AdaGNN: Graph Neural Networks with Adaptive Frequency Response Filter

Yushun Dong[1], Kaize Ding[2], Brian Jalaian[3], Shuiwang Ji[4], Jundong Li[1]

[1]University of Virginia, [2]Arizona State University, [3]Army Research Laboratory, [4]Texas A&M University

09/21/2021

Overview

Background Introduction

Previous Works

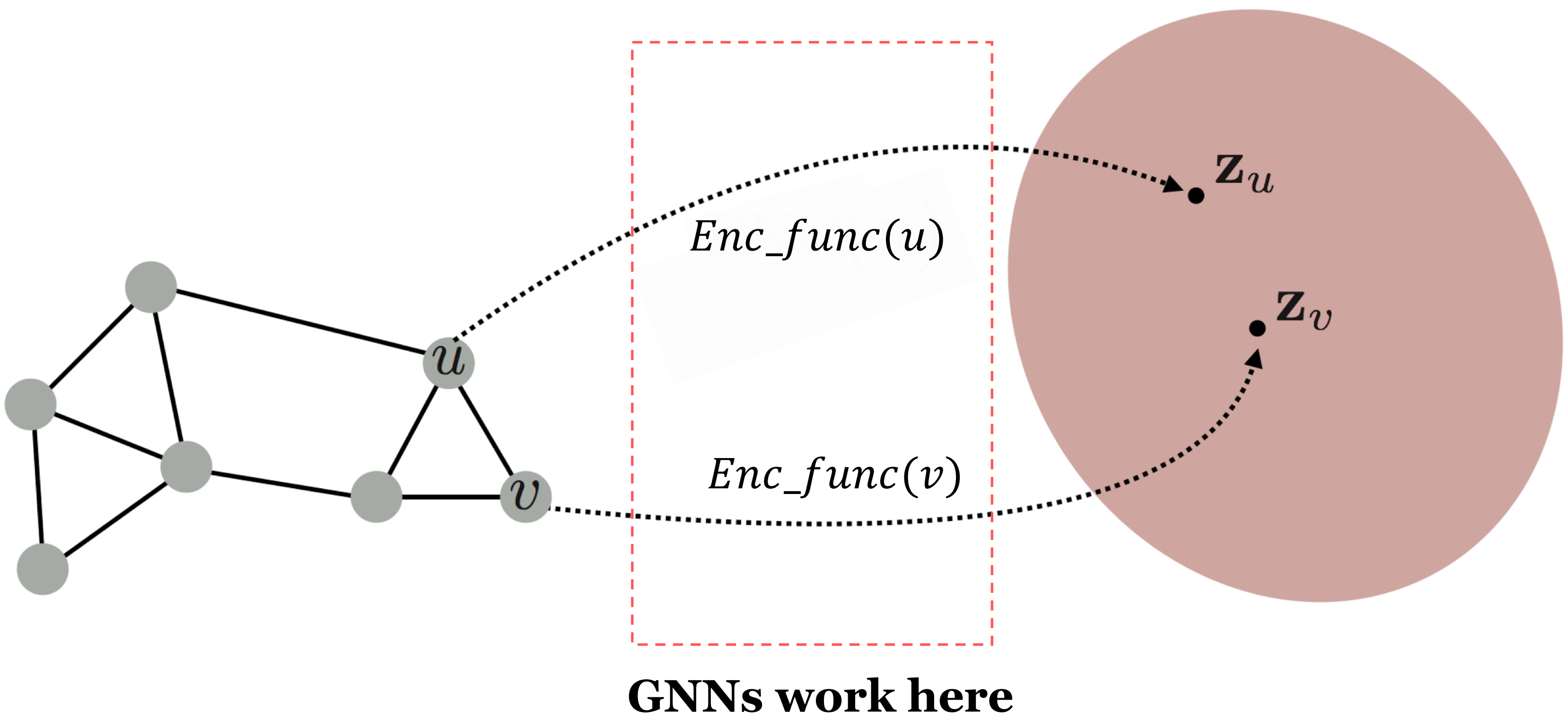
Existing Problems & Challenges

Our Solutions

Experiments & Conclusion

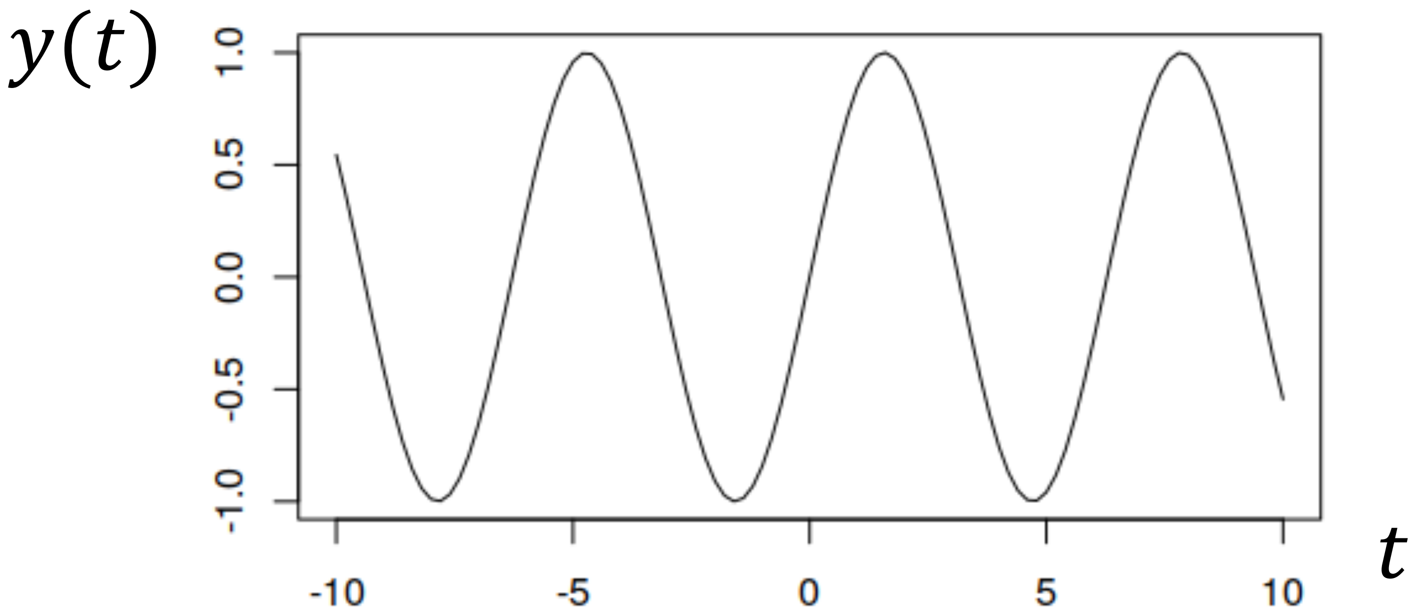
Future Works

Goal of Graph Neural Networks (GNNs): to encode nodes so that similarity in the embedding space (e.g., dot product) approximates similarity in the original network.



Background Introduction: Frequency in Graphs

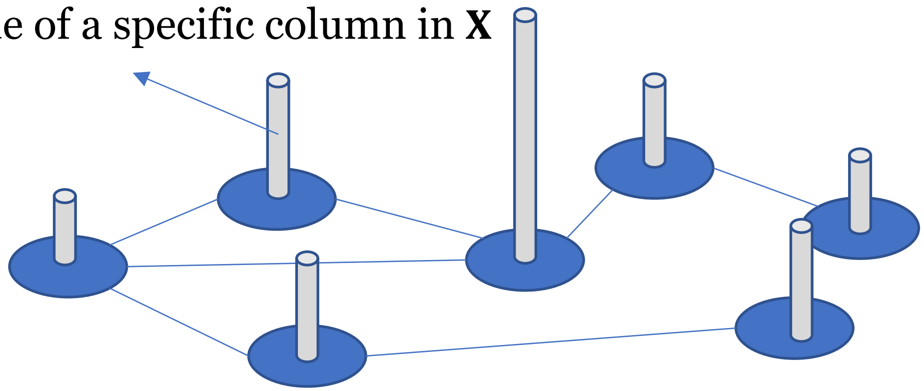
Traditionally, frequency is defined as the number of occurrences of a repeating event per unit of time*, and a basic unit of frequency is Hertz.



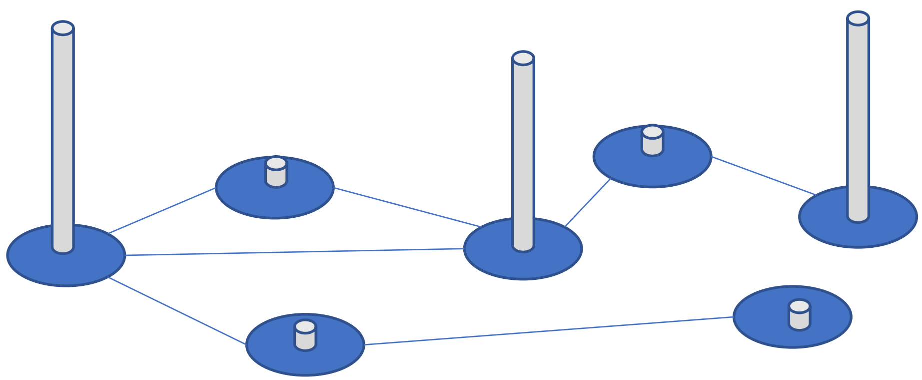
* https://en.wikipedia.org/wiki/Frequency

Background Introduction: Frequency in Graphs

In graphs, we generalize the notion of frequency in the spatial domain to measure how fast a signal changes w.r.t. its graph structure.

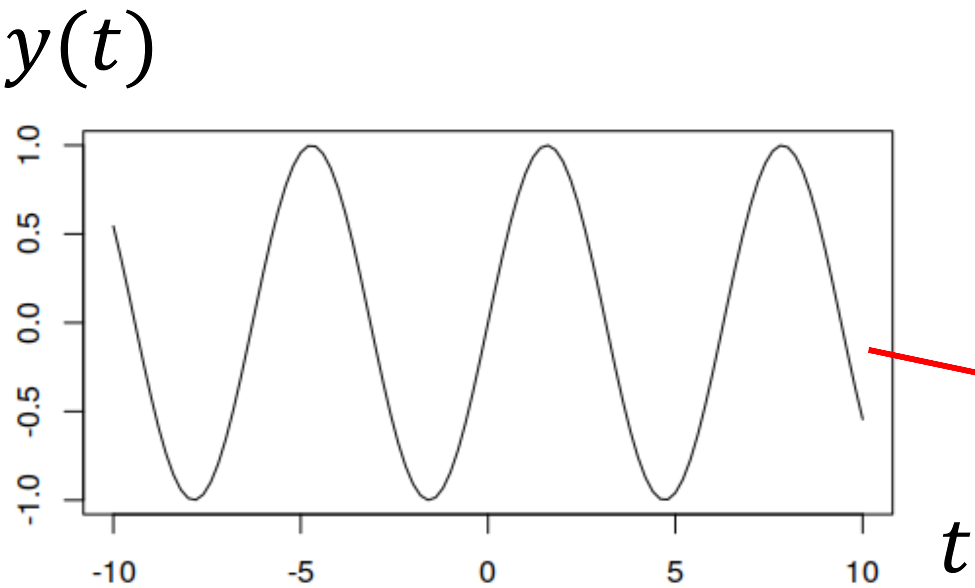


Low frequency: the signal changes slowly across edges.

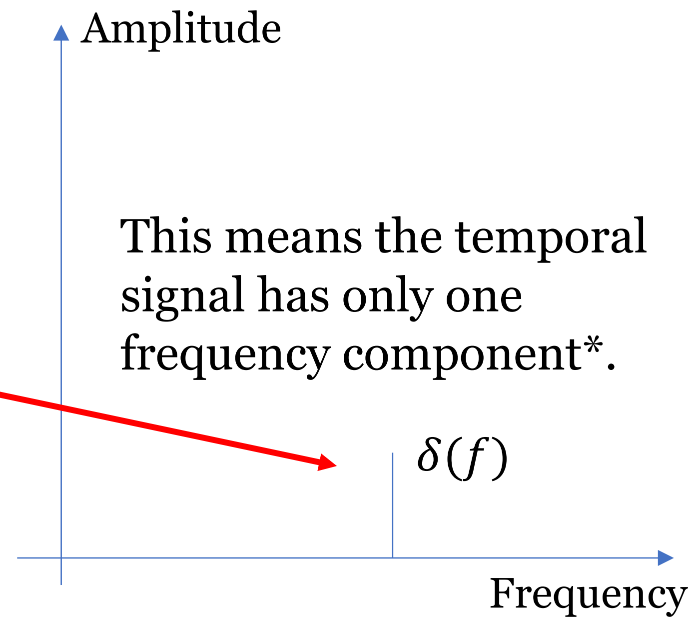


High frequency: the signal changes fast across edges.

For time series signals, we have frequency basis. Cosine function is a commonly utilized basis for time series signal. For example, it is utilized as one of the basis of Fourier Transform.



Time domain



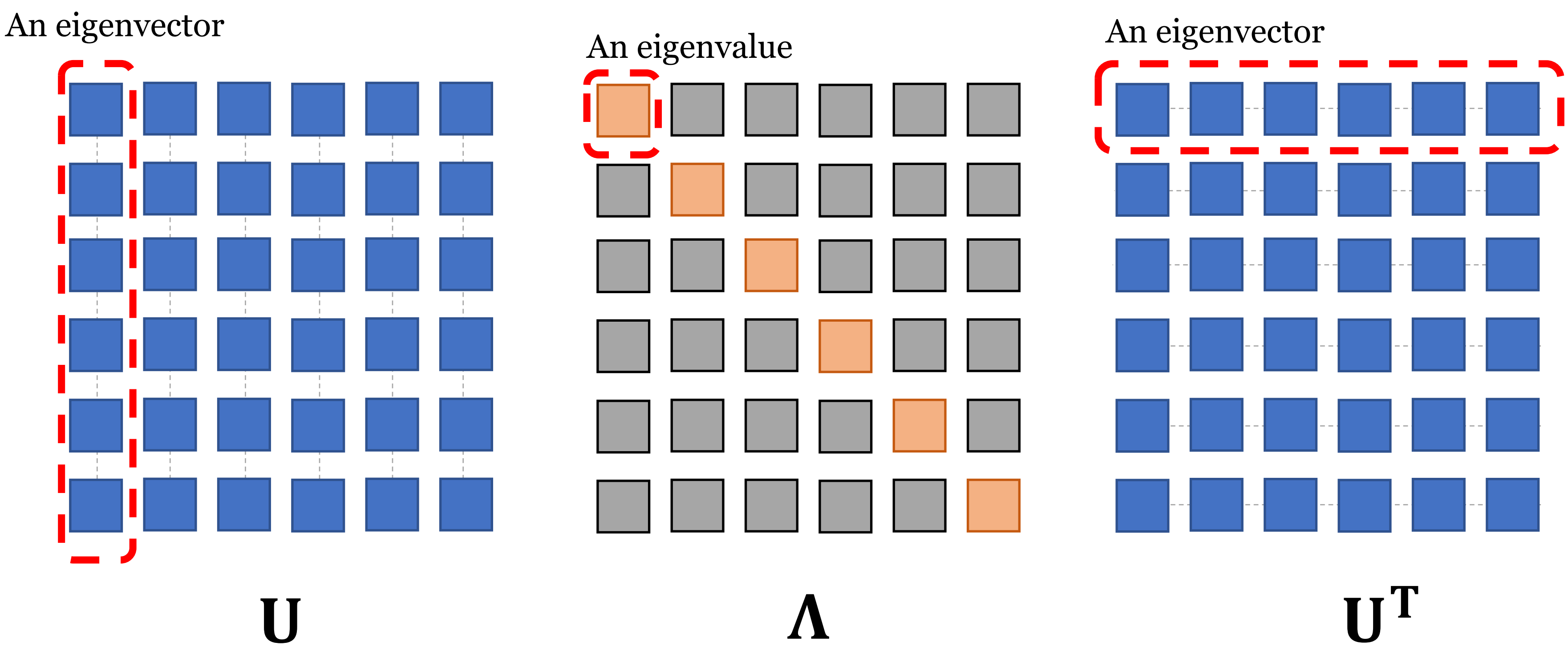
Frequency domain

*Here we only show the positive half axis in the frequency domain.

Background Introduction: Frequency in Graphs

In graphs, we utilize the eigenvectors of graph Laplacian as the basis of different frequencies*.

$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathbf{T}}$$



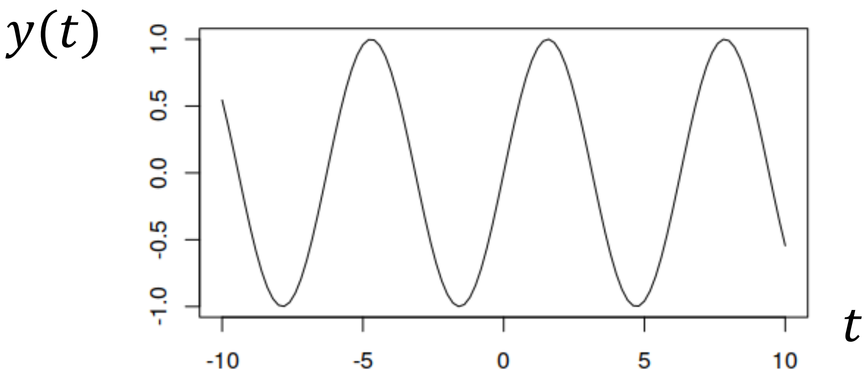
*Here the frequency notion is defined based on graph Laplacian. Similar notion can also be defined based on adjacency matrix, but larger eigenvalues corresponds to lower frequencies.

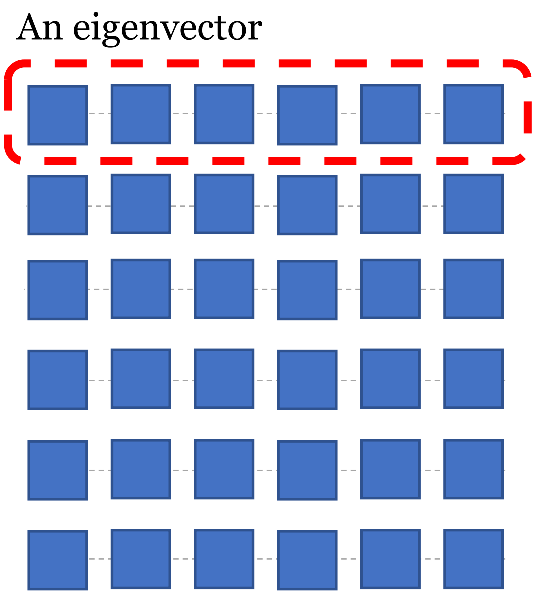
Background Introduction: Frequency in Graphs

In graphs, we utilize the eigenvectors of graph Laplacian as the basis of different frequencies.

$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}$$

We can intuitively understand the functionality of the graph Laplacian eigenvectors as cosine functions.

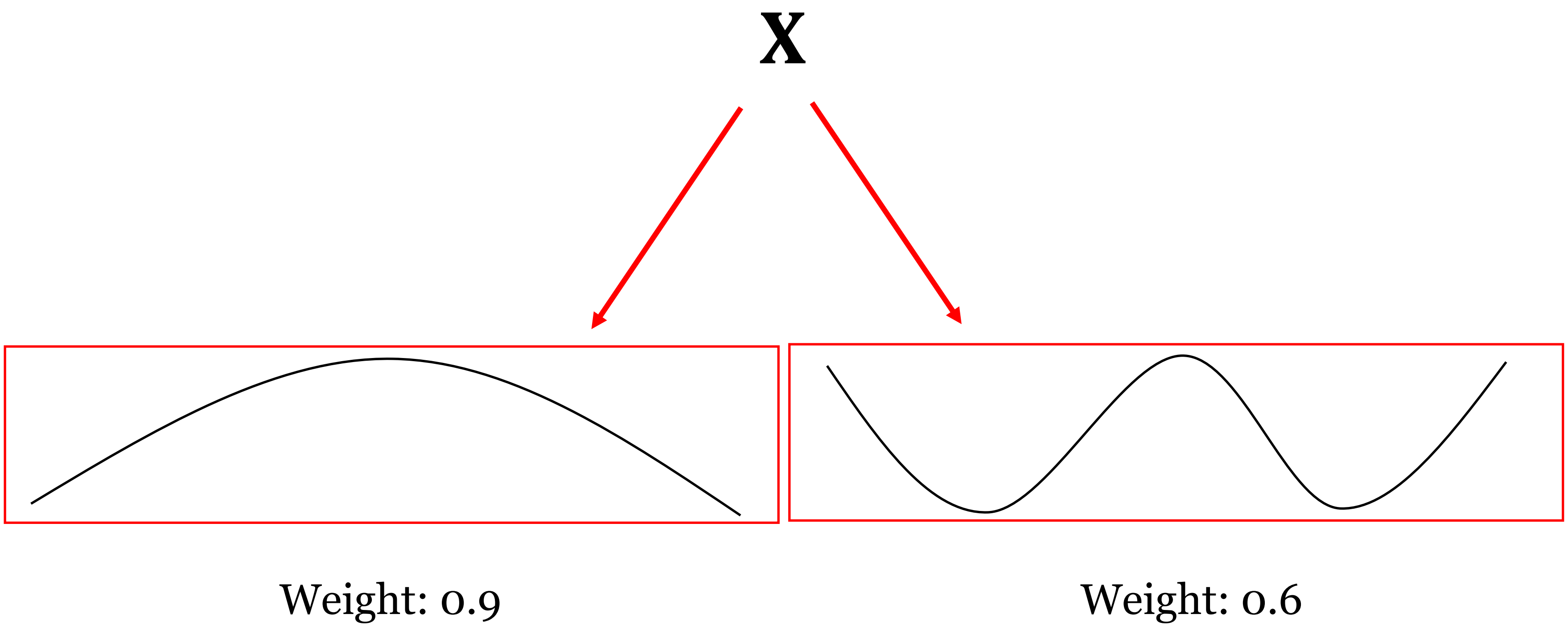




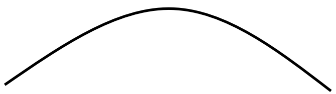
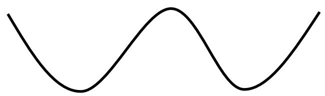
The corresponding eigenvalue is the frequency.

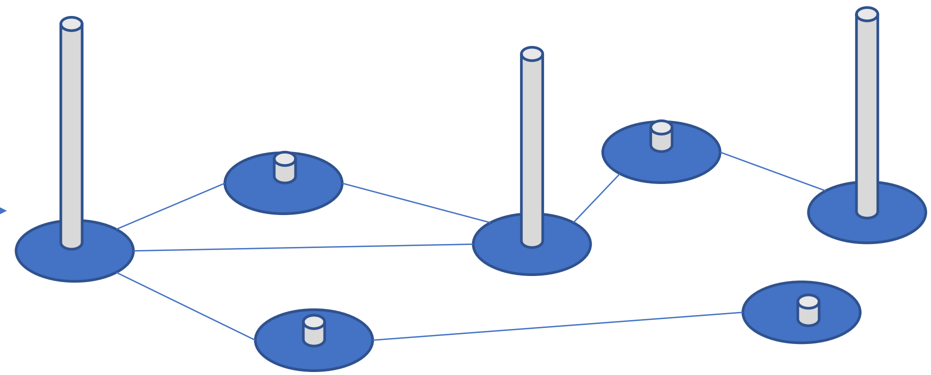
Background Introduction: Graph (low-pass) Filtering

By projecting **X** on different eigenvectors:



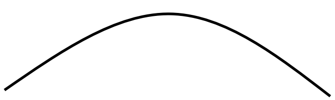
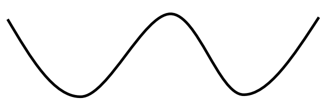
Background Introduction: Graph (low-pass) Filtering

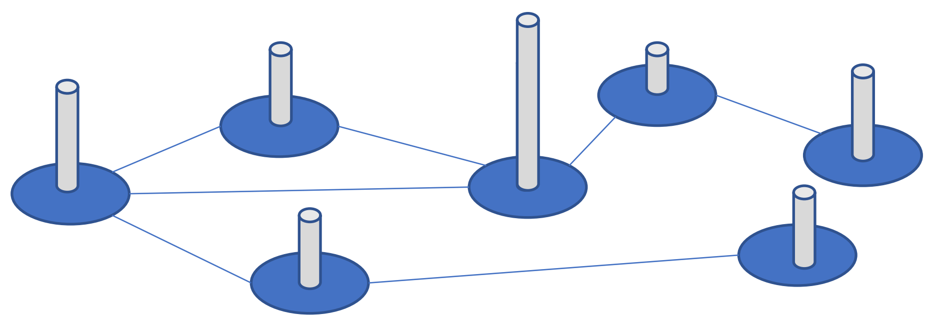
0.9	0.6
	



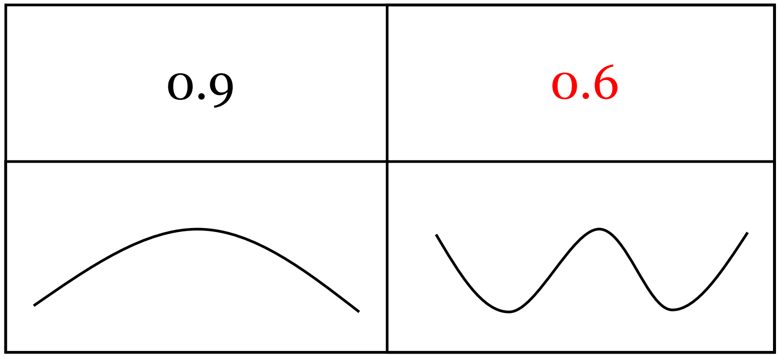
Eigenvectors (of graph Laplacian) are regarded as signal basis.

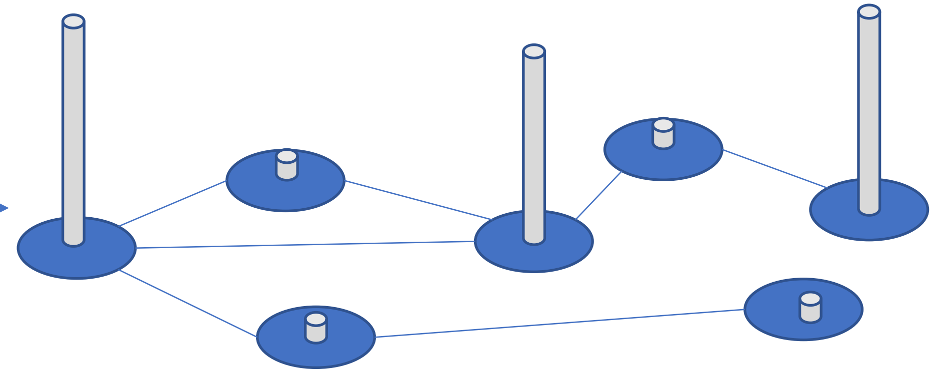
Graph Filter

0.9	0.2
	



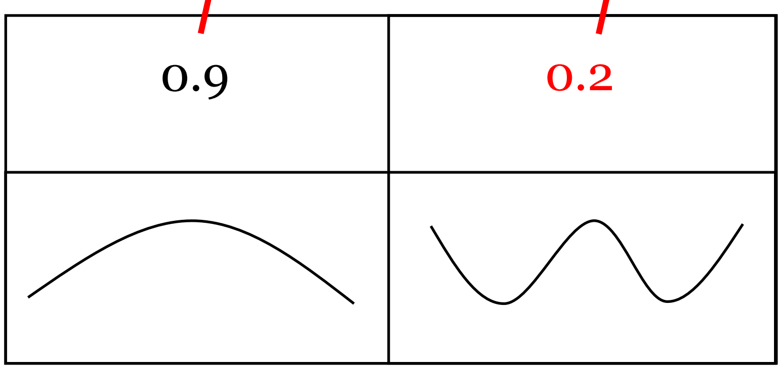
Background Introduction: Graph (low-pass) Filtering

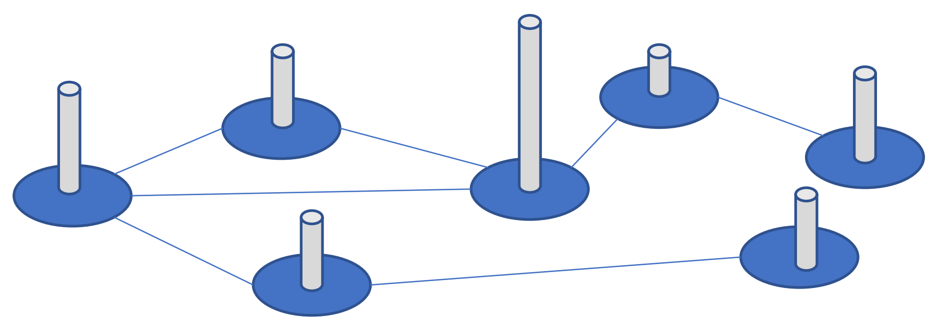




Graph Filter

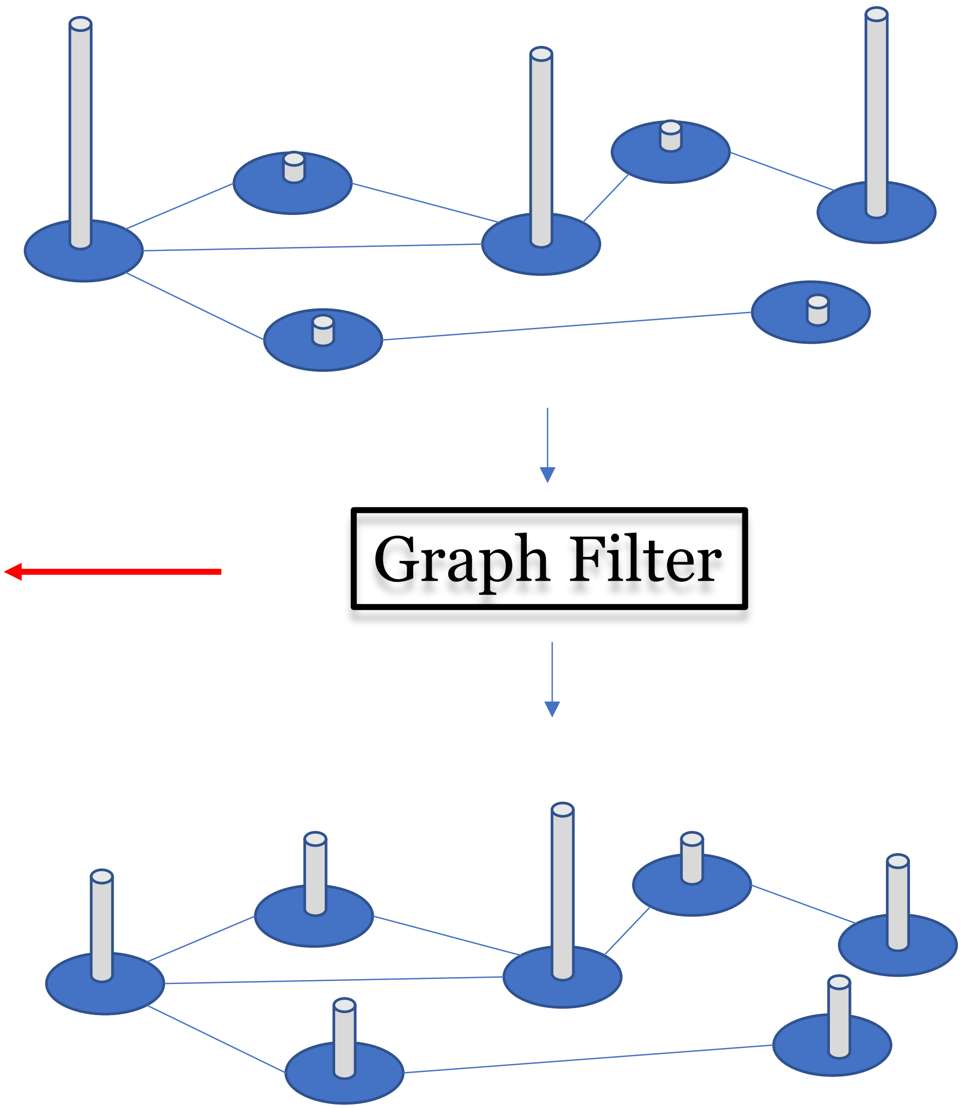
Re-weighted components **X** projected on each basis.





Background Introduction: Graph (low-pass) Filtering

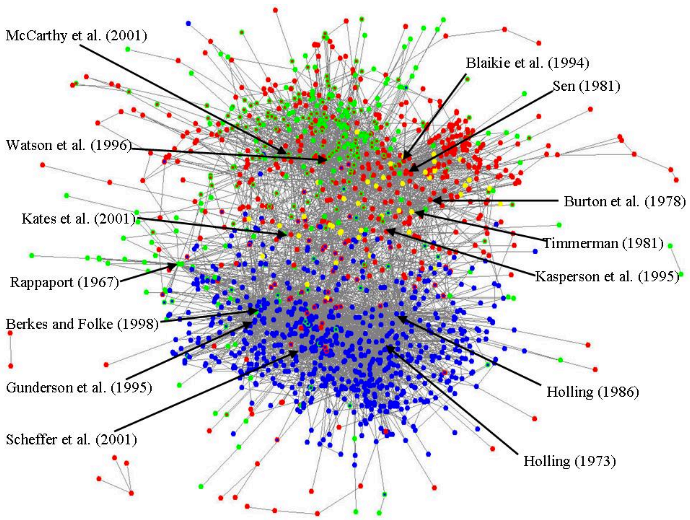
Graph (low-pass) filtering is a process of reducing the weight on eigenvector basis corresponding to large eigenvalues (of the graph Laplacian matrix) in the graph signal (i.e., **X** given **A**).



Background Introduction: Graph (low-pass) Filtering

Why we need to do this?

By filtering out "high-frequency" information (i.e., signals with high variances across the graph), the neighbor nodes are made to be similar. This helps us capture the dependencies between linked nodes.



Background Introduction: Graph (low-pass) Filtering

Why we need to do this?

By filtering out ”high-frequency” information (i.e., signals with high variances across the graph), the neighbor nodes are made to be similar. This helps us capture the dependencies between linked nodes.

Nevertheless, is low frequency all what we need?

Background Introduction: Graph (low-pass) Filtering

Why we need to do this?

By filtering out ”high-frequency” information (i.e., signals with high variances across the graph), the neighbor nodes are made to be similar. This helps us capture the dependencies between linked nodes.

In assortative networks, similar nodes tend to link together; however, in disassortative networks, different nodes tend to link together.

Background Introduction: Graph (low-pass) Filtering

Why we need to do this?

By filtering out ”high-frequency” information (i.e., signals with high variances across the graph), the neighbor nodes are made to be similar. This helps us capture the dependencies between linked nodes.

In assortative networks, similar nodes tend to link together; however, in disassortative networks, different nodes tend to link together.

This indicates that **only preserving low-frequency components cannot fully capture all useful information** [Bo et al. 2021].

Overview

- Background Introduction
- Previous Works
- Existing Problems & Challenges
- Our Solutions
- Experiments & Conclusion
- Future Works

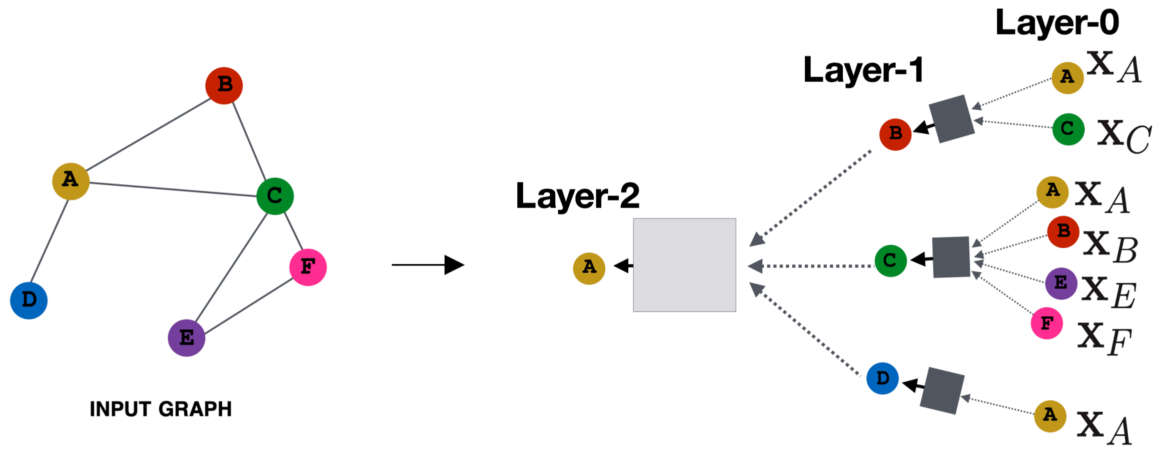
Previous Works: Spatial GNNs

Spatial GNNs: focus on information aggregation between nodes in the spatial domain;

Advantages: explainable and flexible;

Disadvantages: limited supporting theoretical basis; more of an empirical method;

Representative works: Graph Attention Network [Petar et al. 2017], GraphSAGE [Hamilton et al. 2017], etc.



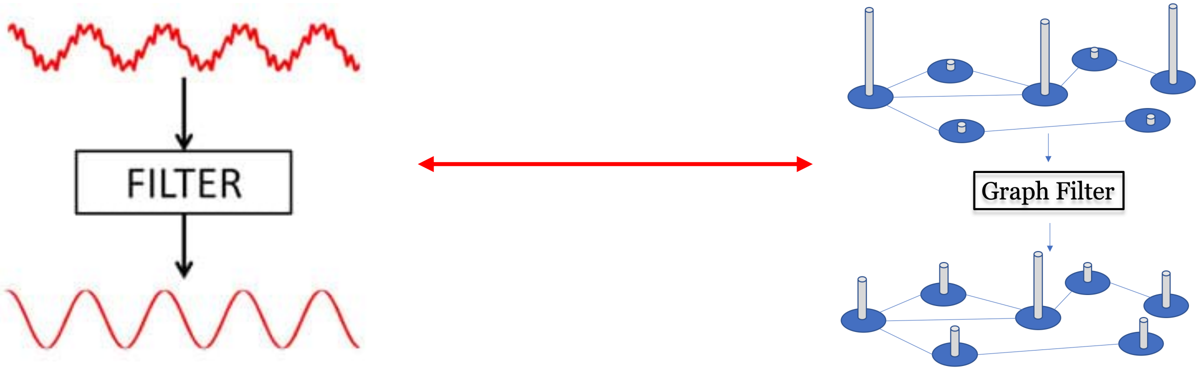
Previous Works: Spectral GNNs

Spectral GNNs: treat graph data as a whole and do signal filtering in the spectral domain;

Advantages: solid theoretical basis; easy to foresee the performance corresponding to certain type of graphs;

Disadvantages: hard to do inductive learning (not impossible though); low localized explainability;

Representative works: Fast Localized Graph Spectral Filtering [Defferrard et al. 2016], Graph Convolutional Network [Kipf et al. 2016], etc.



Overview

Background Introduction

Previous Works

Existing Problems & Challenges

Our Solutions

Experiments & Conclusion

Future Works

Existing Problems: Non-learnable Graph Filter

We focus on the problems of spectral methods: existing models such as GCN can only achieve non-learnable graph filter, which means that it cannot adaptively capture useful information that is not contained in the low-frequency component;

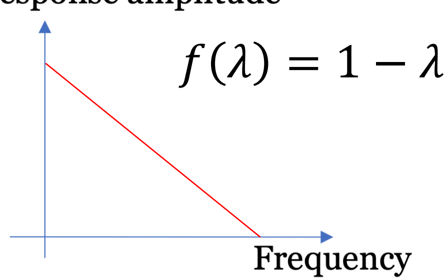
Existing Problems: Non-learnable Graph Filter

We focus on the problems of spectral methods: existing models such as GCN can only achieve non-learnable graph filter, which means that it cannot adaptively capture useful information that is not contained in the low-frequency component;

Layer expression of GCN:

$$\begin{aligned}\mathbf{Z} &= \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}) \\ &= \sigma[(\mathbf{I}-\tilde{\mathbf{L}})\mathbf{X}\mathbf{\Theta}] \\ &= \sigma[\mathbf{U}(\mathbf{I}-\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}}\mathbf{X}\mathbf{\Theta}]\end{aligned}$$

Response amplitude



Frequency-response function

Existing Problems: Non-learnable Graph Filter

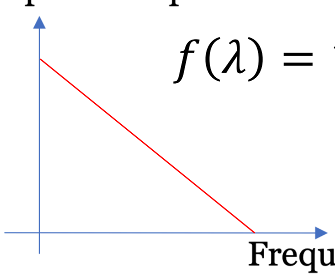
We focus on the problems of spectral methods: existing models such as GCN can only achieve non-learnable graph filter, which means that it cannot adaptively capture useful information that is not contained in the low-frequency component;

Layer expression of GCN:

$$
\begin{aligned}
\mathbf{Z} &= \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\boldsymbol{\Theta}) \\
&= \sigma[(\mathbf{I}-\widetilde{\mathbf{L}})\mathbf{X}\boldsymbol{\Theta}] \\
&= \sigma[\mathbf{U}(\mathbf{I}-\boldsymbol{\Lambda})\mathbf{U}^{\mathrm{T}}\mathbf{X}\boldsymbol{\Theta}]
\end{aligned}
$$

Response amplitude

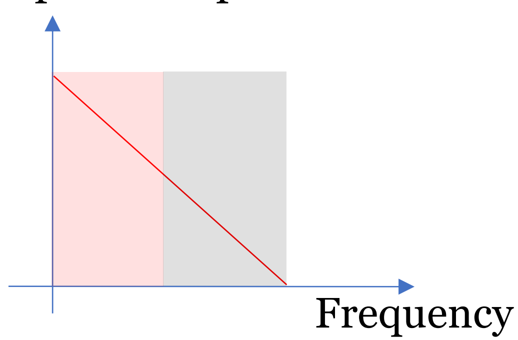
$$f(\lambda) = 1 - \lambda$$



Frequency-response function

Red component can be well-captured:

Response amplitude



Frequency-response function

Existing Problems: Non-learnable Graph Filter

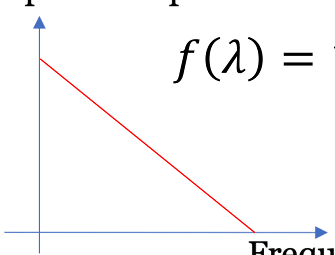
We focus on the problems of spectral methods: existing models such as GCN can only achieve non-learnable graph filter, which means that it cannot adaptively capture useful information that is not contained in the low-frequency component;

Layer expression of GCN:

$$\begin{aligned}\mathbf{Z} &= \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\boldsymbol{\Theta}) \\ &= \sigma[(\mathbf{I}-\tilde{\mathbf{L}})\mathbf{X}\boldsymbol{\Theta}] \\ &= \sigma[\mathbf{U}(\mathbf{I}-\boldsymbol{\Lambda})\mathbf{U}^{\mathrm{T}}\mathbf{X}\boldsymbol{\Theta}]\end{aligned}$$

Response amplitude

$f(\lambda) = 1 - \lambda$

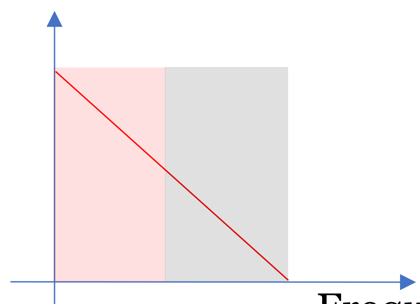


Frequency

Frequency-response function

Red component can be well-captured:

Response amplitude

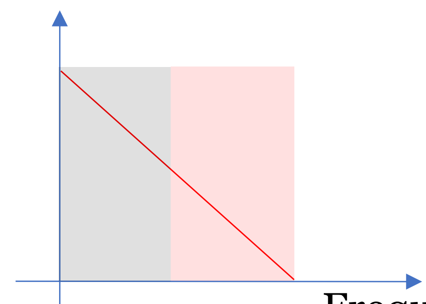


Frequency

Frequency-response function

Red component cannot be well-captured:

Response amplitude



Frequency

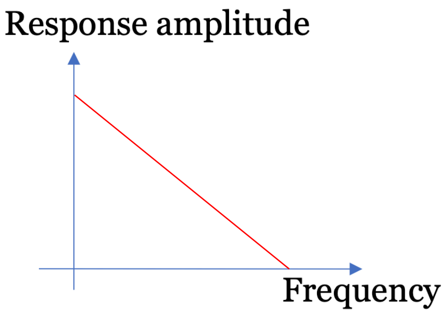
Frequency-response function

Existing Problems: Non-learnable Graph Filter

We focus on the problems of spectral methods: existing models such as GCN can only achieve non-learnable graph filter, which means that it cannot adaptively capture useful information that is not contained in the low-frequency component;

Layer expression of GCN:

$$\begin{aligned}\mathbf{Z} &= \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\boldsymbol{\Theta}) \\ &= \sigma[(\mathbf{I}-\widetilde{\mathbf{L}})\mathbf{X}\boldsymbol{\Theta}] \\ &= \sigma[\mathbf{U}(\mathbf{I}-\boldsymbol{\Lambda})\mathbf{U}^{\mathrm{T}}\mathbf{X}\boldsymbol{\Theta}]\end{aligned}$$

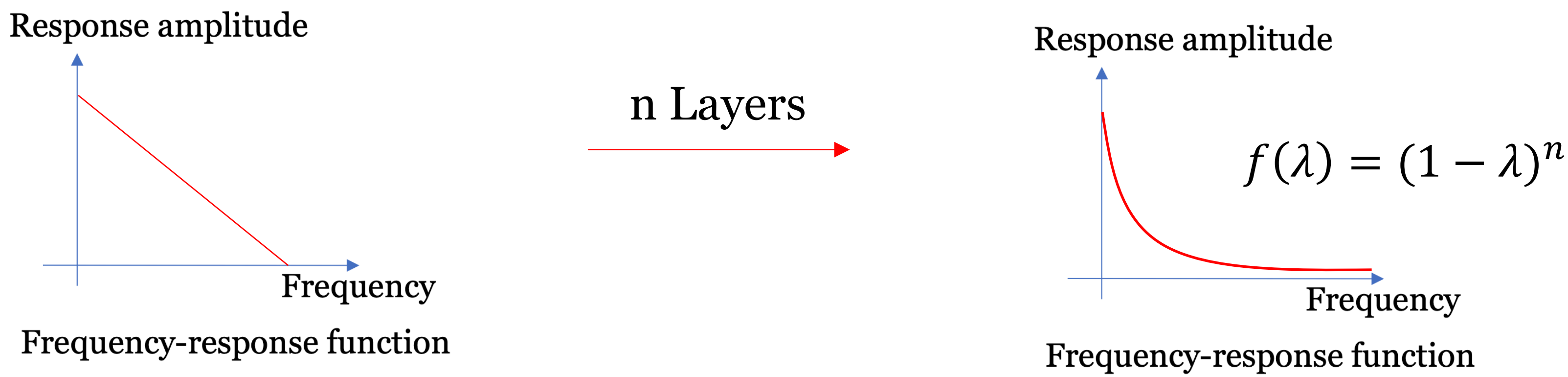


Frequency-response function

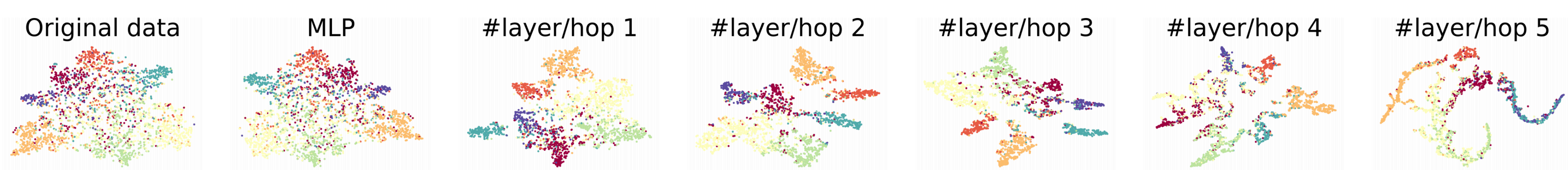
Problem to be tackled: the frequency response function should be **learnable** and **adaptively adjust** itself to capture useful information.

Existing Problems: Over-smoothing

Such fixed filter would greatly reduce most high-frequency components, i.e., making all nodes to be similar to each other.



Node embeddings learned from GCN on Cora dataset [Liu et al. 2020]:



Overview

Background Introduction

Previous Works

Existing Problems & Challenges

Our Solutions

Experiments & Conclusion

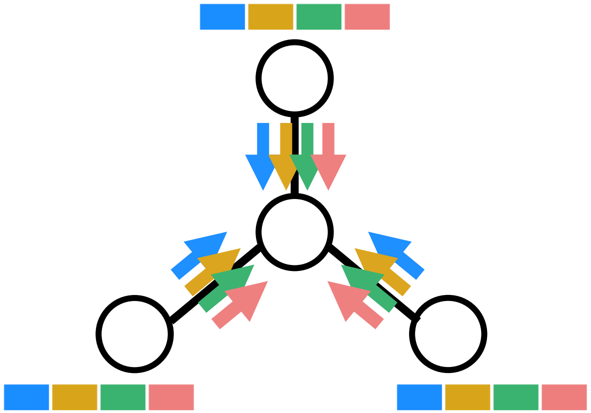
Future Works

Our Solutions: AdaGNN Layer-wise Illustration

Layer-wise signal filtering operation comparison:

GCN (without learnable matrix)

$$\mathbf{E} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X} = \mathbf{X} - \tilde{\mathbf{L}}\mathbf{X}$$



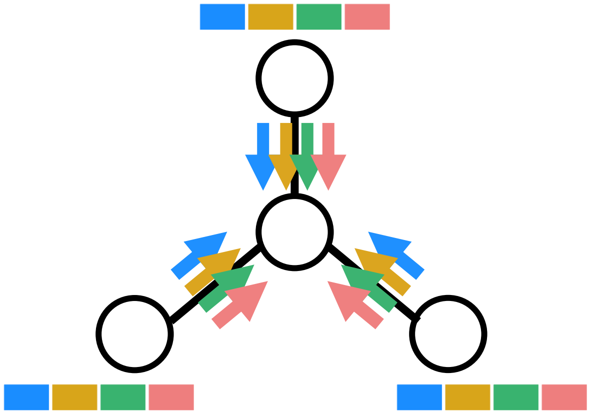
Each channel corresponds to a fixed weight factor for filtering.

Our Solutions: AdaGNN Layer-wise Illustration

Layer-wise signal filtering operation comparison:

GCN (without learnable matrix)

$$\mathbf{E} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X} = \mathbf{X} - \tilde{\mathbf{L}}\mathbf{X}$$

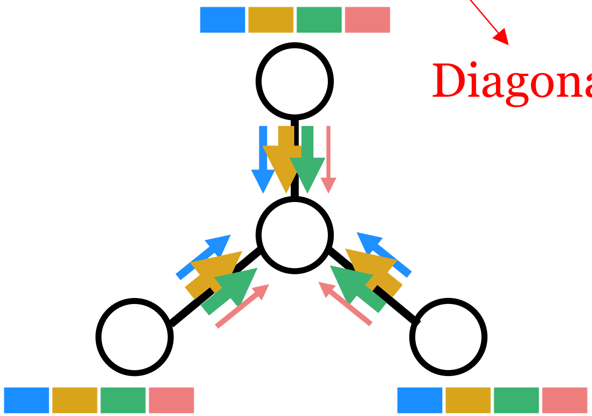


Each channel corresponds to a fixed weight factor for filtering.

AdaGNN

$$\mathbf{E} = \mathbf{X} - \tilde{\mathbf{L}}\mathbf{X}{\color{red}\mathbf{\Phi}}$$

Diagonal matrix

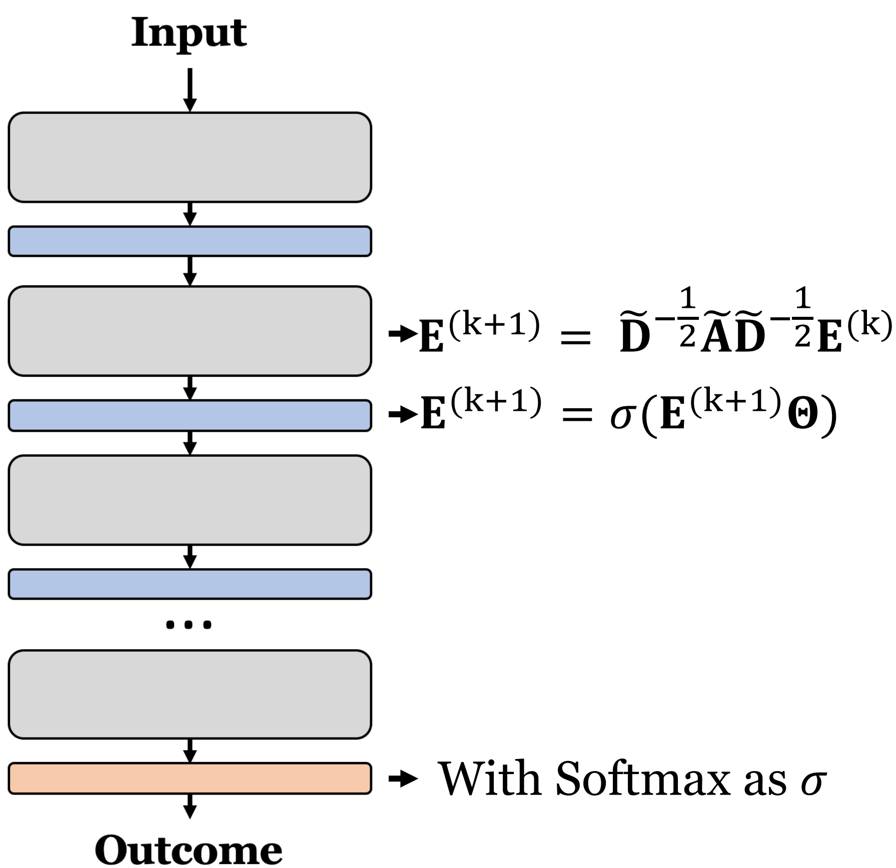


Each channel corresponds to a learnable weight factor ϕ for filtering at each specific layer.

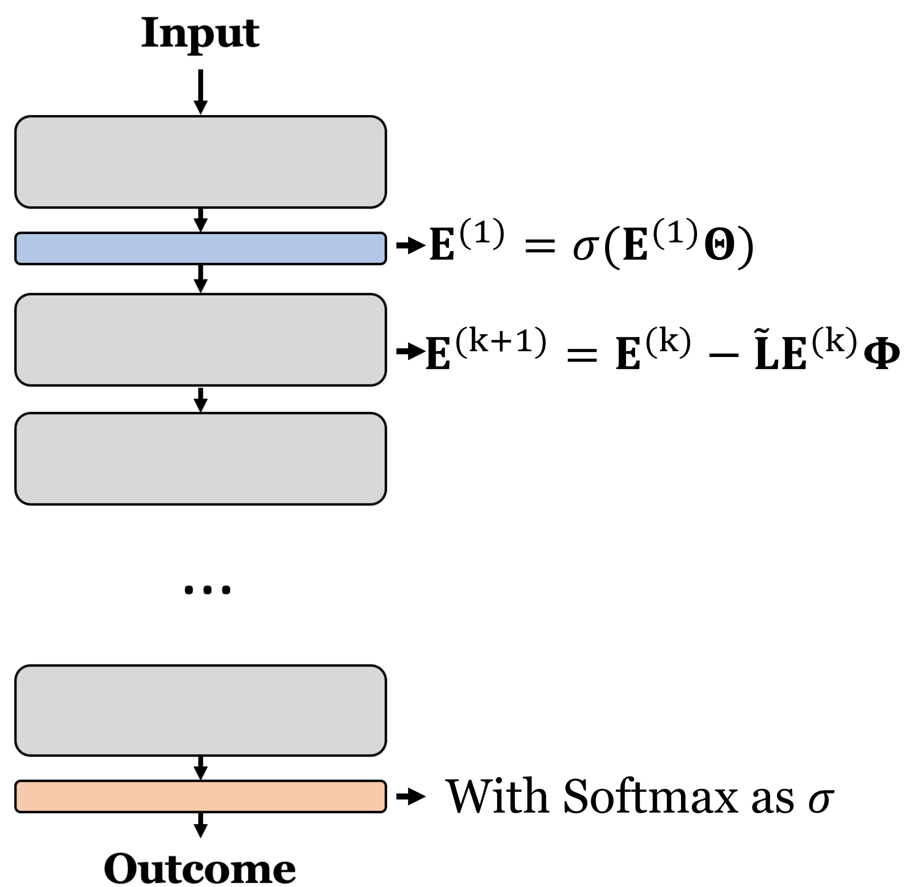
Our Solutions: AdaGNN Model-wise Illustration

Model-wise signal filtering operation comparison:

GCN



AdaGNN



Our Solutions: Learnable Filter in AdaGNN

Question 1: how could this help us to achieve a learnable filter?

	1 Layer	2 Layer	...	n Layer
SGC (without learnable matrix):	$f(\lambda) = 1 - \lambda$	$f(\lambda) = (1-\lambda)^2$		$f(\lambda) = (1-\lambda)^n$
AdaGNN* (one feature dimension):	$f(\lambda) = 1 - \phi_1 \lambda$	$f(\lambda) = \prod_{i=1}^{2}(1-\phi_i\lambda)$		$f(\lambda) = \prod_{i=1}^{n}(1-\phi_i\lambda)$

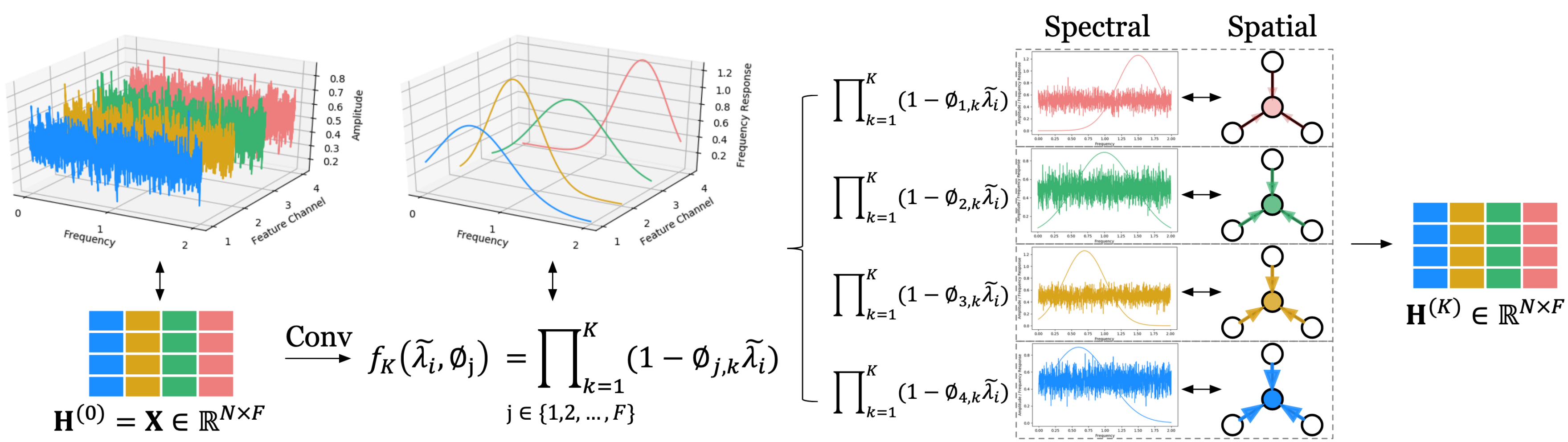
*For simplification purpose, we omit the weight matrix in the first layer.

Our Solutions: Learnable Filter in AdaGNN

Question 1: how could this help us to achieve a learnable filter?

	1 Layer	2 Layer	…	n Layer
SGC (without learnable matrix):	$f(\lambda) = 1 - \lambda$	$f(\lambda) = (1-\lambda)^2$		$f(\lambda) = (1-\lambda)^n$
AdaGNN* (one feature dimension):	$f(\lambda) = 1 - \phi_1\lambda$	$f(\lambda) = \prod_{i=1}^{2}(1-\phi_i\lambda)$		$f(\lambda) = \prod_{i=1}^{n}(1-\phi_i\lambda)$

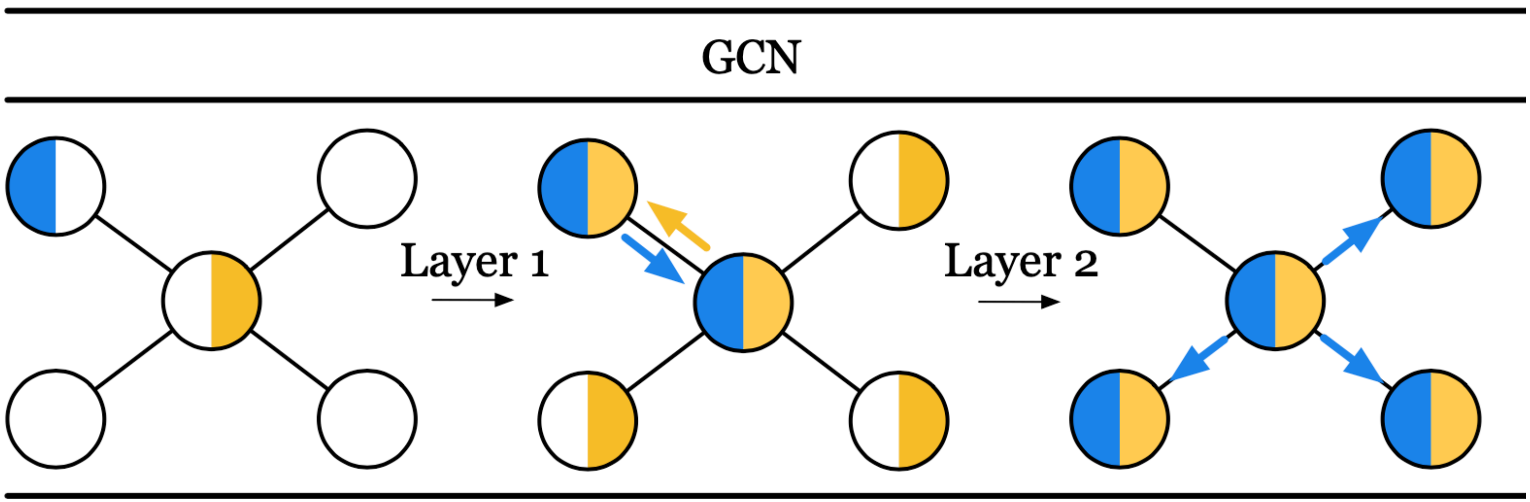
Assume we have four feature dimensions:



*For simplification purpose, we omit the weight matrix in the first layer.

Our Solutions: Toy Example for Over-smoothing Relief

Question 2: how could this help us to relieve over-smoothing?

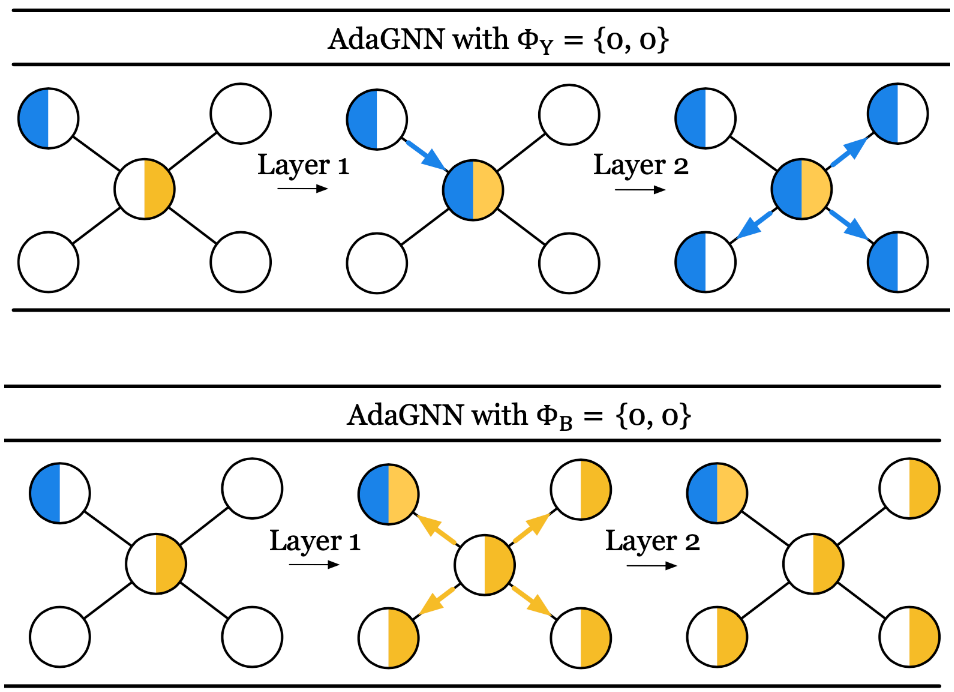


Information is aggregated across different feature dimensions indiscriminately, leading to similar nodes only after 2 layers*.

*In this example, we assume the attribute values can only be binary.

Our Solutions: Toy Example for Over-smoothing Relief

Question 2: how could this help us to relieve over-smoothing?



With learnable ϕs, the embedding of different nodes can be more distinguishable according to their roles after information aggregation.

In AdaGNN, information can be aggregated in a dimension-specific manner*.

*In this example, we assume the attribute values can only be binary.

Overview

Background Introduction

Previous Works

Existing Problems & Challenges

Our Solutions

Experiments & Conclusion

Future Works

Experiments: General Settings

Downstream tasks:

- Node classification;

Datasets:

- BlogCatalog [Tang et al., 2009] , Flickr [Huang et al., 2017], ACM [Tang et al., 2008], Cora, Citeseer and Pubmed [Sen et al., 2008];

Baselines:

- Three state-of-the-art GNNs including GCN [Kipf et al. 2016], SGC [Wu et al. 2019] and GraphSAGE [Hamilton et al. 2017]; Two recent approaches tackling over-smoothness including Dropedge [Rong et al. 2019] and Pairnorm [Zhao et al. 2019].

	BlogCatalog	Flickr	ACM	Cora	Citeseer	Pubmed
# Nodes	5,196	7,575	16,484	2,708	3,327	19,717
# Edges	173,468	242,146	71,980	5,429	4,732	44,338
# Features	8,189	12,047	8,337	1,433	3,703	500
# Average Degree	66.8	63.9	8.7	4.0	2.8	4.5
# Classes	6	9	9	7	6	3

Experiments: Example Results on BlogCatalog

Our model achieves the **best** performance on prediction accuracy in shallow layer.

Dataset	Model	2 Layer	4 Layer	8 Layer	16 Layer
BlogCatalog	GCN	73.98 ± 0.6%	69.71 ± 0.4%	37.61 ± 2.2%	20.61 ± 1.9%
	GraphSAGE	70.41 ± 0.5%	67.03 ± 0.5%	39.15 ± 1.6%	18.34 ± 3.9%
	SGC	73.97 ± 0.6%	68.94 ± 0.8%	47.94 ± 0.9%	29.02 ± 1.7%
	DropEdge-GCN	74.17 ± 0.7%	70.96 ± 1.3%	60.51 ± 2.4%	51.88 ± 0.8%
	Pairnorm-GCN-SI	67.32 ± 0.7%	63.61 ± 0.9%	65.04 ± 0.6%	67.51 ± 0.4%
	Pairnorm-GCN-SCS	71.67 ± 0.3%	67.01 ± 0.2%	69.30 ± 0.7%	69.75 ± 1.2%
	AdaGNN-R	**86.80 ± 0.3%**	**87.04 ± 0.2%**	**86.68 ± 0.1%**	**86.44 ± 0.5%**
	AdaGNN-S	**88.50 ± 0.2%**	**88.79 ± 0.2%**	**88.81 ± 0.1%**	**88.19 ± 0.2%**

AdaGNN-R: model with asymmetrically normalized $\tilde{\mathbf{L}}$;
AdaGNN-S: model with symmetrically normalized $\tilde{\mathbf{L}}$;

Experiments: Example Results on BlogCatalog

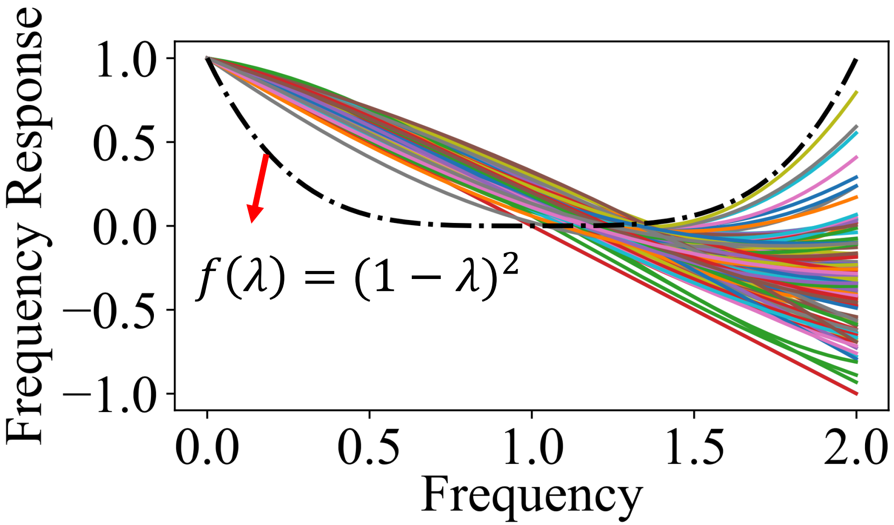
In deeper layers, our model not only achieves the best performance, but also greatly **relieves over-smoothness**.

Dataset	Model	2 Layer	4 Layer	8 Layer	16 Layer
BlogCatalog	GCN	73.98 ± 0.6%	69.71 ± 0.4%	37.61 ± 2.2%	20.61 ± 1.9%
	GraphSAGE	70.41 ± 0.5%	67.03 ± 0.5%	39.15 ± 1.6%	18.34 ± 3.9%
	SGC	73.97 ± 0.6%	68.94 ± 0.8%	47.94 ± 0.9%	29.02 ± 1.7%
	DropEdge-GCN	74.17 ± 0.7%	70.96 ± 1.3%	60.51 ± 2.4%	51.88 ± 0.8%
	Pairnorm-GCN-SI	67.32 ± 0.7%	63.61 ± 0.9%	65.04 ± 0.6%	67.51 ± 0.4%
	Pairnorm-GCN-SCS	71.67 ± 0.3%	67.01 ± 0.2%	69.30 ± 0.7%	69.75 ± 1.2%
	AdaGNN-R	**86.80 ± 0.3%**	**87.04 ± 0.2%**	**86.68 ± 0.1%**	**86.44 ± 0.5%**
	AdaGNN-S	**88.50 ± 0.2%**	**88.79 ± 0.2%**	**88.81 ± 0.1%**	**88.19 ± 0.2%**

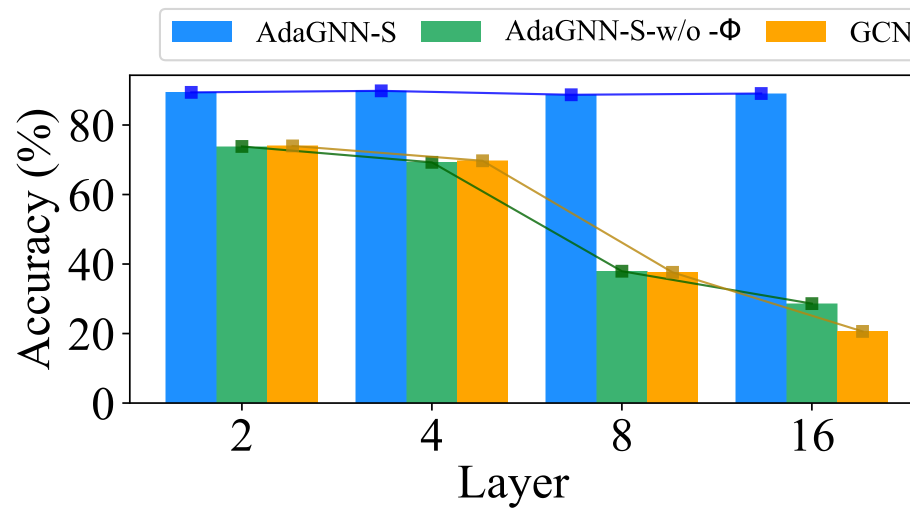
AdaGNN-R: model with asymmetrically normalized $\tilde{\mathbf{L}}$;
AdaGNN-S: model with symmetrically normalized $\tilde{\mathbf{L}}$;

Experiments: Ablation Study

Ablation study with AdaGNN-S as an example:



An visualization of the learned filters across different feature dimension of AdaGNN-S on Flickr dataset.



Model ablation study of AdaGNN-S on BlogCatalog.

Conclusion

- AdaGNN **adaptively** learns the smoothness of each feature dimension, and it achieves a **learnable filter** after multiple layers are stacked together.

- The learnable filter contributes to the **performance superiority** and **over-smoothing relief**.

Overview

Background Introduction

Previous Works

Existing Problems & Challenges

Our Solutions

Experiments & Conclusion

Future Works

Future works

- Fairness issue in spectral GNNs.
- Spectral GNNs with better localized explainability.
- GNNs with learnable and more flexible filter.

References

[Tang et al. 2008] Tang J, Zhang J, Yao L, et al. Arnetminer: extraction and mining of academic social networks[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008: 990-998.

[Tang et al. 2009] Tang L, Liu H. Relational learning via latent social dimensions[C]//Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009: 817-826.

[Huang et al. 2017] Huang X, Li J, Hu X. Label informed attributed network embedding[C]//Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. 2017: 731-739.

[Sen et al. 2008] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. AI magazine, 29(3), 93-93.

[Veličković et al. 2017] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[Hamilton et al. 2017] Hamilton, W. L., Ying, R., & Leskovec, J. (2017, December). Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 1025-1035).

[Defferrard et al. 2016] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29, 3844-3852.

[Kipf et al. 2016] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[Bo et al. 2021] Bo D., Wang X., Shi C., et al. Beyond low-frequency information in graph convolutional networks[J]. arXiv preprint arXiv:2101.00797, 2021.

[Liu et al. 2020] Liu M, Gao H, Ji S. Towards deeper graph neural networks[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020: 338-348.

[Rong et al. 2019] Rong Y, Huang W, Xu T, et al. Dropedge: Towards deep graph convolutional networks on node classification[J]. arXiv preprint arXiv:1907.10903, 2019.

[Zhao et al. 2019] Zhao L, Akoglu L. Pairnorm: Tackling oversmoothing in gnns[J]. arXiv preprint arXiv:1909.12223, 2019.

[Wu et al. 2019] Wu F, Souza A, Zhang T, et al. Simplifying graph convolutional networks[C]//International conference on machine learning. PMLR, 2019: 6861-6871.

[Hamilton et al. 2017] Hamilton W L, Ying R, Leskovec J. Inductive representation learning on large graphs[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017: 1025-1035.

The End

Thanks for listening!