Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks

Sitao Luan & **Mingde Zhao**, Xiao-Wen Chang, Doina Precup

Abstract

We analyze how existing Graph Convolutional Networks (GCNs) have limited expressive power due to the constraint of the activation functions and their architectures. We generalize spectral graph convolution and deep GCN in block Krylov subspace forms and devise 2 architectures, both with the potentials to be extended deeper but each making use of the multi-scale information differently. On several node classification tasks, with or without validation set, the 2 proposed architectures achieve SOTA performance.

Problem of GCN: activation function

Deepened GCN can be described as
$\mathbf{Y}' = \mathrm{L}\ \mathrm{ReLU}(\cdots \mathrm{L}\ \mathrm{ReLU}(\mathrm{L}\ \mathrm{ReLU}(\mathrm{L}\boldsymbol{X}W_0)\ W_1)\ W_2 \cdots)\ W_n$
where $L = \tilde{D}^{-1/2}\tilde{A}\ \tilde{D}^{-1/2}$, $\tilde{A}$=$A + I$ and $\tilde{D}$=$\sum_j \tilde{A}_{ij}$, $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix.

Theorem 1. Suppose the graph $\mathcal{G}$ has k connected components and the diffusion operator L is defined as above. Let $\mathbf{X}\in \mathbb{R}^{N\times F}$ be any block vector and let W_j be any non-negative parameter matrix with $\|W_j\|_2 \leq 1$ for j = 0,1, …. If $\mathcal{G}$ has no bipartite components, then we have $\lim_{n\to\infty} rank(\mathbf{Y}') \leq k$.

Conjecture 1. Theorem 1 still holds without the non-negative constraint on the parameter matrices (see Figure 1).

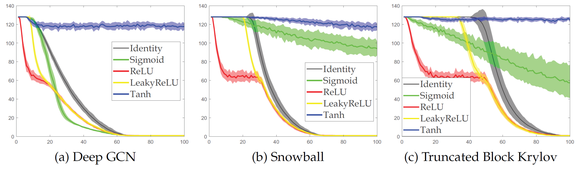
Theorem 2. Suppose the n-dimensional $\boldsymbol{x}$ and $\boldsymbol{y}$ are independently sampled from a continuous distribution and the activation function Tanh is applied to $[\boldsymbol{x}, \boldsymbol{y}]$ pointwisely, then

$$\mathbb{P}(\mathrm{rank}(\mathrm{Tanh}([\boldsymbol{x}, \boldsymbol{y}])) = \mathrm{rank}([\boldsymbol{x}, \boldsymbol{y}])) = 1$$

Problem of GCN: architecture

Definition Let $\mathbb{S}$ be a block vector subspace of $\mathbb{R}^{F\times F}$. Given a set of block vectors $\{\boldsymbol{X}_k\}_{k=1}^m \subset \mathbb{R}^{N\times F}$, the $\mathbb{S}$-span of $\{\boldsymbol{X}_k\}_{k=1}^m$ is defined as $span^{\mathbb{S}}\{\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_m}\} := \{\sum_{k=1}^m \boldsymbol{X_k} C_k : C_k \in \mathbb{S}\}$.

The order-m block Krylov subspace *w.r.t.* $\mathrm{A}\in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{N\times F}$ and $\mathbb{S}$ is defined as $\mathcal{K}_m^{\mathbb{S}}(A, \boldsymbol{B}) := span^{\mathbb{S}}\{\boldsymbol{B}, A\boldsymbol{B}, \ldots, A^{m-1}\boldsymbol{B}\}$. The corresponding block Krylov matrix is defined as $K_m(A, \boldsymbol{B}) := [\boldsymbol{B}, A\boldsymbol{B}, \ldots, A^{m-1}\boldsymbol{B}]$.



(a) Deep GCN (b) Snowball (c) Truncated Block Krylov

Figure 1: Changes in the number of independent features with the increment of network depth

Block Krylov Forms

Graph convolution with any well-defined analytic spectral filter g defined on L can be written as the product of a block Krylov matrix with a specific parameter matrix,

$$g(L)\boldsymbol{X} = \sum_{n=0}^{\infty} \frac{g^{(n)}(0)}{n!} L^n \boldsymbol{X} = [\boldsymbol{X}, L\boldsymbol{X}, L^2\boldsymbol{X}, \cdots][\frac{g^{(n)}(0)}{0!} I_F, \frac{g^{(n)}(0)}{1!} I_F, \frac{g^{(n)}(0)}{2!} I_F, \cdots]^T = A'B'$$

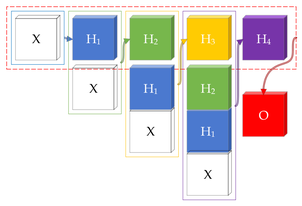
where $\rho(L) < R$, R is the convergence radius and $\rho(L)$ denotes the spectrum radius of L. $Range(A'B') = Range(A')$ and exists a smallest m *s.t.* $span^{\mathbb{S}}\{\boldsymbol{X}, L\boldsymbol{X}, L^2\boldsymbol{X} \ldots\} = span^{\mathbb{S}}\{\boldsymbol{X}, L\boldsymbol{X}, \ldots, L^{m-1}\boldsymbol{X}\}$. Then, the convolution in hidden layer can be written as $g(L)\boldsymbol{X}W' = K_m(L, \boldsymbol{X})W^{\mathbb{S}}$. A deep graph network can be written as

$$\boldsymbol{Y} = \mathrm{softmax}\Big\{K_{m_n}(L, \boldsymbol{H_n})W_n^{\mathbb{S}_n}\Big\}$$

where $\boldsymbol{H_0} = \boldsymbol{X}, \boldsymbol{H_{i+1}} = h_i\Big\{K_{m_i}(L, \boldsymbol{H_i})W_i^{\mathbb{S}_i}\Big\}$, $m_i = m(L, \boldsymbol{H_i})$ and h_i is activation function for $i = 0, \cdots n-1$.

There exist some difficulties when implementing block Krylov subspace algorithm during training, but it provides an inspiration to construct a deep graph network: we need to concatenate multi-scale information in each hidden layer so that multi-hop neighborhood features can be kept for each node.

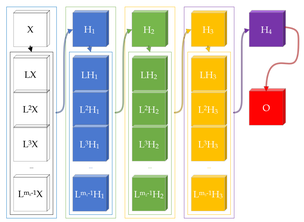
Proposed Architecture: Snowball



- $\boldsymbol{H_0} = \boldsymbol{X}, \boldsymbol{H_{l+1}} = f(L[\boldsymbol{H_0}, \boldsymbol{H_1}, \ldots, \boldsymbol{H_l}]W_l),\ l = 0,1, \ldots, n-1;$
- $\boldsymbol{C} = g([\boldsymbol{H_0}, \boldsymbol{H_1}, \ldots, \boldsymbol{H_n}]W_n);$
- Output = $\mathrm{softmax}(L^p\boldsymbol{C}W_C);$
- $W^l \in \mathbb{R}^{(\sum_{i=0}^{l} F_i)\times F_{l+1}}, W_n \in \mathbb{R}^{(\sum_{i=0}^{n} F_i)\times F_C}, W_C \in \mathbb{R}^{F_C\times F_O};$
- f, g are activation functions;
- $\boldsymbol{C}$ is a classifier of any kind, *e.g.* neural networks or even an identity layer $\boldsymbol{C} = [\boldsymbol{H_0}, \boldsymbol{H_1}, \ldots, \boldsymbol{H_n}]$;
- $p \in \{0,1\}$.

Snowball GCN stacks multi-scale information incrementally.

Proposed Architecture: Truncated Krylov



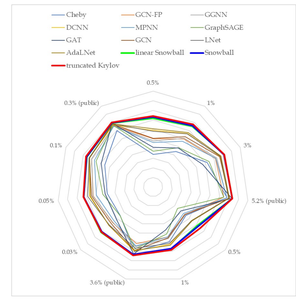
- $\boldsymbol{H_0} = \boldsymbol{X}, \boldsymbol{H_{l+1}} = f(L[\boldsymbol{H_l}, \ldots, L^{m_l-1}\boldsymbol{H_l}]W_l), l = 0,1, \ldots, n-1;$
- $\boldsymbol{C} = g(\boldsymbol{H_n}W_n);$
- Output = $\mathrm{softmax}(L^p\boldsymbol{C}W_C);$
- $W^l \in \mathbb{R}^{(m_l F_l)\times F_{l+1}}, W_n \in \mathbb{R}^{F_n\times F_C}, W_C \in \mathbb{R}^{F_C\times F_O};$
- f, g are activation functions;
- $\boldsymbol{C}$ is a classifier of any kind, e.g. neural networks or even an identity layer $\boldsymbol{C} = [\boldsymbol{H_0}, \boldsymbol{H_1}, \ldots, \boldsymbol{H_n}]$;
- $p \in \{0,1\}$.

In truncated Krylov GCN, the local information will not be diluted because in each layer l, we start the concatenation from $L^0\boldsymbol{H_l}$. It can be shown that snowball GCN with linear activation function is equivalent to a one-layer truncated block Krylov network. The 2 architectures have some inherent connections for feature extractions.

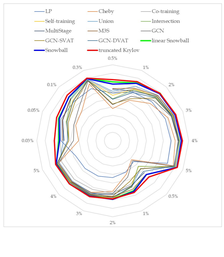
Truncated Krylov GCN stacks multi-scale information directly in each layer.

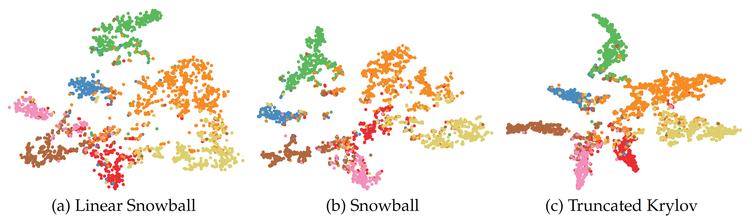
Experiments: Node Classification

Algorithms	Cora				CiteSeer				PubMed			
	0.5%	1%	3%	5.2% *public*	0.5%	1%	3.6% *public*		0.03%	0.05%	0.1%	0.3% *public*
Cheby	33.9	44.2	62.1	78.0	45.3	59.4	70.1		45.3	48.2	55.2	69.8
GCN-FP	50.5	59.6	71.7	74.6	43.9	54.3	61.5		56.2	63.2	70.3	76.0
GGNN	48.2	60.5	73.1	77.6	44.3	56.0	64.6		55.8	63.3	70.4	75.8
DCNN	59.0	66.4	76.7	79.7	53.1	62.2	69.4		60.9	66.7	73.1	76.8
MPNN	46.5	56.7	72.0	78.0	41.8	54.3	64.0		53.9	59.6	67.3	75.6
GraphSAGE	37.5	49.0	64.2	74.5	33.8	51.0	67.2		45.4	53.0	65.4	76.8
GAT	41.4	48.6	56.8	83.0	38.2	46.5	72.5		50.9	50.4	59.6	79.0
GCN	50.9	62.3	76.5	80.5	43.6	55.3	68.7		57.9	64.6	73.0	77.8
LNet	58.1	66.1	77.3	79.5	53.2	61.3	66.2		60.4	68.8	73.4	78.3
AdaLNet	60.8	67.5	77.7	80.4	53.8	63.3	68.7		61.0	66.0	72.8	78.1
linear Snowball	**71.7**	**75.2**	**81.5**	**83.7**	**61.2**	**67.3**	**73.4**		**71.7**	**73.2**	**75.7**	**79.2**
Snowball	**72.9**	**76.2**	**81.1**	**83.3**	**62.4**	**67.0**	**73.2**		**70.9**	**73.0**	**76.1**	**79.5**
truncated Krylov	**74.1**	**77.6**	**81.8**	**83.5**	**65.3**	**68.2**	**74.2**		**71.5**	**73.2**	**77.0**	**80.1**



Algorithms	Cora						CiteSeer						PubMed			
	0.5%	1%	2%	3%	4%	5%	0.5%	1%	2%	3%	4%	5%	0.03%	0.05%	0.1%	0.3%
LP	56.4	62.3	65.4	67.5	69.0	70.2	34.8	40.2	43.6	45.3	46.4	47.3	61.4	66.4	65.4	66.8
Cheby	38.0	52.0	62.4	70.8	74.1	77.6	31.7	42.8	59.9	66.2	68.3	69.3	40.4	47.3	51.2	72.8
Co-training	56.6	66.4	73.5	75.9	78.9	80.8	47.3	55.7	62.1	62.5	64.5	65.5	62.2	68.3	72.7	78.2
Self-training	53.7	66.1	73.8	77.2	79.4	80.0	43.3	58.1	68.2	69.8	70.4	71.0	51.9	58.7	66.8	77.0
Union	58.5	69.9	75.9	78.5	80.4	81.7	46.3	59.1	66.7	66.7	67.6	68.2	58.4	64.0	70.7	79.2
Intersection	49.7	65.0	72.9	77.1	79.4	80.2	42.9	59.1	68.6	70.1	70.8	71.2	52.0	59.3	69.7	77.6
MultiStage	61.1	63.7	74.4	76.1	77.2		53.0	57.8	63.8	68.0	69.0		57.4	64.3	70.2	
M3S	61.5	67.2	75.6	77.8	78.0		56.1	62.1	66.4	70.3	70.5		59.2	64.4	70.6	
GCN	42.6	56.9	67.8	74.9	77.6	79.3	33.4	46.5	62.6	66.9	68.7	69.6	46.4	49.7	56.3	76.6
GCN-SVAT	43.6	53.9	71.4	75.6	78.3	78.5	47.0	52.4	65.8	68.6	69.5	70.7	52.1	56.9	63.5	77.2
GCN-DVAT	49	61.8	71.9	75.9	78.4	78.6	51.5	58.5	67.4	69.2	70.8	71.3	53.3	58.6	66.3	77.3
linear Snowball	**69.5**	**74.1**	**79.4**	**80.4**	**81.3**	**82.4**	**56.8**	**64.3**	**68.8**	**71.0**	**72.2**	**72.3**	**64.1**	**69.5**	**72.9**	**79.3**
Snowball	**67.2**	**73.5**	**78.5**	**80.0**	**81.5**	**82.3**	**56.7**	**65.0**	**69.5**	**71.1**	**72.3**	**72.6**	**62.9**	**68.3**	**73.3**	**79.5**
truncated Krylov	**72.2**	**75.7**	**79.2**	**80.9**	**82.5**	**82.8**	**60.9**	**66.2**	**69.0**	**71.8**	**72.4**	**73.2**	**69.1**	**71.8**	**76.1**	**80.1**





(a) Linear Snowball (b) Snowball (c) Truncated Krylov

Figure 3: t-SNE for the extracted features trained on Cora (7 classes) public (5.2%).

We use RMSprop for tests with validation and Adam for those without. For each case, all the compared methods are fine-tuned.

The performance of tuned instances of the proposed architectures is better than the tuned baselines in EVERY test case.

Surprisingly better performance can be observed in smaller splits: the proposed architectures are particularly well-performing while the train split is small.

References

[1] A. Frommer, K. Lund, and D. B. Szyld. Block Krylov subspace methods for functions of matrices. *Electronic Transactions on Numerical Analysis*, 47:100–126, 2017.

[2] M. H. Gutknecht and T. Schmelzer. The block grade of a block krylov space. *Linear Algebra and its Applications*, 430(1):174–185, 2009.

[3] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *ICLR*, 2019.

[4] Q. Li, Z. Han, and X. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI*, 2018.

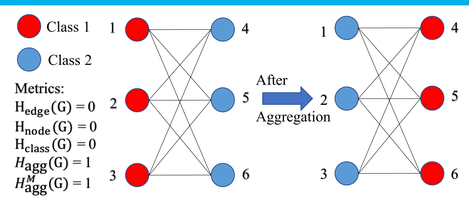
Revisiting Heterophily For Graph Neural Networks

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, Doina Precup

Abstract

We first revisit the widely used homophily metrics and point out their shortcomings. Then, we study heterophily from the perspective of post-aggregation node similarity and define new homophily metrics, which are more informative. Based on this investigation, we prove that some harmful cases of heterophily can be effectively addressed by local diversification operation. Then, we propose the Adaptive Channel Mixing (ACM), a framework to adaptively exploit aggregation, diversification and identity channels node-wisely for diverse node heterophily situations. When evaluated on 10 benchmark node classification tasks, ACM-augmented baselines consistently achieve significant performance gain, exceeding state-of-the-art GNNs on most tasks without incurring significant computational burden.

Shortcomings of Homophily Metrics



Homophily: the edges tend to connect nodes from the same class, and it only describes graph-label consistency. The inconsistency relation, i.e., low homophily or heterophily, is implied to have a negative effect to the performance of GNNs, because nodes from different classes will be falsely mixed and become indistinguishable. Nevertheless, it is not always the case, e.g., the bipartite graph shown above is highly heterophilic after mean aggregation, the nodes in classes 1 and 2 just exchange colors and are still distinguishable.

Aggregation Homophily

We first define the post-aggregation node similarity matrix as $S(\hat{A},X) \equiv \hat{A}X(\hat{A}X)^T \in \mathbb{R}^{N\times N}$, where $\hat{A}$ denotes a general aggregation operator and X denotes the feature matrix. $S(\hat{A},X)$ strongly relates to the gradient of SGC and a large $[S(\hat{A},X)]_{ij}$ means node i tends to be updated to the same class as node j. This indicates that $S(\hat{A},X)$ is closely related to the performance of a single layer GNN.

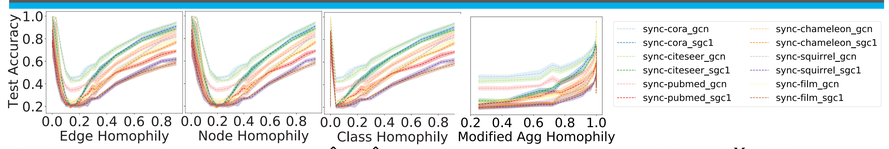
We define the aggregation similarity score $S_{agg}\left(S(\hat{A},X)\right)$ as: $\frac{1}{|V|}|\{v|\text{Mean}_u\left(\{S(\hat{A},X)_{v,u}\middle|Z_{u,:}=Z_{v,:}\}\right) \geq \text{Mean}_u\left(\{S(\hat{A},X)_{v,u}\middle|Z_{u,:}\neq Z_{v,:}\}\right)\}|$

$S_{agg}\left(S(\hat{A},X)\right)$ measures the proportion of nodes v as which the average weights on the set of nodes in the same class (including v) is larger than that in other classes.

Aggregation homophily and its modified version are defined as:

$$H_{agg}(G) = S_{agg}\left(S(\hat{A},Z)\right)$$
$$H^M_{agg}(G) = \left[2\,S_{agg}\left(S(\hat{A},Z)\right) - 1\right]_+$$

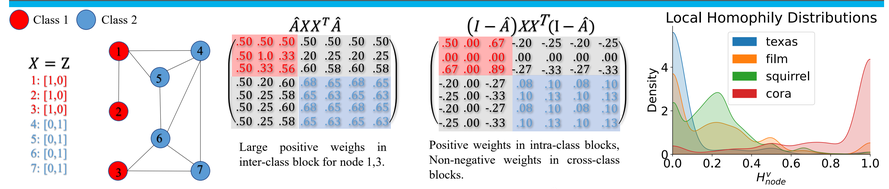
Comparison on Synthetic Graphs



In the previous example, when $\hat{A} = \hat{A}_{rw}$, it is easy to see that $H_{agg}(G) = H^M_{agg}(G) = 1$ and other metrics are 0. Thus, this new metric reflects the fact that nodes in classes 1 and 2 are still highly distinguishable after aggregation, while other metrics fail to capture such information and misleadingly give value 0. This shows the advantage of $H_{agg}(G)$ and $H^M_{agg}(G)$, which additionally exploit information from aggregation operator $\hat{A}$ and the similarity matrix.

To comprehensively compare the metrics, we generate synthetic graphs with different homophily levels and evaluate 1-hop SGC and GCN on them. The performance of SGC-1 and GCN is expected to be monotonically increasing if the homophily metric is informative. However, the above figures show that the performance curves under edge, node and class homophily are U-shaped, while for $H^M_{agg}(G)$, it reveals a nearly monotonic curve with a little numerical perturbation around 1. This indicates that $H^M_{agg}(G)$ provides a better indication of the way in which the graph structure affects the performance of SGC-1 and GCN than existing metrics.

Diversification Distinguishability and High-pass Filters



Diversification operation, i.e., high-pass(HP) filter, $I-\hat{A}$ is empirically found to be useful for heterophily problem. In the above example, $S(\hat{A},X)$ assigns large positive weights to pairs of nodes in different classes, which will make nodes hard to be distinguished from each other. However, $S(I-\hat{A},X)$ assigns positive values to nodes in the same class and negative values to nodes from different classes. This indicates that in some heterophily cases, nodes are still distinguishable through their local surrounding dissimilarities. Thus, we define Diversification Distinguishability (DD) based on $S(I-\hat{A},X)$.

Definition. Given $S(I-\hat{A},X)$, a node v is diversification distinguishable if the following two conditions are satisfied at the same time.

$1.\,Mean_u\left(\{S(I-\hat{A},X)_{v,u}\middle|u\in V, Z_{u,:}=Z_{v,:}\}\right) \geq 0;\ 2.\,Mean_u\left(\{S(I-\hat{A},X)_{v,u}\middle|u\in V, Z_{u,:}\neq Z_{v,:}\}\right) < 0;$

Then, graph diversification distinguishability value is defined as

$$DD_{\hat{A},X}(G) = \frac{1}{|V|}\,|\{v|v\in V, v \text{ is diversification distinguishable}\}|$$

Theorem: For $C=2$, suppose $X=Z$, $\hat{A}=\hat{A}_{rw}$, then all nodes are diversification distinguishable and $DD_{\hat{A},Z}(G)=1$.

Adaptive Channel Mixing (ACM)

HP filter is proved to be effective for some heterophily cases. Therefore, we propose to include low-pass channel, high-pass channel and identity channel together in each GNN layers instead of using the uni-channel architecture. Besides, different nodes may have different local heterophily situations (see the above figure on homophily distributions) and may have different needs for the information from different channels. So we propose a node-wise channel mixing mechanism to adaptively exploit the channel information. We will use GCN as an example to introduce the ACM framework in matrix form, but the framework can be combined in a similar manner to many different GNNs. The ACM framework includes the following steps:

Step 1. Feature Extraction for Each Channel:

Option 1: $H^l_L = \text{ReLU}\left(H_{LP}H^{l-1}W^{l-1}_L\right), H^l_H = \text{ReLU}\left(H_{HP}H^{l-1}W^{l-1}_H\right), H^l_I = \text{ReLU}\left(IH^{l-1}W^{l-1}_I\right);$

Option 1: $H^l_L = H_{LP}\text{ReLU}\left(H^{l-1}W^{l-1}_L\right), H^l_H = H_{HP}\text{ReLU}\left(H^{l-1}W^{l-1}_H\right), H^l_I = I\text{ReLU}\left(H^{l-1}W^{l-1}_I\right);$

$H^0 = \in \mathbb{R}^{N\times F_0}, W^{l-1}_L, W^{l-1}_H, W^{l-1}_I \in \mathbb{R}^{F_{l-1}\times F_l},\ l = 1,\dots,L;$

Step 2. Row-wise Feature-based Weight Learning:

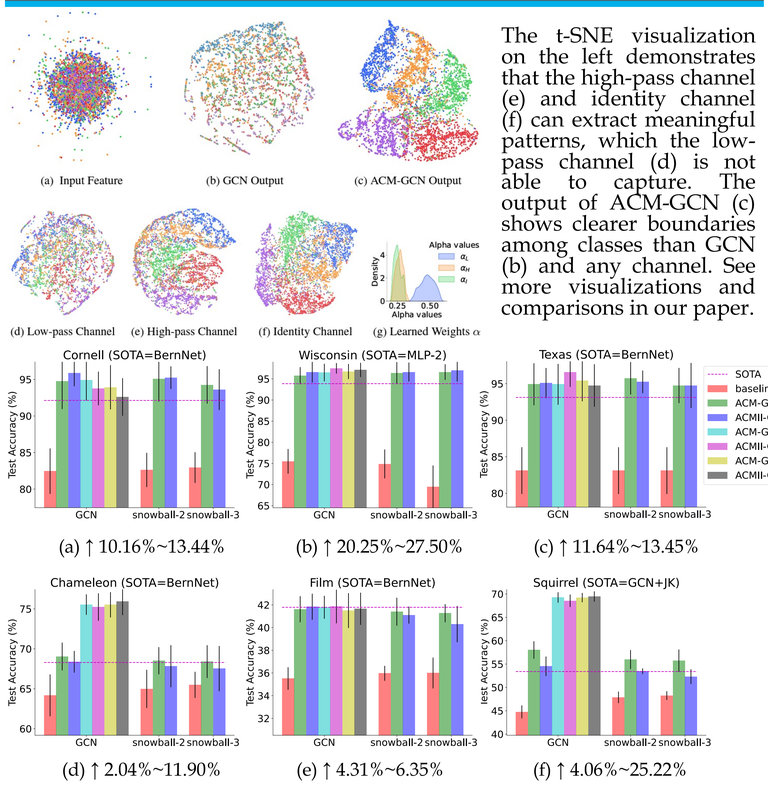
$\tilde{\alpha}^l_L = \text{Sigmoid}\left(H^l_L\widetilde{W}^l_L\right),\ \tilde{\alpha}^l_H = \text{Sigmoid}\left(H^l_H\widetilde{W}^l_H\right),\ \tilde{\alpha}^l_I = \text{Sigmoid}\left(H^l_I\widetilde{W}^l_I\right),\ \widetilde{W}^l_L,\ \widetilde{W}^l_H,\ \widetilde{W}^l_I \in \mathbb{R}^{F_l\times 1};$

$[\alpha^l_L, \alpha^l_H, \alpha^l_I] = \text{Softmax}\left(\left(\frac{[\tilde{\alpha}^l_L, \tilde{\alpha}^l_H, \tilde{\alpha}^l_I]}{T}\right)W^l_{\text{Mix}}\right) \in \mathbb{R}^{N\times 3}, T\in\mathbb{R}$ is temperature, $W^l_{\text{Mix}} \in \mathbb{R}^{3\times 3};$

Step 3. Node-wise Adaptive Channel Mixing:

$H^l = \text{ReLU}\left(\text{diag}(\alpha^l_L)H^l_L + \text{diag}(\alpha^l_H)H^l_H + \text{diag}(\alpha^l_I)H^l_I\right).$

Experiments: Node Classification



The t-SNE visualization on the left demonstrates that the high-pass channel (e) and identity channel (f) can extract meaningful patterns, which the low-pass channel (d) is not able to capture. The output of ACM-GCN (c) shows clearer boundaries among classes than GCN (b) and any channel. See more visualizations and comparisons in our paper.

To visualize the performance, we plot the bar charts of the test accuracy of SOTA models, three selected baselines (GCN, snowball-2, snowball-3), their ACM(II) augmented models, ACM(II)-GCN+ and ACM(II)-GCN++ on the 6 most commonly used benchmark heterophily datasets. From the above figures we can see that (1) after being combined with the ACM or ACMII framework, the performance of the three baseline models is ***significantly boosted, by 2.04%~27.50%*** on all the 6 tasks. The ACM and ACMII in fact achieve SOTA performance. (2) On *Cornell, Wisconsin, Texas, Chameleon* and *Squirrel*, the augmented baseline models ***significantly outperform the current SOTA models***. Overall, these results suggest that the proposed approach can help GNNs to generalize better on node classification tasks on heterophilic graphs.

See our paper for ablation study, more performance comparisons, details of implementation and discussion of limitations.

Complete the Missing Half: Augmenting Aggregation Filtering with Diversification for Graph Convolutional Networks

Sitao Luan, Mingde Zhao, Chenqing Hua, Xiao-Wen Chang, Doina Precup

Abstract

The core operation of current Graph Neural Networks (GNNs) is the aggregation enabled by the graph Laplacian or message passing, which filters the neighborhood information of nodes. Though effective, in this paper, we show that they are potentially a problematic factor underlying all GNN models for learning on certain datasets, as they force the node representations similar, making the nodes gradually lose their identity and become indistinguishable. Hence, we augment the aggregation operations with their dual, i.e., diversification operators that make the node more distinct and preserve the identity. Such augmentation replaces the aggregation with a two-channel filtering process that, in theory, is beneficial for enriching the node representations. In practice, the proposed two-channel filters can be easily patched on existing GNN methods with diverse training strategies, including spectral and spatial methods. In the experiments, we observe desired characteristics of the models and significant performance boost upon the baselines on 9 node classification tasks.

Measure of Smoothness

Dirichlet Energy can measure the global smoothness of the signal. And the signal energy represents the amount of contents in a graph signal. They are defined as follows.

Dirichlet Energy & Graph Signal Energy The Dirichlet energy of vector block $\boldsymbol{X}$ and column vector $\boldsymbol{x}$ defined on $\mathcal{G}$ are separately defined as

$$E_S^{\mathcal{G}}(\boldsymbol{X}) = tr(\boldsymbol{X}^T L \boldsymbol{X}), E_S^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^T L \boldsymbol{x}$$

The signal energy of vector block $\boldsymbol{X}$ and column vector $\boldsymbol{x}$ defined on $\mathcal{G}$ are separately defined as

$$E^{\mathcal{G}}(\boldsymbol{X}) = tr(\boldsymbol{X}^T \boldsymbol{X}), E^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{x}$$

$E_S^{\mathcal{G}}(\boldsymbol{x})$ and $E^{\mathcal{G}}(\boldsymbol{x})$ can be rewritten as

$$E_S^{\mathcal{G}}(\boldsymbol{x}) = \sum_i \lambda_i \left\| u_i^T \boldsymbol{x} \right\|_2^2, E^{\mathcal{G}}(\boldsymbol{x}) = \sum_i \left\| u_i^T \boldsymbol{x} \right\|_2^2$$

where $\boldsymbol{u}_i$ and λ_i are eigenvector and eigenvalue of L. Signal $\boldsymbol{x}$ can be decomposed into smooth and non-smooth components, and the amount the non-smooth component can be measured by

$$E_{NS}^{\mathcal{G}}(\boldsymbol{x}) = E^{\mathcal{G}}(\boldsymbol{x}) - E_S^{\mathcal{G}}(\boldsymbol{x}) = (1-\lambda_i) \sum_i \left\| u_i^T \boldsymbol{x} \right\|_2^2$$

$E_{NS}^{\mathcal{G}}(\boldsymbol{x})$ can be negative and a small $E_{NS}^{\mathcal{G}}(\boldsymbol{x})$ indicates that $\boldsymbol{x}$ is highly non-smooth. Upon the above analysis, we define $S(\boldsymbol{x})$ to measure the smoothness of a signal as follows

$$S(\boldsymbol{x}) = \frac{E_S^{\mathcal{G}}(\boldsymbol{x})}{E^{\mathcal{G}}(\boldsymbol{x})}, S(\boldsymbol{X}) = \frac{E_S^{\mathcal{G}}(\boldsymbol{X})}{E^{\mathcal{G}}(\boldsymbol{X})}$$

Signal with a small S-value means it is smooth on $\mathcal{G}$.

Filterbanks in GNNs: From One Channel to Two

Table 1: Dataset Overview: Network Characteristics and S-values measured by L_{sym}

datasets		Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
Network Info	#nodes	183	251	183	7600	2277	5201	2708	3327	19717
	#edges	295	499	309	33544	36101	217073	5429	4732	44338
	#features	1703	1703	1703	931	2325	2089	1433	3703	500
	#classes	5	5	5	5	5	5	7	6	3
S-values	input feature	0.904	0.873	0.854	0.901	0.99	0.987	0.862	0.799	0.832
	label	0.883	0.877	0.909	0.836	0.747	0.782	0.288	0.35	0.501
	diff (label - feature)	-0.021	0.004	0.055	-0.065	-0.243	-0.205	-0.574	-0.449	-0.331

We use blue and red shades to demonstrate the relation between label and feature: the label of the blue shaded datasets is smoother than its feature and red datasets are less smooth.

We measure the smoothness of 9 frequently used benchmark datasets and present the results with the characteristics for each graph in Table 1. It shows that the input features and ground truth labels of different datasets are all mixtures of smooth and non-smooth graph signals but in different proportions. Besides, it illustrates that different tasks have different demands of learning how to smoothen the input signals. For example, in *Cora, CiteSeer and PubMed*, the ground truth labels are much smoother than the input features, such pattern motivates us to learn how to smoothen the input signals; while in *Wisconsin* and *Texas*, the labels are less smooth than the input features, thus there is no reason that we still learn how to smoothen the input signals. Therefore, to accommodate different situations, we propose that we should learn both smooth and non-smooth components of the input features adaptively instead of merely extracting the smooth part. This motivates us to use filterbanks in GNNs.

Low-pass(LP), High-pass(HP) Graph Filters and Filterbanks The multiplication of L and $\boldsymbol{x}$ acts as a filtering operation over $\boldsymbol{x}$, adjusting the scale of the components of $\boldsymbol{x}$ in frequency domain. To see this, consider

$$\boldsymbol{x} = \sum_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}, \; L\boldsymbol{x} = \sum_i \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x},$$

The projection $\boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}$ corresponding to a large $|\lambda_i|$ will be amplified, while the one corresponding to a small $|\lambda_i|$ will be suppressed. More specifically, a graph filter that filters out smooth (non-smooth) components is called HP (LP) filter. On the node level, left multiplying HP and LP filters on $\boldsymbol{x}$ can be understood as diversification and aggregation operations, respectively. For example, if we implement L_{rw} and A_{rw} on the i^{th} node, we have

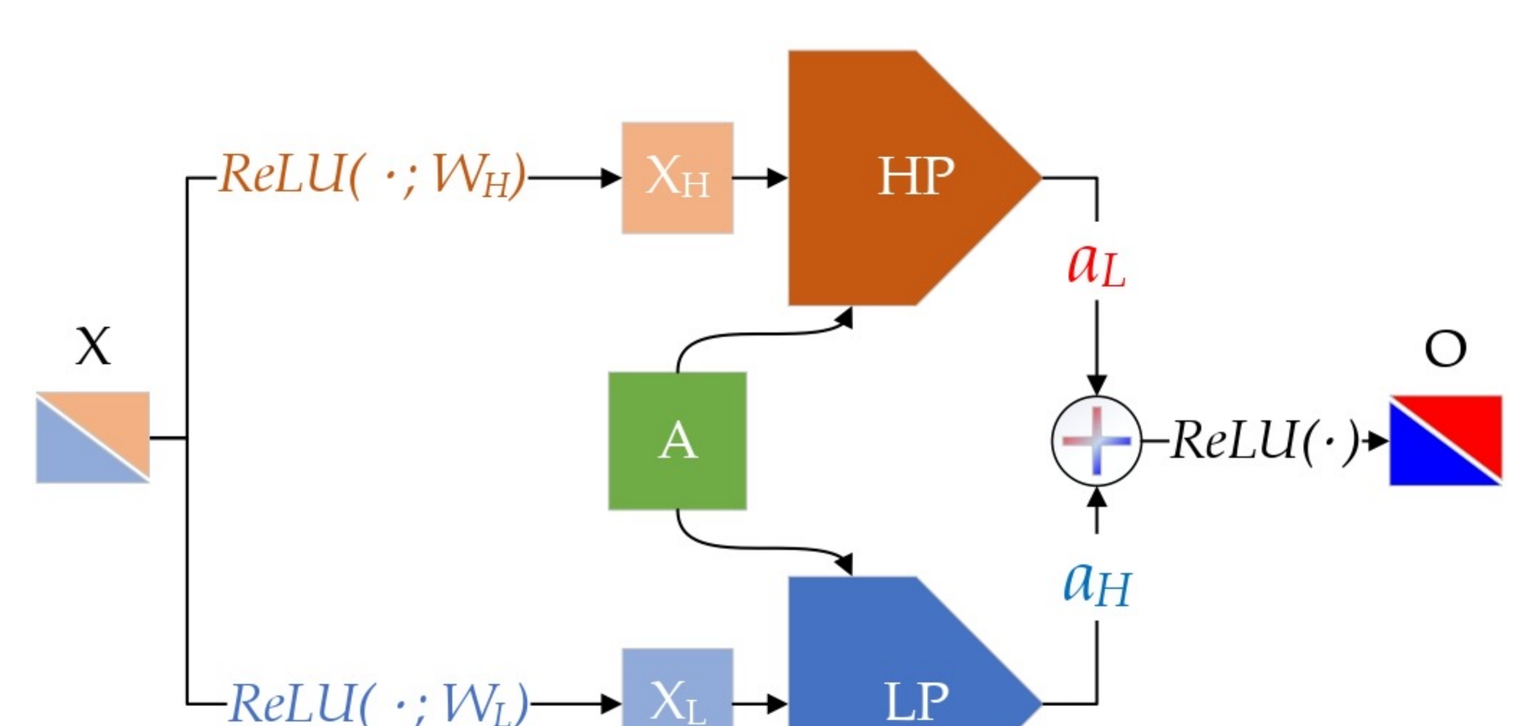
$$(L_{rw}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_j} \frac{1}{D_{ii}} (\boldsymbol{x}_i - \boldsymbol{x}_j), \; (A_{rw}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_j} \frac{1}{D_{ii}} \boldsymbol{x}_j$$

Intuitively, HP filters depict the differences between one node and its neighbors; While LP filters focus on the similarity within a neighborhood. We believe that these two conjugate components are both indispensable to portray a node. Mathematically, multiplying with LP filter (aggregation) is a linear projection, which will project the features to a fixed subspace. We will lose part of feature information by only using LP filter, and the missing half is the HP component of the learned signals, as $L_{LP} + L_{HP} = I$, which satisfies the perfect reconstruction property.

Filterbank assisted GNNs (FB-GNNs)

$$H_L^l = L_{LP}\text{ReLU}(H^{l-1}W_L^{l-1})$$
$$H_H^l = L_{HP}\text{ReLU}(H^{l-1}W_H^{l-1})$$
$$\text{H}^l = \text{ReLU}(\alpha_L^l \cdot H_L^l + \alpha_H^l \cdot H_H^l)$$
$$l = 1, \dots, n, H^0 = \boldsymbol{X}, \alpha_L^l, \alpha_H^l \in (0,1)$$

W_L^{l-1} and W_H^{l-1} are learnable parameter matrices for the non-linear feature extractors focusing on disentangling the smooth and non-smooth information from input H^{l-1}.



Experiments: Node Classification

Table 2: Supervised Learning of Shallow GNNs

Models/Datasets	Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
Diff of S-values	-0.021	0.004	0.055	-0.065	-0.243	-0.205	-0.574	-0.449	-0.330
Spatial Methods(%)									
GraphSAINT	70.27	71.35	72.97	17.89	43.86	33.27	84.69	73.2	89.42
FB-GraphSAINT	78.38(8.11)	80(8.65)	75.68(2.71)	19.08(1.19)	46.05(2.19)	36.06(2.79)	87.5(2.81)	74.76(1.56)	89.88(0.46)
GraphSAGE	54.05	66	56.76	14.67	40.13	24.14	80.64	72.36	85.49
FB-GraphSAGE	63.14(9.09)	70(4)	58.05(1.29)	23.27(8.6)	39.74(-0.39)	24.6(0.46)	83.7(3.06)	72.58(0.22)	86.31(0.82)
Spectral Methods(%)									
GAT	54.32	49.41	58.38	28.45	42.93	30.03	86.37	74.32	87.62
FB-GAT	64.86(10.54)	60.78(11.37)	64.86(6.48)	30.66(2.21)	47.37(4.44)	31.8(1.77)	**88.73(2.36)**	77.12(2.8)	88.16(0.54)
GCN	52.7	45.88	52.16	26.86	28.18	23.96	85.77	73.68	88.13
FB-GCN	62.16(9.46)	56.86(10.98)	62.16(10.00)	31.21(4.35)	32.89(4.71)	24.73(0.77)	85.92(0.15)	75.24(1.56)	88.54(0.41)
Geom-GCN-P	60.81	64.12	67.57	**31.63**	60.9	38.14	84.93	75.14	88.09
FB-Geom-GCN-P	64.86(4.05)	72.55(8.43)	70.27(2.70)	31.02(-0.61)	**67.20(6.30)**	**49.66(11.52)**	85.17(0.24)	76.23(1.09)	88.25(0.16)
Geom-GCN-S	55.68	56.67	59.73	30.3	59.96	36.24	85.27	74.71	84.75
FB-Geom-GCN-S	56.54(0.86)	56.94(0.27)	62.16(2.43)	31.25(0.95)	61.49(1.53)	37.27(1.03)	85.43(0.16)	75.21(0.5)	85.88(1.13)
Geom-GCN-I	56.76	58.24	57.58	29.09	60.31	33.32	85.19	**77.99**	90.05
FB-Geom-GCN-I	57.38(0.62)	60.68(2.44)	62.21(4.63)	31.45(2.36)	60.76(0.45)	35.27(1.95)	85.45(0.26)	77.69(-0.3)	**90.48(0.43)**
GWNN	70.67	72.22	69.44	20.92	33.63	29.13	84.49	72.47	83.6
FB-GWNN	**80.11(9.44)**	**84.67(12.45)**	**77.78(8.34)**	22.24(1.32)	37.36(3.73)	30.6(1.47)	85.6(1.11)	72.83(0.36)	85.92(2.32)
Baseline Average	59.41	60.49	62.32	26.2	46.46	31.44	85.15	74.33	87.3
FB-Baseline Average	65.93(6.52)	67.81(**7.32**)	66.65(4.33)	27.52(1.32)	49.11(2.65)	33.75(2.31)	85.94(1.27)	75.21(0.97)	87.93(0.63)

The results are averaged from 10 independent runs. The (values) represent the difference of performance brought by patching FB.

Table 4: Supervised Learning of Deep Multi-scale GNNs

Models\Datasets	Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
GCNII-8	70.54	73.88	71.08	33.7	60.61	37.49	85.69	75.54	88.62
FB-GCNII-8	75.95(5.41)	82.35(8.47)	74.59(3.51)	35.37(1.67)	60.43(-0.18)	39.69(2.2)	86.04(0.35)	75.51(-0.03)	89.97(1.35)
GCNII-16	74.86	74.12	69.46	33.62	55.48	35.98	87.3	76.54	88.28
FB-GCNII-16	**77.57(2.71)**	82.55(8.43)	77.03(7.57)	35.12(1.5)	56.78(1.3)	39.38(3.4)	87.5(0.2)	76.67(0.13)	89.39(1.11)
GCNII-32	72.7	70.2	69.46	31.61	53.71	35.92	88.13	76.08	87.89
FB-GCNII-32	72.81(0.11)	78.63(8.43)	80.27(10.81)	32.99(1.38)	54.98(1.27)	36.81(0.89)	88.33(0.2)	76.49(0.41)	89.1(1.21)
GCNII-64	71.89	68.84	66.49	28.76	54.14	36.1	**88.49**	77.08	89.57
FB-GCNII-64	76.49(4.6)	76.27(7.43)	76.22(9.73)	29.57(0.81)	54.39(0.25)	36.79(0.69)	87.92(-0.57)	77(-0.08)	89.65(0.08)
GCNII*-8	72.97	78.82	72.7	34.89	62.48	40.72	86.14	75.06	89.7
FB-GCNII*-8	76.76(3.79)	**82.94(4.12)**	78.11(5.41)	**35.87(0.98)**	**65.11(2.63)**	41.19(0.47)	86.94(0.8)	76.32(1.26)	90.2(0.5)
GCNII*-16	76.49	81.57	75.41	34.18	58.86	39.88	87.46	75.8	86.69
FB-GCNII*-16	76.95(0.46)	82.39(0.82)	76.76(1.35)	35.4(1.22)	59.98(1.12)	40.08(0.2)	87.48(0.02)	76.43(0.63)	89.95(3.26)
GCNII*-32	74.32	77.06	77.84	33.78	56.27	37.69	88.35	76.55	89.37
FB-GCNII*-32	74.51(0.19)	80.78(3.72)	**84.86(7.02)**	34.73(0.95)	57.65(1.38)	**41.24(3.55)**	88.16(-0.19)	76.89(0.34)	89.92(0.55)
GCNII*-64	72.43	73.53	75.41	32.72	53.82	36.83	88.01	**77.13**	**90.3**
FB-GCNII*-64	75.84(3.41)	81.57(8.04)	80.54(5.13)	34.89(2.17)	57.52(3.7)	39.81(2.98)	87.44(-0.57)	77.03(-0.1)	89.98(-0.32)
Baseline Average	73.28	74.75	72.23	32.91	56.92	37.58	87.45	76.22	88.8
FB-Baseline Average	75.86(2.58)	80.94(6.19)	78.55(**6.32**)	34.24(1.33)	58.36(1.44)	39.37(1.79)	87.48(0.03)	76.54(0.32)	89.77(0.97)

The results are averaged from 10 independent runs. The (values) represent the difference of performance brought by patching FB.

Shallow GNNs From the results we can see that, our proposed method generally boosts the performance of baseline GNNs in almost all cases, especially when the labels are not much smoother than the input features, considering the S-values in Table 1. Table 3 shows that the S-values of FB-GCN outputs are closer to the S-values of the ground truth labels (see the absolute differences in the bracket) compared with those of GCN outputs. This indicates that FB-GCN is able to learn better representations which can reconstruct both the smooth and non-smooth part of the ground truth labels.

Deep GNNs We build deep multi-hop filterbank models based on the architecture of GCNII and GCNII* to see if the two-channel method is capable to assist deep GNN models. We report mean accuracy, highlight best performing depth, and record performance difference in brackets in Table 4. In general, FB-GCNII and FB-GCNII* achieve better results than the unpatched GCNII and GCNII* at different depths, especially on *Wisconsin* and *Texas*, where the non-smooth part of representations are desirable.

See ablation study, t-SNE visualization of the output layer and more discussion of filters in our paper.

Table 3: Statistics of Datasets (measured by $\hat{L}_{\text{sym}}$ instead of L_{sym}) and Comparison of the Output Smoothness

datasets		Cornell	Wisconsin	Texas	Actor	Chameleon	Squirrel	Cora	CiteSeer	PubMed
S-values	input feature	0.172	0.385	0.205	0.567	0.831	0.87	0.617	0.515	0.529
	label	*0.139*	*0.328*	*0.301*	*0.511*	*0.638*	*0.681*	*0.188*	*0.209*	*0.272*
	diff (label - feature)	-0.033	-0.057	0.096	-0.056	-0.193	-0.189	-0.429	-0.306	-0.257
	GCN output	*0.037 (0.102)*	*0.124 (0.204)*	*0.139 (0.162)*	*0.397 (0.114)*	*0.595 (0.043)*	*0.578 (0.103)*	*0.156 (0.032)*	*0.112 (0.097)*	*0.234 (0.038)*
	FB-GCN output	***0.099 (0.040)***	***0.269 (0.059)***	***0.201 (0.100)***	***0.531 (0.020)***	***0.655 (0.017)***	***0.683 (0.002)***	***0.172 (0.016)***	***0.148 (0.061)***	***0.247 (0.025)***

These results are obtained from 10 independent runs. The stds are negligible so they are not presented (mostly < 0.002).This table shows how FB- patched baseline could better reconstruct the label smoothness, *i.e.,* we want the S-value of the output to be closer to that of the labels. The (values) stand for the absolute difference between the S-values of the output of the methods and those of the ground truths. Better reconstruction between the two methods on each task is marked **bold**.

References

[1] Luan, S., Zhao, M., Chang, X. W., & Precup, D. (2019). Break the ceiling: Stronger multi-scale deep graph convolutional networks. *Advances in neural information processing systems*, *32*.

[2] Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X., & Precup, D. (2022). Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, *35*, 1362-1375.

[3] Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X., & Precup, D. (2021). Is heterophily a real nightmare for graph neural networks to do node classification?. *arXiv preprint arXiv:2109.05641*.

[4] Luan, S., Hua, C., Lu, Q., Zhu, J., Chang, X. W., & Precup, D. (2022). When Do We Need GNN for Node Classification?. *arXiv preprint arXiv:2210.16979*.

[5] Ekambaram, V. N. (2014). *Graph-structured data viewed through a Fourier lens*. University of California, Berkeley.

[6] Ekambaram, V. N., Fanti, G., Ayazifar, B., & Ramchandran, K. (2013, December). Critically-sampled perfect-reconstruction spline-wavelet filterbanks for graph signals. In *2013 IEEE Global Conference on Signal and Information Processing* (pp. 475-478). IEEE.

High-Order Pooling For Graph Neural Network With Tensor Decomposition

Chenqing Hua, Guillaume Rabusseau, Jian Tang

ABSTRACT

We propose the Tensorized Graph Neural Network (tGNN), a highly expressive GNN architecture relying on tensor decomposition to model high-order non-linear node interactions. tGNN leverages the symmetric CP decomposition to efficiently parameterize permutation-invariant multilinear maps for modeling node interactions. Theoretical and empirical analysis on both node and graph classification tasks show the superiority of tGNN over competitive baselines. In particular, tGNN achieves the most solid results on two OGB node classification datasets and one OGB graph classification dataset.

TENSOR

A k-th order tensor $\mathcal{T} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_k}$ can simply be seen as a multidimensional array.
The mode-i fibers of $\mathcal{T}$ are the vectors obtained by fixing all indices except the i-th one: $\mathcal{T}_{n_1,n_2,\ldots,n_{i-1},:,n_{i+1},\ldots,n_k} \in \mathbb{R}^{N_i}$.
The i-th mode matricization of a tensor is the matrix having its mode-i fibers as columns and is denoted by $\mathcal{T}_{(i)}$.
A Rank R CP decomposition factorizes a k-th order tensor $\mathcal{T} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_k}$ into the sum of R rank one tensors as
$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \boldsymbol{v_{1r}} \circ \boldsymbol{v_{2r}} \circ \cdots \circ \boldsymbol{v_{kr}}, v_{1r} \in \mathbb{R}^{N_1}, \ldots, v_{kr} \in \mathbb{R}^{N_k}$.
For mode-i product between a tensor $\mathcal{T} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_k}$ and a vector $v \in \mathbb{R}^{N_i}$,

$$(\boldsymbol{\mathcal{T}} \times_i \boldsymbol{v})_{n_1,\ldots,n_{i-1},n_{i+1},\ldots,n_k} = \sum_{n_i=1}^{N_i} \boldsymbol{\mathcal{T}}_{n_1,\ldots,n_k} \boldsymbol{v}_{n_i}.$$

And the mode-i product with the Kronecker product:

$$\boldsymbol{\mathcal{T}} \times_1 \boldsymbol{v_1} \times_2 \boldsymbol{v_2} \cdots \times_{k-1} \boldsymbol{v_{k-1}} = \boldsymbol{\mathcal{T}}_{(k)} (\boldsymbol{v_{k-1}} \otimes \cdots \otimes \boldsymbol{v_1}).$$

CP DECOMPOSITION

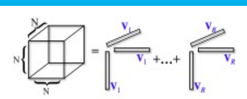


Figure 1: Example of a rank R symmetric CP decomposition of a symmetric 3-order tensor $\mathcal{T} \in \mathbb{R}^{N \times N \times N}$ such that $\mathcal{T} = \sum_{r=1}^{R} \mathbf{v}_r \circ \mathbf{v}_r \circ \mathbf{v}_r$.

The decomposition vectors, $v_{:r}$ for $r = 1, \ldots, R$, are equal in length, thus can be naturally gathered into factor matrices $\mathrm{M}_1 = [v_{11}, \ldots, v_{1R}] \in \mathbb{R}^{N_1 \times R}, \mathrm{M}_k = [v_{k1}, \ldots, v_{kR}] \in \mathbb{R}^{N_k \times R}$.
Using the factor matrices, we denote the CP decomposition of $\mathcal{T}$ as

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \boldsymbol{v_{1r}} \circ \boldsymbol{v_{2r}} \circ \cdots \circ \boldsymbol{v_{kr}} = [\![\boldsymbol{M_1}, \boldsymbol{M_2}, \ldots, \boldsymbol{M_k}]\!].$$

The k-th order tensor $\mathcal{T}$ is cubical if all its modes have the same size, i.e., $N_1 = N_2 = \cdots = N_k := N$.
A tensor $\mathcal{T}$ is symmetric if it is cubical and is invariant under permutation of its indices:

$$\boldsymbol{\mathcal{T}}_{\mathbf{n}_{\phi(1)},\ldots,\mathbf{n}_{\phi(k)}} = \boldsymbol{\mathcal{T}}_{\mathbf{n}_1,\ldots,\mathbf{n}_k} \quad \mathbf{n}_1, \ldots, \mathbf{n}_k \in [N]$$

for any permutation $\phi: [k] \to [k]$.

TENSORIZED GRAPH NEURAL NETWORK

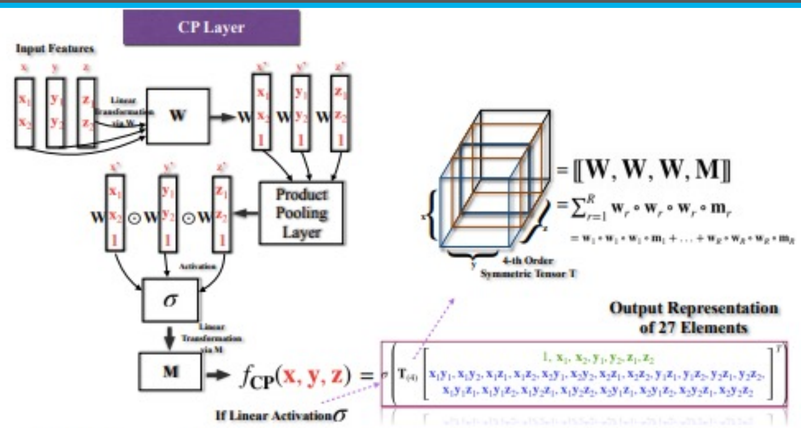


Figure 2: (Left) Sum pooling followed by a FC layer: the output takes individual components of the input into account. (Right) The CP layer can be interpreted as a combination of product pooling with linear layers (with weight matrices $\mathbf{W}$ and $\mathbf{M}$) and non-linearities. The weight matrices of a CP layer corresponds to a partially symmetric CP decomposition of a weight tensor $\mathcal{T} = [\mathbf{W}, \mathbf{W}, \mathbf{W}, \mathbf{M}]$. It shows that the output of a CP layer takes high-order multiplicative interactions of the inputs' components into account (in contrast with sum pooling that only considers 1st order terms).

Let $\mathcal{T} \in \mathbb{R}^{N \times N \times \cdots \times N \times M}$ of order k+1 be a tensor which is partially symmetric w.r.t. its first k modes. We can parameterize $\mathcal{T}$ using a rank R partially symmetric CP decomposition: $\mathcal{T} = [\![\mathrm{W}, \mathrm{W}, \ldots, \mathrm{W}, \mathrm{M}]\!], \mathrm{W} \in \mathbb{R}^{N \times R}, \mathrm{M} \in \mathbb{R}^{M \times R}$. Such a tensor naturally defines a map from $(\mathbb{R}^N)^k \to \mathbb{R}^M$ using contractions over the first k modes:

$$\boldsymbol{f}(\boldsymbol{x_1}, \cdots, \boldsymbol{x_k}) = \boldsymbol{\mathcal{T}} \times_1 \boldsymbol{x_1} \cdots \times_{k-1} \boldsymbol{x_{k-1}} = [\![\mathrm{W}, \mathrm{W}, \ldots, \mathrm{W}, \mathrm{M}]\!] \times_1 \boldsymbol{x_1} \cdots \times_{k-1} \boldsymbol{x_{k-1}}.$$

This map satisfies two very important properties for GNNs: it is permutation-invariant (due to the partial symmetry of T) and its number of parameters is independent of k (due to the partially symmetric CP parameterization.

Definition 1: (CP layer) Given parameter matrices $\mathrm{M} \in \mathbb{R}^{d \times R}, \mathrm{W} \in \mathbb{R}^{(F+1) \times R}$, and activation functions σ, σ', a rank R CP layer computes the function f_{CP} defined by:

$$\boldsymbol{f_{CP}}(\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}) = \boldsymbol{\sigma'}\left(\mathrm{M}\left(\boldsymbol{\sigma}\left(\mathrm{W}^T \begin{bmatrix} \boldsymbol{x_1} \\ \mathbf{1} \end{bmatrix} \odot \cdots \odot \mathrm{W}^T \begin{bmatrix} \boldsymbol{x_k} \\ \mathbf{1} \end{bmatrix}\right)\right)\right)$$

for any $k \geq 1, x_1, \ldots, x_k \in \mathbb{R}^F$.
To avoid the numerical instability and counterbalance the bias of high-order CP terms, we use

$$\boldsymbol{f_{CP}}(\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}) = \boldsymbol{\sigma'}\left(\mathrm{M}\left(\boldsymbol{\sigma}\left(\mathrm{W}_1^T \begin{bmatrix} \boldsymbol{x_1} \\ \mathbf{1} \end{bmatrix} \odot \cdots \odot \mathrm{W}_1^T \begin{bmatrix} \boldsymbol{x_k} \\ \mathbf{1} \end{bmatrix}\right)\right)\right) + \boldsymbol{\sigma''}(\mathrm{W}_2^T \boldsymbol{x_1} + \cdots + \mathrm{W}_2^T \boldsymbol{x_k}),$$

where the first term corresponds to the CP layer and the second one to a standard sum pooling layer (with σ, σ' and σ'' being activation functions).

Theorem 1: The function computed by a CP layer is permutation-invariant. In addition, any permutation-invariant multilinear polynomial $f: (\mathbb{R}^F)^k \to \mathbb{R}^d$ can be computed by a CP layer (with a linear activation function).
The following theorem shows that CP layers can compute any permutation-invariant multilinear polynomial map. The figure illustrates that the class of functions computed by CP layer subsumes multilinear polynomials (including sum and mean aggregation functions).
Also, the CP layer is strictly more expressive than permutation invariant multilinear polynomials due to the non-linear activation functions.

Figure 3: Visualization of relations of {permutation-invariant function space} $\supseteq$ {CP function space} $\supseteq$ {permutation-invariant multilinear polynomial space} $\supseteq$ {sum and mean aggregation functions}.

NODE CLASSIFICATION

Table 1: Results of node-level tasks. **Left Table**: tGNN in comparison with GNN architectures on citation networks. **Right Table**: tGNN in comparison with GNN architectures on *OGB* datasets.

DATASET / MODEL	*Cora* Acc	*Citeseer* Acc	*Pubmed* Acc
GCN	0.8778±0.0096	0.8139±0.0123	0.8890±0.0032
GAT	0.8686±0.0042	0.6720±0.0046	0.8328±0.0012
GraphSAGE	0.8658±0.0026	0.7624±0.0030	0.8658±0.0011
H_2GCN	0.8752±0.0061	0.7997±0.0069	0.8778±0.0028
GPRGNN	0.7951±0.0036	0.6763±0.0038	0.8507±0.0009
APPNP	0.7941±0.0038	0.6859±0.0030	0.8502±0.0009
MixHop	0.6565±0.1131	0.4952±0.1335	0.8704±0.0410
tGNN	0.8808±0.0131	0.80.51±0.0192	0.9080±0.0018

DATASET / MODEL	*PRODUCTS* Acc	*ARXIV* Acc	*PROTEINS* AUC
MLP	0.6106±0.0008	0.5550±0.0023	0.7204±0.0048
Node2vec	0.7249±0.0010	0.7007±0.0013	0.6881±0.0065
GCN	0.7564±0.0021	0.7174±0.0029	0.7251±0.0035
GraphSAGE	0.7850±0.0016	0.7149±0.0027	0.7768±0.0020
DeeperGCN	0.8098±0.0020	0.7192±0.0016	0.8580±0.0017
tGNN	0.8179±0.0054	0.7538±0.0015	0.8255±0.0049

We can observe that tGNN outperforms all classic baselines on Cora, Pubmed, PRODUCTS, and ARXIV, and have slight improvements on the other datasets but underperforms GCN on Citeseer and DeeperGCN on PROTEINS. Regarding the citation networks, tGNN outperforms others on 2 out of 3 datasets. Moreover, on the OGB node datasets, even when tGNN is not ranked first, it is still very competitive (top 3 for all datasets).

GRAPH CLASSIFICATION

Table 2: Results of tGNN on graph-level tasks in comparison with GNN architectures.

DATASET		*ZINC* No edge features MAE	*CIFAR10* No edge features Acc	*MNIST* No edge features Acc	*MolHIV* No edge features AUC
Dwivedi et al. and Hu et al.	MLP	0.710±0.001	0.560±0.009	0.945±0.003	
	GCN	0.469±0.002	0.545±0.001	0.899±0.002	0.761±0.009
	GIN	0.408±0.008	0.533±0.037	0.939±0.013	0.756±0.014
	DiffPool	0.466±0.006	0.579±0.005	0.950±0.004	
	GAT	0.463±0.002	0.655±0.003	0.956±0.001	
	MoNet	0.407±0.007	0.534±0.004	0.904±0.005	
	GatedGCN	0.422±0.006	0.692±0.003	0.974±0.001	
Corso et al.	PNA	0.320±0.032	0.702±0.002	0.972±0.001	0.791±0.013
Le et al.	PHM-GNN				0.793±0.012
Beaini et al.	DGN	0.219±0.010	0.727±0.005		0.797±0.009
Ours	tGNN	0.301±0.008	0.684±0.006	0.965±0.002	0.799±0.016

We present tGNN performance on graph property prediction tasks. tGNN achieves state-of-the-art results on MolHIV and has slight improvements on the other three Benchmarking graph datasets.

EFFICIENCY

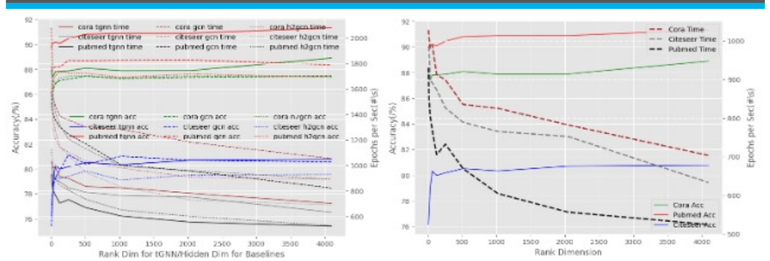


Figure 4: Results of node classification with increasing rank dimension on three citation datasets. **Left Figure**: Left axis shows accuracy, right axis shows #training epochs second, and horizontal axis indicates rank dim of tGNN or hidden dim of baselines. **Right Figure**: Left axis shows accuracy, right axis shows #training epochs second, and horizontal axis indicates rank dim of tGNN.

We can see that the model performance can be improved with higher ranks (i.e., Tensor T is more accurately computed as the rank R gets larger), but training time is also increased, thus it is a trade-off between classification accuracy and computation efficiency. In comparison with baselines, tGNN still has marginal improvements with higher ranks while the baselines stop improving with larger hidden dimensions.

When Do Graph Neural Networks Help with Node Classification?

-Investigating the Impact of Homophily Principle on Node Distinguishability

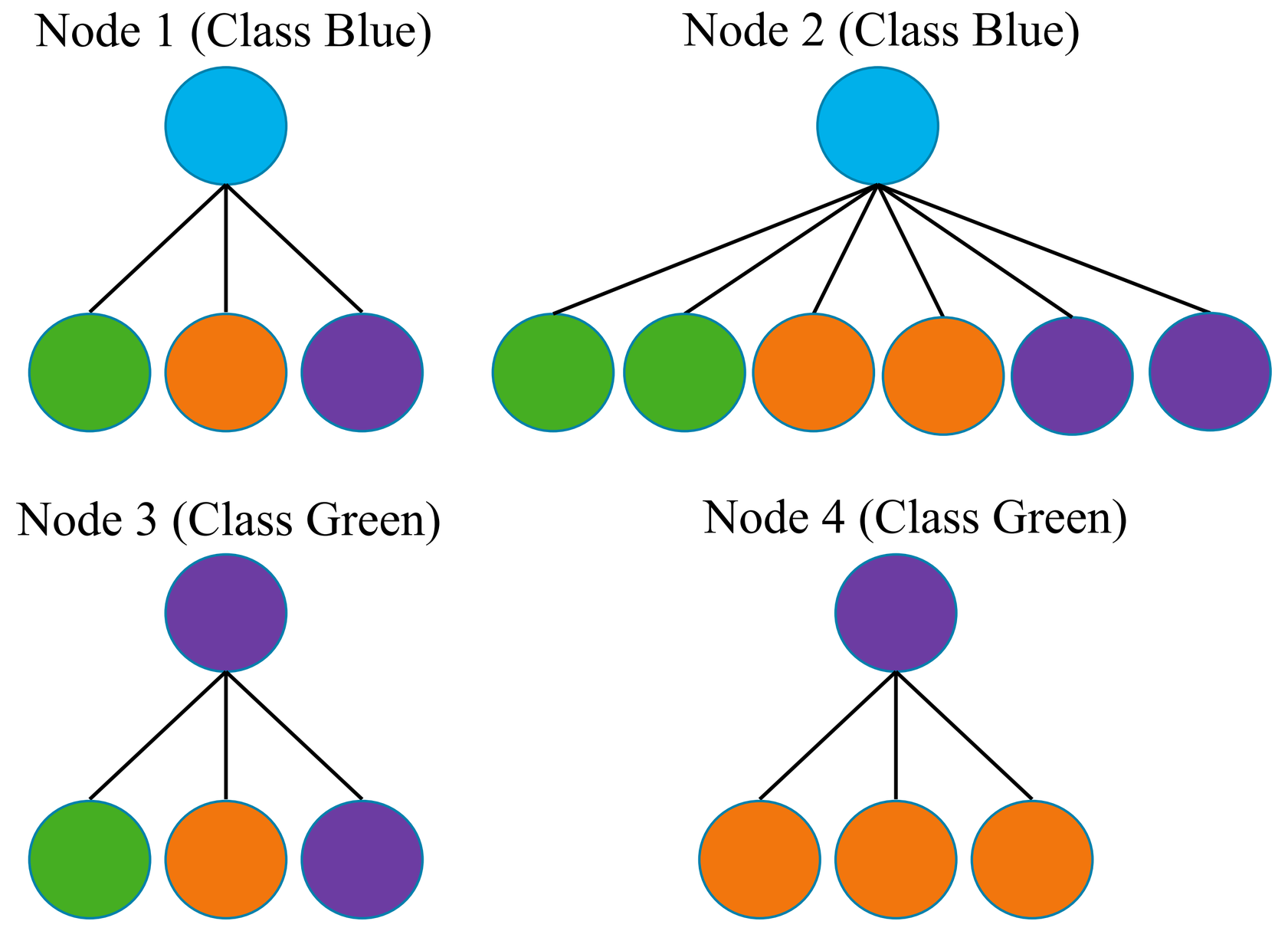
Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, Doina Precup

Introduction

In this paper, we point out the deficiency of the current understanding of homophily on Node Distinguishability (ND). To study ND deeply, we propose Contextual Stochastic Block Model for Homophily (CSBM-H) and define two metrics to quantify ND. With the metrics, we visualize and analyze how **graph filters, node degree distributions and class variances** influence ND, and investigate the combined effect of intra- and inter-class ND. Besides, we verified that, in real-work tasks, the superiority of GNNs is indeed closely related to both intra- and inter-class ND regardless of homophily levels. Grounded in this observation, we propose a new hypothesis-testing based performance metric beyond homophily, which is non-linear, feature-based and can provide statistical threshold value for GNNs' the superiority. Experiments indicate that it is significantly more effective than the existing homophily metrics on revealing the advantage and disadvantage of GNNs on both synthetic and benchmark real-world datasets.

Background & Motivation

The success of GNNs is believed to be rooted in the homophily assumption, which states that connected nodes tend to have similar attributes. On the other hand, the lack of homophily, i.e., heterophily, is considered as the main cause of the inferiority of GNNs on heterophilic graphs. Recently, both empirical and theoretical studies indicate that **the relationship between homophily and GNN performance is more complicated than "homophily wins, heterophily loses".**



Current understanding: The authors in [1] stated that, as long as nodes within the same class share similar neighborhood patterns, their embeddings will be similar after aggregation. For example, nodes {1,2} are from class blue and both have the same heterophilic neighborhood patterns. As a result, their aggregated features will still be similar, and they can be classified into the same class.

Problem: The above statement is partially true because it only discusses intra-class ND but forgets inter-class ND. For example, node 3 is from class green and has the same neighborhood pattern as {1,2}, which means node 3 and nodes {1,2} will become indistinguishable after aggregation.

Takeaway: This highlights the necessity for careful consideration of both intra- and inter-class ND when evaluating the impact of homophily on the performance of GNNs. An ideal case would be node {1,2,4}, where we have smaller intra-class "distance" than inter-class "distance". We will formulate it.

Quantify Node Distinguishability

CSBM-H The features of 2 classes of nodes, $i \in \mathcal{C}_0$ and $j \in \mathcal{C}_1$, are generated independently, with $\boldsymbol{x_i} \sim N(\boldsymbol{\mu_0}, \sigma_0^2 \mathrm{I})$ and $\boldsymbol{x_j} \sim N(\boldsymbol{\mu_1}, \sigma_1^2 \mathrm{I})$, where $\boldsymbol{\mu_0}, \boldsymbol{\mu_1} \in \mathbb{R}^{F_h}$; d_0, d_1 are node degree for i, j. For $i \in \mathcal{C}_0$, its neighbors are generated by independently sampling from $h \cdot d_0$ intra-class nodes and $(1-h) \cdot d_0$ inter-class nodes. The neighbors of $j \in \mathcal{C}_1$ are generated in the same way. The centers $(\tilde{\mu}_0, \tilde{\mu}_1, \tilde{\mu}_0^{HP}, \tilde{\mu}_1^{HP})$ and variances $(\tilde{\sigma}_0, \tilde{\sigma}_1, (\tilde{\sigma}_0^{HP})^2, (\tilde{\sigma}_1^{HP})^2)$ of low- and high-pass filtered features can be computed accordingly (See our paper).

Theorem 1 (Optimal Bayes Classifier $\mathrm{CL_{Bayes}}$ for CSBM-H) Suppose the prior distribution for $\boldsymbol{x}$ is $\mathbb{P}(\boldsymbol{x} \in \mathcal{C}_0) = \mathbb{P}(\boldsymbol{x} \in \mathcal{C}_1) = 1/2$, then we have

$$\mathrm{CL_{Bayes}}(\boldsymbol{x}) = \begin{cases} 1, \eta(\boldsymbol{x}) \geq 0.5 \\ 0, \eta(\boldsymbol{x}) < 0.5 \end{cases}, \eta(\boldsymbol{x}) = \mathbb{P}(z=1|\boldsymbol{x}) = \frac{1}{1+\exp(Q(\boldsymbol{x}))}$$

where $Q(\boldsymbol{x}) = a\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} + \mathrm{c}$, z_i is the label of node i. See the calculation of $a, \boldsymbol{b}, \mathrm{c}$ in the paper.

Bayes Error Rate (BE) for $\mathrm{CL_{Bayes}}$ can be written as

$$\mathbf{BE} = \mathbb{P}(\boldsymbol{x} \in \mathcal{C}_0)\left(1 - \mathbb{P}(\mathrm{CL_{Bayes}}(\boldsymbol{x}) = 0 | \boldsymbol{x} \in \mathcal{C}_0)\right) + \mathbb{P}(\boldsymbol{x} \in \mathcal{C}_1)\left(1 - \mathbb{P}(\mathrm{CL_{Bayes}}(\boldsymbol{x}) = 1 | \boldsymbol{x} \in \mathcal{C}_1)\right)$$

Probabilistic Bayes Error (PBE) Since $Q(\boldsymbol{x})$ in Theorem 1 follows generalized chi-square distribution, BE can be estimated by PBE as follows

$$\frac{\mathrm{CDF}_{\tilde{\chi}(\omega_0, F_h, \lambda_0)}(-\xi) + (1 - \mathrm{CDF}_{\tilde{\chi}(\omega_1, F_h, \lambda_1)}(-\xi))}{2}$$

See the parameters and the calculation in our paper. PBE can be numerically calculated to show the relation between h and ND precisely. However, it doesn't have an analytic expression, which makes it less explainable.

Generalized Jeffreys Divergence D_{GJ} The KL-divergence based statistical measure offers us a tool to define an explainable ND measure as follows

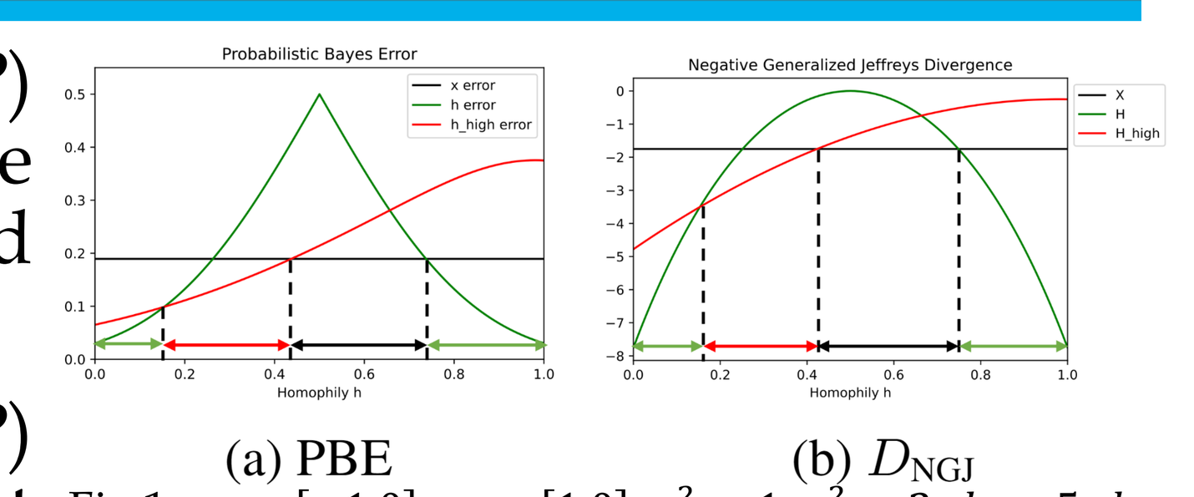
$$D_{GJ}(P, Q) = \mathbb{P}(\boldsymbol{x} \in P)\mathbb{E}_{x \sim P}\left[ln \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}\right] + \mathbb{P}(\boldsymbol{x} \in Q)\mathbb{E}_{x \sim P}\left[ln \frac{Q(\boldsymbol{x})}{P(\boldsymbol{x})}\right]$$

The negative generalized Jeffreys divergence CSBM-H can be computed by

$$D_{NGJ}(CSBM\text{-}H) = -d_X^2\left(\frac{1}{4\sigma_1^2} + \frac{1}{4\sigma_0^2}\right) - \frac{F_h}{4}\left[\rho^2 + \frac{1}{\rho^2} - 2\right]$$

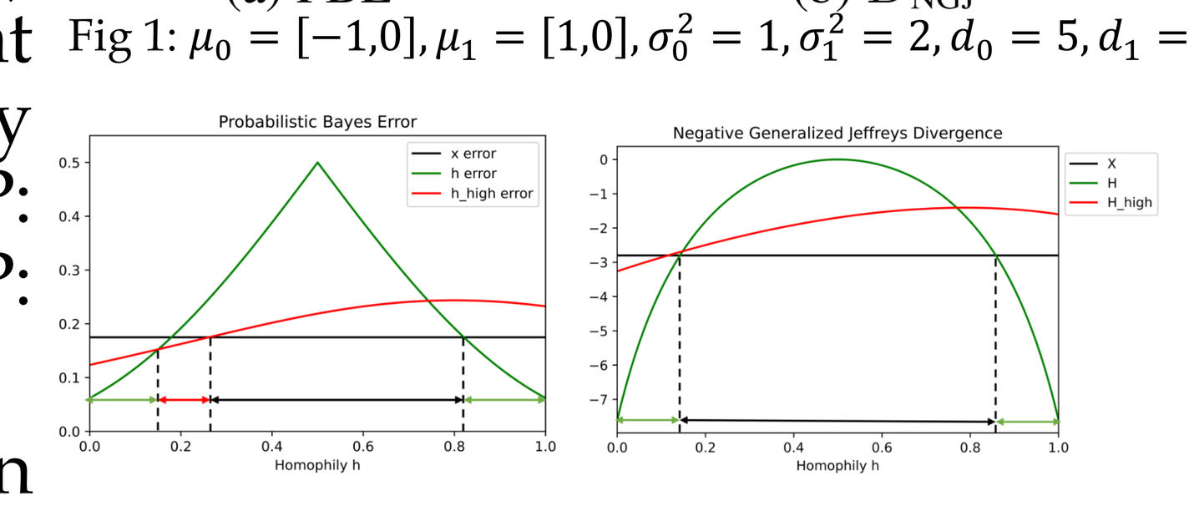
where $d_X^2 = (\mu_0 - \mu_1)^{\mathrm{T}}(\mu_0 - \mu_1), \rho = \sigma_0/\sigma_1$. PBE and $\boldsymbol{D_{GJ}}$ can quantify ND.

Visualization & Analysis



(a) PBE (b) D_{NGJ}

Fig 1: $\mu_0 = [-1,0], \mu_1 = [1,0], \sigma_0^2 = 1, \sigma_1^2 = 2, d_0 = 5, d_1 = 5$



(a) PBE (b) D_{NGJ}

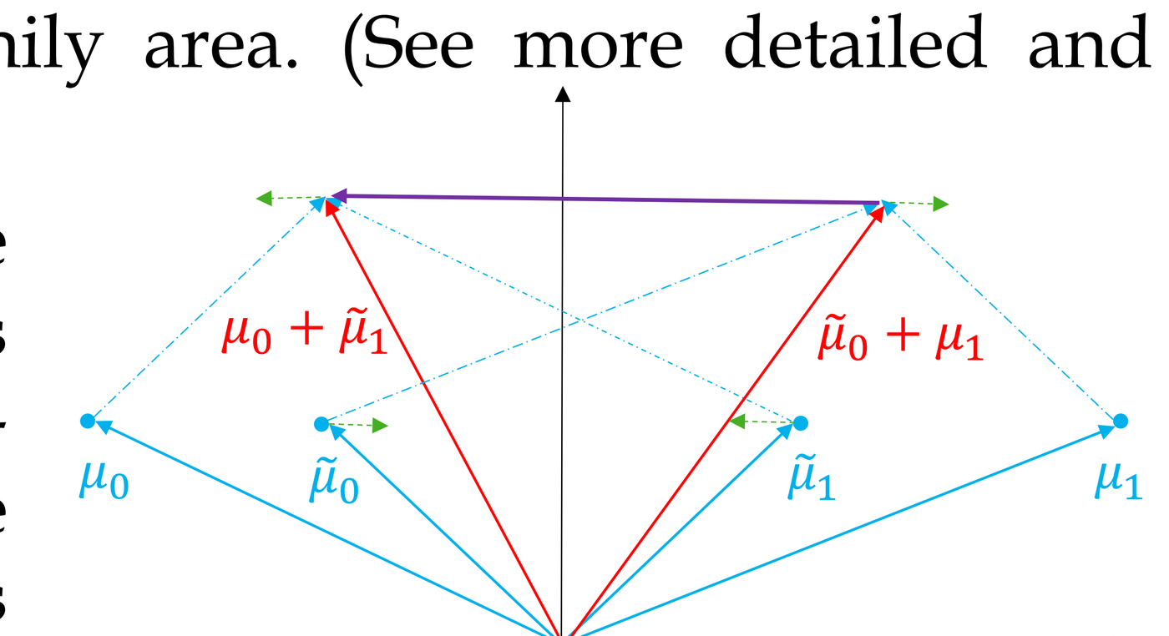
Fig 2: $\mu_0 = [-1,0], \mu_1 = [1,0], \sigma_0^2 = 1, \sigma_1^2 = 5, d_0 = 5, d_1 = 5$

Mid-homophily pitfall: For low-pass (LP) filtered features, the most indistinguishable area is the medium homophily level, instead of the extreme low homophily levels.

3 regimes: low-pass (LP), high-pass (HP) and full-pass (FP) filters dominate different homophily areas. LP: very low and very high homophily intervals (two ends); HP: low to medium homophily levels; FP: medium to high homophily areas.

HP filter: high-pass filter works better in heterophily areas than in homophily areas

Ablation Study: From the comparison of Fig 1 and Fig 2, it is observed that as σ_1^2 increases, the PBE and D_{NGJ} of the three curves all go up, which means the node embeddings become less distinguishable under HP, LP and FP filters. The significant shrinkage of the HP regimes and the expansion of the FP regime indicates that the original features are more robust to imbalanced variances, especially in the low homophily area. (See more detailed and comprehensive analysis in our paper)



More General Theoretical Analysis we investigate how significant the intra-class embedding distance is smaller than inter-class embedding distance under more general conditions. The theorem verifies that ND is related to the combined effect of inter- and intra-class ND. Besides, we found that the impact of HP filter depends on the relative center distance (See the above Fig), which is novel.

Empirical Study of ND

We conduct hypothesis tests with GCN vs. MLP-2 and SGC-1 vs. MLP-1 to verify whether "intra-class embedding distance is smaller than the inter-class embedding distance" strongly relates to the superiority of Graph-aware models to their coupled Graph-agnostic models. We compute the proportion of nodes whose intra-class node distance is significantly smaller than inter-class node distance (Prop) and use it to quantify ND. Then, we test

$\mathrm{H_0}$: Prop(G−aware)≥Prop(G−agnostic); $\mathrm{H_1}$: Prop(G−aware)<Prop(G−agnostic)

We compare the results 100 times on 9 benchmark datasets. With the 100 samples, we conduct T-test to obtain the p-values. It is observed that, in most cases (except for GCN vs. MLP-2 on PubMed), when $\mathrm{H_1}$ significantly holds, G-aware models will underperform the coupled G-agnostic models and vice versa, no matter the homophily levels. This confirms our claim.

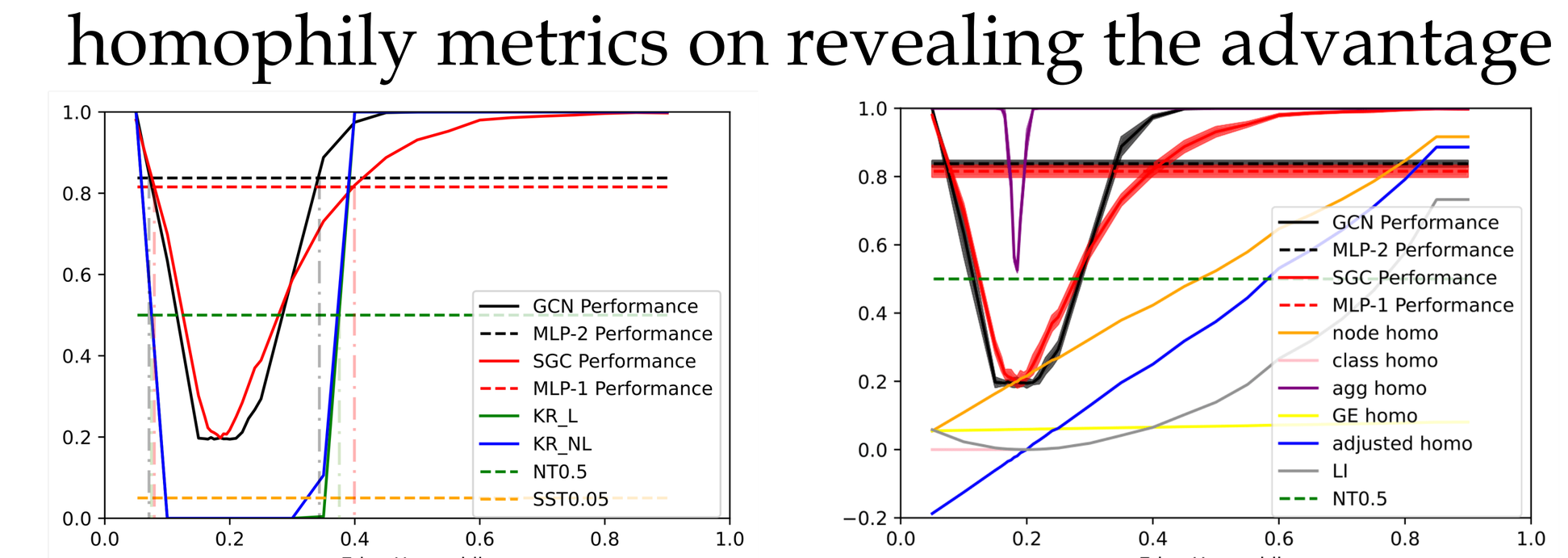
Moreover, the p-value can provide a statistical threshold, which is a property that is not present in existing homophily metrics. However, it is required to train and fine-tune the models to obtain the p-values, which make it less practical because of computational costs.

		Cornell	Wisconsin	Texas	Film	Chameleon	Squirrel	Cora	CiteSeer	PubMed
Baseline Homophily Metrics	$\mathrm{H_{edge}}$	0.5669	0.4480	0.4106	0.3750	0.2795	0.2416	0.8100	0.7362	0.8024
	$\mathrm{H_{node}}$	0.3855	0.1498	0.0968	0.2210	0.2470	0.2156	0.8252	0.7175	0.7924
	$\mathrm{H_{class}}$	0.0468	0.0941	0.0013	0.0110	0.0620	0.0254	0.7657	0.6270	0.6641
	$\mathrm{H_{agg}}$	0.8032	0.7768	0.694	0.6822	0.61	0.3566	0.9904	0.9826	0.9432
	$\mathrm{H_{GE}}$	0.31	0.34	0.35	0.16	0.0152	0.0157	0.17	0.19	0.27
	$\mathrm{H_{adj}}$	0.1889	0.0826	0.0258	0.1272	0.0663	0.0196	0.8178	0.7588	0.7431
	LI	0.0169	0.1311	0.1923	0.0002	0.048	0.0015	0.5904	0.4508	0.4093
Classifier-based Performance Metrics	$\mathrm{KR_{NNGP}}$	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
	GNB	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
SGC v.s. MLP-1	p-value	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	0.00
	ACC SGC	70.98 ± 8.39	70.38 ± 2.85	83.28 ± 5.43	25.26 ± 1.18	64.86 ± 1.81	47.62 ± 1.27	85.12 ± 1.64	79.66 ± 0.75	85.5 ± 0.76
	ACC MLP-1	93.77 ± 3.34	93.87 ± 3.33	93.77 ± 3.34	34.53 ± 1.48	45.01 ± 1.58	29.17 ± 1.46	74.3 ± 1.27	75.51 ± 1.35	86.23 ± 0.54
	Diff Acc	-22.79	-23.49	-10.49	-9.27	19.85	18.45	10.82	4.15	-0.73
GCN v.s. MLP-2	p-value	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	0.00
	ACC GCN	82.46 ± 3.11	75.5 ± 2.92	83.11 ± 3.2	35.51 ± 0.99	64.18 ± 2.62	44.76 ± 1.39	87.78 ± 0.96	81.39 ± 1.23	88.9 ± 0.32
	ACC MLP-2	91.30 ± 0.70	93.87 ± 3.33	92.26 ± 0.71	38.58 ± 0.25	46.72 ± 0.46	31.28 ± 0.27	76.44 ± 0.30	76.25 ± 0.28	86.43 ± 0.13
	Diff Acc	-8.84	-18.37	-9.15	-3.07	17.46	13.48	11.34	5.14	2.47

Table 1: P-values, homophily values and classifier-based performance metrics on 9 real-world benchmark datasets. Cells marked by grey are incorrect results for both SGC v.s. MLP-1 and GCN v.s. MLP-2 and cells marked by blue are incorrect for 1 of the 2 tests. We use 0.5 as the threshold value of the homophily metrics.

Beyond Homophily: A New Performance Metrics

We use the simple classifiers (Gaussian Naïve Bayes (GNB) and Kernel Regression (KR) with Neural Network Gaussian Process (NNGP)) that doesn't require iterative training for hypothesis testing. From Table 1 and the figures below on synthetic graphs, we can see that the new classifier-based performance metrics are much more effective than the existing homophily metrics on revealing the advantage and disadvantage of GNNs.



Performance Metrics	Linear or Non-linear	Feature Dependency	Sparse Labels	Statistical Threshold
$\mathrm{H_{node}}$	linear	✗	✗	✗
$\mathrm{H_{edge}}$	linear	✗	✗	✗
$\mathrm{H_{class}}$	linear	✗	✗	✗
$\mathrm{H_{agg}}$	linear	✗	✓	✗
$\mathrm{H_{GE}}$	linear	✓	✓	✗
$\mathrm{H_{adj}}$	linear	✗	✗	✗
LI	linear	✗	✗	✗
Classifier	both	✓	✓	✓

References

[1] Is Homophily a Necessity for Graph Neural Networks?. In *International Conference on Learning Representations*.

[2] Revisiting heterophily for graph neural networks. *Advances in neural information processing systems, 35*, 1362-1375.

[3] When Do We Need GNN for Node Classification?. *arXiv preprint arXiv:2210.16979*.

Paper

When Do We Need Graph Neural Networks for Node Classification?

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Doina Precup

Abstract

Graph Neural Networks (GNNs) extend basic Neural Networks (NNs) by additionally making use of graph structure based on the relational inductive bias (edge bias), rather than treating the nodes as collections of independent and identically distributed i.i.d. samples. Though GNNs are believed to outperform basic NNs in real-world tasks, it is found that in some cases, GNNs have little performance gain or even underperform graph-agnostic NNs. To identify these cases, based on graph signal processing and statistical hypothesis testing, we propose two measures which analyze the cases in which the edge bias in features and labels does not provide advantages. Based on the measures, a threshold value can be given to predict the potential performance advantages of graph-aware models over graph-agnostic models.

Introduction

Table 1. Accuracy (%) Comparison of Baseline GNNs and MLP

Datasets\Models	MLP Acc	GCN Acc	GAT Acc	GraphSAGE Acc	Baseline Average	Diff(MLP, Baseline)	Edge Homophily
Cornell	85.14	60.81	59.19	82.97	67.66	**17.48**	0.3
Wisconsin	87.25	63.73	60.78	87.84	70.78	**16.47**	0.21
Texas	84.59	61.62	59.73	82.43	67.93	**16.66**	0.11
Film	36.08	30.98	29.71	35.28	31.99	**4.09**	0.22
Chameleon	46.21	61.34	61.95	47.32	56.87	-10.66	**0.23**
Squirrel	29.39	41.86	43.88	30.16	38.63	-9.24	**0.22**
Cora	74.81	87.32	88.07	85.98	87.12	-12.31	0.81
Citeseer	73.45	76.70	76.42	77.07	76.73	-3.28	0.74
Pubmed	87.86	88.24	87.81	88.59	88.21	**-0.35**	**0.80**
DBLP	77.39	85.87	85.89	81.19	84.32	-6.93	0.81
Coauthor CS	93.72	93.91	93.41	94.38	93.90	**-0.18**	**0.81**
Coauthor Phy	95.77	96.84	96.32	OOM	96.58	**-0.81**	**0.93**
AMZ Comp	83.89	87.03	89.74	83.70	86.82	-2.93	0.78
AMZ Photo	90.87	93.61	94.12	87.97	91.90	-1.03	**0.83**

Growing evidence shows that GNNs do not always gain advantages over traditional NNs on relational data, e.g., as shown in table 1, MLP outperform baseline GNNs on *Cornell, Wisconsin, Texas* and *Film* and perform almost the same as baseline GNNs on *PubMed, Coauthor CS* and *Coauthor Phy*. This makes us wonder when do we really need GNNs?

Most existing graph filters can be viewed as operators that aggregate node information from its direct neighbors and they are considered as low-pass (LP) filters mainly capturing the low-frequency components of the input, i.e., the locally smooth features across the whole graph. The use of LP graph filters relies on the assumption that nodes tend to share attributes with their neighbors, a tendency called **homophily** that is widely exploited in node classification tasks. GNNs that are built on the homophily assumption learn to assign similar labels to nodes that are closely connected, which corresponds to an assumption of intrinsic smoothness on latent label distribution. We call this kind of relational inductive bias the edge bias. We believe it is a key factor leading to GNNs superiority over NNs.

However, the existing homophily metrics are not appropriate to display the edge bias, e.g. as shown in table 1, MLP does not necessarily outperform baseline GNNs on some low homophily datasets (*Chameleon and Squirrel*) and does not significantly underperform baseline GNNs on some high homophily datasets (*PubMed, Coauthor CS, Coauthor Phy and AMZ Photo*). Thus, a metric that is able to indicate whether or not the graph-aware models can outperform graph-agnostic models is needed.

Measuring the Effect of Edge Bias

Normalized Total Variation (NTV): it measures the distance between node features and its aggregated neighborhood features

$$NTV(X) = \frac{\|X - \hat{A}X\|_2^2}{\|X\|_2^2}$$

Normalized Smoothness Value (NSV): it measures how different is the average pairwise attribute distance of connected nodes from that of unconnected nodes. First, we have he total pairwise attribute distance of connected nodes (Dirichlet energy)

$$E_D^{\mathcal{G}}(X) = Trace(X^\top LX) = \sum_{i \leftrightarrow j} \|X_{i,:} - X_{j,:}\|_2^2$$

The total pairwise distance of unconnected nodes can be derived from the Laplacian $L^C = D^C - A^C = NI - \mathbf{1}^\top\mathbf{1} - L$ of the complementary graph $\mathcal{G}^C$

$$E_D^{\mathcal{G}^C}(X) = Trace(X^\top L^C X) = \sum_{i \nleftrightarrow j} \|X_{i,:} - X_{j,:}\|_2^2$$

Then, the average pairwise distance of connected and unconnected nodes can be calculated by normalizing $E_D^{\mathcal{G}}(X)$ and $E_D^{\mathcal{G}^C}(X)$

$$E_N^{\mathcal{G}}(X) = \frac{E_D^{\mathcal{G}}(X)}{2|\mathcal{E}|}, E_N^{\mathcal{G}^C}(X) = \frac{E_D^{\mathcal{G}^C}(X)}{N(N-1) - 2|\mathcal{E}|}$$

Then, the Normalized Smoothness Value (NSV) is defined as

$$NSV(X) = \frac{E_N^{\mathcal{G}}(X)}{E_N^{\mathcal{G}}(X) + E_N^{\mathcal{G}^C}(X)}$$

Analysis $0 \leq NSV(X) \leq 1$ and it can help to interpret the edge bias: (1) For labels Z, $NSV(Z) \geq 0.5$ means that the proportion of connected nodes that share different labels is larger than that of unconnected nodes, which implies that edge bias is harmful for the prediction of Z and the homophily assumption is invalid; (2) For features X, $NSV(X) \geq 0.5$ means that the average pairwise feature distance of connected nodes is greater than that of unconnected nodes, which suggests that the feature is non-smooth. On the contrary, small $NSV(Z)$ and $NSV(X)$ indicates that the homophily assumption holds and the edge bias is potentially beneficial.

The above analysis raises another question: how much does NSV deviating from 0.5 or what is the exact NSV to indicate the edge bias is statistically significantly beneficial or harmful?

Hypothesis Testing for Edge Bias

Consider the following distributions of labels and features. For labels Z,

$P_1 = \mathbb{P}(Z_{i,:} \neq Z_{j,:} | e_{ij} \in \mathcal{E}) =$ The proportion of connected nodes that share different labels;
$P_2 = \mathbb{P}(Z_{i,:} \neq Z_{j,:} | e_{ij} \notin \mathcal{E}) =$ The proportion of unconnected nodes that share different labels.

For features X,

$D_1 = \|X_{i,:} - X_{j,:}\|_2^2 \mid e_{ij} \in \mathcal{E} =$ Distribution of pairwise feature distance of connected nodes;
$D_2 = \|X_{i,:} - X_{j,:}\|_2^2 \mid e_{ij} \notin \mathcal{E} =$ Distribution of pairwise feature distance of unconnected nodes.

Suppose P_1, P_2, D_1, D_2 follow

$P_1 \sim \text{Binom}(n_1, p_1), P_2 \sim \text{Binom}(n_2, p_2);\ D_1 \sim N(d_1, \sigma_1^2),\ D_2 \sim N(d_2, \sigma_2^2)$

Consider the hypotheses for labels

$H_0^L: p_1 = p_2;\ H_1^L: p_1 \neq p_2;\ H_2^L: p_1 \geq p_2;\ H_3^L: p_1 \leq p_2$

and hypotheses for features

$H_0^F: d_1 = d_2;\ H_1^F: d_1 \neq d_2;\ H_2^F: d_1 \geq d_2;\ H_3^F: d_1 \leq d\_2$

To conduct the hypothesis tests, we use Welch's t-test for features and χ^2 test for labels. We can see $E_N^{\mathcal{G}}(Z)$ and $E_N^{\mathcal{G}^C}(Z)$ are sample estimation of the p_1, p_2 for label Z; $E_N^{\mathcal{G}}(X)$ and and $E_N^{\mathcal{G}^C}(X)$ are sample estimation of d_1, d_2 for X. Thus, the p-values of hypothesis tests can suggest if NSV statistically significantly deviates from 0.5.

Results and Analysis

Analysis For feature X:

- p-value (H_0^F *vs.* H_1^F): > 0.05, H_0^F holds, feature is non-smooth; ≤ 0.05, TBD.
- p-value(H_0^F *vs.* H_2^F): ≤ 0.05, feature is significantly non-smooth.
- p-value(H_0^F *vs.* H_3^F): ≤ 0.05, feature is statistically significantly smooth.

For label Z:

- p-value(H_0^L *vs.* H_1^L): > 0.05, H_0^L holds, label is non-smooth; ≤ 0.05, TBD.
- p-value(H_0^L *vs.* H_2^L): ≤ 0.05, label is statistically significantly non-smooth.
- p-value(H_0^L *vs.* H_3^L): ≤ 0.05, label is statistically significantly smooth.

Table 2. Statistics of Datasets and the Performance Differences

Datasets\Measures	Features					Labels					Baseline Average - MLP
	NTV	NSV	H_0^F vs H_1^F	H_0^F vs H_2^F	H_0^F vs H_3^F	NTV	NSV	H_0^L vs H_1^L	H_0^L vs H_2^L	H_0^L vs H_3^L	
Cornell	0.33	0.48	0.00	1.00	**0.00**	0.33	0.53	0.0003	**0.00**	1.00	-17.48
Texas	0.33	0.48	0.00	1.00	**0.00**	0.42	0.60	0.00	**0.00**	1.00	-16.66
Wisconsin	0.38	0.51	**0.72**	0.36	0.64	0.40	0.55	0.00	**0.00**	1.00	-16.47
Film	0.39	0.50	**0.19**	0.90	0.10	0.37	0.50	**0.05**	0.97	0.03	-4.09
Coauthor CS	0.36	0.36	0.00	1.00	**0.00**	0.19	0.18	0.00	1.00	**0.00**	0.18
Pubmed	0.33	0.44	0.00	1.00	**0.00**	0.25	0.24	0.00	1.00	**0.00**	0.35
Coauthor Phy	0.35	0.36	0.00	1.00	**0.00**	0.16	0.09	0.00	1.00	**0.00**	0.81
AMZ Photo	0.41	0.39	0.00	1.00	**0.00**	0.23	0.17	0.00	1.00	**0.00**	1.03
AMZ Comp	0.41	0.38	0.00	1.00	**0.00**	0.25	0.22	0.00	1.00	**0.00**	2.93
Citeseer	0.35	0.45	0.00	1.00	**0.00**	0.22	0.24	0.00	1.00	**0.00**	3.28
DBLP	0.37	0.46	0.00	1.00	**0.00**	0.21	0.20	0.00	1.00	**0.00**	6.93
Squirrel	0.47	0.54	0.00	**0.00**	1.00	0.44	0.49	0.00	1.00	**0.00**	9.24
Chameleon	0.45	0.45	0.00	1.00	**0.00**	0.45	0.49	0.00	1.00	**0.00**	10.66
Cora	0.38	0.47	0.00	1.00	**0.00**	0.20	0.19	0.00	1.00	**0.00**	12.31

Results of hypothesis testing are summarized in Table 2. We can see that for the datasets where baseline GNNs underperform MLP, *Cornell, Texas* and *Wisconsin* has statistically significantly non-smooth labels and *Film* has non-smooth labels (p=0.05). In these datasets, the edge bias will provide harmful information no matter the features are smooth or not. For other datasets, they have statistically significantly smooth labels, which means the edge bias can statistically provide benefits to the baseline GNNs and lead them to have superiority performance over MLP.

Furthermore, NSV makes a comparison to see if the current edges in $\mathcal{E}$ have significantly less probability of indicating different pairwise labels than the rest edges. For most cases in Table 2, if the NSV of labels are larger than or equal to those of features, GNN will underperform MLP because the smoothing effect of aggregation operation has less benefit on labels, and vice versa.