Less is More: SlimG for Accurate, Robust, and Interpretable Graph Mining

J Yoo, MC Lee, S Shekhar, C Faloutsos

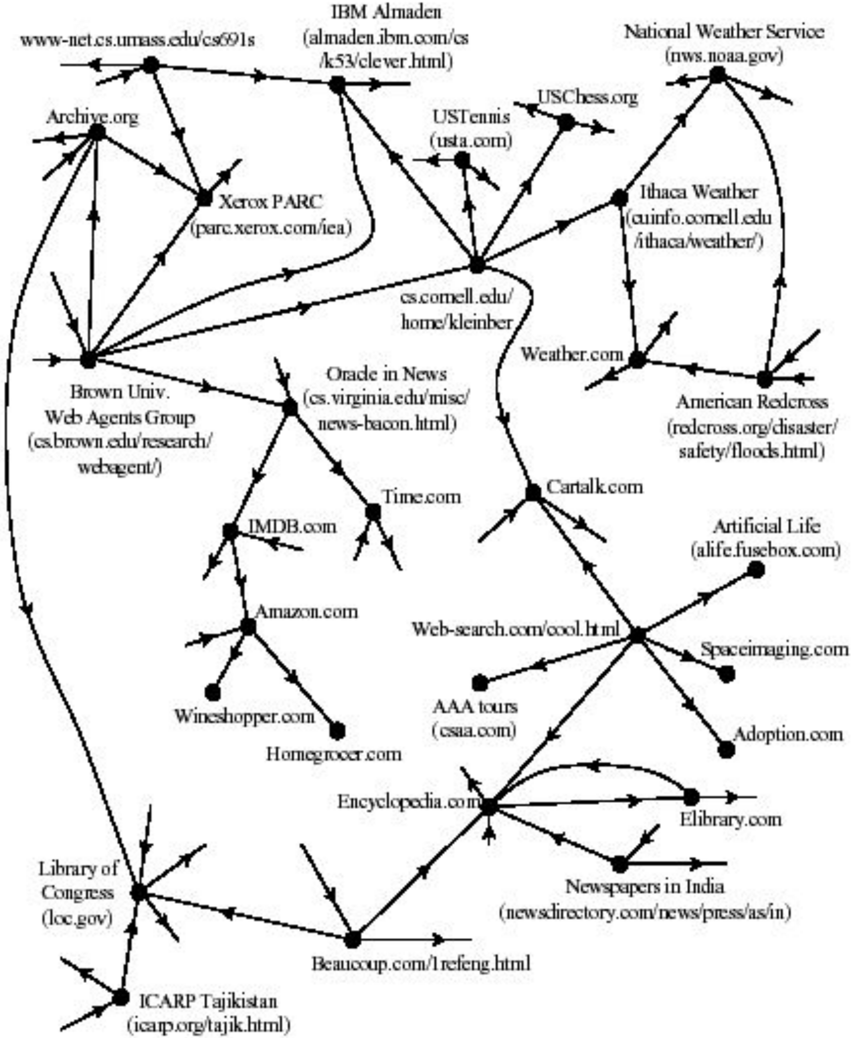
KDD 2023

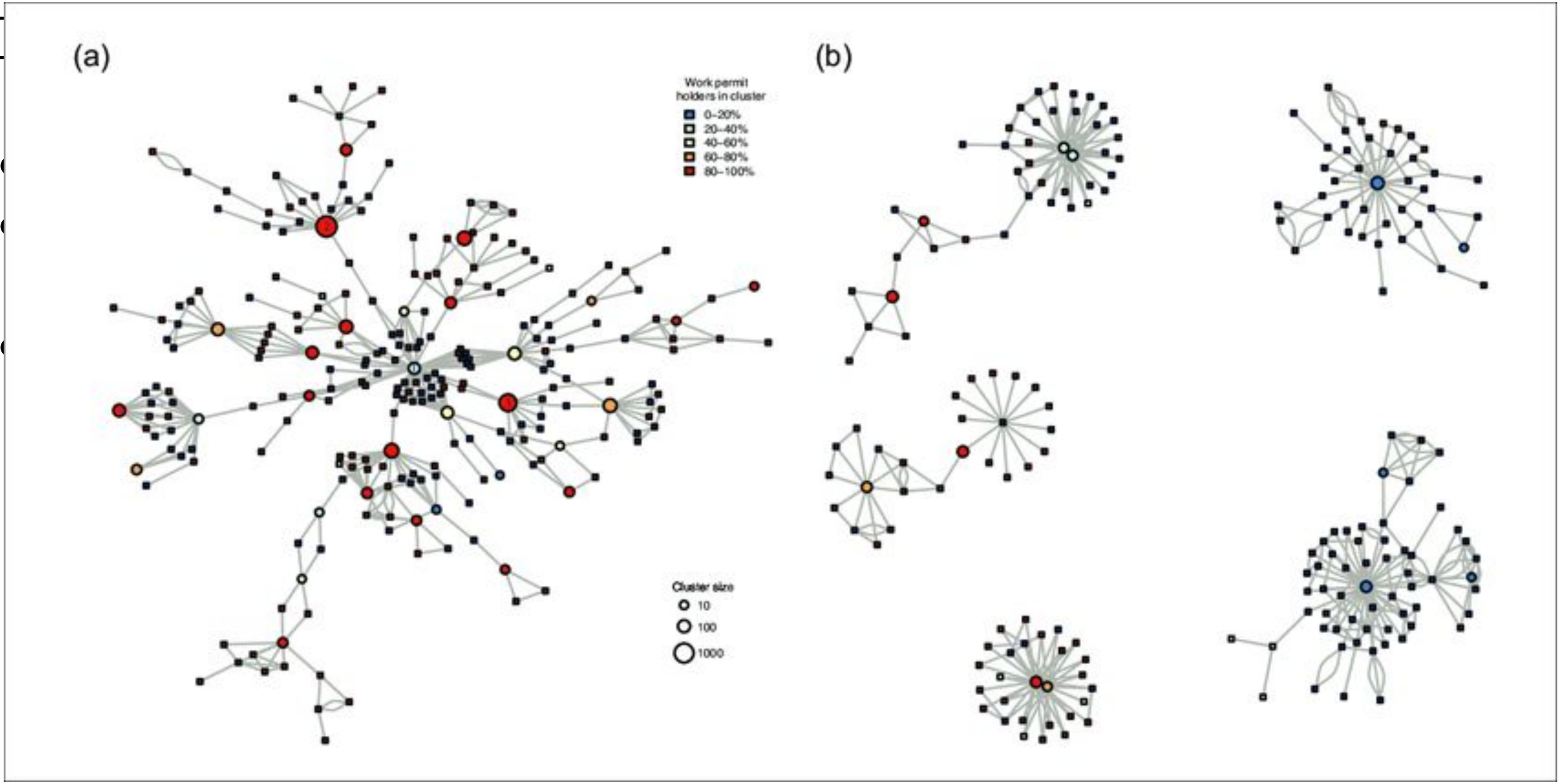
그래프 마이닝

- 그래프는 관계 정보가 포함되는 데이터를 표현할때 적합한 자료구조임
- 웹 네트워크, SNS, 감염 네트워크, 인용 그래프 등 다양한 분야의 데이터가 그래프로 표현됨
- 이런 그래프에서 유의미한 정보를 도출하는 것이 그래프 마이닝임

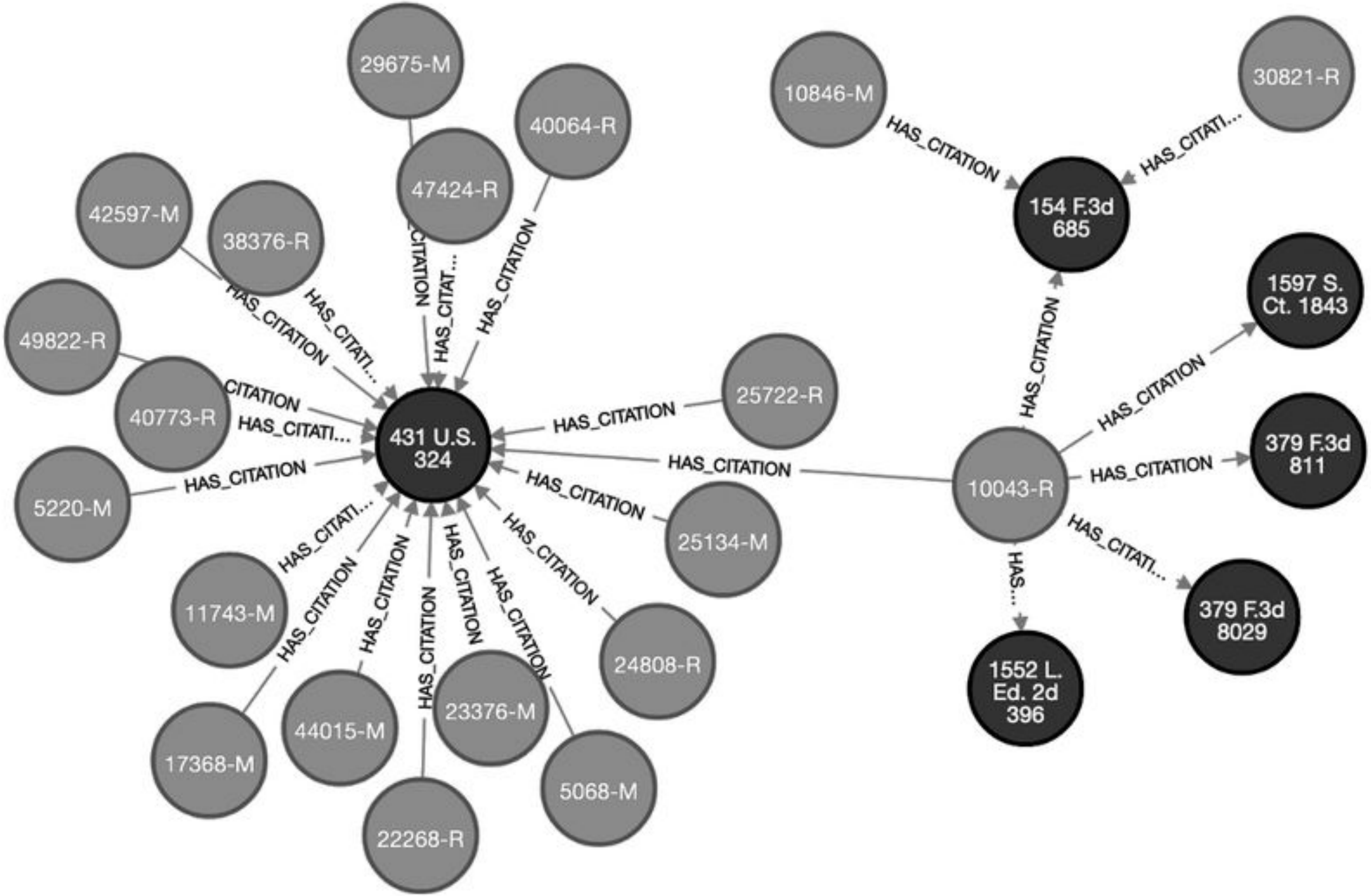
그래프 마이닝

- 그래프는 관계 … 임
- 웹 네트워크, SN … 이터가 그래프로 표현됨
- 이런 그래프에서





(a)
(b)
Work permit holders in cluster
0–20%
20–40%
40–60%
60–80%
80–100%
Cluster size
10
100
1000

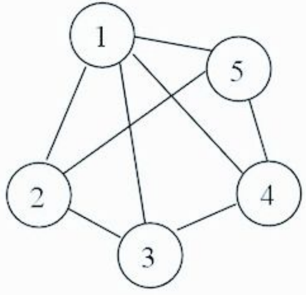

29675-M
40064-R
47424-R
42597-M
38376-R
49822-R
40773-R
5220-M
431 U.S. 324
25722-R
25134-M
24808-R
11743-M
17368-M
44015-M
22268-R
23376-M
5068-M
10846-M
30821-R
154 F.3d 685
1597 S. Ct. 1843
10043-R
379 F.3d 811
379 F.3d 8029
1552 L. Ed. 2d 396
HAS_CITATION
HAS_CITATI...

그래프 마이닝

- 그래프 마이닝의 주제로 노드 분류, 클러스터링, 링크 예측 등의 문제가 있음
- 노드 분류는 해당 노드가 어떤 그룹에 속하는지 파악하는 문제임. SNS에서 악성 유저 판별, 남녀판별, 감염 그래프에서 감염자/정상인 판별, 인용 그래프에서 논문의 주제 예측 등의 문제가 노드 분류 문제에 해당함
- 클러스터링은 그래프의 연결 정보와 노드의 특징으로만 해당 그래프 내의 노드를 그룹으로 만드는 문제임. SNS에서 커뮤니티 감지 문제가 있음
- 링크 예측은 그래프에 추가되어야 하는 연결 정보가 있는지, 또는 잘못된 연결을 제거해야 하는지 판별하는 문제임. 인용 그래프에서 인용을 예측하는 문제가 있음

그래프

- 그래프는 수학적으로 다음과 같이 정의할 수 있음
- 그래프 G는 노드 Feature 행렬 X와 관계 인접 행렬 A를 요소로 가지는 집합임
- X는 n*d의 행렬이며, n은 노드 수, d는 Feature의 차원 수를 의미함
- X의 행이 노드의 Feature Vector임
- A는 n*n의 행렬이며, 두 노드 (i, j)가 인접한(연결된) 경우 $A_{ij} = 1$, 인접하지 않은 경우 $A_{ij} = 0$임
- 라벨은 각 노드마다 가지는 [1, c] 사이의 하나의 실수 값임



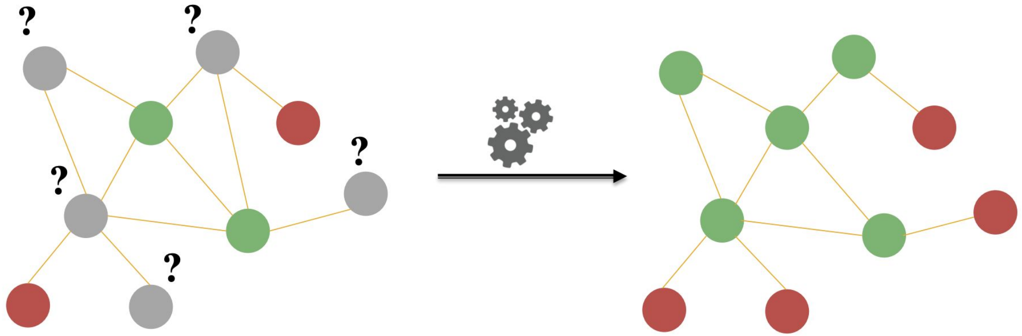
$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$G = (\mathbf{A}, \mathbf{X}) \quad \mathbf{y} \in \{1, \cdots, c\}^m$$

$$\mathbf{A} \in \mathbb{R}^{n \times n} \quad \mathbf{X} \in \mathbb{R}^{n \times d}$$

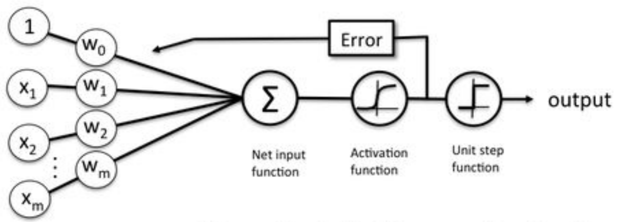
노드 분류 문제 - 준지도학습

- 해당 논문은 노드 분류 문제를 해결하는 프레임워크를 제안하며, 이 노드 분류 문제의 학습 설정은 준지도학습 설정임
- 준지도 학습 은 레이블이 있는 데이터가 매우 적고, 레이블이 없는 데이터가 많은 환경이 주어져서 데이터의 레이블을 추정하도록 학습하는 설정임
- 노드 분류 문제의 준지도 학습 설정은 레이블이 있는 노드가 매우 적고, 레이블이 없는 노드가 많은 환경에서 노드의 레이블을 추정하는 설정임

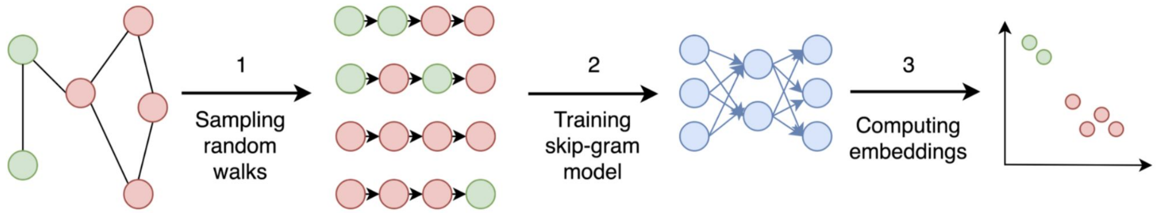


기존 접근 방법과 GNN

- 그래프의 연결 정보를 고려하지 않고 노드의 Feature만 이용하여 기존 머신러닝(또는 딥러닝) 방법론으로도 노드 분류를 수행할 수 있지만, 성능이 매우 낮음
- 이는 그래프의 연결 정보가 노드 분류에 있어서 중요하다는 것을 의미함
- DeepWalk(2014)는 딥러닝을 통해 연결 정보를 반영하도록 만든 초기 프레임워크임
- 이후 GCN의 등장으로 GNN이라는 명칭으로 분야가 활발한 관심을 받게 됨

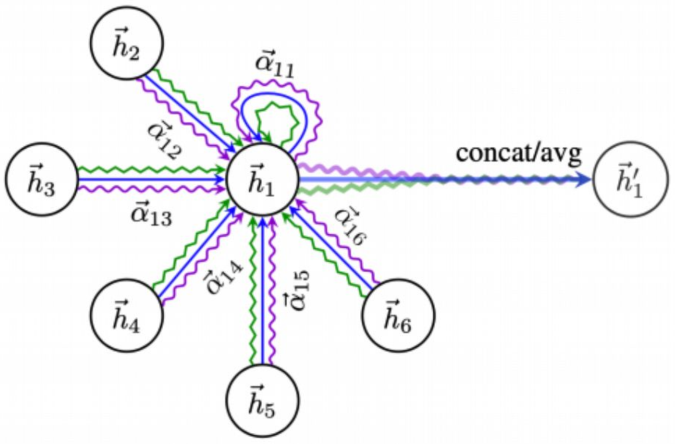


Schematic of a logistic regression classifier.



기존 접근 방법과 GNN

- GCN, GraphSAGE, GAT 등 강력한 GNN 모델의 등장으로 그래프 마이닝이 활발해짐
- 이후의 모델들은 대부분 해당 모델들을 기반으로 변형시킨 모델들임
- 그러나, 이러한 GNN 모델들은 단일 특성의 그래프에서만 좋은 성능을 내며, 몇몇 모델은 메모리와 처리 성능을 과도하게 사용하여 대규모 그래프 데이터를 처리하기 어려움



GnnExp

- 저자는 GNN 모델들이 특정한 그래프만 잘 처리한다는 점을 분석하기 위해 GnnExp 프레임워크를 설계함
- 이는 GNN의 특성(Pros and Cons)를 분석하기 위한 방법론이 포함됨
- GnnExp는 GNN 모델에 있는 비선형성을 무시하고 선형화하여 GNN 모델을 표현함
- 이때, 선형화는 GNN 모델이 가진 Propagator Function에서 비선형성을 제거하고 학습되는 파라미터를 제거하는 것을 의미함
- 이 과정을 통해 GnnExp로 표현되는 GNN 모델은 표현력을 유지하면서, 해당 GNN 모델의 해석을 가능하게 함
- GnnExp는 선형인 GNN의 일반화된 표현으로도 해석할 수 있음

GnnExp

- 선형화된 GNN 모델들을 저자는 다음과 같이 분류함

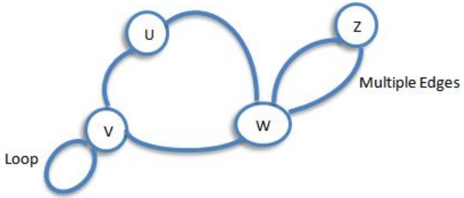
Model	Type	Propagator function $\mathcal{P}(\mathbf{A}, \mathbf{X})$
LR	Linear	$\mathbf{X}$
SGC	Linear	$\tilde{\mathbf{A}}_{\text{sym}}^{K}\,\mathbf{X}$
DGC	Linear	$[(1 - T/K)\mathbf{I} + (T/K)\tilde{\mathbf{A}}_{\text{sym}}]^{K}\,\mathbf{X}$
S^2GC	Linear	$\sum_{k=1}^{K}(\alpha\mathbf{I} + (1-\alpha)\tilde{\mathbf{A}}_{\text{sym}}^{k})\,\mathbf{X}$
G^2CN	Linear	$\Vert_{i=1}^{N}[\mathbf{I} - (T_i/K)((b_i - 1)\mathbf{I} + \mathbf{A}_{\text{sym}})^2]^{K}\,\mathbf{X}$
PPNP*	Decoupled	$(\mathbf{I} - (1-\alpha)\tilde{\mathbf{A}}_{\text{sym}})^{-1}\,\mathbf{X}$
APPNP*	Decoupled	$[\sum_{k=0}^{K-1}\alpha(1-\alpha)^{k}\tilde{\mathbf{A}}_{\text{sym}}^{k} + (1-\alpha)^{K}\tilde{\mathbf{A}}_{\text{sym}}^{K}]\,\mathbf{X}$
GDC*	Decoupled	$\mathbf{S} = \text{sparse}_{\epsilon}(\sum_{k=0}^{\infty}(1-\alpha)^{k}\tilde{\mathbf{A}}_{\text{sym}}^{k})$ for $\tilde{\mathbf{S}}_{\text{sym}}\,\mathbf{X}$
GPR-GNN*	Decoupled	$\Vert_{k=0}^{K}\,\tilde{\mathbf{A}}_{\text{sym}}^{k}\,\mathbf{X}$
ChebNet*	Coupled	$\Vert_{k=0}^{K-1}\,\mathbf{A}_{\text{sym}}^{k}\,\mathbf{X}$
GCN*	Coupled	$\tilde{\mathbf{A}}_{\text{sym}}^{K}\,\mathbf{X}$
SAGE*	Coupled	$\Vert_{k=0}^{K}\,\mathbf{A}_{\text{row}}^{k}\,\mathbf{X}$
GCNII*	Coupled	$\Vert_{k=0}^{K-2}\,\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X} \,\Vert\, ((1-\alpha)\tilde{\mathbf{A}}_{\text{sym}}^{K} + \alpha\tilde{\mathbf{A}}_{\text{sym}}^{K-1})\,\mathbf{X}$
H_2GCN*	Coupled	$\Vert_{k=0}^{2K}\,\mathbf{A}_{\text{sym}}^{k}\mathbf{X}$
GAT**	Attention	$\prod_{k=1}^{K}[\text{diag}(\mathbf{X}\mathbf{w}_{k,1})\tilde{\mathbf{A}} + \tilde{\mathbf{A}}\text{diag}(\mathbf{X}\mathbf{w}_{k,2})]\,\mathbf{X}$
DA-GNN**	Attention	$\sum_{k=0}^{K}\text{diag}(\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X}\mathbf{w})\tilde{\mathbf{A}}_{\text{sym}}^{k}\,\mathbf{X}$

GnnExp

- 해당 행렬식에서 Notation에 대한 설명
- A와 **Ã**는 self-loop의 유무 차이임
- sym은 symmetrical normalization으로 GCN에서 사용되는 normalization 방법
- row는 row normalization으로 해당 행 기준으로만 normalization

$$\mathbf{A}_{\mathrm{sym}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \qquad \mathbf{A}_{\mathrm{row}} = \mathbf{D}^{-1}\mathbf{A}$$

$$\tilde{\mathrm{A}}=\mathrm{I}+\mathrm{A}$$



Model	Type	Propagator function $\mathcal{P}(\mathbf{A},\mathbf{X})$
LR	Linear	$\mathbf{X}$
SGC	Linear	$\tilde{\mathbf{A}}_{\mathrm{sym}}^{K}\,\mathbf{X}$
DGC	Linear	$[(1-T/K)\mathbf{I}+(T/K)\tilde{\mathbf{A}}_{\mathrm{sym}}]^{K}\,\mathbf{X}$
S^2GC	Linear	$\sum_{k=1}^{K}(\alpha\mathbf{I}+(1-\alpha)\tilde{\mathbf{A}}_{\mathrm{sym}}^{k})\,\mathbf{X}$
G^2CN	Linear	$\Vert_{i=1}^{N}[\mathbf{I}-(T_i/K)((b_i-1)\mathbf{I}+\mathbf{A}_{\mathrm{sym}})^2]^{K}\,\mathbf{X}$
PPNP*	Decoupled	$(\mathbf{I}-(1-\alpha)\tilde{\mathbf{A}}_{\mathrm{sym}})^{-1}\,\mathbf{X}$
APPNP*	Decoupled	$[\sum_{k=0}^{K-1}\alpha(1-\alpha)^{k}\tilde{\mathbf{A}}_{\mathrm{sym}}^{k}+(1-\alpha)^{K}\tilde{\mathbf{A}}_{\mathrm{sym}}^{K}]\,\mathbf{X}$
GDC*	Decoupled	$\mathbf{S}=\mathrm{sparse}_{\epsilon}(\sum_{k=0}^{\infty}(1-\alpha)^{k}\tilde{\mathbf{A}}_{\mathrm{sym}}^{k})$ for $\tilde{\mathbf{S}}_{\mathrm{sym}}\,\mathbf{X}$
GPR-GNN*	Decoupled	$\Vert_{k=0}^{K}\tilde{\mathbf{A}}_{\mathrm{sym}}^{k}\,\mathbf{X}$
ChebNet*	Coupled	$\Vert_{k=0}^{K-1}\mathbf{A}_{\mathrm{sym}}^{k}\,\mathbf{X}$
GCN*	Coupled	$\tilde{\mathbf{A}}_{\mathrm{sym}}^{K}\,\mathbf{X}$
SAGE*	Coupled	$\Vert_{k=0}^{K}\mathbf{A}_{\mathrm{row}}^{k}\,\mathbf{X}$
GCNII*	Coupled	$\Vert_{k=0}^{K-2}\tilde{\mathbf{A}}_{\mathrm{sym}}^{k}\mathbf{X}\Vert((1-\alpha)\tilde{\mathbf{A}}_{\mathrm{sym}}^{K}+\alpha\tilde{\mathbf{A}}_{\mathrm{sym}}^{K-1})\,\mathbf{X}$
H_2GCN*	Coupled	$\Vert_{k=0}^{2K}\mathbf{A}_{\mathrm{sym}}^{k}\mathbf{X}$
GAT**	Attention	$\prod_{k=1}^{K}[\mathrm{diag}(\mathbf{X}\mathbf{w}_{k,1})\tilde{\mathbf{A}}+\tilde{\mathbf{A}}\mathrm{diag}(\mathbf{X}\mathbf{w}_{k,2})]\,\mathbf{X}$
DA-GNN**	Attention	$\sum_{k=0}^{K}\mathrm{diag}(\tilde{\mathbf{A}}_{\mathrm{sym}}^{k}\mathbf{X}\mathbf{w})\tilde{\mathbf{A}}_{\mathrm{sym}}^{k}\,\mathbf{X}$

GNN의 문제점

- 저자가 파악한 기존 GNN 모델들의 문제점은 다음과 같음
- 강건함 부족 - 모델들은 다양한 그래프 시나리오에 동시에 대응하지 못함
- 노이즈 취약성 - 모델들은 노드 Feature에 의존하므로 Feature가 Nosiy한 경우 그래프 구조를 전부 이용하지 못함
- 비효율적, 비유효성 - Aggregation이 Summation이 아닌 Concatenation 기반인 모델은 지나치게 파라미터가 많아 학습이 어려우며, 이웃과 2-hop 이웃 인접 행렬이 상관 관계가 높아지게됨
- 많은 하이퍼파라미터 - 많은 하이퍼파라미터는 모델 해석의 어려움과 성능을 위해 방대한 튜닝이 필요함

개선의 고려사항

- 앞서 분석한 문제점들을 기반으로 저자는 모델을 개선하기 위해 3가지 고려 사항을 정리
- 노드 Feature, 이웃 Feature, 장거리 이웃 Feature를 어떻게 통합해야 할까?
- 인접 행렬 변형과 정규화 - 어떤 정규화와 인접 행렬 변형이 유효할까?
- 이종성 - 기존 GNN은 특히 Heterophily에서 성능이 저하되는데, 이를 극복하려면?

SlimG

- 저자는 앞서 설명한 고려사항들을 바탕으로 새로운 Propagator를 설계함
- SlimG의 Propagator는 아래와 같이 구성됨
- 먼저 Feature, 이웃 Feature, 2-hop 이웃 Feature, Structure Feature를 통합하여 모델이 노드 Feature Noise, 혹은 Structure Noise가 있는 경우 중요한 Feature를 스스로 선택하도록 함
- 이는 각 Feature를 Concat하여 가능함(이웃 Feature의 Concat이 아님)

$$\mathcal{P}(\mathrm{A}, \mathrm{X}) = \underbrace{\mathbf{U}}_{\text{Structure}} \parallel \underbrace{g(\mathrm{X})}_{\text{Features}} \parallel \underbrace{g(\mathrm{A}^2_{\text{row}}\mathrm{X})}_{\text{2-step neighbors}} \parallel \underbrace{g(\tilde{\mathrm{A}}^2_{\text{sym}}\mathrm{X})}_{\text{Neighbors}}$$

SlimG

- 2-hop 이웃 Feature는 Heterophily에 대응하기 위해 반영되는데, 이때 이웃 Feature가 바로 반영되는 것을 막기 위해 Self-loop가 없음
- row normalization은 normalization이 단순함을 위해 사용됨
- 또한 각 Feature의 Concat은 단일 hop이 아닌 다중 hop을 통합하는 효과를 부여함
- 그러나, 이렇게 직접 Feature들을 Concat하는 것은 너무 큰 메모리를 요구함

$$\mathcal{P}(\mathrm{A}, \mathrm{X}) = \underbrace{\mathbf{U}}_{\text{Structure}} \parallel \underbrace{g(\mathrm{X})}_{\text{Features}} \parallel \underbrace{g(\mathrm{A}^2_{\text{row}}\mathrm{X})}_{\text{2-step neighbors}} \parallel \underbrace{g(\tilde{\mathrm{A}}^2_{\text{sym}}\mathrm{X})}_{\text{Neighbors}}$$

SlimG

- 저자는 이 Propagation의 메모리를 감소시키고 유용한 특징만 남기기 위해 전통적인 방법으로 행렬을 압축함
- U는 인접행렬 A에 저수준 특이값분해(SVD)를 적용하여 A를 작은 수준으로 감소시킴
- 특이값 분해는 행렬을 구성하는데 필요한 성분들을 중요도(Rank) 순서로 가지는 대각행렬을 구하는데, 여기서 성분들을 제거하여 행렬을 압축할 수 있음
- g(·)는 L2 Normalization+주성분분석(PCA)임
- PCA는 직교화를 위해 사용되지만, 차원 감소로도 사용할 수 있음

$$\mathcal{P}(\mathrm{A}, \mathrm{X}) = \underbrace{\mathbf{U}}_{\text{Structure}} \parallel \underbrace{g(\mathrm{X})}_{\text{Features}} \parallel \underbrace{g(\mathrm{A}_{\text{row}}^{2}\mathrm{X})}_{\text{2-step neighbors}} \parallel \underbrace{g(\tilde{\mathrm{A}}_{\text{sym}}^{2}\mathrm{X})}_{\text{Neighbors}}$$

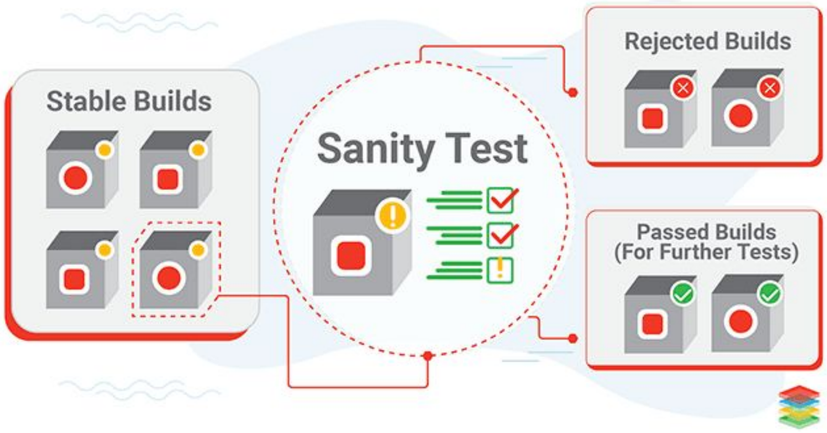
SlimG

- SlimG의 또다른 장점은 SVD, PCA를 이용하므로 하이퍼파라미터가 매우 적다는 점임

$$\mathcal{P}(\mathbf{A}, \mathbf{X}) = \underbrace{\mathbf{U}}_{\text{Structure}} \parallel \underbrace{g(\mathbf{X})}_{\text{Features}} \parallel \underbrace{g(\mathbf{A}_{\text{row}}^2 \mathbf{X})}_{\text{2-step neighbors}} \parallel \underbrace{g(\tilde{\mathbf{A}}_{\text{sym}}^2 \mathbf{X})}_{\text{Neighbors}}$$

Sanity Check

- 저자는 SlimG의 강건함을 입증하기 위해 Sanity Check를 설계함
- Sanity Check (Sanity Test)란?
- 소프트웨어 개발에서, 전체 시스템을 테스트 하는 것이 아닌, 구성 요소를 테스트하여 빠르게 오류를 검사하는 것
- Sanity Check를 통해 GNN들의 강점과 약점을 빠르게 평가할 수 있음



Sanity Check

- 구체적으로, Sanity Check의 시나리오는 9개 (실제 7개)로 구성됨
- Node Classification 문제에서 구성될 수 있는 그래프의 특성은 크게 두 가지 요소에 존재함
- Edge - 노드 간 연결이 어떤 특성을 가지는가?
- Features and Labels - 노드의 Feature가 Label과 어떤 관계가 있는가? 또는 Feature가 유의미한가?

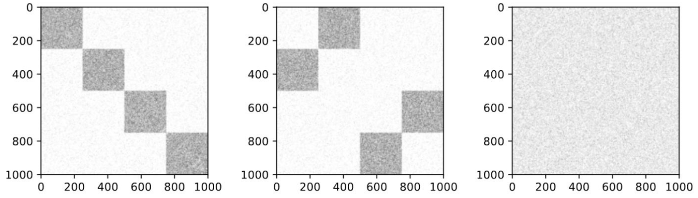
Sanity Check - Edge Charicteristics

- 에지의 특성은 3가지로 나뉨
- Uniform - 노드의 레이블과 이웃 연결 관계에 연관성이 없음, 즉 랜덤
- Homophily - 노드는 이웃과 보통 같은 레이블을 가짐, 전통적인 그래프 데이터 특성
- Heterophily - 노드는 보통 이웃과 다른 레이블을 가짐, 흔하지 않지만 real-world 그래프에서 자주 보임

Uniform: $P(Y_i = y \mid A_{ij} = 1, Y_j = y) = P(Y_i = y)$
Homophily: $P(Y_i = y \mid A_{ij} = 1, Y_j = y) > P(Y_i = y)$
Heterophily: $P(Y_i = y \mid A_{ij} = 1, Y_j = y) < P(Y_i = y)$

Sanity Check - Feature and Label Charicteristics

- Feature와 Label의 특성은 3가지로 나뉨
- Random - Feature가 Label에 대한 정보를 포함하지 않음
- Structural - Feature가 그래프 구조에 대한 정보를 포함함
- Semantic - Feature가 Label에 대한 정보를 포함함

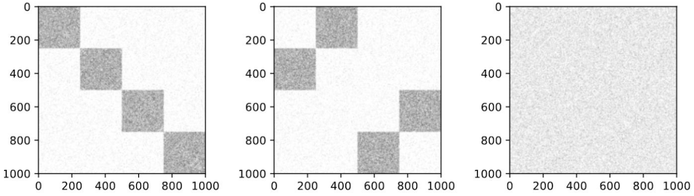


(b) Homophily A (c) Heterophily A (d) Uniform A

Random: $p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j, a_{ij}) = p(\mathbf{x}_i, \mathbf{x}_j)$
Structural: $p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j, a_{ij}) \neq p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j)$
Semantic: $p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j, a_{ij}) \neq p(\mathbf{x}_i, \mathbf{x}_j \mid a_{ij})$

Sanity Check

- Edge 특성과 Feature+Label 특성을 조합하면 총 9가지의 시나리오가 생성됨
- 이 중 무의미한 2가지 시나리오를 제외하고 7가지 시나리오에 대해 테스트
- 해당 시나리오에 맞추어 생성되는 4클래스 인공 그래프에서 실험함

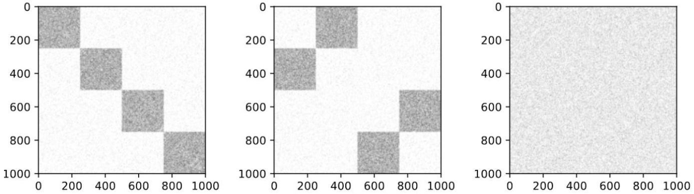


(b) Homophily A (c) Heterophily A (d) Uniform A

Structure	Feature		
	Semantic **X**	Structural **X**	Random **X**
Homophily **A**	Both help	Both help	**A** helps
Heterophily **A**	Both help	Both help	**A** helps
Uniform **A**	**X** helps	None helps	None helps

Sanity Check

- Edge 특성과 Feature+Label 특성을 조합하면 총 9가지의 시나리오가 생성됨
- 이 중 무의미한 2가지 시나리오를 제외하고 7가지 시나리오에 대해 테스트
- 해당 시나리오에 맞추어 생성되는 4클래스 인공 그래프에서 실험함



(b) Homophily A (c) Heterophily A (d) Uniform A

Structure	Feature		
	Semantic **X**	Structural **X**	Random **X**
Homophily **A**	Both help	Both help	**A** helps
Heterophily **A**	Both help	Both help	**A** helps
Uniform **A**	**X** helps	None helps	None helps

Sanity Check - Result

- Feature만 의미가 있는 경우, LR, SlimG에서 좋은 정확도를 보이며, 대부분의 GNN이 낮은 정확도를 보임
- 연결 구조가 유의미하고 Feature가 노이즈로 작동하는 경우 SlimG만 좋은 결과를 냄
- 연결 구조와 Feature가 모두 의미가 있는 경우 SlimG는 Homophily와 Heterophily 모두 좋은 성능을 보임

Model	Only X helps	Only A helps		Both X and A help				Avg. Acc	Avg. Rank
	Semantic X Uniform A	Random X Homophily	Random X Heterophily	Structural X Homophily	Structural X Heterophily	Semantic X Homophily	Semantic X Heterophily		
LR	83.7±0.6	24.2±0.7	24.2±0.7	71.4±0.9	66.8±2.2	83.4±0.6	83.4±0.6	62.4 (26.9)	10.7 (5.4)
Reg. Kernel	82.7±0.5	27.9±0.4	24.3±1.0	75.7±0.2	65.3±1.6	91.5±0.5	79.5±0.3	63.8 (27.0)	10.4 (4.3)
Diff. Kernel	26.8±1.7	38.0±8.7	37.6±7.5	79.5±0.3	73.5±0.6	70.9±23.	56.1±27.	54.6 (20.7)	10.6 (4.0)
RW Kernel	72.2±0.7	37.0±0.4	24.5±1.3	81.3±1.2	51.0±1.1	94.5±0.9	57.8±0.7	59.8 (24.7)	10.4 (3.6)
SGC	44.6±9.8	64.3±0.7	50.2±14.	87.1±0.6	84.3±0.5	93.9±0.9	91.5±0.5	73.7 (20.4)	5.7 (3.1)
DGC	63.8±1.0	50.5±13.	26.0±0.9	88.6±1.0	45.3±1.3	96.2±0.4	54.0±0.6	60.6 (24.6)	8.3 (5.9)
S^2GC	79.9±0.6	38.5±12.	25.4±0.9	88.4±1.0	67.9±1.5	95.9±0.6	78.0±0.5	67.7 (26.2)	7.4 (3.4)
G^2CN	25.2±0.3	24.2±1.1	25.0±0.1	88.5±1.0	88.6±1.2	24.3±1.1	50.7±31.	46.6 (30.2)	11.6 (6.3)
GCN	36.3±3.5	46.7±8.0	43.7±1.9	83.3±1.3	72.2±1.7	91.2±1.2	80.3±3.9	64.8 (22.1)	8.1 (3.0)
SAGE	80.3±1.1	31.1±0.7	34.6±2.1	83.9±0.8	81.3±0.7	94.4±0.5	94.4±0.9	71.4 (27.0)	5.7 (2.9)
GCNII	73.5±1.2	30.7±0.7	27.1±1.3	84.2±0.8	69.0±1.4	90.6±0.9	80.4±1.2	65.1 (25.7)	8.7 (1.8)
H^2GCN	80.2±1.5	27.0±1.0	27.5±0.8	78.0±0.9	74.6±1.3	91.9±0.7	92.2±0.9	67.3 (28.2)	8.0 (3.9)
APPNP	66.0±2.6	30.3±1.2	25.2±0.7	71.2±4.9	43.8±2.0	83.2±3.8	58.7±4.5	54.1 (21.6)	12.9 (2.0)
GPR-GNN	73.4±0.4	74.6±0.7	65.9±2.1	89.9±0.6	87.6±1.2	95.0±1.1	91.9±1.1	82.6 (11.2)	3.3 (2.1)
GAT	32.7±5.5	42.6±4.8	36.8±5.7	64.0±5.7	55.6±6.8	68.5±7.1	67.0±12.	52.5 (15.0)	11.6 (4.1)
SlimG (Ours)	81.0±1.1	87.1±1.4	89.2±1.2	88.1±0.5	88.9±0.7	94.4±0.6	93.9±0.5	88.9 (4.5)	2.6 (1.8)

Experiments

- 저자는 실험을 통해 각종 성능을 테스트하고 모듈의 특성을 분석함
- 주요한 실험만 설명함
- 정확도 - SlimG가 다양한 그래프에서 타 모델 대비 좋은 정확도를 보이는가?
- 속도 - SlimG는 빠르면서 정확도가 유지되는가?
- non-linearity - SlimG는 Linear Propagator인데, 여기에 non linearity를 추가하는 것이 더 나은가?
- 고차 Aggregation 필요성 - Propagation은 몇 hop을 사용하는 것이 좋은가? 직관적으로, 더 많은 hop을 사용할 수록 좋지 않을까?

Experiments - Settings

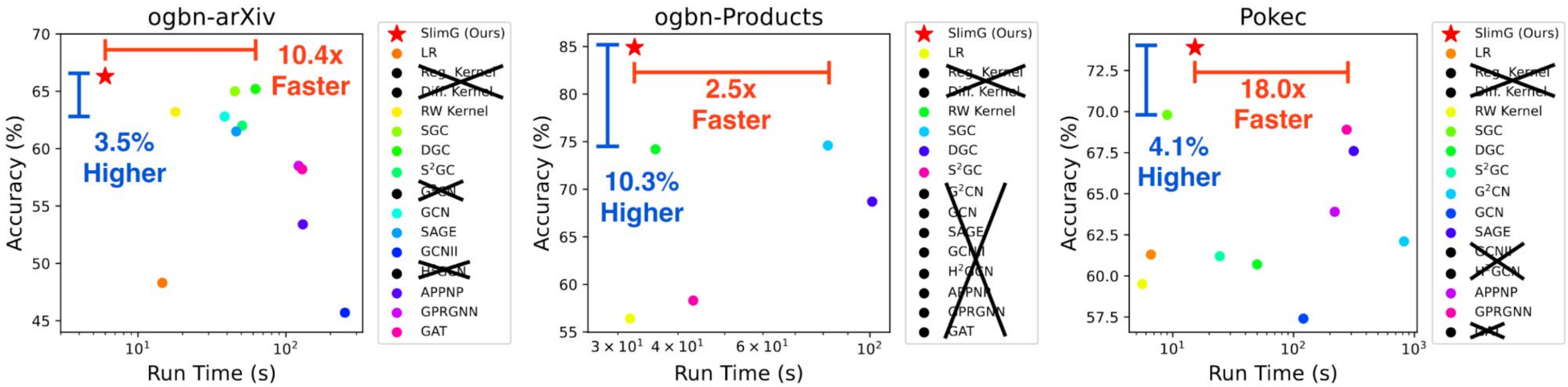
- 실험 데이터셋은 13개의 실제 데이터셋이며, 7개의 Homophily 데이터셋과 6개의 Heterophily 데이터셋을 사용함
- 준지도학습 설정으로 학습용 데이터의 2.5%만 라벨을 가짐
- 데이터셋의 학습, 검증과 시험 분배율은 2.5%/2.5%/95%임
- ogbn-Products, Pokec은 각각 Homophily, Heterophily의 대규모 그래프 데이터임

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1433	7
CiteSeer	3,327	4,732	3703	6
PubMed	19,717	44,338	500	3
Computers	13,752	245,861	767	10
Photo	7,650	119,081	745	8
ogbn-arXiv	169,343	1,166,243	128	40
ogbn-Products	2,449,029	61,859,140	100	30
Chameleon	2,277	36,101	2325	5
Squirrel	5,201	216,933	2089	5
Actor	7,600	29,926	931	5
Penn94	41,554	1,362,229	4814	2
Twitch	168,114	6,797,557	7	2
Pokec	1,632,803	30,622,564	65	2

Experiments - Accuracy

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec	Avg. Rank
LR	51.5±1.2	52.9±4.5	79.9±0.5	73.9±1.2	79.3±1.5	48.3±1.9	56.4±0.5	24.9±1.7	26.7±1.9	27.8±0.8	63.5±0.5	53.0±0.1	61.3±0.0	11.7 (4.2)
Reg. Kernel	67.8±2.5	62.1±4.4	83.4±1.4	80.3±1.4	87.1±1.2	O.O.M.	O.O.M.	29.4±2.6	24.3±2.3	29.6±1.4	O.O.M.	O.O.M.	O.O.M.	12.2 (3.8)
Diff. Kernel	70.6±1.5	62.7±3.8	82.1±0.4	83.1±1.0	89.8±0.6	O.O.M.	O.O.M.	34.5±7.9	28.3±1.5	24.7±0.9	53.5±0.8	O.O.M.	O.O.M.	11.8 (2.5)
RW Kernel	72.7±1.7	64.1±3.9	83.1±0.7	84.2±0.7	90.6±0.7	63.2±0.2	74.2±0.0	34.9±3.5	25.0±1.6	26.4±1.1	63.1±0.7	57.6±0.1	59.5±0.0	8.3 (3.3)
SGC	76.2±1.1	65.8±3.9	84.1±0.8	83.7±1.6	90.1±0.9	65.0±3.4	74.6±5.1	38.1±4.5	33.1±1.0	24.6±0.8	64.0±1.1	56.5±0.1	69.8±0.0	6.6 (4.2)
DGC	77.8±1.4	66.1±4.2	84.3±0.6	83.9±0.7	90.4±0.2	65.2±4.0	68.7±13.	37.2±3.7	29.2±1.2	25.2±2.1	62.5±0.4	58.2±0.2	60.7±0.1	6.6 (3.2)
S^{2}GC	78.3±1.5	66.9±4.4	84.3±0.3	83.1±0.8	90.1±0.8	62.0±7.4	58.3±18.	34.9±4.9	27.6±1.8	26.7±1.8	63.1±0.5	58.7±0.1	61.2±0.0	6.6 (2.7)
G^{2}CN	76.6±1.5	64.2±3.3	81.4±0.6	82.8±1.6	88.8±0.5	O.O.M.	O.O.M.	40.7±2.9	32.1±1.5	24.3±0.5	O.O.M.	O.O.M.	O.O.M.	10.5 (4.5)
GCN	76.0±1.2	65.0±2.9	84.3±0.5	85.1±0.9	91.6±0.5	62.8±0.6	O.O.M.	38.5±3.0	31.4±1.8	26.8±0.4	62.9±0.7	57.0±0.1	63.9±0.4	6.3 (2.4)
SAGE	74.6±1.3	63.7±3.6	82.9±0.4	83.8±0.5	90.6±0.5	61.5±0.6	O.O.M.	39.8±4.3	27.0±1.3	27.8±0.9	O.O.M.	56.6±0.4	68.9±0.1	8.5 (3.5)
GCNII	77.8±1.7	63.4±3.0	84.9±0.8	82.3±1.8	90.8±0.6	45.7±0.5	O.O.M.	30.5±2.5	21.9±3.0	29.0±1.3	64.5±0.5	56.9±0.6	62.1±0.3	8.4 (4.6)
H^{2}GCN	77.6±0.9	64.7±3.8	85.4±0.4	49.5±16.	75.8±11.	O.O.M.	O.O.M.	31.9±2.6	25.0±0.5	28.9±0.6	63.9±0.4	58.7±0.0	O.O.M.	8.9 (4.9)
APPNP	80.0±0.6	67.1±2.8	84.6±0.5	84.2±1.7	92.5±0.3	53.4±1.3	O.O.M.	30.9±4.7	23.9±3.2	26.1±1.0	63.7±0.9	47.3±0.3	57.4±0.4	7.6 (4.8)
GPR-GNN	78.8±1.3	64.2±4.0	85.1±0.7	85.0±1.0	92.6±0.3	58.5±0.8	O.O.M.	31.7±4.7	26.2±1.6	29.5±1.1	64.5±0.4	57.6±0.2	67.6±0.1	5.4 (3.7)
GAT	78.2±1.2	65.8±4.0	83.6±0.2	85.4±1.4	91.7±0.5	58.2±1.0	O.O.M.	39.1±4.1	28.6±0.6	26.4±0.4	60.5±0.8	O.O.M.	O.O.M.	7.5 (3.7)
SlimG	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1	1.9 (1.5)

Experiments - Speed



Experiments - Non-linearity

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec
w/ MLP-2	65.9±1.1	54.2±5.3	83.3±0.2	84.8±0.5	90.1±1.9	65.8±0.1	<u>85.7±0.0</u>	40.0±2.5	30.5±0.5	28.8±1.0	66.7±1.7	60.3±0.3	<u>76.5±0.1</u>
w/ MLP-3	66.1±2.0	50.3±3.6	80.9±0.9	85.2±0.8	90.0±0.8	63.0±0.1	84.9±0.9	38.5±5.5	30.9±0.7	28.9±1.2	65.1±0.6	60.3±0.2	76.4±0.2
w/ NL Trans.	70.7±2.3	57.5±5.1	81.0±0.4	71.4±10.	77.9±2.2	57.0±0.6	O.O.M.	<u>41.3±3.2</u>	30.0±1.6	27.6±2.4	61.8±1.6	<u>61.5±0.3</u>	75.7±0.5
SlimG	<u>77.8±1.1</u>	<u>67.1±2.3</u>	<u>84.6±0.5</u>	<u>86.3±0.7</u>	<u>91.8±0.5</u>	<u>66.3±0.3</u>	84.9±0.0	40.8±3.2	<u>31.1±0.7</u>	<u>30.9±0.6</u>	<u>68.2±0.6</u>	59.7±0.1	73.9±0.1

Experiments - Aggregation

k_{row}	k_{sym}	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec
$\{2, 4, 6\}$	$\{2, 3, 4\}$	79.4±1.1	66.0±4.4	84.4±0.4	85.9±0.5	91.3±0.5	67.9±0.2	84.8±2.1	41.0±3.9	31.4±0.6	29.8±0.6	68.2±0.6	60.7±0.1	76.6±0.2
$\{2, 4\}$	$\{2, 3\}$	78.9±0.9	66.9±2.5	84.0±0.5	86.2±0.7	91.5±0.5	67.4±0.1	73.9±23.	41.4±4.0	31.0±0.7	30.4±0.4	68.3±0.5	60.8±0.1	76.0±0.1
6	4	79.2±0.7	66.2±3.4	84.0±0.3	85.2±0.7	91.0±0.8	67.3±0.2	84.7±1.7	38.2±6.3	29.2±1.5	31.2±0.8	67.7±0.7	59.8±0.1	74.2±0.2
4	3	79.2±0.8	66.1±3.5	84.2±0.5	85.8±0.6	91.3±0.4	67.2±0.1	85.4±0.0	38.6±7.1	29.5±1.8	30.4±0.7	67.5±0.6	60.0±0.1	74.6±0.1
2 (ours)	2 (ours)	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1

Conclusion

- 준지도 노드 분류를 위한 간단하고 효과적인 모델인 SlimG를 제안함
- SOTA GNN들과의 벤치마킹을 통해 SlimG는 다양한 그래프에서 모두 우월한 성능을 내어 SlimG의 강건함, 일반화 능력을 보임
- 또한 SlimG는 우월한 성능을 내면서도 메모리와 처리 속도 측면에서도 좋은 성능을 가짐
- GnnExp를 통해 기존 GNN 모델들이 수행에 문제가 있는 부분을 밝혀냄
- Sanity Checks를 통해 각 GNN 모델의 강점과 단점을 파악해냄