Diffusion Improves Graph Learning

(J. Gasteiger et al. NeurIPS 2019)

Jihyeong Jung

Department of Industrial & Systems Engineering, KAIST

2022.11.03.

KSE801 Paper Presentation

Contents

Introduction

– Why Graph Diffusion Convolution?

- Message Passing Framework (Spatial methods)
  - Prevailing approach
  - Cons: Only pass messages b/w neighbors
  - Limiting messages' source to one-hop neighbors can be arbitrary
    - Real-world graphs are often **noisy** or defined **using an arbitrary threshold**
    - e.g. Spectral Clustering: defining similarity matrix required.
- Spectral-Based Methods
  - Capture complex properties via filtering spectral signals
  - Cons: outperformed by MPNN, cannot generalize to unseen graphs
- Goal of this Paper
  - Harmonize two approaches and combine their strengths

Introduction

- Contribution of the paper

- Proposing Graph Diffusion Convolution(GDC)
  - **Generate 'larger' adjacency** by graph diffusion, which is scalable and can be combined to any Graph-based models
  - General, Powerful, yet spatially localized
- Analyze spectral properties of GDC and Graph Diffusion
  - Graph Diffusion can be expressed as an Polynomial Filter in spectral GNN
  - Analyze GDC's effect on the spectrum of the graph
- Extensive experiments about GDC

Methodology

– Generalized Graph Diffusion

- Notation
  - $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: undirected graph, $N = |\mathcal{V}|$: # of nodes, $\boldsymbol{A} \in \mathbb{R}^{N \times N}$: adjacency matrix
- Definition: Generalized Graph Diffusion
  - Defined by diffusion matrix $\boldsymbol{S} = \sum_{k=0}^{\infty} \theta_k \boldsymbol{T}^k$, θ_k: weighting coefficients, $\boldsymbol{T}$: generalized transition matrix
  - Choice of coefficients and transition matrix should ensure that given series converges
  - In this paper, stricter conditions used
    - $\sum_{k=0}^{\infty} \theta_k = 1$, $\theta_k \in [0,1]$ and eigenvalues of $\boldsymbol{T}$ are bounded by some $\lambda_i \in [0,1]$
    - This strict condition does guarantee convergence of the series; so that definition is valid
  - Graph diffusion commonly requires $\boldsymbol{T}$ to be column/row-stochastic
- Transition Matrix
  - Examples: $\boldsymbol{T}_{rw} = \boldsymbol{A}\boldsymbol{D}^{-1}$ (Random Walk), $\boldsymbol{T}_{sym} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ (Symmetric)
  - Self-loop added version can be used; $\widetilde{\boldsymbol{T}}_{sym} = (w_{loop}\boldsymbol{I}_N + \boldsymbol{D})^{-1/2}(w_{loop}\boldsymbol{I}_N + \boldsymbol{A})(w_{loop}\boldsymbol{I}_N + \boldsymbol{D})^{-1/2}$

Methodology

- Generalized Graph Diffusion

- Special Cases
  - Personalized PageRank(PPR) & heat kernel can be expressed as a graph diffusion;
  - PPR: $\boldsymbol{T} = \boldsymbol{T}_{rw}$, $\theta_k^{PPR} = \alpha(1-\alpha)^k$, $\alpha \in (0,1) =$ teleport probability.
  - heat kernel: $\boldsymbol{T} = \boldsymbol{T}_{rw}$, $\theta_k^{HK} = e^{-t}\frac{t^k}{k!}$, $t =$ diffusion time
  - GCN: $\boldsymbol{T} = \widetilde{\boldsymbol{T}}_{sym}$, $\theta_1 = 1$, $\forall k \neq 1$: $\theta_k = 0$, uses $w_{loop} = 1$

- Weighting Coefficients
  - PPR/HK coefficients give closed-form solution of the series
  - Another transition matrix $\boldsymbol{T}$ can be used with PPR/HK coefficients since it still converges
  - Another graph-specific coefficients θ_k can be used
    - Coefficients learned by label propagation and node embedding models are possible
    - But using simple PPR/HK coefficients have shown better performance than those coefficients

Methodology

- Graph Diffusion Convolution

- Overview

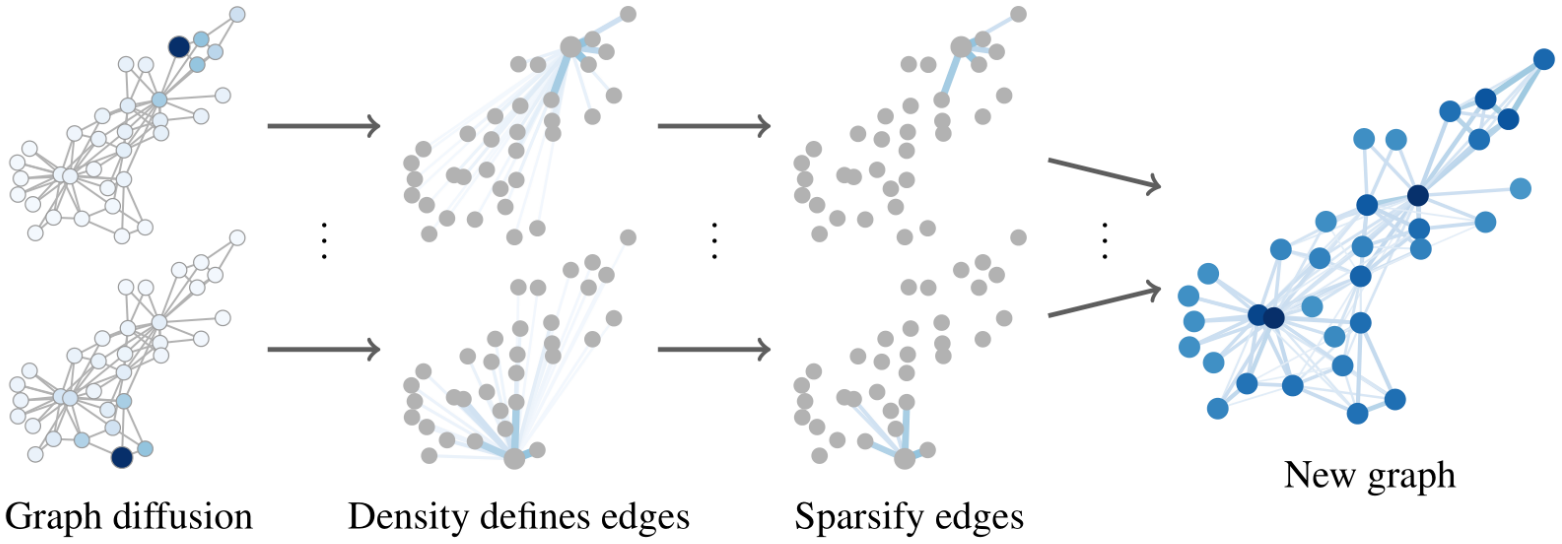


Figure 1: Illustration of graph diffusion convolution (GDC). We transform a graph $\boldsymbol{A}$ via graph diffusion and sparsification into a new graph $\tilde{\boldsymbol{S}}$ and run the given model on this graph instead.

GDC generates new graph via diffusion with sparsification

Methodology

– Graph Diffusion Convolution

- Intuition behind the GDC
  - Graph diffusion smooths out the neighborhood over the graph
  - Acting like denoising filter; improves graph learning since real-world graph's features/edges are often noisy
- GDC creates new graph based on given graph
  - GDC replaces $\boldsymbol{A}$ with sparsified version $\widetilde{\boldsymbol{S}}$ of the diffusion matrix $\boldsymbol{S}$
  - $\widetilde{\boldsymbol{S}}$: weighted & directed; **we use this matrix for the model we want to augment**
  - Obtained edge weight is beneficial but can be ignored when model require unweighted graph
  - Obtained graph can be made undirected by using $(\widetilde{\boldsymbol{S}} + \widetilde{\boldsymbol{S}}^T)/2$
  - With some modifications, GDC can be plugged into any graph-based models

Methodology

– Graph Diffusion Convolution

- Sparsification
  - Graph diffusion results Dense $\boldsymbol{S}$; even for finite sum due to 'Small-world property'
  - Two ways for sparsification suggested
    - 1) Top-k: use k entries with highest weight for each column
    - 2) Threshold ε: truncate entries smaller than the threshold
  - Though after sparsification, $\widetilde{\boldsymbol{S}}$ might be dense
    - But there are fast approximations for PPR and Heat Kernel, so that GDC only require $O(N)$ time.
  - Sparsification acts like spatial localization.
  - Sparsification leads slight improvement in model performance empirically
- After sparsification, $\boldsymbol{T}_{sym}^{\tilde{\boldsymbol{S}}} = \boldsymbol{D}_{\tilde{\boldsymbol{S}}}^{-1/2}\widetilde{\boldsymbol{S}}\boldsymbol{D}_{\tilde{\boldsymbol{S}}}^{-1/2}$ or $\boldsymbol{T}_{rw}^{\tilde{\boldsymbol{S}}} = \widetilde{\boldsymbol{S}}\boldsymbol{D}_{\tilde{\boldsymbol{S}}}^{-1}$ are calculated for application
- Limitation
  - GDC relies on homophily;

Spectral Analyses

– Graph Diffusion as a Polynomial Filter

- GDC can be analyzed in Spectral Domain
  - GDC is spatial-based
  - But, generalized graph diffusion can be expressed as a polynomial filter, and vice versa
- Spectral Filter $g(L)$
  - Used in Spectral GNN having form of $Z = \phi\big(g(L)\big)\psi(X)$
  - Scales given signal X's frequency component in the spectral domain
  - Usually, **many Spectral GNNs use polynomial filter to approximate arbitrary filters**
- Graph Diffusion as Polynomial Filter
  - We have relation b/w transition matrix and normalized graph Laplacian as; $\boldsymbol{L}_{rw} = \boldsymbol{I}_N - \boldsymbol{T}_{rw}$, $\boldsymbol{L}_{sym} = \boldsymbol{I}_N - \boldsymbol{T}_{sym}$
  - Finite sum of diffusion $\boldsymbol{S} = \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k \rightarrow \boldsymbol{S} = \sum_{j=0}^{J} \xi_j \boldsymbol{L}^j$ where $\xi_j = \sum_{k=j}^{K} \binom{k}{j} (-1)^k \theta_k$, $\theta_k = \sum_{j=k}^{J} \binom{j}{k} (-1)^j \xi_j$ by binomial expansion
  - So, Graph Diffusion <-> Spectral Polynomial Filter

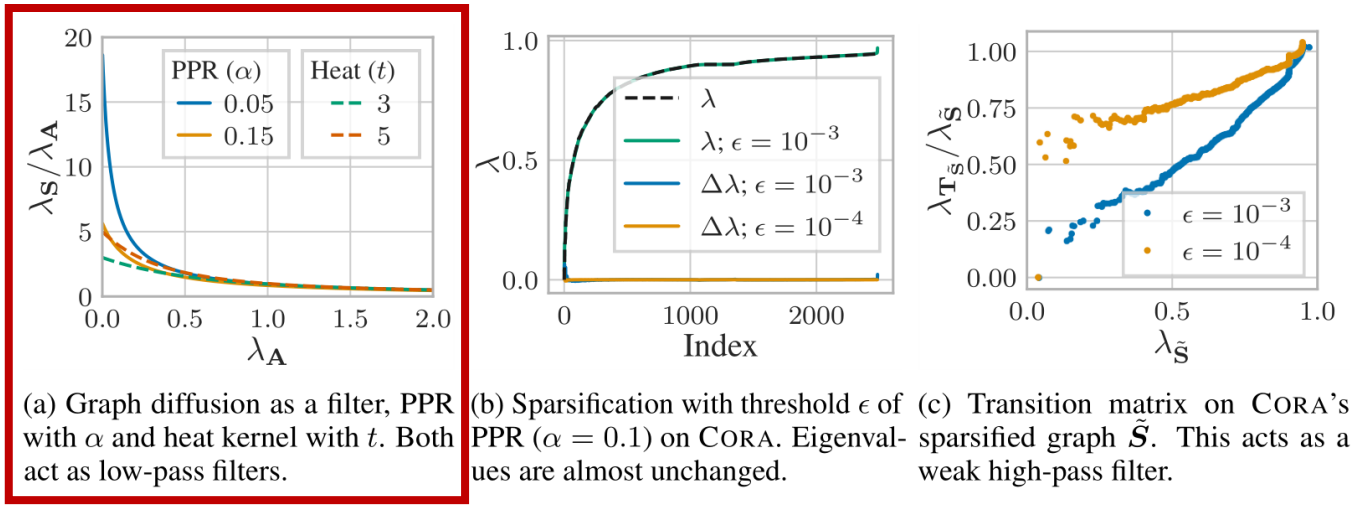
Spectral Analyses

– Spectral Properties of GDC

- GDC has 4 steps
  - 1) Calculate Transition matrix $\boldsymbol{T}$
  - 2) Summation to get $\boldsymbol{S}$
  - 3) Sparsification to get $\widetilde{\boldsymbol{S}}$
  - 4) Calculate transition matrix $\boldsymbol{T}^{\widetilde{\boldsymbol{S}}}$
- Transition matrix
  - Only changes which Laplacian($\boldsymbol{L}_{rw}$ or $\boldsymbol{L}_{sym}$) we use for analyze spectrum of the graph;
  - If self-loops are added, then eigenvectors are not preserved

Spectral Analyses

- Spectral Properties of GDC



(a) Graph diffusion as a filter, PPR with α and heat kernel with t. Both act as low-pass filters.

(b) Sparsification with threshold ϵ of PPR ($\alpha = 0.1$) on CORA. Eigenvalues are almost unchanged.

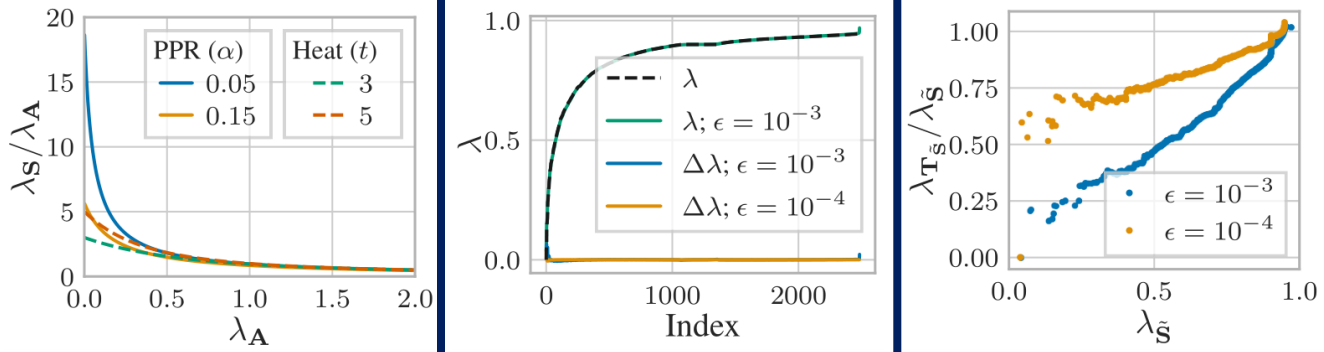
(c) Transition matrix on CORA's sparsified graph $\tilde{S}$. This acts as a weak high-pass filter.

Figure 2: Influence of different parts of GDC on the Laplacian's eigenvalues λ.

- Summation over $\boldsymbol{T}^k$
  - Has **no effect the eigenvectors** of the original matrix; since $\boldsymbol{T}^k\boldsymbol{v}_i = \lambda_i\boldsymbol{T}^{k-1}\boldsymbol{v}_i = \cdots = \lambda_i^k\boldsymbol{v}_i$ it only transforms eigenvalues as; $\widetilde{\lambda}_i = \sum_{k=0}^{\infty}\theta_k\lambda_i^k$
  - Eigenvalues of $\boldsymbol{T}$ are bounded by 1 → $\widetilde{\lambda}_i^{PPR} = \alpha/(1-(1-\alpha)\lambda_i)$ for PPR, $\widetilde{\lambda}_i^{HK} = \exp\big(t(\lambda_i - 1)\big)$
  - So that PPR, HK act as Low-Pass Filters
  - Low eigenvalues corresponding to large-scale structure are amplified, while high values of fine details are suppressed (also noise are suppressed)

Spectral Analyses

- Spectral Properties of GDC

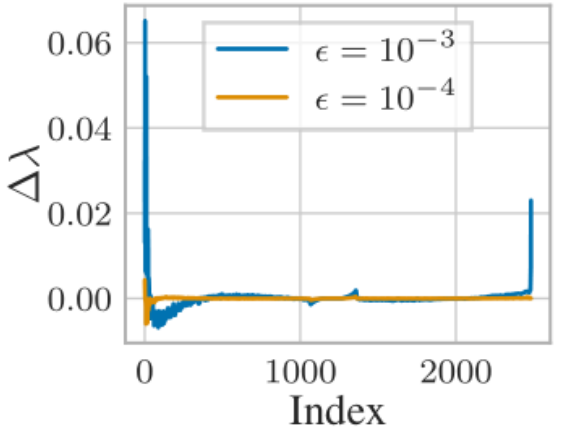


(a) Graph diffusion as a filter, PPR with α and heat kernel with t. Both act as low-pass filters.

(b) Sparsification with threshold ϵ of PPR ($\alpha = 0.1$) on CORA. Eigenvalues are almost unchanged.

(c) Transition matrix on CORA's sparsified graph $\tilde{S}$. This acts as a weak high-pass filter.

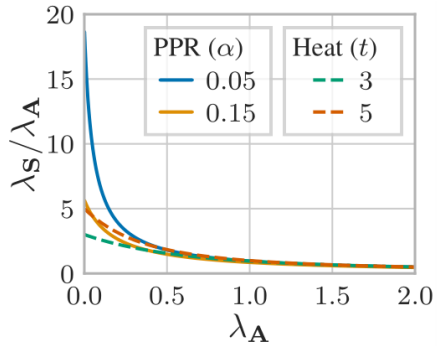
Figure 2: Influence of different parts of GDC on the Laplacian's eigenvalues λ.



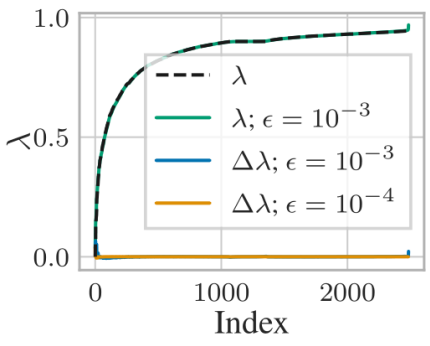
- Sparsification of $\boldsymbol{S}$
  - Changes both eigenvectors & eigenvalues
  - But, typical thresholds for sparsification has only little effect on the eigenvalues
  - Sparsification affects mostly on Highest & Lowest eigenvalues
  - Both eigenvalues are not helpful for graph learning

Spectral Analyses

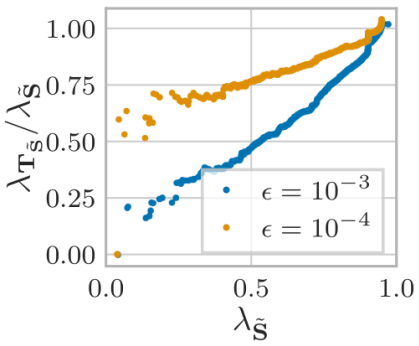
- Spectral Properties of GDC



(a) Graph diffusion as a filter, PPR with α and heat kernel with t. Both act as low-pass filters.



(b) Sparsification with threshold ϵ of PPR ($\alpha = 0.1$) on CORA. Eigenvalues are almost unchanged.



(c) Transition matrix on CORA's sparsified graph $\tilde{S}$. This acts as a weak high-pass filter.

Figure 2: Influence of different parts of GDC on the Laplacian's eigenvalues λ.

- Transition matrix $T^{\tilde{S}}$
  - Empirically, **acts like a weak high-pass filter**
  - May seem counter-productive
  - Main purpose: ensuring sparsification does not cause nodes to be treated differently by losing adjacent edges
  - So, filtering is just a side-effect.
  - Opinion: limited explanation on this transition matrix over diffusion matrix; it was hard to understand, also caused mis-explanation

Spectral Analyses

- Strengths of GDC over Spectral-based models

- GDC's properties
  - GDC is spatial-based, so that does not share limitations of Spectral-based models
  - Does not compute expensive eigen-decomposition
  - Preserve locality on the graph
  - Not limited to a single graph
    - Same coefficients θ_k can be used over different graphs
    - Choice of θ_k depends on the type of graph; does not change significantly when using similar graphs
    - Hyperparameters of PPR and HK is quite insensitive to both the graph and the model

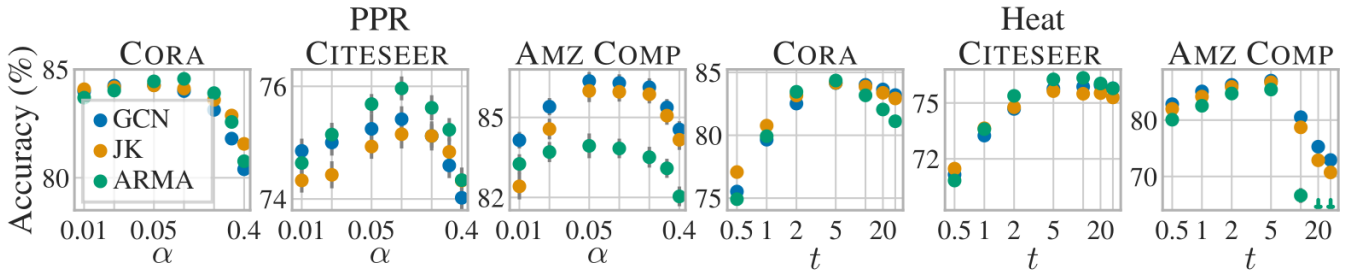


Figure 8: Accuracy achieved by using GDC with varying hyperparameters of PPR (α) and the heat kernel (t). Optimal values fall within a narrow range that is consistent across datasets and models.

Experiments

– Settings

Table 1: Dataset statistics.

Dataset	Type	Classes	Features	Nodes	Edges	Label rate
CORA	Citation	7	1433	2485	5069	0.056
CITESEER	Citation	6	3703	2120	3679	0.057
PUBMED	Citation	3	500	19 717	44 324	0.003
COAUTHOR CS	Co-author	15	6805	18 333	81 894	0.016
AMZ COMP	Co-purchase	10	767	13 381	245 778	0.015
AMZ PHOTOS	Co-purchase	8	745	7487	119 043	0.021

- GDC settings
  - Transition matrix for diffusion: $\widetilde{\boldsymbol{T}}_{sym} = (\boldsymbol{I}_N + \boldsymbol{D})^{-1/2}(\boldsymbol{I}_N + \boldsymbol{A})(\boldsymbol{I}_N + \boldsymbol{D})^{-1/2}$
  - Transition matrix after diffusion: $\boldsymbol{T}_{rw}^{\widetilde{\boldsymbol{S}}} = \widetilde{\boldsymbol{S}}\boldsymbol{D}_{\widetilde{\boldsymbol{S}}}^{-1}$ on $\widetilde{\boldsymbol{S}}$
  - Weight coefficients θ_k: PPR and Heat Kernel
  - Sparsification: using ε-threshold or top-k
- Datasets and Models
  - Dataset
    - Total 6 datasets
    - Citation graphs Citeseer, Cora, PubMed
    - Co-author graph Coauthor CS
    - Co-purchase graph Amazon Computers, Amazon Photo
  - Models
    - Total 9 Models
    - Supervised models GCN, GAT, GIN, JK-Net, ARMA
    - Unsupervised models DCSBM, Spectral Clustering, DeepWalk, DGI

Experiments

- Results: GDC's effect on Model Performance

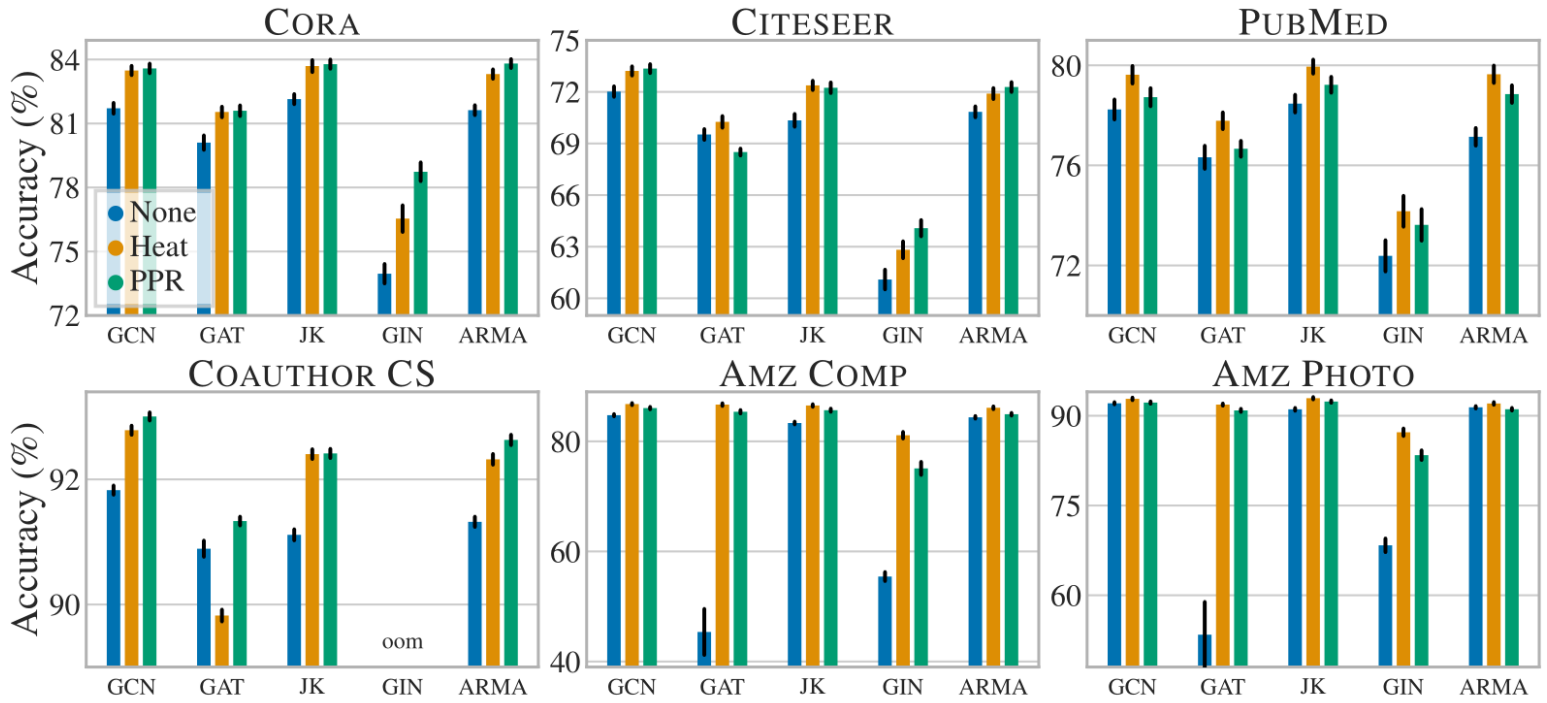


Figure 3: Node classification accuracy of GNNs with and without GDC. GDC consistently improves accuracy across models and datasets. It is able to fix models whose accuracy otherwise breaks down.

Application of GDC leads to Performance Improvement for Supervised models

Experiments

- Results: GDC's effect on Model Performance

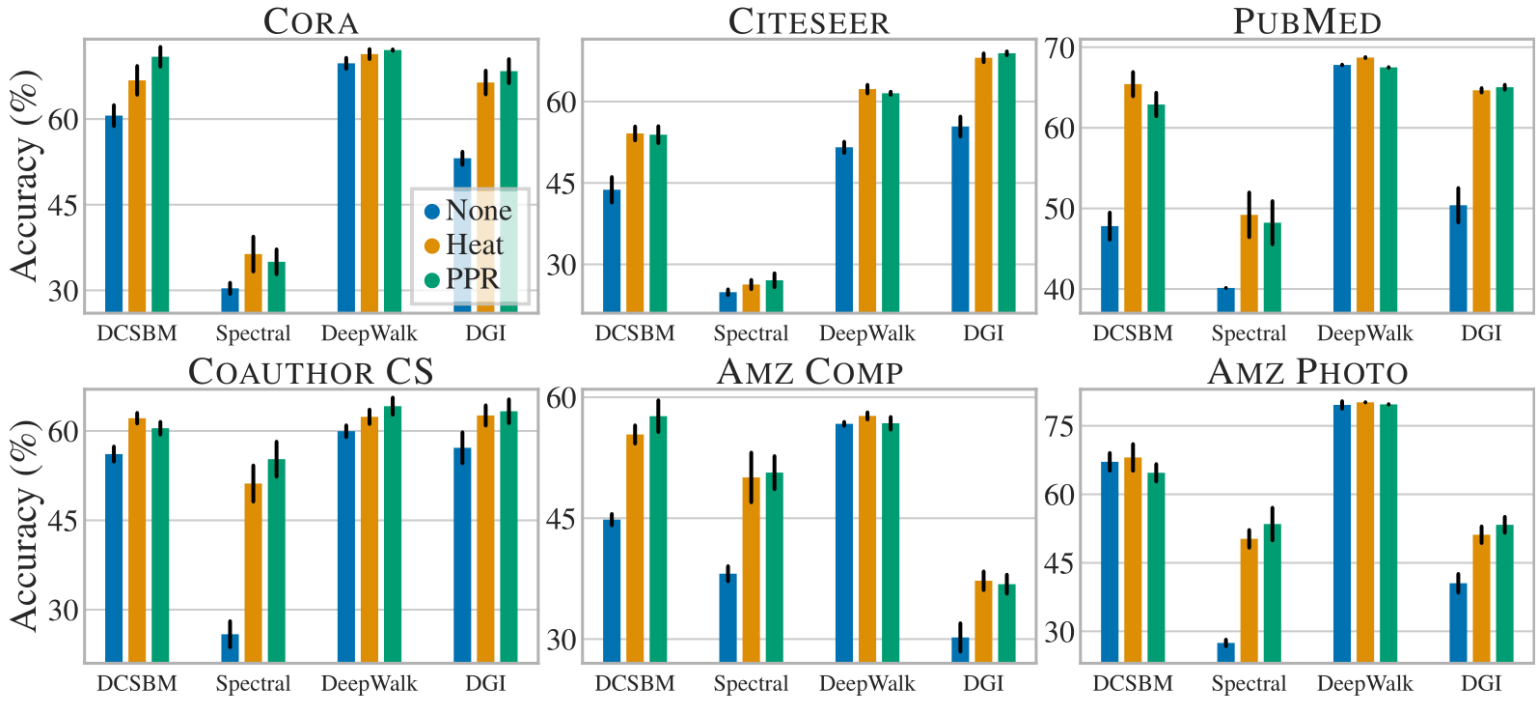


Figure 4: Clustering accuracy with and without GDC. GDC consistently improves the accuracy across a diverse set of models and datasets.

Application of GDC leads to Performance Improvement also for Unsupervised Models

Experiments

- Results: Does GDC make graph dense?

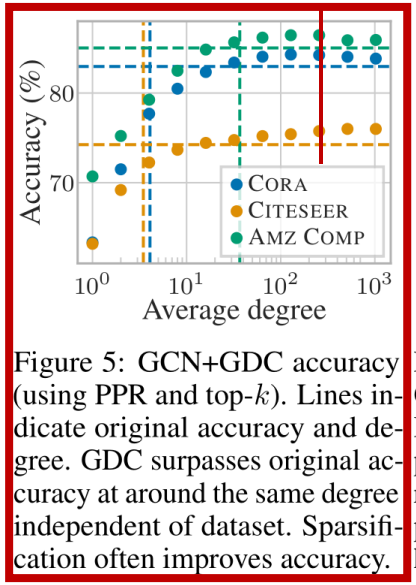


Figure 5: GCN+GDC accuracy (using PPR and top-k). Lines indicate original accuracy and degree. GDC surpasses original accuracy at around the same degree independent of dataset. Sparsification often improves accuracy.

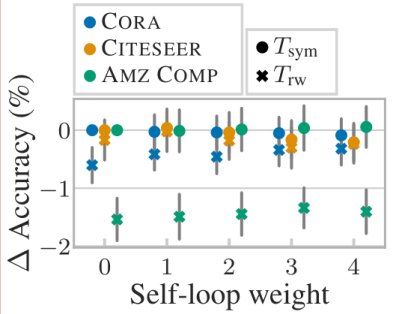


Figure 6: Difference in GCN+GDC accuracy (using PPR and top-k, percentage points) compared to the symmetric $\boldsymbol{T}_{\text{sym}}$ without self-loops. $\boldsymbol{T}_{\text{rw}}$ performs worse and self-loops have no significant effect.

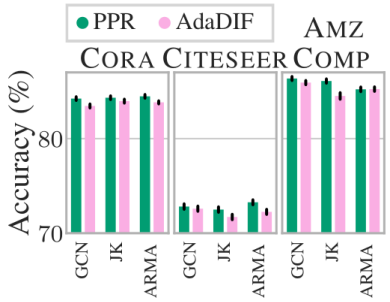


Figure 7: Accuracy of GDC with coefficients θ_k defined by PPR and learned by AdaDIF. Simple PPR coefficients consistently perform better than those obtained by AdaDIF, even with optimized regularization.

- GDC's effect on Graph Density
  - GDC requires almost same avg. degree to be improved, independent of the Dataset and Avg. degree
  - → Sparsification Hyperparameter can be obtained from that fixed avg. degree
  - Graphs are usually become denser, but Optimal avg. degree where the performance starts to decrease exists
  - → Sparsification improves model performance, not only benefits computations

Experiments

- Results: About choosing Transition Matrix $\boldsymbol{T}$ & Self-loop weight

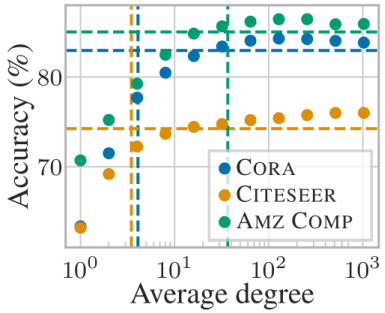


Figure 5: GCN+GDC accuracy (using PPR and top-k). Lines indicate original accuracy and degree. GDC surpasses original accuracy at around the same degree independent of dataset. Sparsification often improves accuracy.

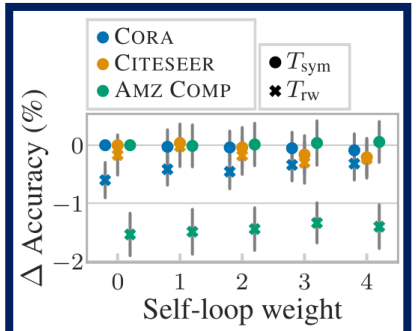


Figure 6: Difference in GCN+GDC accuracy (using PPR and top-k, percentage points) compared to the symmetric $\boldsymbol{T}_{\text{sym}}$ without self-loops. $\boldsymbol{T}_{\text{rw}}$ performs worse and self-loops have no significant effect.

Figure 7: Accuracy of GDC with coefficients θ_k defined by PPR and learned by AdaDIF. Simple PPR coefficients consistently perform better than those obtained by AdaDIF, even with optimized regularization.

- Choosing Transition Matrix for GDC & self-loop weights
  - Symmetric $\boldsymbol{T}_{sym}$ outperforms Random Walk transition matrix $\boldsymbol{T}_{rw}$
  - GCN is usually insensitive to self-loop weight; But, other models like GAT performed better with Self-loops

Experiments

- Results: Choosing weighting coefficients θ_k; Fixed or Learnable

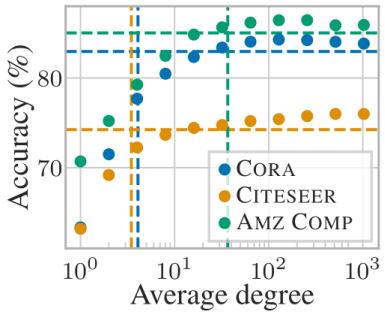


Figure 5: GCN+GDC accuracy (using PPR and top-k). Lines indicate original accuracy and degree. GDC surpasses original accuracy at around the same degree independent of dataset. Sparsification often improves accuracy.

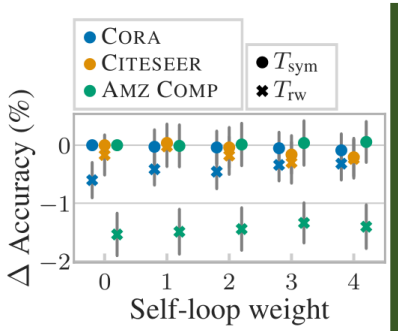


Figure 6: Difference in GCN+GDC accuracy (using PPR and top-k, percentage points) compared to the symmetric $\boldsymbol{T}_{\text{sym}}$ without self-loops. $\boldsymbol{T}_{\text{rw}}$ performs worse and self-loops have no significant effect.

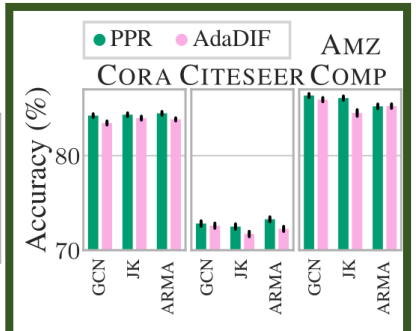


Figure 7: Accuracy of GDC with coefficients θ_k defined by PPR and learned by AdaDIF. Simple PPR coefficients consistently perform better than those obtained by AdaDIF, even with optimized regularization.

- Choosing weighting coefficients θ_k
  - Coefficients learned by other models collapse to minimal neighborhood; $\theta_0 \approx 1$ or $\theta_1 \approx 1$ and $\theta_k = 0$
  - These situations are happened due to those models' training loss decreases when neighborhood shrinks
  - Regularization might be helpful, but it requires lots of hand-tuning
  - Moreover, fixed coefficients from PPR and Heat Kernel perform better

Experiments

- Results: Which nodes benefit from GDC?

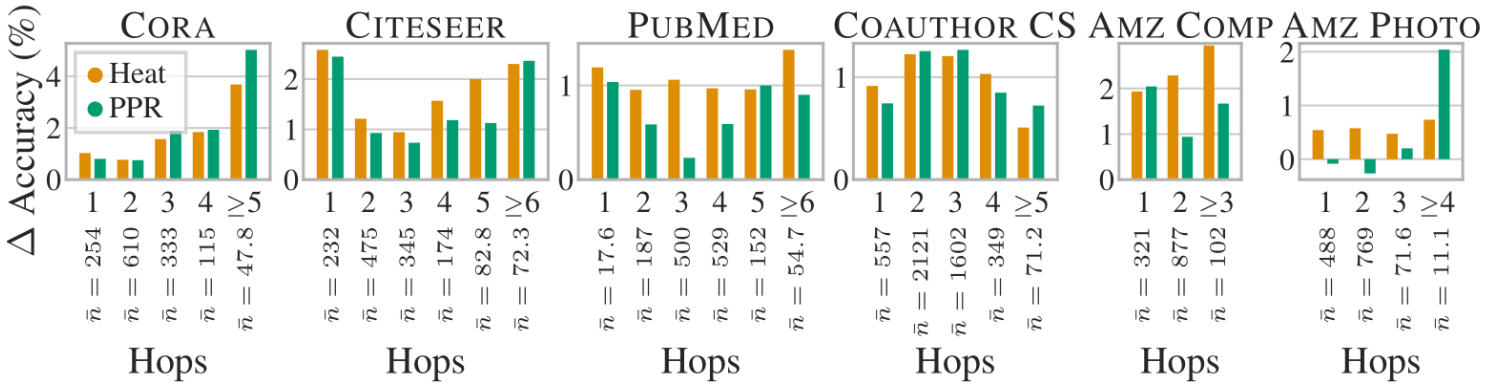


Figure 10: Improvement (percentage points) in GCN accuracy by adding GDC depending on distance (number of hops) from the training set. Nodes further away tend to benefit more from GDC.

- Nodes benefit from GDC
  - There's no correlation of improvement with common node properties
  - However, nodes far away from the training set tend to benefit from GDC
  - Since GDC smoothing out the neighborhood, it has effect of increasing the Model's range
  - So that nodes far from the training set can influence the training, and can benefit from the improved model

Conclusion

- Comments

- Summaries
  - GDC is kind of Pre-processing, smoothing out the neighborhood so that can improve model's performance
  - GDC can be interpreted as a Spectral Polynomial Filter, and have some spectral properties

- Pros
  - GDC is simple but powerful framework, that can improve performance
  - GDC can be plugged into any graph-based models & algorithms
  - Spatial-based method that has some of spectral-based model's properties
  - Extensive experiments to support GDC's strength and the effect

- Cons
  - GDC cannot be used directly for Heterophily dataset
  - Explanation about **why** Transition Matrix over diffusion matrix $\tilde{\boldsymbol{T}}$ are calculated and **where** it will be used are omitted

Research Idea

- Spectral to Spatial: Polynomial Filters to Graph Diffusion

- Polynomial Filters <-> Graph Diffusion
  - Based on the formula $\xi_j = \sum_{k=j}^{K} \binom{k}{j} (-1)^k \theta_k$, $\theta_k = \sum_{j=k}^{J} \binom{j}{k} (-1)^j \xi_j$

- How about utilizing suggested other polynomial filters?
  - As presented in the paper, we can replace fixed GDC's θ_k into learnable one
  - Also, we can transform several polynomial filters into graph diffusion via above formula
  - Expressive enough polynomial filters might lead to better Graph Diffusion

Table 5. The filter form of spectral GNNs.

MODEL	g	HYPERPARAMS	LEARNABLE	PFME
SGC (Wu et al., 2019)	$(1-\lambda)^K$	α, K		$\times$
APPNP (Klicpera et al., 2019a)	$\sum_{k=0}^{K} \frac{\alpha^k}{1-\alpha}(1-\lambda)^k$	α, K		$\times$
GNN-LF (Zhu et al., 2021)	$\frac{1-(1-\mu)(1-\lambda)}{1-(2-\mu+\frac{1}{\alpha})(1-\lambda)}$	α, μ		$\times$
GNN-HF (Zhu et al., 2021)	$\frac{1+\beta(1-\lambda)}{1-(1-\beta-\frac{1}{\alpha})(1-\lambda)}$	a, b		$\times$
CHEBYNET (Defferrard et al., 2016)	$\sum_{k=0}^{K} \alpha_k \cos(k \arccos(1-\lambda))$	K	α_k, K	$\surd$
GPRGNN (Chien et al., 2021)	$\sum_{k=0}^{K} \alpha_k (1-\lambda)^k$	K	α_k	$\surd$
ARMA (Bianchi et al., 2021)	$\sum_{k=0}^{K} \frac{b_k}{1-a_k(1-\lambda)}$	K	a_k, b_k	$\surd$
BERNNET (He et al., 2021)	$\sum_{k=0}^{K} \alpha_k \binom{K}{k} (1-\frac{\lambda}{2})^{K-k} (\frac{\lambda}{2})^k$	K	α_k	$\surd$
JACOBICONV (OUR MODEL)	$\sum_{k=0}^{K} \alpha_k \sum_{s=0}^{k} \frac{(k+a)!(k+b)!(-\lambda)^{k-s}(2-\lambda)^s}{2^k s!(k+a-s)!(b+s)!(k-s)!}$	K, a, b	α_k	$\surd$

Thank you!

Reference

- Xiyuan Wang and Muhan Zhang. *How Powerful are Spectral Graph Neural Networks*. ICML, 2022.
- *Eigenvalues and eigenvectors*. Wikipedia, 2022.
- *Laplacian Matrix*. Wikipedia, 2022.