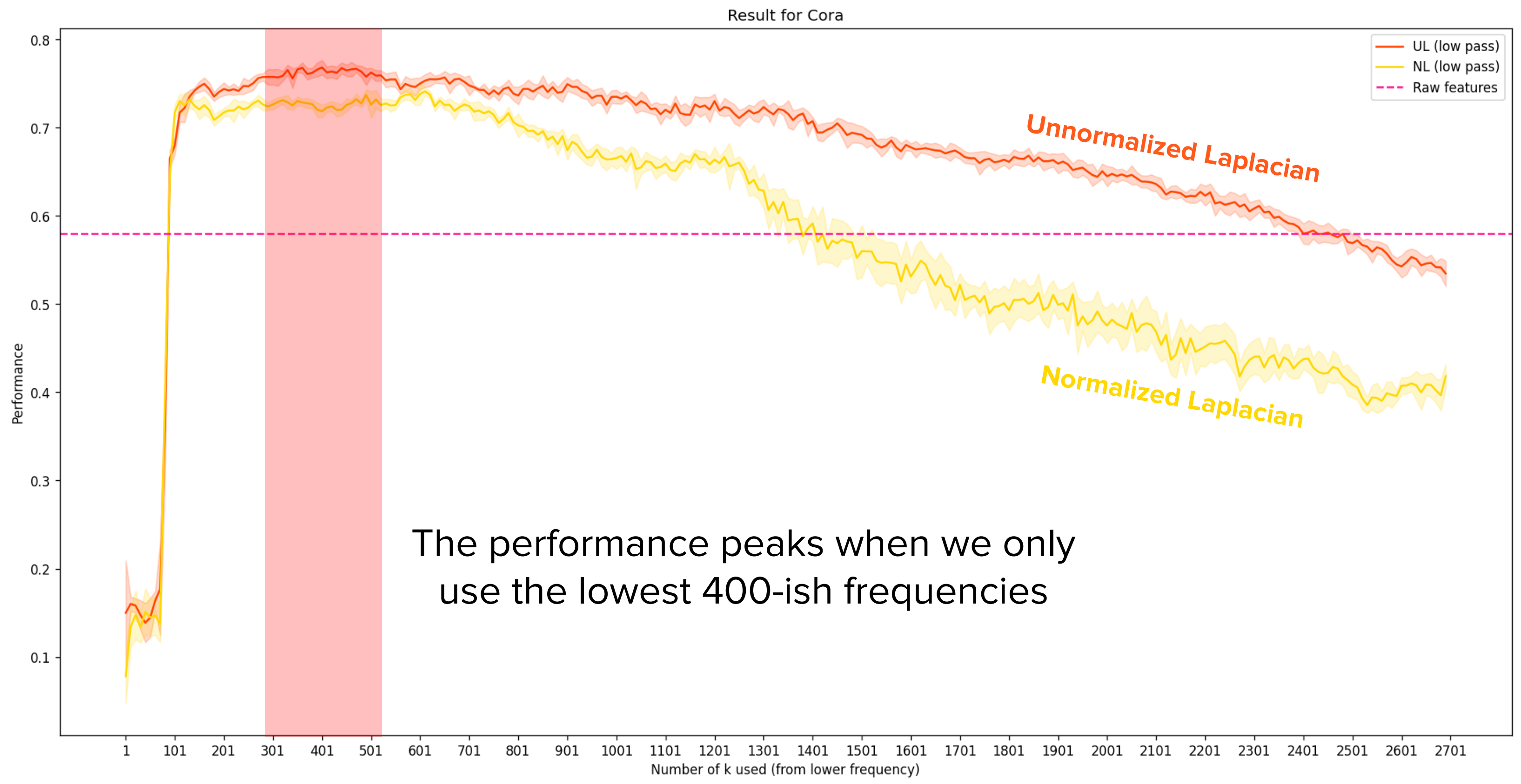
$$F^* = (1-\alpha)(I - \alpha S)^{-1}$$

GNNs and LPs are natural low-pass filters

Theoretical connections between LP & GNN

NT & Maehara, Revisiting graph neural networks: All we have is low-pass filters. arXiv 2019 (citation > 500)

Assumption 1. *Input features consist of low-frequency true features and noise. The true features have sufficient information for the machine learning task.*



The performance peaks when we only use the lowest 400-ish frequencies

Send f in vertex space to frequency space

$$\mathcal{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top \qquad \mathcal{L} = \mathbf{D} - \mathbf{A}$$

$$\hat{f}(\lambda_l) = < f, u_i > = \mathbf{U}^T f$$

Apply a **filtering function** h to eigenvalues

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_0 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \lambda_{N-1} \end{bmatrix} \Rightarrow h(\boldsymbol{\Lambda}) = \begin{bmatrix} h(\lambda_0) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & h(\lambda_{N-1}) \end{bmatrix}$$

In this experiment, we use an ideal low-pass filter, which leaves the lowest k frequencies untouched and **everything else to zero**.

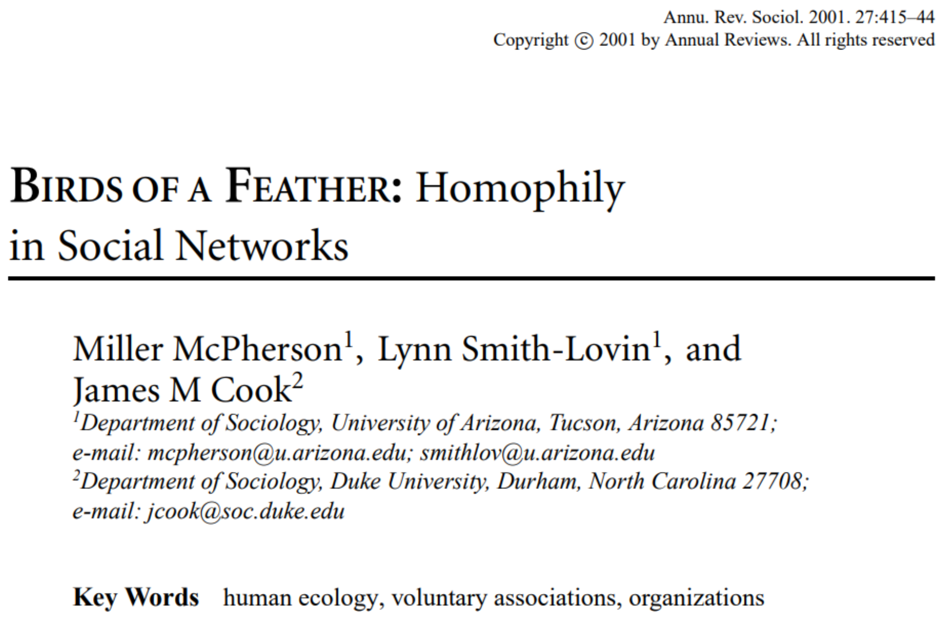
Send back to vertex space

$$f_{new} = \mathbf{U}h(\boldsymbol{\Lambda})\mathbf{U}^T f$$

Experimental results are my own, see the following link for more details: https://jordan7186.github.io/blog/2022/Cora_spectral/

Understanding how homophily interacts with graph neural networks

Definition of homophily



Annu. Rev. Sociol. 2001. 27:415–44
Copyright © 2001 by Annual Reviews. All rights reserved

BIRDS OF A FEATHER: Homophily in Social Networks

Miller McPherson[1], Lynn Smith-Lovin[1], and James M Cook[2]
[1]*Department of Sociology, University of Arizona, Tucson, Arizona 85721; e-mail: mcpherson@u.arizona.edu; smithlov@u.arizona.edu*
[2]*Department of Sociology, Duke University, Durham, North Carolina 27708; e-mail: jcook@soc.duke.edu*

Key Words human ecology, voluntary associations, organizations

"Homophily is the principle that a contact between similar people occurs at a higher rate than among dissimilar people."

- In social networks, race/ethnicity-based homophily create the most distinctive divides among people
- Sex, age, religion, and education is also a strong source of homophily
- Occupation, network position etc. also show homophily properties but somewhat limited

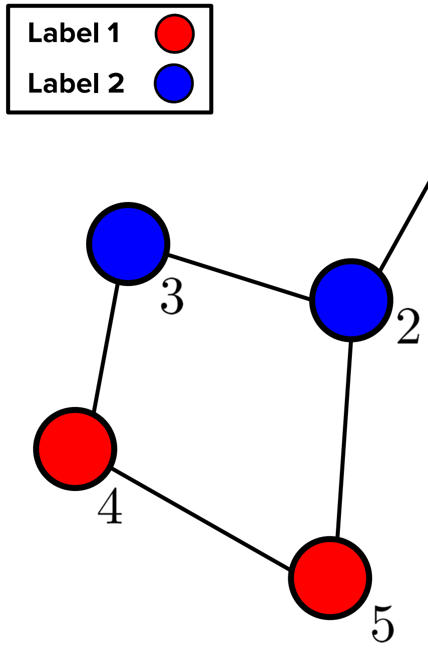
Miller McPherson et al., Birds of a feather: Homophily in social networks, Annual Review of Sociology 2001 27:1, pp. 415-444

Measurement of homophily

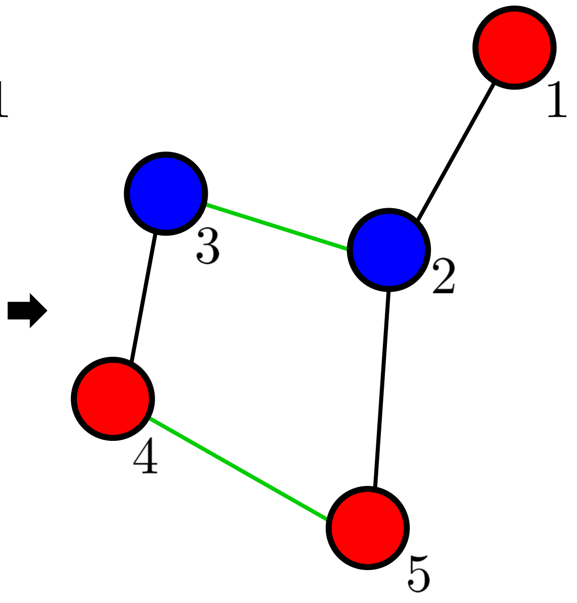
Zhu et al., Beyond homophily in graph neural networks: Current limitations and effective designs, NeurIPS'20

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency matrix



Given graph



Count edge with same labels

$$h = \frac{|\{(u,v) | (u,v) \in \mathcal{E} \wedge \boxed{y_u = y_v}\}|}{|\mathcal{E}|}$$

$$= \frac{2}{5} = 0.4$$

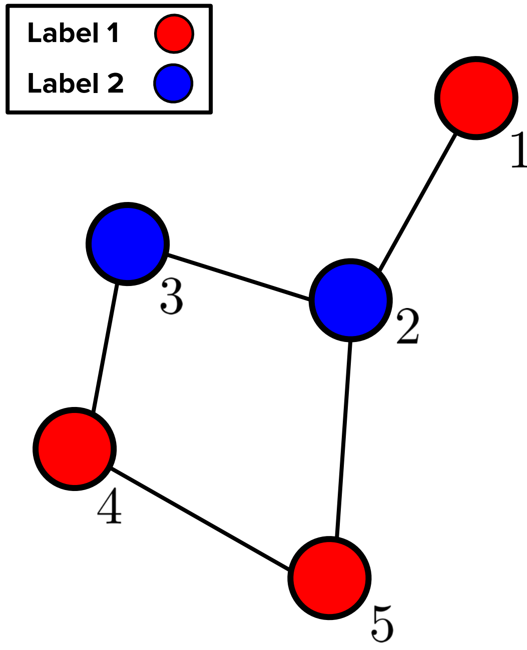
Edge homophily ratio

Measurement of homophily

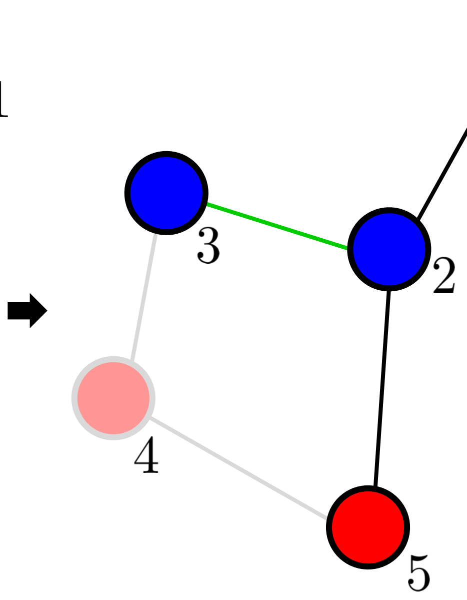
Pei et al., Geom-GCN: Geometric graph convolutional networks, ICLR'20

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency matrix



Given graph



(Foucsing on node 2)

$$h = \frac{1}{|V|} \sum_{v \in V} \frac{\text{Num. of neighbors with same labels}}{\text{Num. of neighbors}}$$

$$= \frac{1}{5}(0 + \frac{1}{3} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2}) \approx 0.367$$

Node homophily ratio

Due to the low-pass characteristics, GNNs implicitly assume homophily

Zhu et al., Beyond homophily in graph neural networks: Current limitations and effective designs, NeurIPS'20

"GNNs model the homophily principle by propagating features and aggregating them within various graph neighborhoods via different mechanisms (e.g., averaging, LSTM)"

Bo et al., Beyond low-frequency information in graph convolutional networks, AAAI'21

"In general, GNNs update node representations by aggregating information from neighbors, which can be seen as a special form of low-pass filter. … this mechanism may work well for assortative networks, i.e., similar nodes tend to connect with each other."

Pei et al., Geom-GCN: Geometric graph convolutional networks, ICLR'20

"The MPNNs with such aggregation are inclined to learn similar representations for proximal nodes in a graph. … they are probably desirable methods for assortative graphs (e.g., citation networks and community networks) where node homophily holds (i.e., similar nodes are more likely to be proximal, and vice versa), … ."

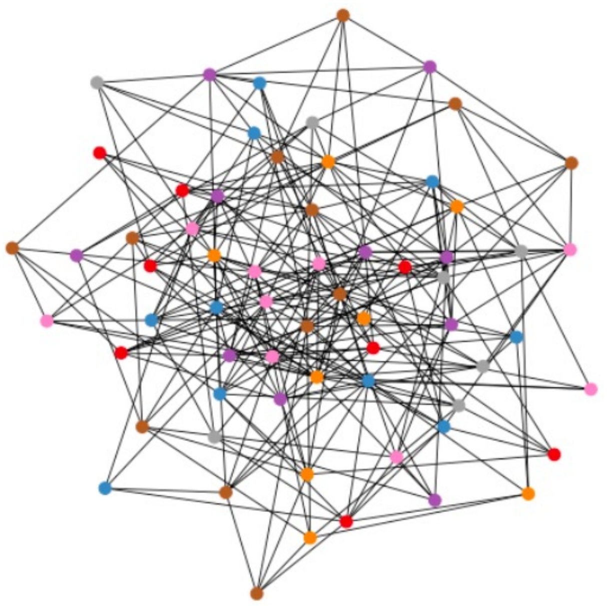
Empirical trend: Homophily and GNN performance

An experiment inspired by (Zhu et al., NeurIPS 2020)

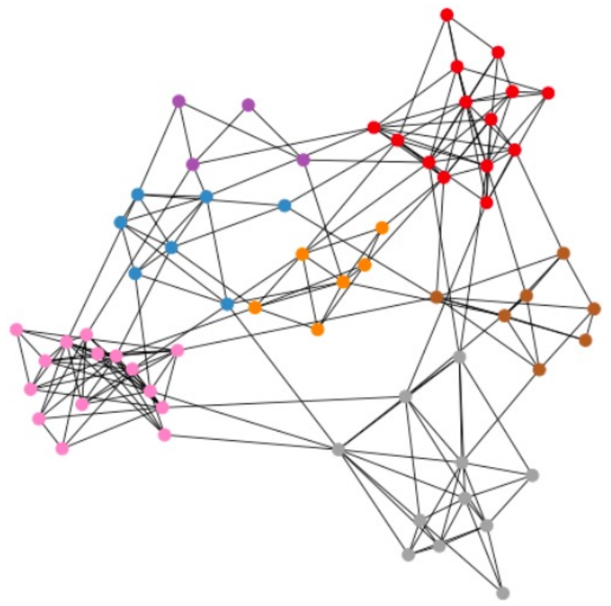
Synthetic-cora dataset

- Cora (Yang et al., ICML 2016) is the most widely-used benchmark dataset in graph learning.
- Node: Papers, Edge: Citations, Node features: Binary encoding of abstract.
- Typical task: Node classification. Correctly classify 2708 nodes according to one of seven classes (research area).
- **Synthetic-cora**: Based on the original Cora dataset, re-generate a new graph according to a target homophily ratio.
    - Node features are all sampled from the original dataset.
    - Edges are all reconstructed: Based on the BA model (Barabási–Albert model / Preferrential attachment model)
    - In this way, **we can generate realistic graphs with various homophily ratios**

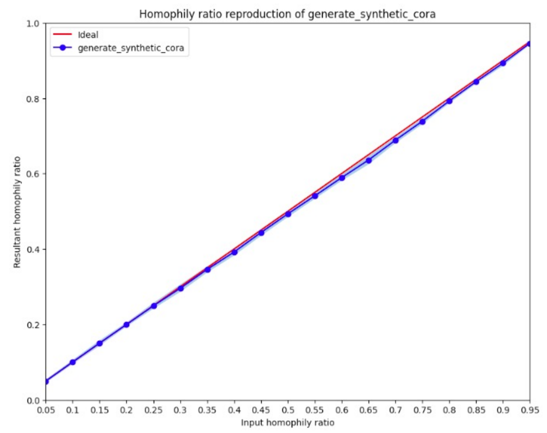
Example of BA-model generating a graph



Syn-cora (h=0.0929)



Syn-cora (h=0.8325)

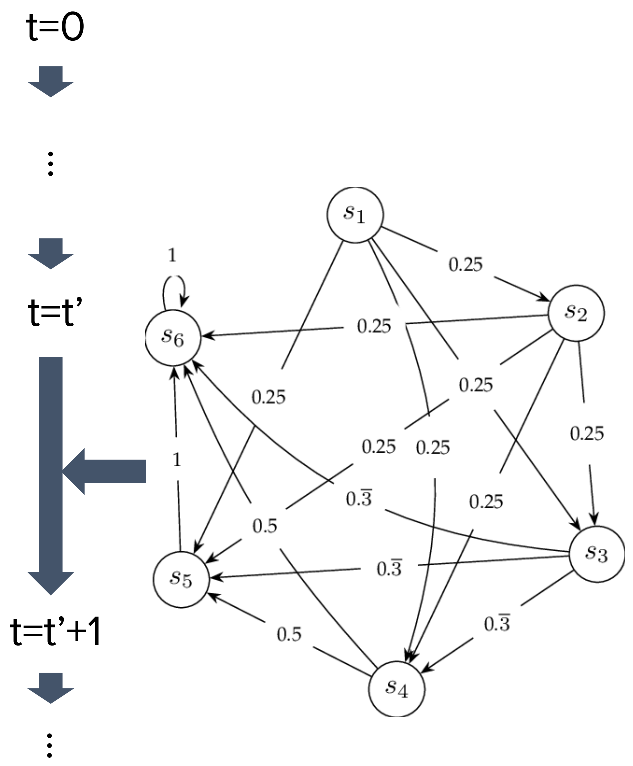


Ideal homophily ratio (x, red) vs. Realized homophily ratio (y, blue)

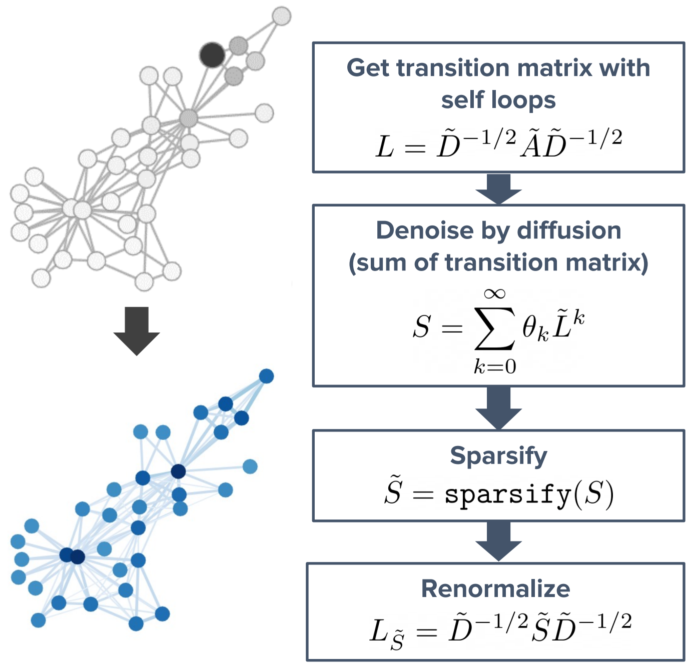
Zhu et al., Beyond homophily in graph neural networks: Current limitations and effective designs, NeurIPS 2020
Yang et al., Revisiting semi-supervised learning with graph embeddings, ICML 2016 (citations >2600)
BA-model example: https://en.wikipedia.org/wiki/Barab%C3%A1si%E2%80%93Albert_model

Empirical trend: Homophily and GNN performance

An experiment inspired by (Zhu et al., NeurIPS 2020)



Imagine the adjacency matrix = transition matrix.
As time goes by, we apply the transition matrix
Therefore, powers of the adjacency matrix shows
the diffusion process



Get transition matrix with self loops

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

Denoise by diffusion (sum of transition matrix)

$$S = \sum_{k=0}^{\infty} \theta_k \tilde{L}^k$$

Sparsify

$$\tilde{S} = \texttt{sparsify}(S)$$

Renormalize

$$L_{\tilde{S}} = \tilde{D}^{-1/2}\tilde{S}\tilde{D}^{-1/2}$$

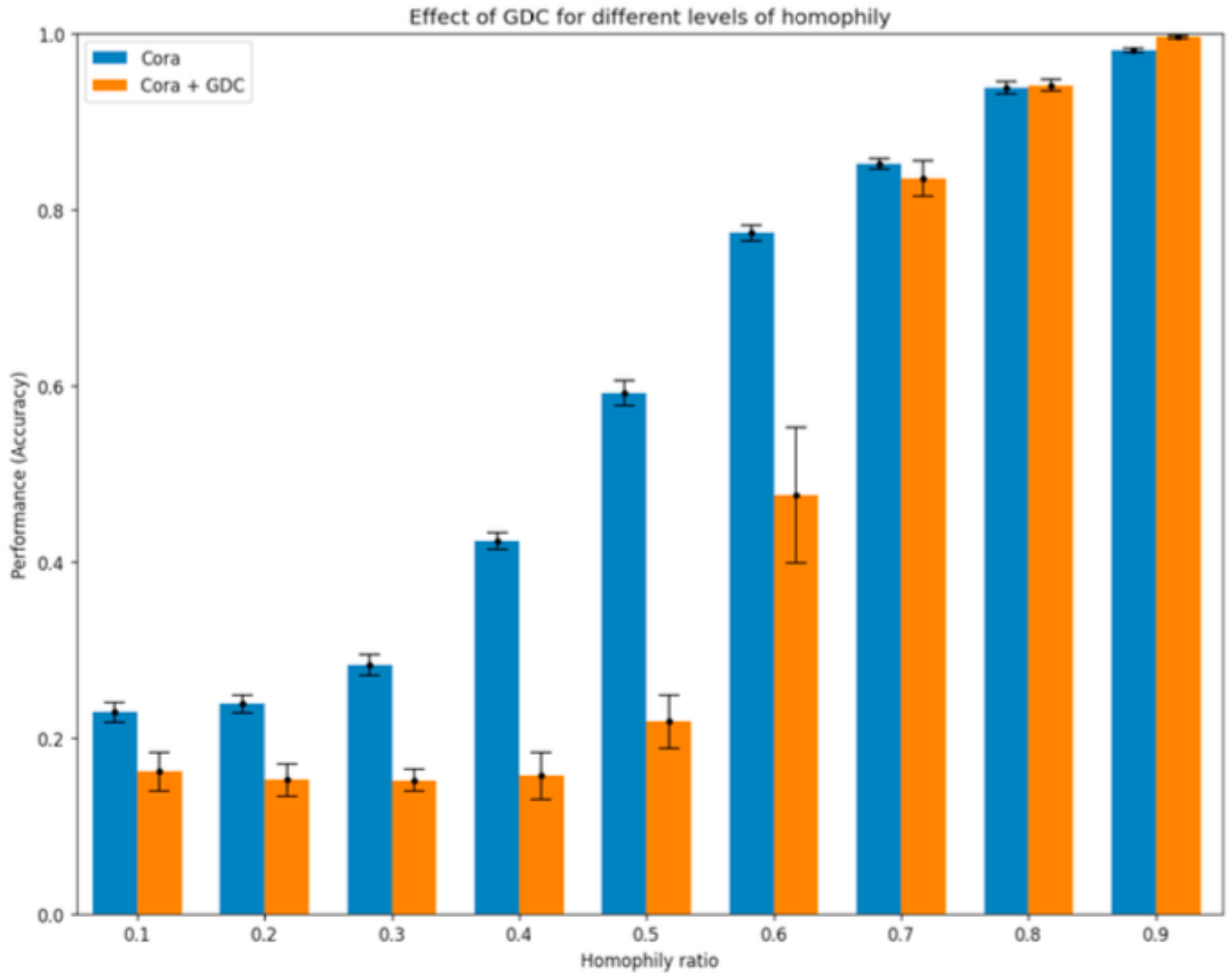
One more thing: GDC (Graph Diffusion Convolution)

- A pre-processing step directly on the graph
- Diffusion as in "heat diffusion": Image heat sources as nodes and steel edges
  - Each single heat source will spread over time
- Why is the power of the transition (normalized adjacency matrix) important?
  - Think of the heat spreading simulation in a discretized time frame
- GDC is also based on the assumption that the given graph is strong in homophily: Intuitively, adds more connection to (structurally) nearby nodes.
  - Therefore, it can improve the node classification performance when applied to datasets with already high levels of homophily
  - Then how about the opposite (heterophily)?

If you are interested in GDC (Graph Diffusion Convolution): Gasteiger et al., Diffusion improves graph learning, NeurIPS 2019
Transition matrix example: Aleti et al., Is perturbation an effective restart strategy?, arXiv 2019

Empirical trend: Homophily and GNN performance

An experiment inspired by (Zhu et al., NeurIPS 2020)



Running GCN on Syn-Cora with various homophily ratios

- GCN performs bad in heterophilic regions
- And its even worse with GDC
- And vice versa!

Going beyond homophily: H2GCN

Motivation: We need a GNN that is also good in heterophilc graphs

How does homophily ratios affect node classification for other architectures?

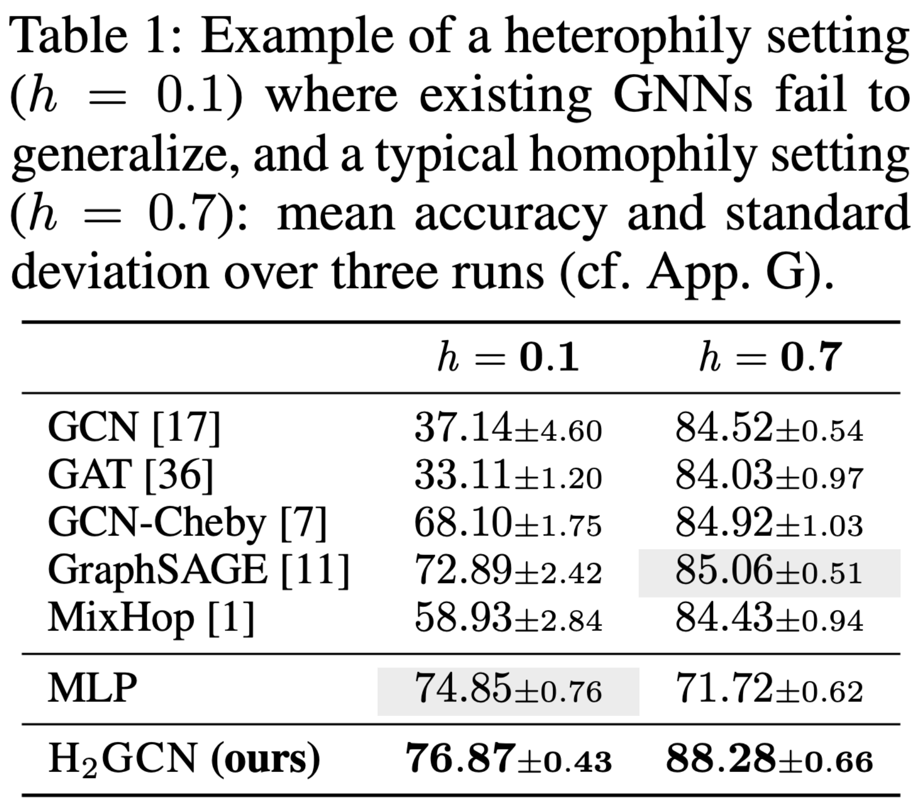


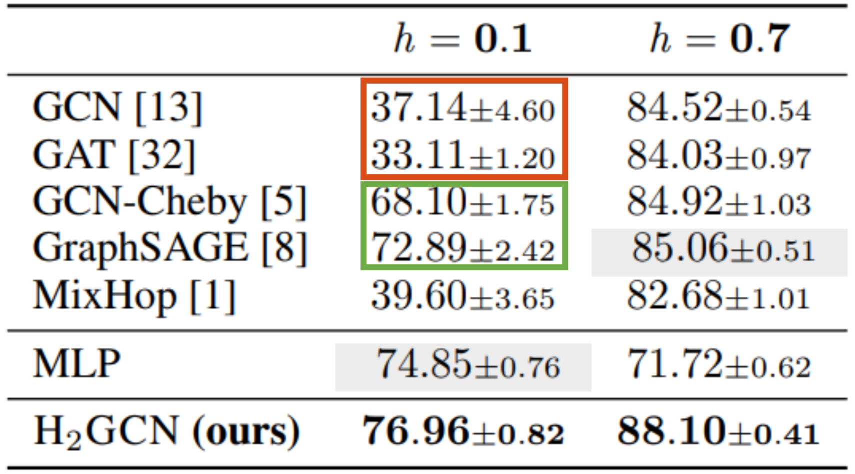
Table 1: Example of a heterophily setting ($h = 0.1$) where existing GNNs fail to generalize, and a typical homophily setting ($h = 0.7$): mean accuracy and standard deviation over three runs (cf. App. G).

	$h = \mathbf{0.1}$	$h = \mathbf{0.7}$
GCN [17]	37.14 ± 4.60	84.52 ± 0.54
GAT [36]	33.11 ± 1.20	84.03 ± 0.97
GCN-Cheby [7]	68.10 ± 1.75	84.92 ± 1.03
GraphSAGE [11]	72.89 ± 2.42	85.06 ± 0.51
MixHop [1]	58.93 ± 2.84	84.43 ± 0.94
MLP	74.85 ± 0.76	71.72 ± 0.62
H_2GCN **(ours)**	$\mathbf{76.87} \pm \mathbf{0.43}$	$\mathbf{88.28} \pm \mathbf{0.66}$

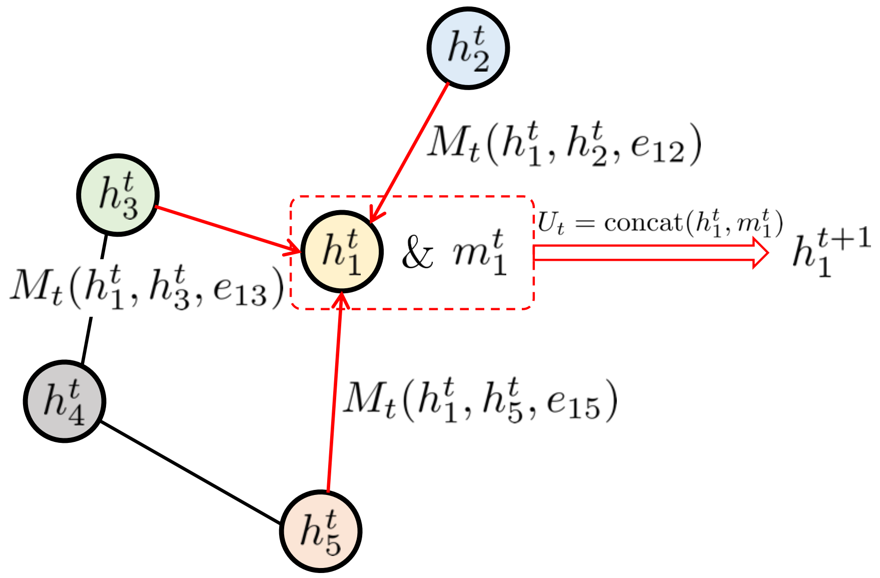
- It’s not just GCNs, its all general message-passing GNNs
- **MLP is an important baseline**, since it only uses node features as input (i.e., no graph structure information)
- H2GCN achieves good results on both homophilic and heterophilic datasets, but how do they achieved this?

3 modifications of the original message-passing mechanism

Modification 1. Ego (self) and neighbor embeddings are separated



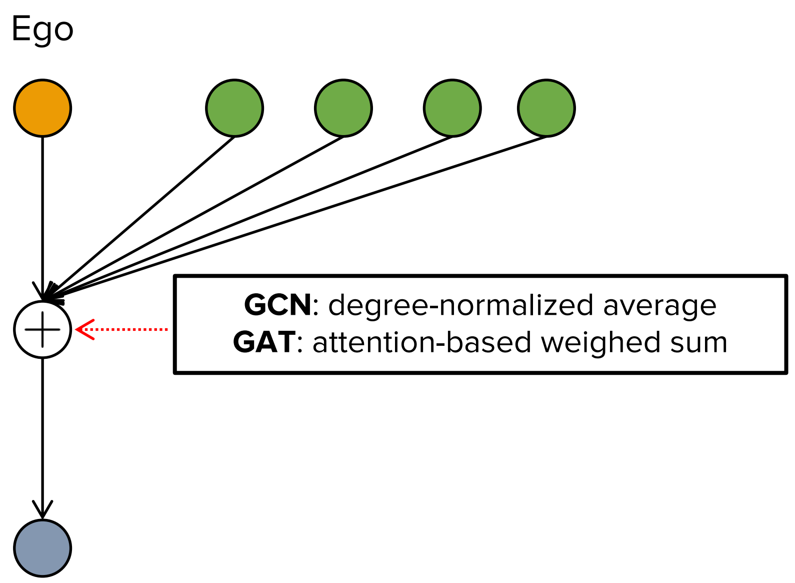
	$h = \mathbf{0.1}$	$h = \mathbf{0.7}$
GCN [13]	$37.14_{\pm 4.60}$	$84.52_{\pm 0.54}$
GAT [32]	$33.11_{\pm 1.20}$	$84.03_{\pm 0.97}$
GCN-Cheby [5]	$68.10_{\pm 1.75}$	$84.92_{\pm 1.03}$
GraphSAGE [8]	$72.89_{\pm 2.42}$	$85.06_{\pm 0.51}$
MixHop [1]	$39.60_{\pm 3.65}$	$82.68_{\pm 1.01}$
MLP	$74.85_{\pm 0.76}$	$71.72_{\pm 0.62}$
$\mathrm{H_2GCN}$ **(ours)**	$\mathbf{76.96_{\pm 0.82}}$	$\mathbf{88.10_{\pm 0.41}}$


h_2^t
$M_t(h_1^t, h_2^t, e_{12})$
h_3^t
h_1^t & m_1^t
$U_t = \mathrm{concat}(h_1^t, m_1^t)$
h_1^{t+1}
$M_t(h_1^t, h_3^t, e_{13})$
h_4^t
$M_t(h_1^t, h_5^t, e_{15})$
h_5^t

Observation 1

GCN and GATs perform poorly

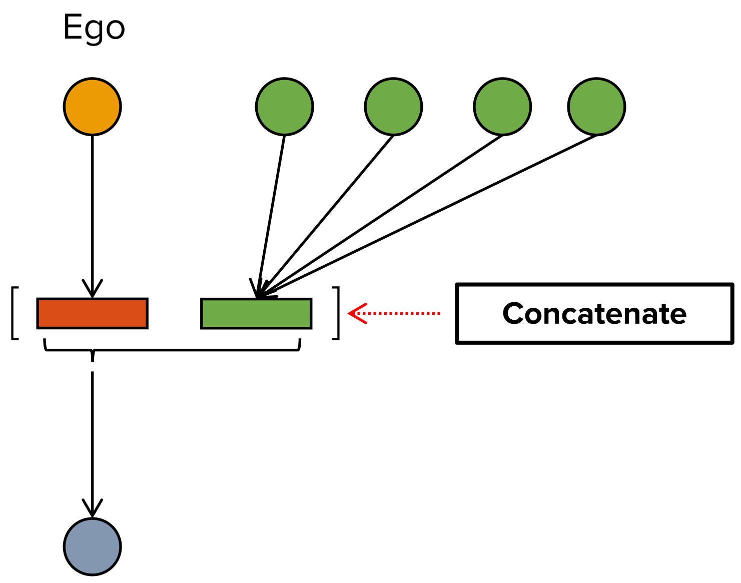
GCN, GAT


Ego
GCN: degree-normalized average
GAT: attention-based weighed sum

Observation 2

GraphSAGE is kind of okay

GraphSAGE



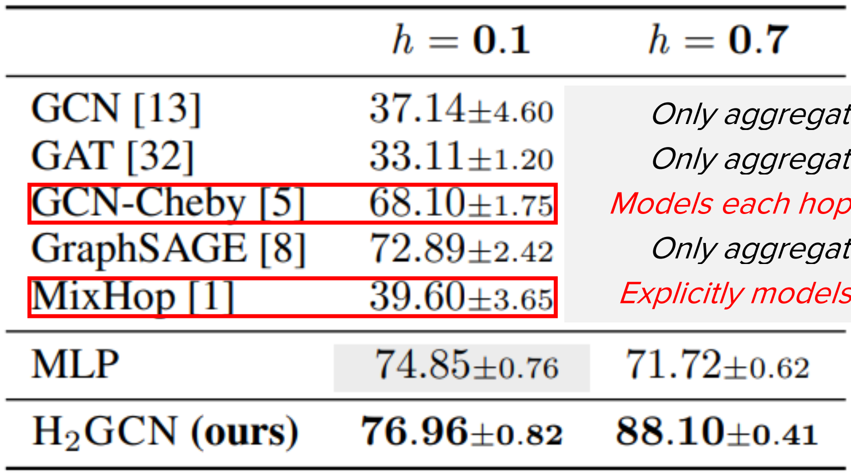
Mixing the sum (or variant) of neighbor embeddings with ego embedding via update function harms expressiveness in heterophily settings.

Fix the update function as the concatenation function.

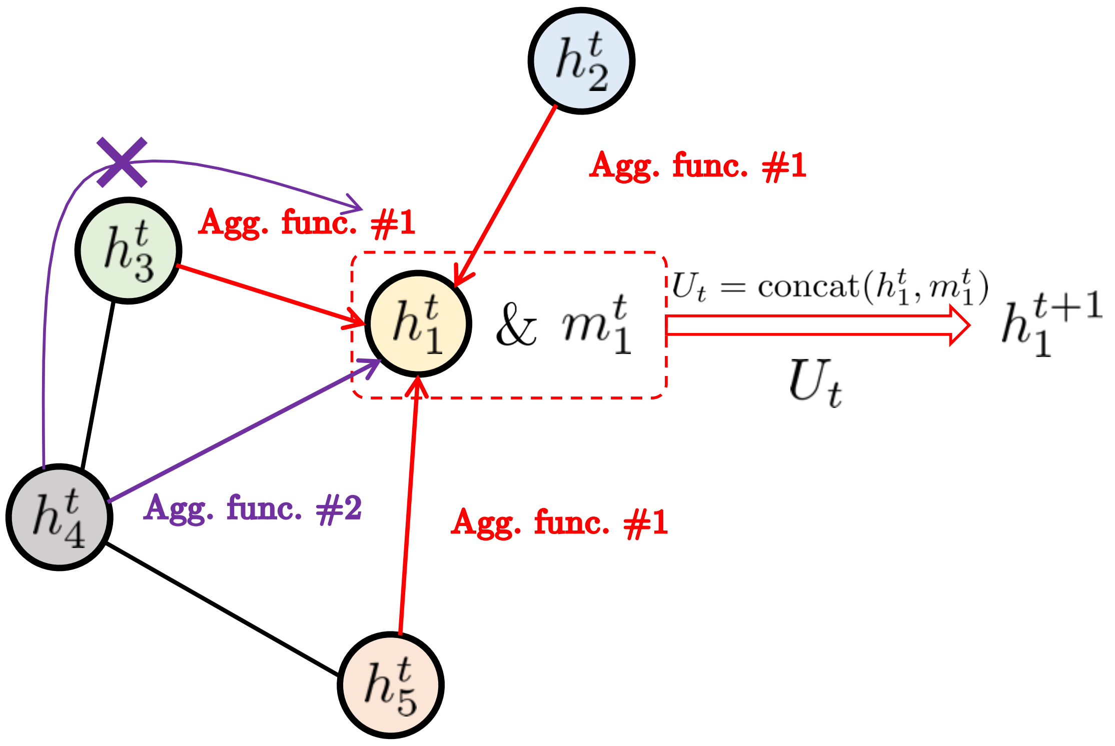
$$U_t = \mathrm{concat}(h_1^t, m_1^t)$$

3 modifications of the original message-passing mechanism

Modification 2. Aggregating from higher-order neighborhoods



	$h = \mathbf{0.1}$	$h = \mathbf{0.7}$	
GCN [13]	37.14±4.60		*Only aggregates over immediate*
GAT [32]	33.11±1.20		*Only aggregates over immediate*
GCN-Cheby [5]	68.10±1.75		*Models each hop neighbors differently*
GraphSAGE [8]	72.89±2.42		*Only aggregates over immediate*
MixHop [1]	39.60±3.65		*Explicitly models 1 & 2 hop neighbors*
MLP	74.85±0.76	71.72±0.62	
H_2GCN (**ours**)	**76.96**±0.82	**88.10**±0.41	

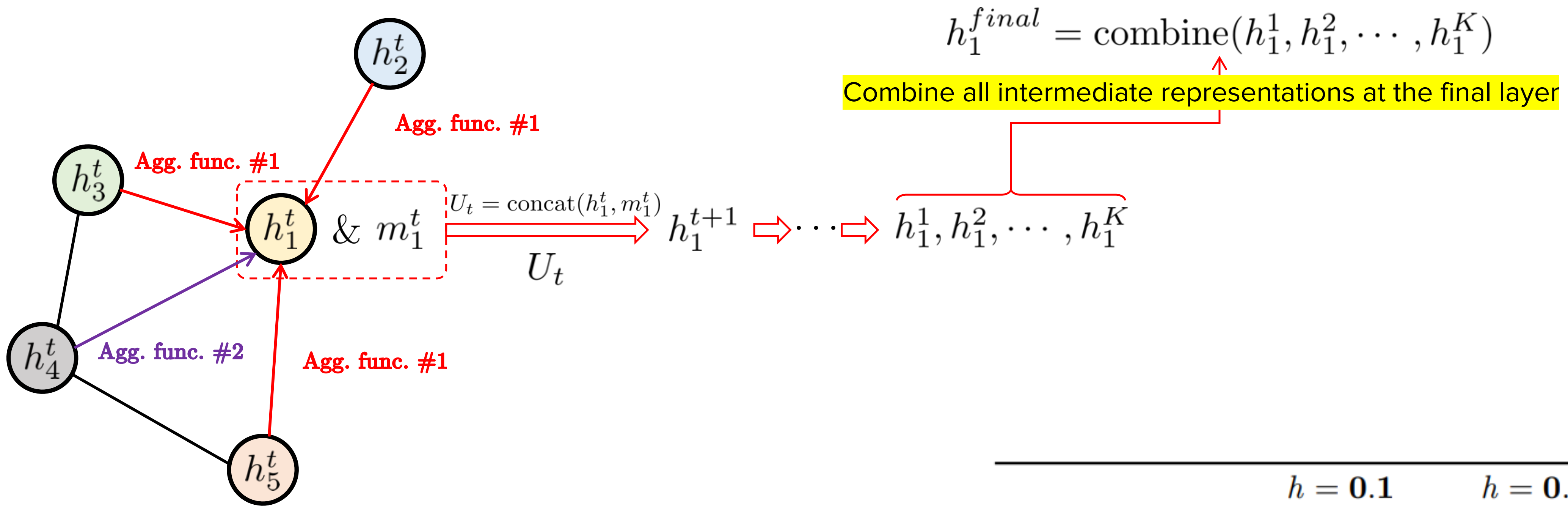


Theorem 2 (informal) Under certain conditions, the 2-hop neighbors will be homophily-dominant in expectation.

Explicitly aggregate 1,2,3, … hop neighbor information without traversing through lower hop neighbors

3 modifications of the original message-passing mechanism

Modification 3. Incorporate jumping knowledge



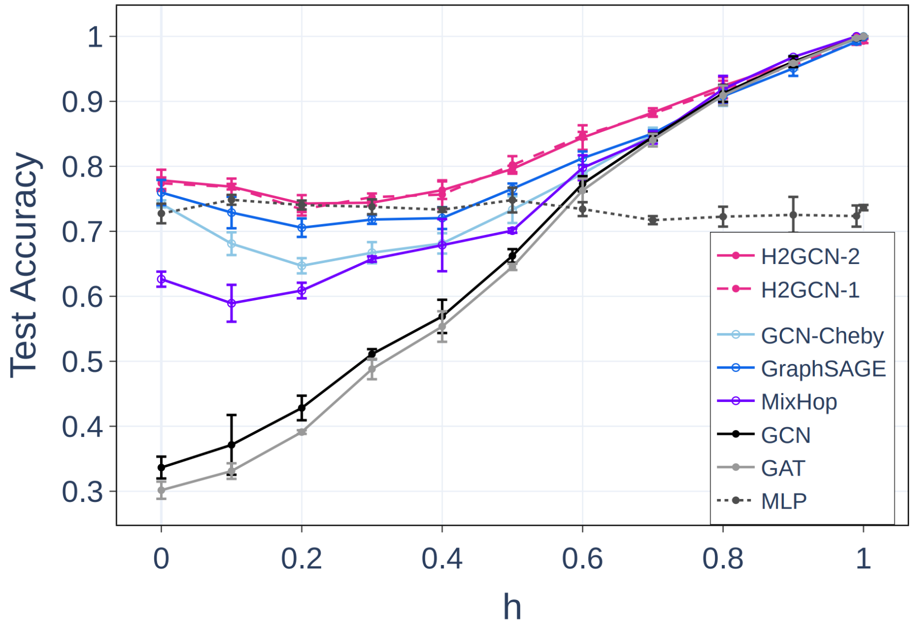
$h_1^{final} = \text{combine}(h_1^1, h_1^2, \cdots, h_1^K)$
Combine all intermediate representations at the final layer
h_2^t
Agg. func. #1
h_3^t
Agg. func. #1
h_1^t & m_1^t
$U_t = \text{concat}(h_1^t, m_1^t)$
h_1^{t+1}
U_t
$h_1^1, h_1^2, \cdots, h_1^K$
h_4^t
Agg. func. #2
Agg. func. #1
h_5^t

	$h = \mathbf{0.1}$	$h = \mathbf{0.7}$
GCN [13]	$37.14{\pm}4.60$	$84.52{\pm}0.54$

"By concatenating the intermediate representations from two rounds with the embedded ego-representation (following the jumping knowledge framework), GCN's accuracy increases to 58.93%±3.17 for h = 0.1, a 20% improvement over its counterpart without design D3 (Table 1)."

Jumping knowledge is first introduced in the graph learning literature by:
Xu et al,. Representation learning on graphs with jumping knowledge networks, ICML 2018

Final form: H2GCN



Mod 2: Explicitly aggregate 1-hop and 2-hop neighbors separately

$$\mathbf{r}_v^{(k)} = \texttt{COMBINE}\left(\texttt{AGGR}\{\mathbf{r}_u^{(k-1)} : u \in \bar{N}_1(v)\}, \texttt{AGGR}\{\mathbf{r}_u^{(k-1)} : u \in \bar{N}_2(v)\}\right)$$

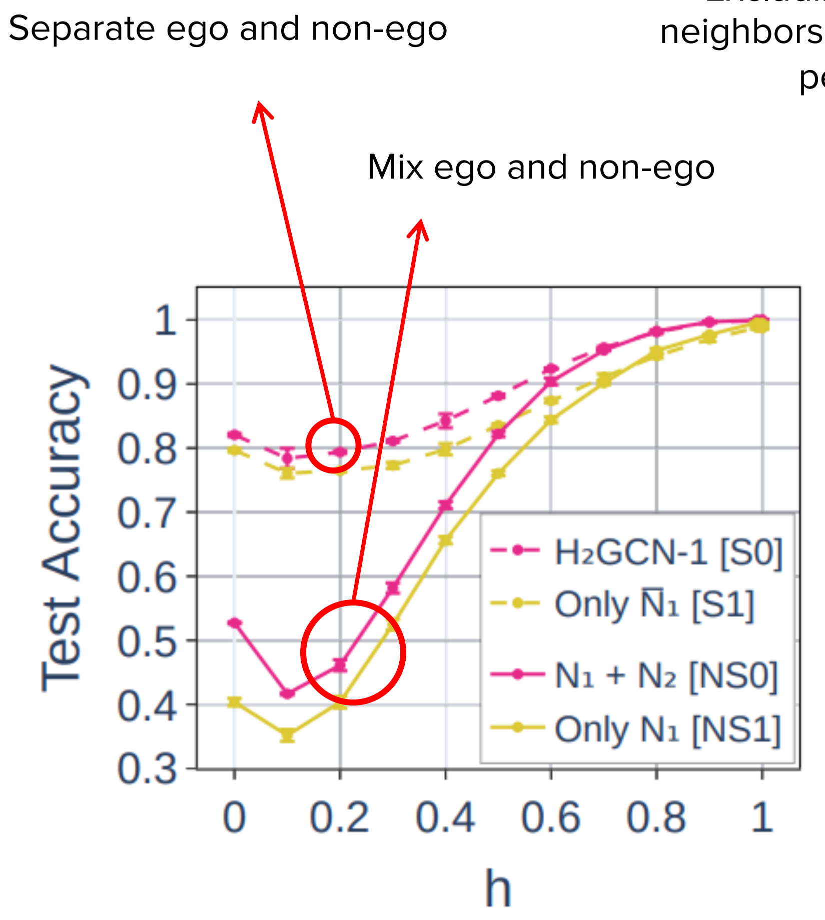
$$\mathbf{r}_v^{(\text{final})} = \boxed{\texttt{COMBINE}\left(\boxed{\mathbf{r}_v^{(0)}}, \mathbf{r}_v^{(1)}, \ldots, \mathbf{r}_v^{(K)}\right)}$$

Mod 1: Separate processing of the ego (self) node

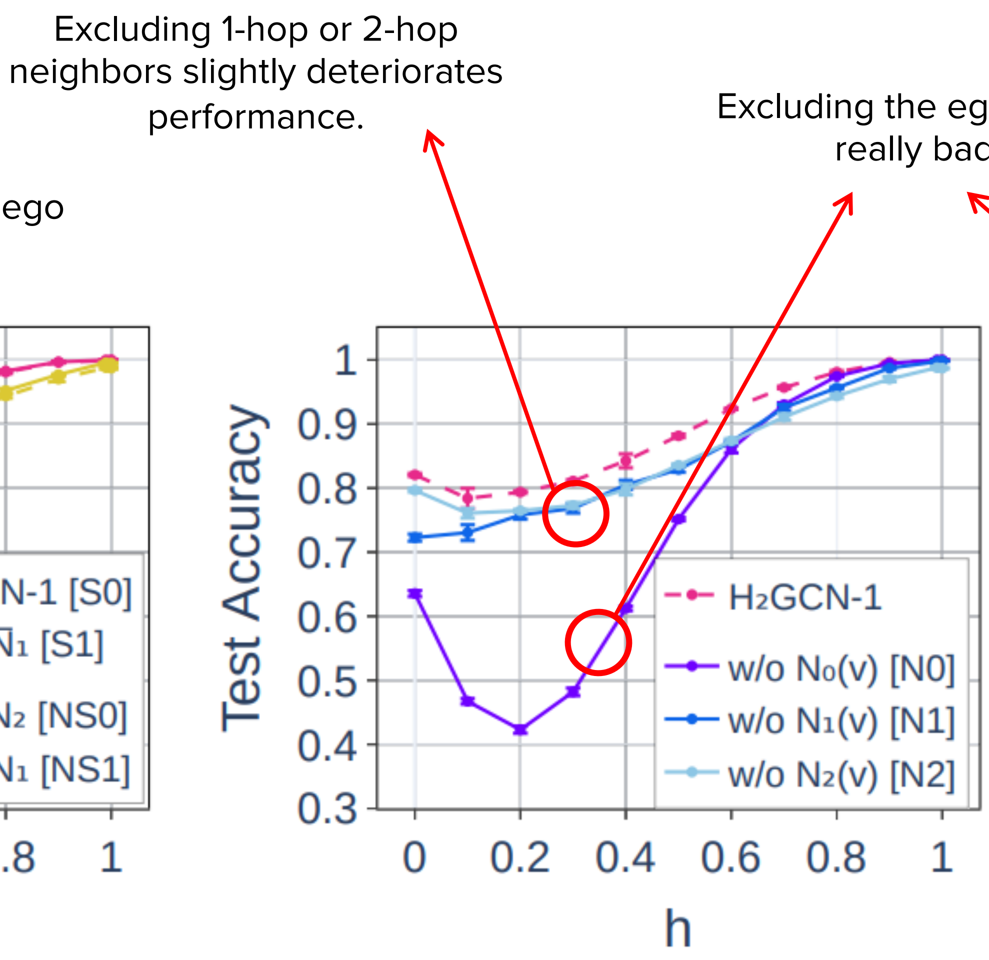
Mod 3: Jumping knowedge

Analysis of the design choices

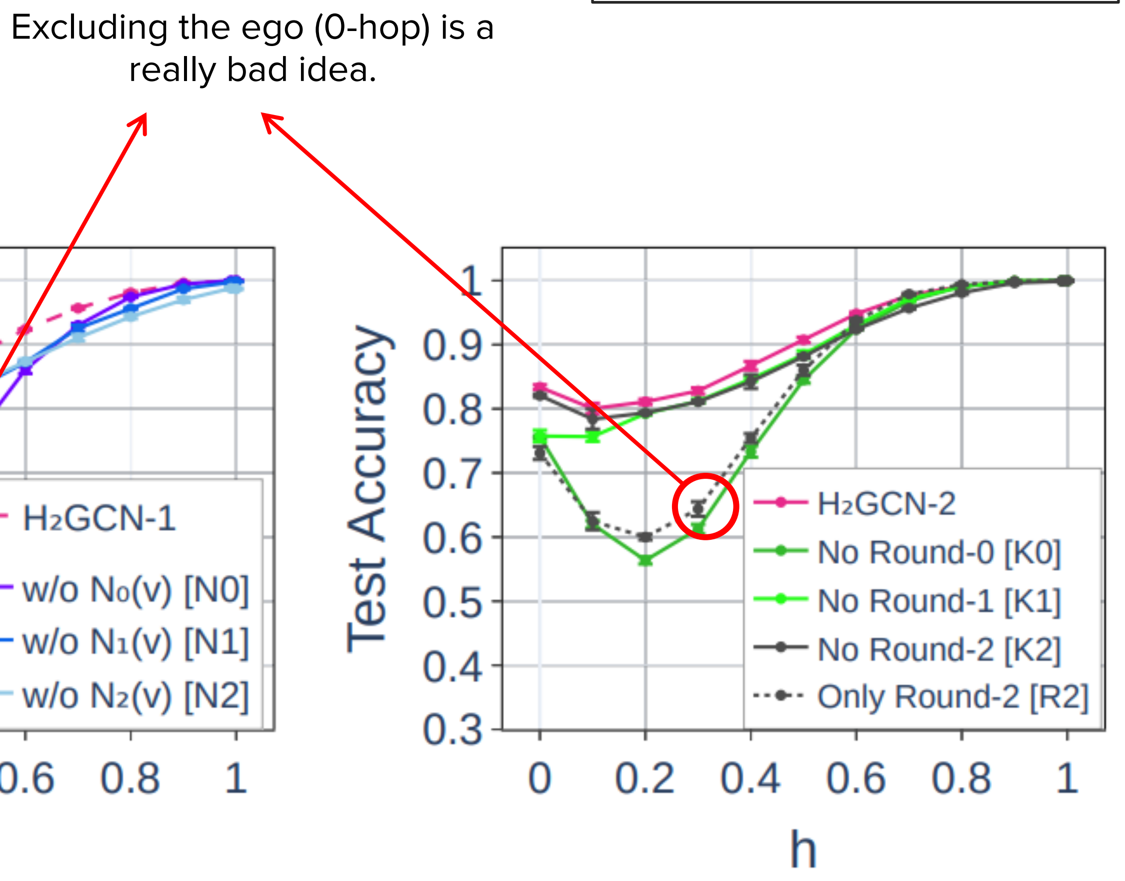
$N_0(v)$: Ego (self) node
$N_1(v)$: Immediate neighbors (1-hop)
$N_2(v)$: Neighbor's neighbor (2-hop)



Mixing ego and non-ego fails at heterophily



Ego information is very important in neighbor modeling



Ego information is very important in JK.

Takeaways

1. Label propagation is highly based on the local & global consistency assumption of the dataset

2. GNNs and label propagation are related, also aligning with the low-pass filter discussion

3. Therefore, GNNs are natural for homophilic datasets.

4. H2GCN: How to modify the message-passing architecture to bypass the homophily limitations?

*Although not included in this presentation, there are multiple studies that goes beyond homophily in GNNs, including:
Pei et al,. Geo-GCN: Gemetric graph convolutional netwokrs, ICLR 2020
Deyu Bo et al., Beyond low-frequency information in graph convolutional networks, AAAI 2021
Chien et al., Adaptime universal generalized PageRank graph neural network, ICLR 2021
and more…

Thank you!

Please feel free to ask any questions :)

jordan7186.github.io