AI Drive

大规模图机器学习的可扩展性

演讲人：张文涛
学校/实验室：北京大学计算机学院

AI Drive

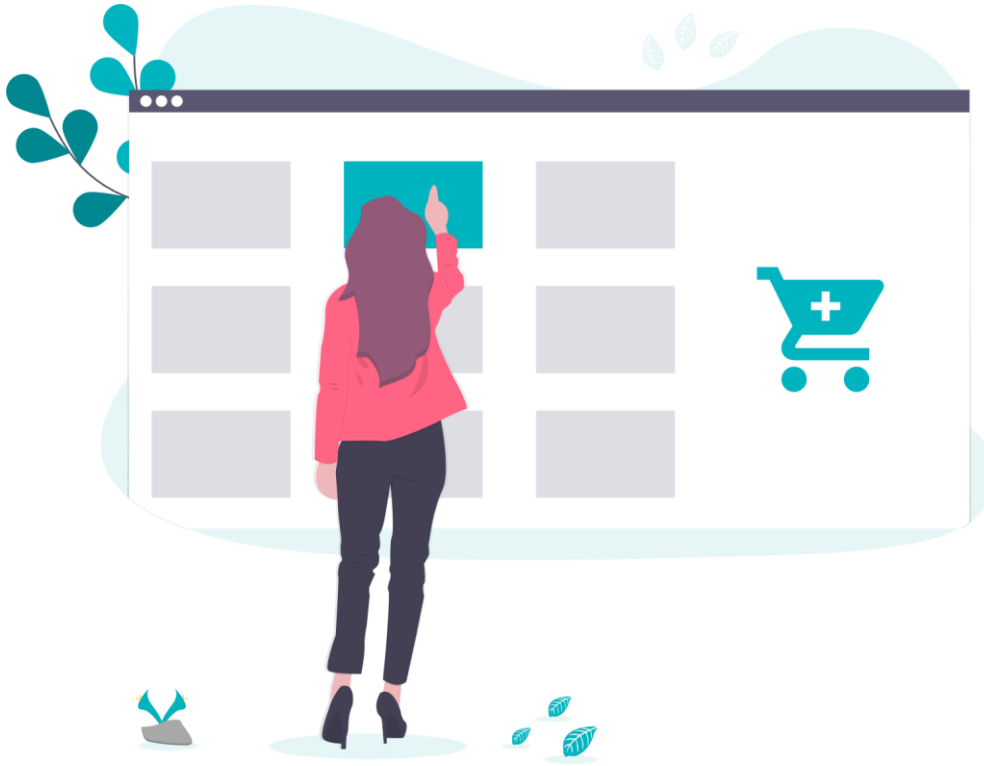
CONTENTS

AI Drive

/01

Motivation篇

GNN的应用和局限



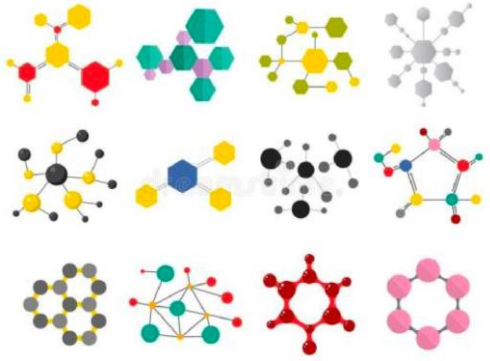
Applications of GNN

AI Drive

Many real-world data are graphs.

Social Network

Knowledge Graph



Drugs and New materials

- Graph Neural Network (GNN) has achieved great success in many graph-based applications.

- Compared with CNN, GNN has the ability to enhance its feature (or representation in each layer) with the adjacent nodes, and thus improve the model performance.

Low Scalability of GNN

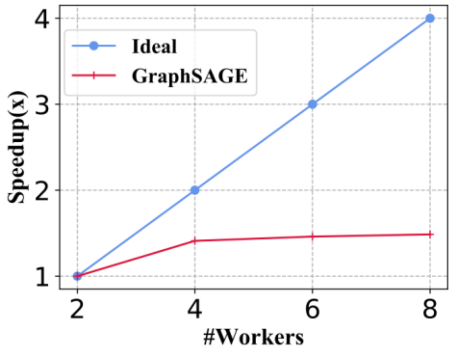
AI Drive

1. High Memory/Time Cost in Single Machine

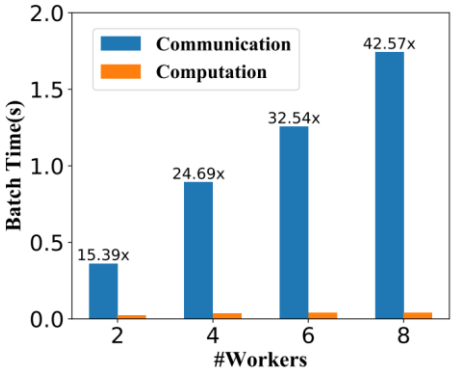
- Hard to load the feature and adjacent matrix due to limited GPU memory.
- The matrix multiplication is time-consuming.

2. High Communication Cost in Distributed Setting

- k-layer GNNs have to pull and aggregate the graph embedding of the k-hop neighbors of nodes in each batch training.
- The speedup is unsatisfactory due to the high communication cost.



Speedup

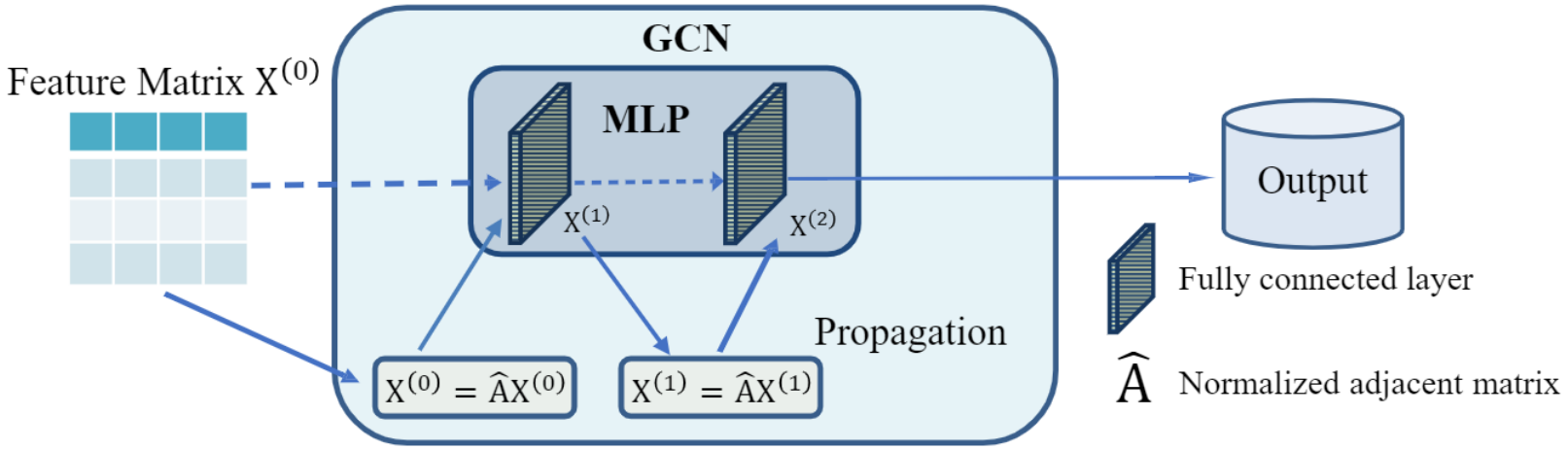


Bottleneck

Low Flexibility of GNN

AI Drive

1. Restricting $D_p = D_t$

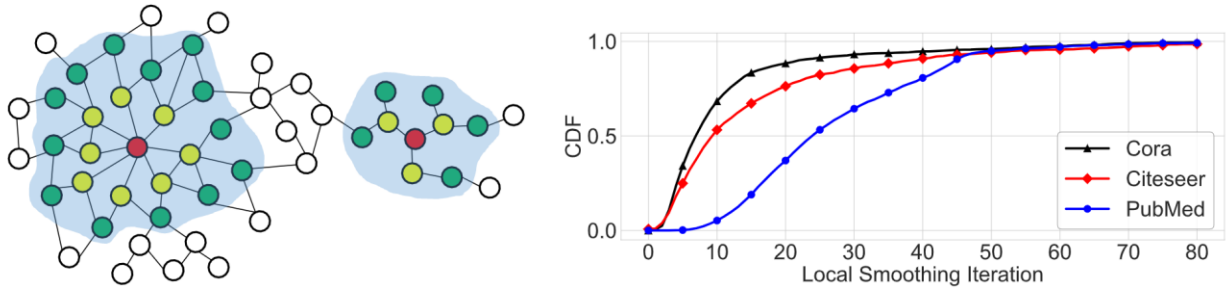


- GNN has two depth, the propagation depth D_p and transformation depth D_t
- We need large D_p for sparse graph and large D_t for large graph

Low Flexibility of GNN

AI Drive

2. Inconsistent smoothing speed



(a) Two nodes with different local structures (b) The CDF of LSI in different graphs

Figure 1: (Left) The local graph structures for two nodes in different regions; the node in dense region has larger smoothed area within two iterations of propagation. (Right) The CDF of LSI in three citation networks.

- Smoothing speed in dense region is extremely faster than the sparse region and noise is introduced if we propagate so many times.
- Different nodes require different propagation steps.

Over-Smoothing issue

AI Drive

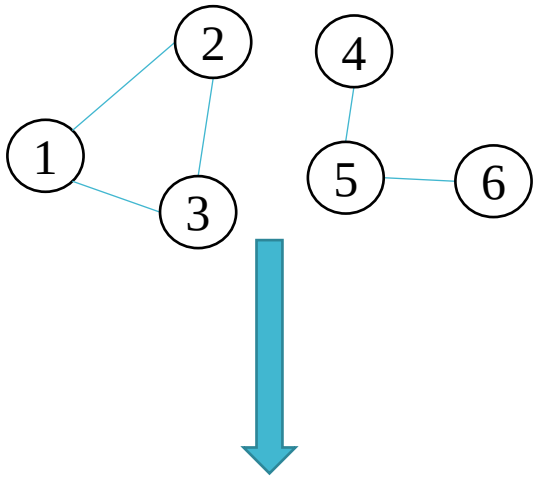
- GNN smooths the representation of each node via aggregating its own representations and the ones of its neighbors.
- Suppose $\widetilde{\mathrm{D}}$ is the diagonal node degree matrix with self loops, and $\widehat{\mathrm{A}} = \widetilde{\mathrm{D}}^{r-1}\widetilde{\mathrm{A}}\widetilde{\mathrm{D}}^{-r}$ is the normalized adjacency matrix. By continually smoothing the node feature with infinite number of propagation in SGC, the final smoothed feature is

$$\mathrm{X}^{(\infty)} = \widehat{\mathrm{A}}^{\infty}\mathrm{X}, \qquad \widehat{\mathrm{A}}^{\infty}_{i,j} = \frac{(d_i+1)^r\left(d_j+1\right)^{1-r}}{2m+n}$$

The final feature is over-smoothed and unable to capture the full graph structure information since it only relates with the node degrees of target nodes and source nodes.

An example of over-smoothing

AI Drive

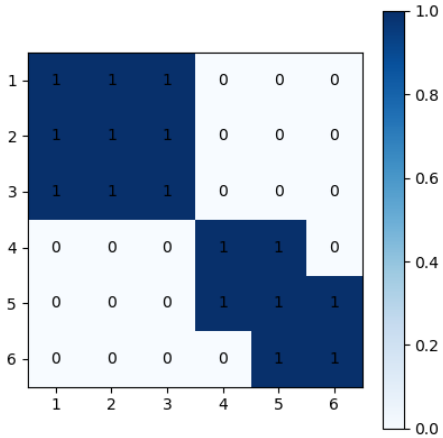


$$\lim_{k\to\infty}(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^k_{i,j} = \frac{d_j+1}{2m+n}$$

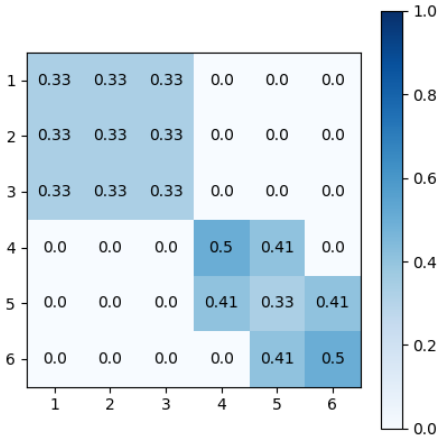
$$\lim_{k\to\infty}(\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1})^k_{i,j} = \frac{d_i+1}{2m+n}$$

$$\lim_{k\to\infty}(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2})^k_{i,j} = \frac{\sqrt{(d_i+1)(d_j+1)}}{2m+n}$$

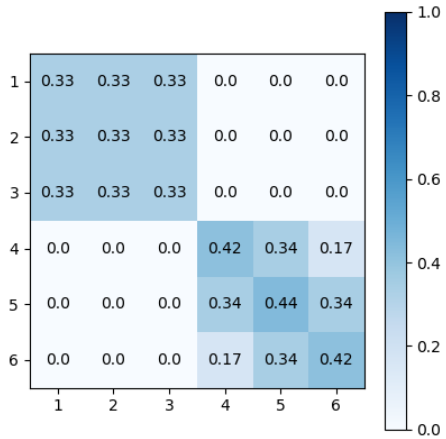
where $n = |V|$, $m = |E|$ *and* $d_i = \sum_j \mathbf{A}_{ij}$.



A

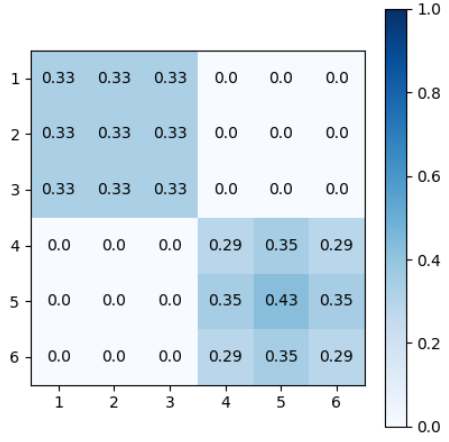


$\left(\widetilde{D}^{-1/2}A\widetilde{D}^{-1/2}\right)^1$



$\left(\widetilde{D}^{-1/2}A\widetilde{D}^{-1/2}\right)^2$

...



$\left(\widetilde{D}^{-1/2}A\widetilde{D}^{-1/2}\right)^{26}$

limitations

AI Drive

1. Over-smoothing

- Deep GNN faces the over-smoothing problem that the node representation is less discriminative.

2. Under-smoothing

- Each node in a k-layer GNN can only capture the node attribute and node dependency information in its k-hop neighbors. For sparse graphs, lots of valuable information in unlabeled nodes is ignored when we use shallow architecture.

AI Drive

/02 Related Works篇

Scalable GNNs

AI Drive

1. Sampling

- Graph-wise sampling: Cluster-GCN, GraphSAINT
- Layer-wise sampling: Fast-GCN, AS-GCN
- Node-wise sampling: GraphSAGE, VR-GCN

2. Model decoupling

- SGC
- SIGN
- GBP
- S2GC

Our method focuses on the second type since it is more efficient and scalable.

SGC(ICML 19)

AI Drive

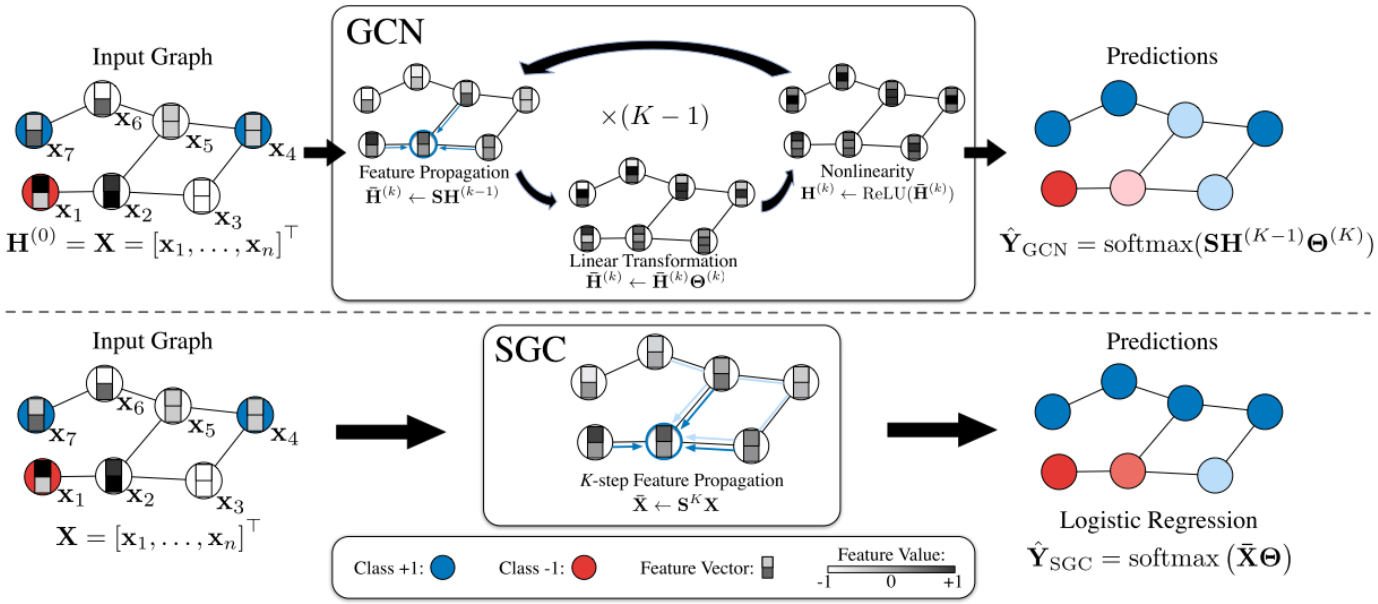


Figure 1. Schematic layout of a GCN v.s. a SGC. *Top row:* The GCN transforms the feature vectors repeatedly throughout K layers and then applies a linear classifier on the final representation. *Bottom row:* the SGC reduces the entire procedure to a simple feature propagation step followed by standard logistic regression.

- SGC firstly propagates the feature to high order and then keeping a small number of transformation.
- It faces over-smoothing issue when the propagation step is large.

SIGN

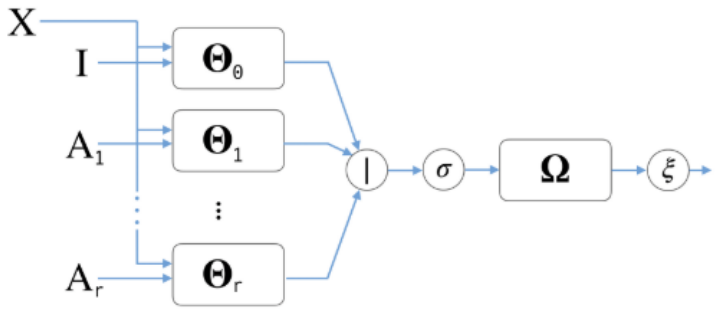


Figure 1: The *SIGN* architecture for r generic graph filtering operators. $\mathbf{\Theta}_k$ represents the k-th dense layer transforming node-wise features downstream the application of operator k, $|$ is the concatenation operation and $\mathbf{\Omega}$ refers to the dense layer used to compute final predictions.

SIGN further improve SGC by adding more diffusion operators, such as PageRank-based and triangle-based adjacency matrices.

- No flexibility: some node needs less receptive field while other nodes need more. SIGN simply concatenate the global and local representation, ignoring the personalization of each node.

DAGNN(KDD 20)

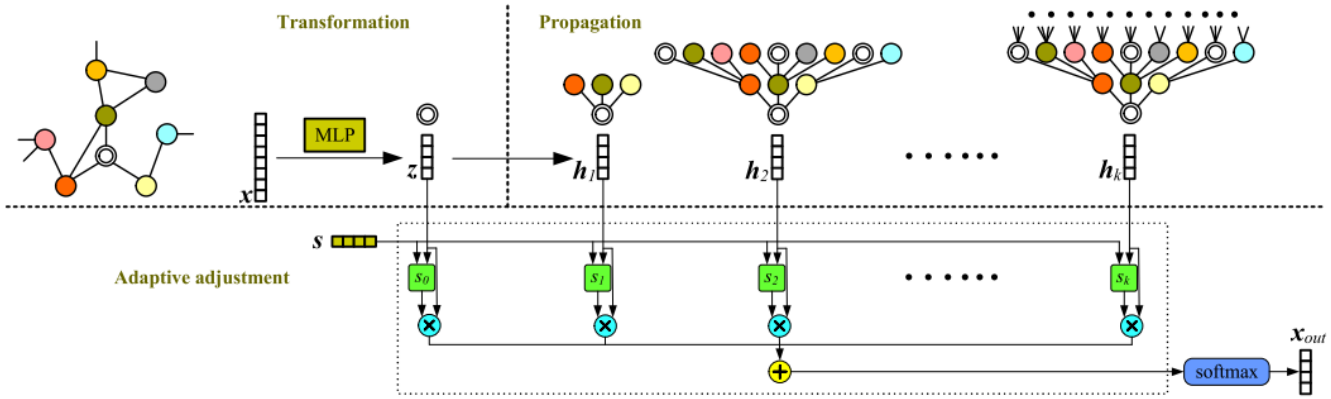


Figure 5: An illustration of the proposed Deep Adaptive Graph Neural Network (DAGNN) . For clarity, we show the pipeline to generate the prediction for one node. Notation letters are consistent with Eq.(8) but bold lowercase versions are applied to denote representation vectors. s is the projection vector that computes retainment scores for representations generating from various receptive fields. s_0, s_1, s_2, and s_k represent the retainment scores of z, h_1, h_2, and h_k, respectively.

It can be seen as a type of label propagation since it just weighted ensemble the outputs of each propagated MLP predictions.

- The MLP prediction and the label propagation can not be decoupled and DAGNN is trained in an end-to-end manner. Correspondingly, the scalability is limited.
- One global vector is adopted to train the weight of different propagated label, such process may be hard to train.

/03 Method篇

NDLS(NeurIPS 2021, Spotlight)

AI Drive

Problem: how to control the smoothness in a node dependent way ?

Definition Local-Smoothing Iteration(LSI, parameterized by ϵ) is defined as

$$K(i,\epsilon) = \min\left\{k: \left\|\tilde{I}_i - I(k)_i\right\|_2 < \epsilon\right\},$$

Where ϵ is an arbitrary small constant with $\epsilon > 0$,

$$I(k)_{i,j} = \frac{\partial \widehat{\mathrm{X}}_{ih}^{(k)}}{\partial \widehat{\mathrm{X}}_{jh}^{(0)}}, \forall h \in \{1,2,\cdots,f\}, \text{and } \tilde{I} = I(\infty).$$

Node Dependent Local Smoothing

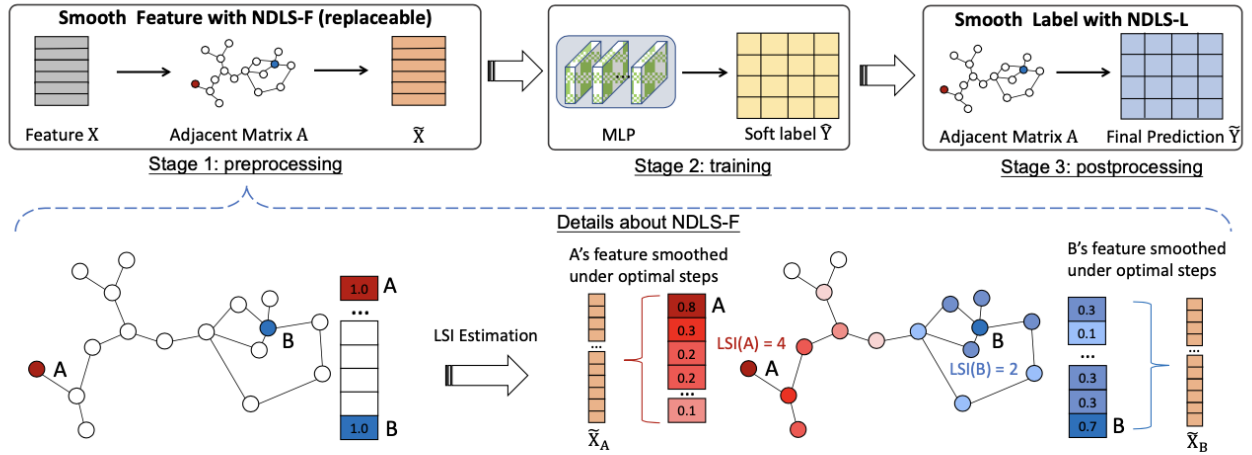


Figure 2: Overview of the proposed NDLS method, containing 1) feature smoothing with NDLS (NDLS-F); 2) model training with smoothed features; 3) label smoothing with NDLS (NDLS-L). NDLS-F and NDLS-L are pre-processing and post-processing steps, respectively.

Node Dependent Propagation → P(NDLS-F)-T-P(NDLS-L) ← Node Dependent Propagation

All ML models (MLP, LR, XGB, ect.) can be applied

Method

AI Drive

Vector Form

$$\widetilde{\mathbf{X}}_i(\epsilon) = \frac{1}{K(i,\epsilon)+1} \sum_{k=0}^{K(i,\epsilon)} \mathbf{X}_i^{(k)}.$$

NDLS-F

$$\widetilde{\mathbf{Y}}_i(\epsilon) = \frac{1}{K(i,\epsilon)+1} \sum_{k=0}^{K(i,\epsilon)} \hat{\mathbf{Y}}_i^{(k)}.$$

NDLS-L

Matrix Form

$$\widetilde{\mathbf{X}}(\epsilon) = \sum_{k=0}^{\max_i K(i,\epsilon)} \mathbf{M}^{(k)}\mathbf{X}^{(k)}, \qquad \mathbf{M^{(k)}}_{ij} = \begin{cases} \frac{1}{K(i,\epsilon)+1}, & i=j \quad and \quad k \le K(i,\epsilon) \\ 0, & \text{otherwise} \end{cases}$$

T

$$\widetilde{\mathbf{Y}}(\epsilon) = \sum_{k=0}^{\max_i K(i,\epsilon)} \mathbf{M}^{(k)}\hat{\mathbf{Y}}^{(k)},$$

Table 1: Algorithm analysis for existing scalable GNNs. n, m, c, and f are the number of nodes, edges, classes, and feature dimensions, respectively. b is the batch size, and k refers to the number of sampled nodes. L corresponds to the number of times we aggregate features, K is the number of layers in MLP classifiers. For the coupled GNNs, we always have $K = L$.

Type	Method	Preprocessing and postprocessing	Training	Inference	Memory
Node-wise sampling	GraphSAGE	-	$\mathcal{O}(k^L nf^2)$	$\mathcal{O}(k^L nf^2)$	$\mathcal{O}(bk^L f + Lf^2)$
Layer-wise sampling	FastGCN	-	$\mathcal{O}(kLnf^2)$	$\mathcal{O}(kLnf^2)$	$\mathcal{O}(bkLf + Lf^2)$
Graph-wise sampling	Cluster-GCN	$\mathcal{O}(m)$	$\mathcal{O}(Lmf + Lnf^2)$	$\mathcal{O}(Lmf + Lnf^2)$	$\mathcal{O}(bLf + Lf^2)$
Linear model	SGC	$\mathcal{O}(Lmf)$	$\mathcal{O}(nf^2)$	$\mathcal{O}(nf^2)$	$\mathcal{O}(bf + f^2)$
	S^2GC	$\mathcal{O}(Lmf)$	$\mathcal{O}(nf^2)$	$\mathcal{O}(nf^2)$	$\mathcal{O}(bf + f^2)$
	SIGN	$\mathcal{O}(Lmf)$	$\mathcal{O}(Knf^2)$	$\mathcal{O}(Knf^2)$	$\mathcal{O}(bLf + Kf^2)$
	GBP	$\mathcal{O}(Lnf + L\frac{\sqrt{m \lg n}}{\varepsilon})$	$\mathcal{O}(Knf^2)$	$\mathcal{O}(Knf^2)$	$\mathcal{O}(bf + Kf^2)$
Linear model	NDLS	$\mathcal{O}(Lmf + Lmc)$	$\mathcal{O}(Knf^2)$	$\mathcal{O}(Knf^2)$	$\mathcal{O}(bf + Kf^2)$

NDLS has a smaller or competitive training and inference complexity comparing with other scalable GNNs.

What influences LSIs?

AI Drive

Theorem 3.1. *Given feature smoothing* $\mathbf{X}^{(k)} = \hat{\mathbf{A}}^k\mathbf{X}$ *with* $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, *we have*

$$K(i,\epsilon) \leq \log_{\lambda_2}\left(\epsilon\sqrt{\frac{\tilde{d}_i}{2m+n}}\right),$$

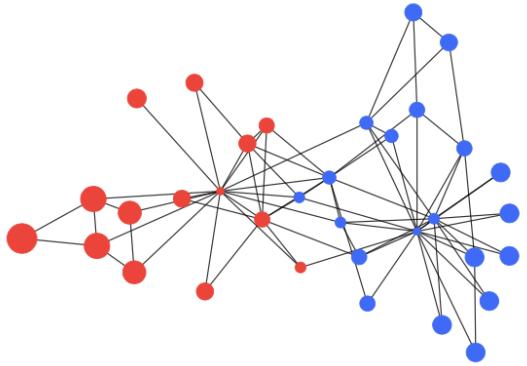
where λ_2 *is the second largest eigenvalue of* $\hat{\mathbf{A}}$, $\tilde{d}_i$ *denotes the degree of node* v_i *plus 1 (i.e.,* $\tilde{d}_i = d_i + 1$*), and* m, n *denote the number of edges and nodes respectively.*

Positively related with

- The scale of graph(m, n)
- The sparsity of graph (λ_2)

Negatively related with

- The node degree (d_i)



(b) The visualization of LSI

- Adjacent nodes have similar LSIs
- Nodes with a super-node as neighbors (or neighbor's neighbors) may have small LSIs

Theorem 3.2. *For any nodes* i *in a graph* $\mathcal{G}$,

$$K(i,\epsilon) \leq \max\{K(j,\epsilon), j \in N(i)\} + 1,$$

where $N(i)$ *is the set of node* v_i*'s neighbours.*

AI Drive

/04 Experiment篇

Setups

AI Drive

➢ Datasets

Table 2: Overview of datasets and task types (T/I represents Transductive/Inductive).

Dataset	#Nodes	#Features	#Edges	#Classes	#Train/Val/Test	Type	Description
Cora	2,708	1,433	5,429	7	140/500/1,000	T	citation network
Citeseer	3,327	3,703	4,732	6	120/500/1,000	T	citation network
Pubmed	19,717	500	44,338	3	60/500/1,000	T	citation network
Industry	1,000,000	64	1,434,382	253	5K/10K/30K	T	short-form video network
ogbn-papers100M	111,059,956	128	1,615,685,872	172	1,207K/125K/214K	T	citation network
Flickr	89,250	500	899,756	7	44K/22K/22K	I	image network
Reddit	232,965	602	11,606,919	41	155K/23K/54K	I	social network

➢ Baselines

1. **Decoupled GNNs:** APPNP, AP-GCN, PPRGo, DAGNN
2. **Coupled GNNs**: GCN, GAT, JK-Net
3. **Linear Models**: MLP, SGC, SIGN, S2GC, GBP

https://github.com/zwt233/NDLS

Accuracy and Efficiency Comparison

AI Drive

Table 3: Results of transductive settings. OOM means "out of memory".

Type	Models	Cora	Citeseer	PubMed	Industry	ogbn-papers100M
Coupled	GCN	81.8±0.5	70.8±0.5	79.3±0.7	45.9±0.4	OOM
	GAT	83.0±0.7	72.5±0.7	79.0±0.3	46.8±0.7	OOM
	JK-Net	81.8±0.5	70.7±0.7	78.8±0.7	47.2±0.3	OOM
Decoupled	APPNP	83.3±0.5	71.8±0.5	80.1±0.2	46.7±0.6	OOM
	AP-GCN	83.4±0.3	71.3±0.5	79.7±0.3	46.9±0.7	OOM
	PPRGo	82.4±0.2	71.3±0.5	80.0±0.4	46.6±0.5	OOM
	DAGNN (Gate)	84.4±0.5	73.3±0.6	80.5±0.5	47.1±0.6	OOM
	DAGNN (NDLS-L)*	84.4±0.6	73.6±0.7	80.9±0.5	47.2±0.7	OOM
Linear	MLP	61.1±0.6	61.8±0.8	72.7±0.6	41.3±0.8	47.2±0.3
	SGC	81.0±0.2	71.3±0.5	78.9±0.5	45.2±0.3	63.2±0.2
	SIGN	82.1±0.3	72.4±0.8	79.5±0.5	46.3±0.5	64.2±0.2
	S^2GC	82.7±0.3	73.0±0.2	79.9±0.3	46.6±0.6	64.7±0.3
	GBP	83.9±0.7	72.9±0.5	80.6±0.4	46.9±0.7	65.2±0.3
Linear	NDLS-F+MLP*	84.1±0.6	73.5±0.5	81.1±0.6	47.5±0.7	65.3±0.5
	MLP+NDLS-L*	83.9±0.6	73.1±0.8	81.1±0.6	46.9±0.7	64.6±0.4
	SGC+NDLS-L*	84.2±0.2	73.4±0.5	81.1±0.4	47.1±0.6	64.9±0.3
	NDLS*	**84.6±0.5**	**73.7±0.6**	**81.4±0.4**	**47.7±0.5**	**65.6±0.3**

Models	Flickr	Reddit
GraphSAGE	50.1±1.3	95.4±0.0
FastGCN	50.4±0.1	93.7±0.0
ClusterGCN	48.1±0.5	95.7±0.0
GraphSAINT	51.1±0.1	96.6±0.1
NDLS-F+MLP*	51.9±0.2	96.6±0.1
GraphSAGE+NDLS-L*	51.5±0.4	96.3±0.0
NDLS*	**52.6±0.4**	**96.8±0.1**

Table 4: Results of inductive settings.

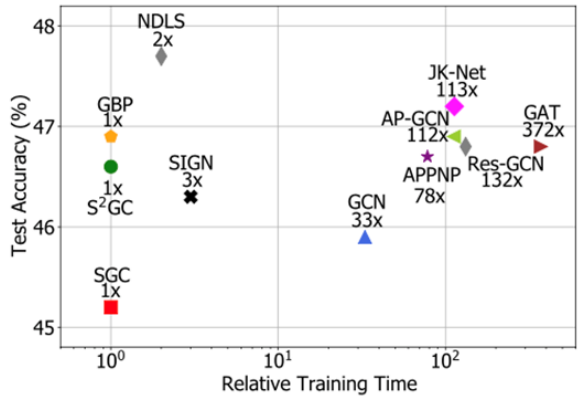
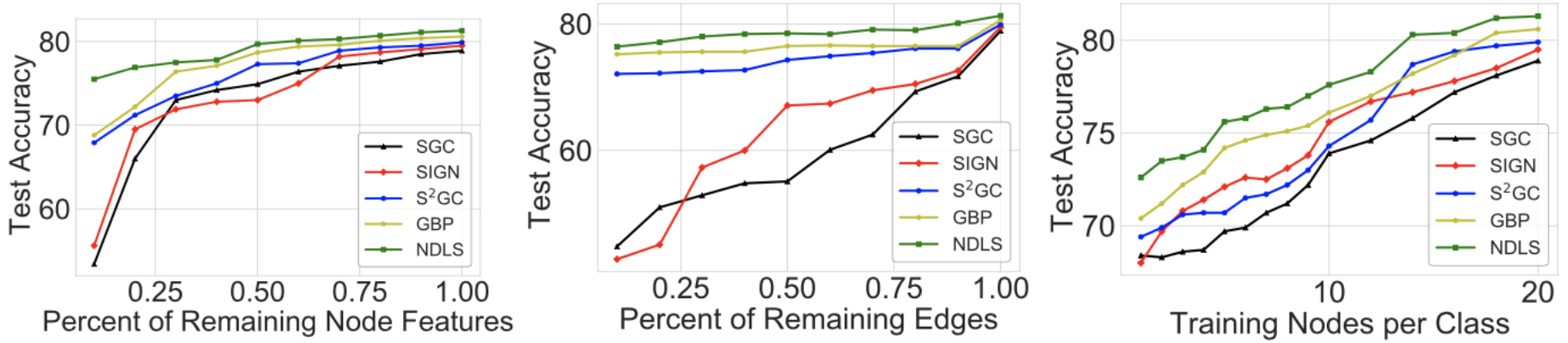


Figure 3: Performance along with training time on the Industry dataset.

- MLP with FS or LS achieve competitive performance.
- LS can be cooperated and improve the current model, such as SGC and DAGNN.
- Both FS and LS contributes to the performance.
- NDLS can get the best tradeoff between training time and test accuracy.

Other Experiments



(a) Feature Sparsity (b) Edge Sparsity (c) Label Sparsity

Figure 4: Test accuracy on PubMed dataset under different levels of feature, edge and label sparsity.

NDLS performs well and robustly on sparse graphs.

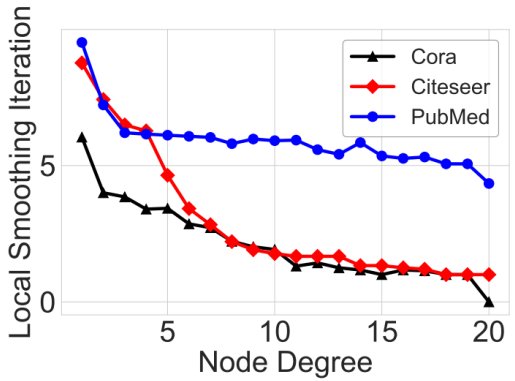
Table 1: Results of different base models on PubMed.

Base Models	Models	Accuracy	Gain
MLP	Base	72.7±0.6	-
	+ NDLS-F	81.1±0.6	+ 8.4
	+ NDLS-L	81.1±0.6	+ 8.4
	+ NDLS (both)	**81.4±0.4**	+ 8.7
RF	Base	74.4±0.2	-
	+ NDLS-F	80.3±0.1	+ 5.9
	+ NDLS-L	80.0±0.2	+ 5.6
	+ NDLS (both)	**80.5±0.4**	+ 6.1
XGB	Base	74.1±0.2	-
	+ NDLS-F	81.0±0.3	+ 6.9
	+ NDLS-L	79.8±0.2	+ 5.7
	+ NDLS (both)	**81.6±0.3**	+ 7.5

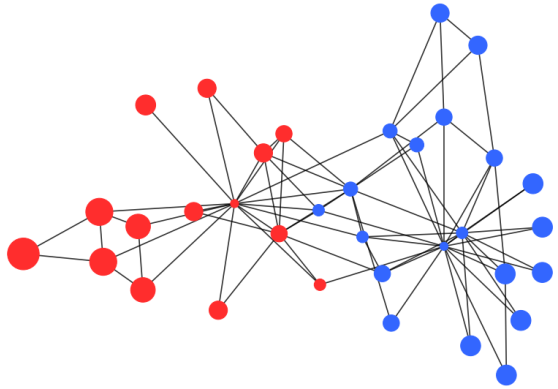
NDLS works with tree based models.

Interpretability

AI Drive



(a) LSI along with the node degree



(b) The visualization of LSI

Figure 5: (Left) LSI distribution along with the node degree in three citation networks. (Right) The visualization of LSI in Zachary's karate club network. Nodes with larger radius have larger LSIs.

Theorem 3.1. *Given feature smoothing* $\mathbf{X}^{(k)} = \hat{\mathbf{A}}^k\mathbf{X}$ *with* $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$*, we have*

$$K(i,\epsilon) \leq \log_{\lambda_2}\left(\epsilon\sqrt{\frac{\tilde{d}_i}{2m+n}}\right), \tag{5}$$

where λ_2 *is the second largest eigenvalue of* $\hat{\mathbf{A}}$*,* $\tilde{d}_i$ *denotes the degree of node* v_i *plus 1 (i.e.,* $\tilde{d}_i = d_i + 1$*), and* m*,* n *denote the number of edges and nodes respectively.*

Theorem 3.2. *For any nodes* i *in a graph* $\mathcal{G}$*,*

$$K(i,\epsilon) \leq \max\{K(j,\epsilon), j \in N(i)\} + 1, \tag{6}$$

where $N(i)$ *is the set of node* v_i*'s neighbours.*

- Nodes with a higher degree have a smaller LSI (**Theorem 3.1**)
- Adjacent nodes have similar LSIs (**Theorem 3.2**)
- Nodes with a super-node as neighbors (or neighbor's neighbors) may have small LSIs (**Theorem 3.2**)

Implementation

AI Drive

Table 6: URLs of baseline codes.

Type	Baselines	URLs
Coupled	GCN	https://github.com/rusty1s/pytorch_geometric
	GAT	https://github.com/rusty1s/pytorch_geometric
Decoupled	APPNP	https://github.com/rusty1s/pytorch_geometric
	PPRGo	https://github.com/TUM-DAML/pprgo_pytorch
	AP-GCN	https://github.com/spindro/AP-GCN
	DAGNN	https://github.com/divelab/DeeperGNN
Sampling	GraphSAGE	https://github.com/williamleif/GraphSAGE
	GraphSAINT	https://github.com/GraphSAINT/GraphSAINT
	FastGCN	https://github.com/matenure/FastGCN
	Cluster-GCN	https://github.com/benedekrozemberczki/ClusterGCN
Linear	SGC	https://github.com/Tiiiger/SGC
	SIGN	https://github.com/twitter-research/sign
	S^2GC	https://github.com/allenhaozhu/SSGC
	GBP	https://github.com/chennnM/GBP
	NDLS	https://github.com/zwt233/NDLS

AI Drive

/05 Conclusion篇

Conclusion

AI Drive

Advantages

1. Deep

- NDLS is able to go very deep without the over-smoothing issue

2. Scalable

- MLP (or tree-based model) is easy for distributed learning (no feature pulling communication)

3. Efficient

- MLP (or tree-based model) is simple and acquires less computation cost compared with GNNs

4. Flexible

- NDLS can be applied to any ML models

Applications

1. Sparse Graph

- NDLS can help to propagate the graph information of sparse graph by increasing the propagation steps.

2. Large Graph

- NDLS is scalable in distributed settings for extremely large graph in industrial production environment.

PKU-DAIR实验室

AI Drive

北京大学数据与智能实验室(Data and Intelligence Research Lab at Peking University)：
实验室由北京大学计算机系长江学者特聘教授崔斌老师带领，多年来主要在人工智能、大数据等领域进行前沿研究，在理论和技术创新以及系统研发上取得多项成果，已在国际顶级学术会议和期刊发表学术论文100余篇。

开源项目：实验室围绕机器学习系统已经展开了多方面的研究工作，包括机器学习/深度学习系统优化、 AutoML、图机器学习、AI4DB等，发布了多个开源项目：

黑盒优化系统OpenBox
https://github.com/PKU-DAIR/open-box

OpenBox

自动化机器学习系统MindWare
https://github.com/PKU-DAIR/mindware

MindWare
TOOLS FOR DATA INTELLIGENCE

分布式深度学习系统河图（Hetu）
https://github.com/PKU-DAIR/hetu

Hetu 河图

分布式机器学习平台Angel
https://github.com/Angel-ML

Angel

欢迎感兴趣的同学联系实习或交流问题！

微信：z1299799152

邮箱：wentao.zhang@pku.edu.cn

企业合作:
2017年，课题组与腾讯公司成立北京大学-腾讯协同创新实验室，深度合作并开源了分布式机器学习平台Angel。另外，实验室还与阿里巴巴、苹果、百度、快手、中兴通讯等多家知名企业开展项目合作和前沿探索，解决实际问题，进行科研成果的转化落地。

AI Drive

Thanks