SIMPLE AND DEEP GRAPH CONVOLUTIONAL NETWORKS

Paper Authors: Ming Chen, Zhewei Wei,
Zengfeng Huang, Bolin Ding, Yaliang Li

Presented by: Hauton J Tsang

UNIVERSITY OF
WATERLOO

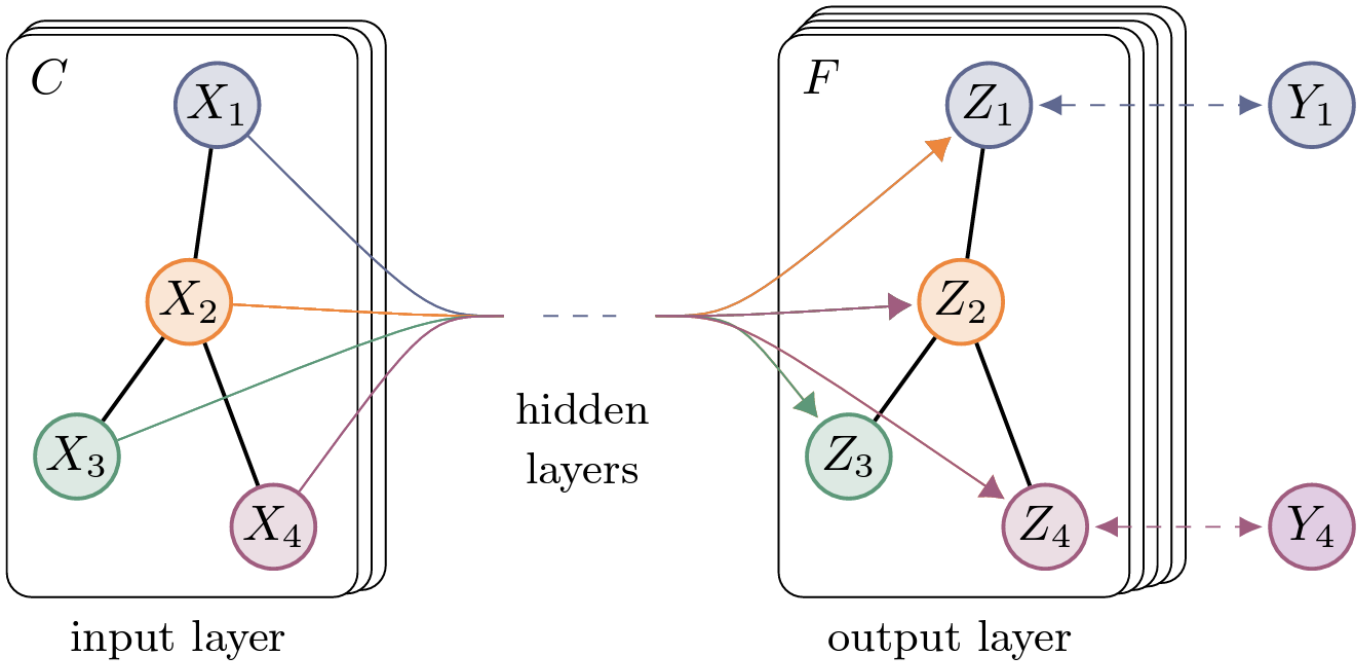
Outline

- Introduction
- Importance of problem
- Prior work
- Authors' solutions
- Empirical results
- Interesting questions
- Future directions

INTRODUCTION

Introduction

- Graph convolutional networks (GCN) (Kipf & Welling)



Introduction

- Problem: Deep GCNs are not very effective
  - Best GCNs are shallow 2-layer models
  - Shouldn't extracting info from higher-order neighbors improve effectiveness?
- Reason: over-smoothing (Li et al.)
  - As layers increase, the nodes in GCN tend to converge to a certain value

WHY IS IT IMPORTANT?

Why is it important?

- Definitively answer the question of whether deep GCNs can perform better

- Would information from higher-order neighbors make the model more effective?

PRIOR WORK

Why don't previous methods address this problem?

- Mitigate the over-smoothing problem of deep GCNs
  - Residual connections (He et al.)
    - Slows down over-smoothing but they still occur
  - Dense skip connections (Xu et al.)
  - Removing random edges of input graph (Rong et al.)

- Even with these methods, state-of-the-art results for semi-supervised tasks are achieved using shallow GCNs

Why don't previous methods address this problem?

- Incorporate higher-order information in shallow networks
  - Applying K-th power of convolution matrix in one layer (Wu et al.)
  - Replace convolution matrix with Personalized PageRank (Klicpera et al.)

$$\boldsymbol{H}^{(\ell+1)} = (1-\alpha)\tilde{\boldsymbol{P}}\boldsymbol{H}^{(\ell)} + \alpha\boldsymbol{H}^{(0)}$$

- Authors consider the above methods as using shallow networks as the neighbor features are combined linearly
  - Would deep nonlinearity provide more powerful expressivity in practice?

WHAT IS THE SOLUTION?

Solution: GCNII - Extending vanilla GCN

- Vanilla GCN equation:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right)$$

- GCNII extends vanilla GCN equation with two simple additions:
  - Initial residual
    - Add skip connection from input layer
  - Identity mapping
    - Add an identity matrix to weight matrix

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell\mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell\mathbf{W}^{(\ell)}\right)\right)$$

Solution: Extending vanilla GCN

- Graph convolution matrix with the renormalization trick from vanilla GCN

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right)$$

- Combined residual connection $\mathbf{H}^{(0)}$ with $\widetilde{\mathbf{P}}\mathbf{H}^{(l)}$
- Combined the identity mapping $\mathbf{I}_n$ with the weight matrix $\mathbf{W}^{(l)}$
- α_l and β_l are hyper-parameters, setting $\alpha_l = 0$ and $\beta_l = 1$ gives vanilla GCN

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell\mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell\mathbf{W}^{(\ell)}\right)\right)$$

Solution: Extending vanilla GCN

- Initial residual:
  - Residual connection, but to the initial representation rather than previous layer

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell \mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}\right)\right)$$

  - Linearly combined with smooth representation $\tilde{\mathbf{P}}\mathbf{H}^{(l)}$, where $\tilde{\mathbf{P}}$ is the graph convolution matrix with renormalization trick

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell \mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}\right)\right)$$

Solution: Extending vanilla GCN

- Why is initial residual alone not enough?
  - Prior work on Personalized PageRank (Klicpera et al.) has also tried an initial residual method
  - However, adding non-linearity caused overfitting
  - Initial residuals alone are *not* sufficient for creating a deep GCN
  - $\mathbf{H}^{(0)}$ could be either the feature matrix or a transformation of the feature matrix (e.g. after applying dimensional reduction)

Solution: Extending vanilla GCN

- Identity mapping:
  - Combine identity matrix $\mathbf{I}_n$ with the weight matrix $\mathbf{W}^{(l)}$

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell \mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}\right)\right)$$

  - Why?

Solution: Extending vanilla GCN

- Identity mapping:
  - Why add identity mapping?
    - Idea borrowed from ResNet: ensures deep GCN achieves at least same performance as shallow models
    - Mitigates interactions with other dimensions can degrade performance
    - Identity mapping has been used in semi-supervised tasks in linear models
    - Mitigating theoretical information loss by increasing the maximum singular value

Solution: Extending vanilla GCN

- How does GCNII achieve at least same performance as shallow models?
- Recall Personalized PageRank (Klicpera et al.):

$$\boldsymbol{H}^{(\ell+1)} = (1-\alpha)\tilde{\boldsymbol{P}}\boldsymbol{H}^{(\ell)} + \alpha\boldsymbol{H}^{(0)}$$

- Equation of GCNII (if we set $\beta_l = 0$) :

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell\mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell\mathbf{W}^{(\ell)}\right)\right)$$

Solution: Extending vanilla GCN

- How does identity mapping mitigate theoretical information loss?
  - Ono & Suzuki proved that K-layer GCNs converge and incur information loss
  - Rate depends on s^K, maximum singular value of weight matrix $\mathbf{W}^{(l)}$
  - Authors show we can ensure s^K is close to 1 if we replace $\mathbf{W}^{(l)}$ with

$$(1-\beta_l)\mathbf{I}_n + \beta_l \mathbf{W}^{(l)}$$

and applying regularization to $\mathbf{W}^{(l)}$

Solution: Extending vanilla GCN

- β_l should be set differently for each layer so impact of weight matrix decreases the deeper the layer is
- In practice authors chose $\beta_l = \log(\frac{\lambda}{l} + 1)$, where λ is a fixed hyperparameter
- β_l decreases as layers increase, so weight matrix decays at lower layers

$$\left((1-\beta_\ell)\mathbf{I}_n+\beta_\ell \mathbf{W}^{(\ell)}\right)$$

Solution: Extending GCNII

- GCNII*: independent weight matrices for initial residual and smoothed representation

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \alpha_\ell \mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}\right)\right)$$

$$\mathbf{H}^{(\ell+1)} = \sigma\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}_1^{(\ell)}\right) + \right.$$
$$\left. + \alpha_\ell \mathbf{H}^{(0)}\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}_2^{(\ell)}\right)\right).$$

EMPIRICAL RESULTS

Results: Semi-Supervised Tasks

Dataset	Method	Layers 2	4	8	16	32	64
Cora	GCN	**81.1**	80.4	69.5	64.9	60.3	28.7
	GCN(Drop)	**82.8**	82.0	75.8	75.7	62.5	49.5
	JKNet	-	80.2	80.7	80.2	**81.1**	71.5
	JKNet(Drop)	-	**83.3**	82.6	83.0	82.5	83.2
	Incep	-	77.6	76.5	81.7	**81.7**	80.0
	Incep(Drop)	-	82.9	82.5	83.1	83.1	**83.5**
	GCNII	82.2	82.6	84.2	84.6	85.4	**85.5**
	GCNII*	80.2	82.3	82.8	83.5	84.9	**85.3**

Dataset	Method	Layers 2	4	8	16	32	64
Citeseer	GCN	**70.8**	67.6	30.2	18.3	25.0	20.0
	GCN(Drop)	**72.3**	70.6	61.4	57.2	41.6	34.4
	JKNet	-	68.7	67.7	**69.8**	68.2	63.4
	JKNet(Drop)	-	72.6	71.8	**72.6**	70.8	72.2
	Incep	-	69.3	68.4	**70.2**	68.0	67.5
	Incep(Drop)	-	**72.7**	71.4	72.5	72.6	71.0
	GCNII	68.2	68.9	70.6	72.9	**73.4**	73.4
	GCNII*	66.1	67.9	70.6	72.0	**73.2**	73.1

Dataset	Method	Layers 2	4	8	16	32	64
Pubmed	GCN	**79.0**	76.5	61.2	40.9	22.4	35.3
	GCN(Drop)	**79.6**	79.4	78.1	78.5	77.0	61.5
	JKNet	-	78.0	**78.1**	72.6	72.4	74.5
	JKNet(Drop)	-	78.7	78.7	79.1	**79.2**	78.9
	Incep	-	77.7	**77.9**	74.9	OOM	OOM
	Incep(Drop)	-	**79.5**	78.6	79.0	OOM	OOM
	GCNII	78.2	78.8	79.3	**80.2**	79.8	79.7
	GCNII*	77.7	78.2	78.8	**80.3**	79.8	80.1

Results: Fully-Supervised Tasks

Method	Cora	Cite.	Pumb.	Cham.	Corn.	Texa.	Wisc.
GCN	85.77	73.68	88.13	28.18	52.70	52.16	45.88
GAT	86.37	74.32	87.62	42.93	54.32	58.38	49.41
Geom-GCN-I	85.19	**77.99**	90.05	60.31	56.76	57.58	58.24
Geom-GCN-P	84.93	75.14	88.09	60.90	60.81	67.57	64.12
Geom-GCN-S	85.27	74.71	84.75	59.96	55.68	59.73	56.67
APPNP	87.87	76.53	89.40	54.3	73.51	65.41	69.02
JKNet	85.25 (16)	75.85 (8)	88.94 (64)	60.07 (32)	57.30 (4)	56.49 (32)	48.82 (8)
JKNet(Drop)	87.46 (16)	75.96 (8)	89.45 (64)	62.08 (32)	61.08 (4)	57.30 (32)	50.59 (8)
Incep(Drop)	86.86 (8)	76.83 (8)	89.18 (4)	61.71 (8)	61.62 (16)	57.84 (8)	50.20 (8)
GCNII	**88.49** (64)	77.08 (64)	89.57 (64)	60.61 (8)	74.86 (16)	69.46 (32)	74.12 (16)
GCNII*	88.01 (64)	77.13 (64)	**90.30** (64)	**62.48** (8)	**76.49** (16)	**77.84** (32)	**81.57** (16)

Results: Inductive Learning

Method	PPI
GraphSAGE (Hamilton et al., 2017)	61.2
VR-GCN (Chen et al., 2018b)	97.8
GaAN (Zhang et al., 2018)	98.71
GAT (Veličković et al., 2018)	97.3
JKNet (Xu et al., 2018)	97.6
GeniePath (Liu et al., 2019)	98.5
Cluster-GCN (Chiang et al., 2019)	99.36
GCNII	99.53 ± 0.01
GCNII*	$\mathbf{99.56 \pm 0.02}$

Ablation Study

- Isolating effects of initial residuals and identity mapping independently

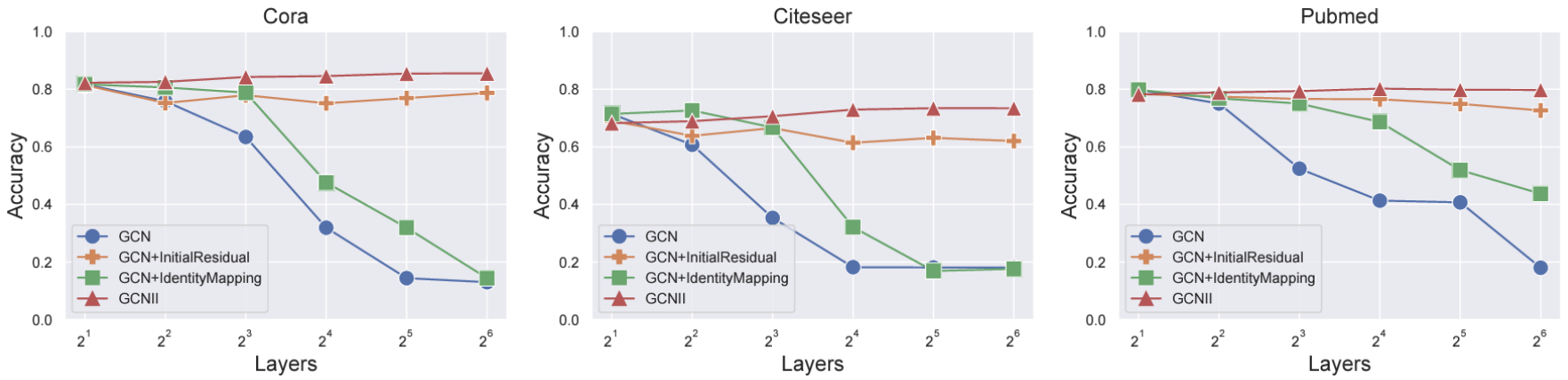


Figure 2. Ablation study on initial residual and identity mapping.

INTERESTING QUESTIONS

What makes over-smoothing more severe?

- Conjecture: nodes with higher degree are more affected
- Authors begin with a GCN model with residual connection of the self-looped graph:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell)} + \mathbf{H}^{(\ell)}\right)\mathbf{W}^{(\ell)}\right)$$

- Wang et al. previously derived that this simulates a random walk with transition matrix:

$$\frac{\mathbf{I}_n + \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}}{2}$$

What makes over-smoothing more severe?

- Therefore, the representation after applying a K-layer renormalized graph convolution with residual connection to $\mathbf{x}$ is:

$$\mathbf{h}^{(K)} = \left(\frac{\mathbf{I}_n + \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2}}{2} \right)^K \cdot \mathbf{x}$$

where $\tilde{\mathbf{D}}$ is the diagonal degree matrix

and $\tilde{\mathbf{A}}$ is the adjacency matrix

What makes over-smoothing more severe?

- Authors prove that this can be rewritten to:

$$\mathbf{h}^{(K)} = \frac{\langle \tilde{\mathbf{D}}^{1/2}\mathbf{1}, \mathbf{x} \rangle}{2m+n} \pm \left(\sum_{i=1}^{n} x_i \right) \cdot \left(1 - \frac{\lambda_{\tilde{G}}^2}{2} \right)^K \cdot \mathbf{1}.$$

where $\lambda_{\tilde{G}}^2$ is the spectral gap of the self-looped graph

What makes over-smoothing more severe?

- This can be rewritten into the following form:
  - d_i: degree of some node

$$\mathbf{h}^{(K)}(j) = \sqrt{d_j + 1}\left(\sum_{i=1}^{n} \frac{\sqrt{d_i+1}}{2m+n} x_i \pm \frac{\sum_{i=1}^{n} x_i \left(1 - \frac{\lambda_{\tilde{G}}^2}{2}\right)^K}{\sqrt{d_j + 1}}\right)$$

- Equation converges faster as $\sqrt{d_j + 1}$ increases to a steady state

$$\mathbf{h}^{(K)}(j) = \sqrt{d_j + 1}\left(\sum_{i=1}^{n} \frac{\sqrt{d_i+1}}{2m+n} x_i \pm \frac{\sum_{i=1}^{n} x_i \left(1 - \frac{\lambda_{\tilde{G}}^2}{2}\right)^K}{\sqrt{d_j + 1}}\right)$$

What makes over-smoothing more severe?

- Equation converges faster as $\sqrt{d_j + 1}$ increases to a steady state

$$\mathbf{h}^{(K)}(j) = \sqrt{d_j + 1}\left(\sum_{i=1}^{n} \frac{\sqrt{d_i+1}}{2m+n} x_i \pm \frac{\sum_{i=1}^{n} x_i \left(1 - \frac{\lambda_{\tilde{G}}^2}{2}\right)^K}{\sqrt{d_j + 1}}\right)$$

$$\mathbf{h}^{(K)}(j) = \sqrt{d_j + 1}\left(\sum_{i=1}^{n} \frac{\sqrt{d_i+1}}{2m+n} x_i \right)$$

What makes over-smoothing more severe?

- Empirical evidence:
  - High node degree for shallow models improves performance as nodes can obtain more information from its neighbors
  - However, accuracy of high degree nodes drops as depth increases
  - Accuracy is ~0% when degree is more than 100

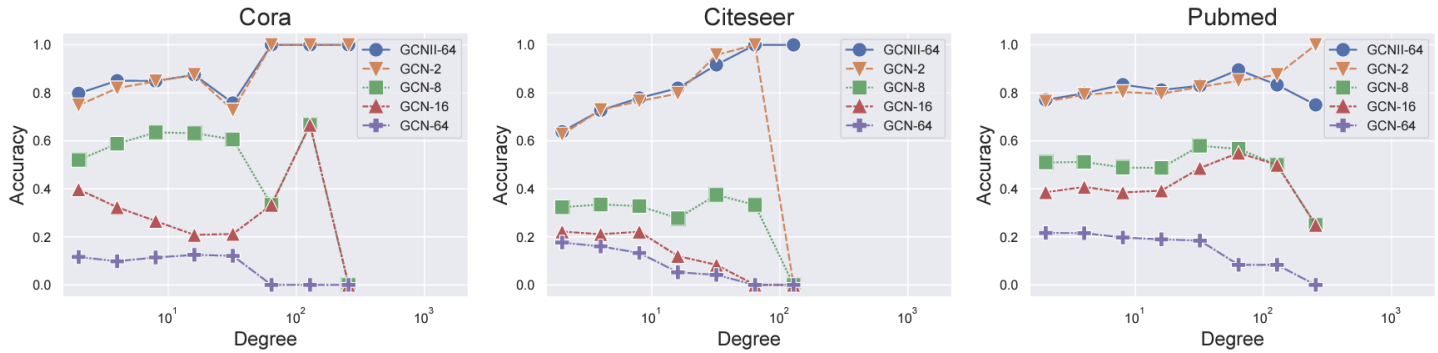


Figure 1. Semi-supervised node classification accuracy v.s. degree.

Expressive power and over-smoothing in GCNII

- Spectral analysis of GCNII:
  - Vanilla GCN simulates a polynomial filter of order K:

$$\left(\sum_{\ell=0}^{K} \theta_\ell \tilde{\mathbf{L}}^\ell\right) \cdot \mathbf{x}$$

$\tilde{\mathbf{L}}$ is the normalized Laplacian matrix of the self-looped graph

θ are fixed coefficients

  - GCNII can express a K-order polynomial filter with *arbitrary* coefficients (i.e. higher expressive power)

Expressive power and over-smoothing in GCNII

- Spectral analysis of GCNII:
  - Arbitrary coefficients are also what enable GCNII to overcome over-smoothing
  - Since we know that $\widetilde{\mathbf{P}} = \mathrm{I}_n - \widetilde{\mathbf{L}}$, we can rewrite

$$\left(\sum_{\ell=0}^{K} \theta_\ell \tilde{\mathbf{L}}^\ell\right) \cdot \mathbf{x}$$

into

$$\left(\sum_{\ell=0}^{K} \theta'_\ell \tilde{\mathbf{P}}^\ell\right) \cdot \mathbf{x}$$

Expressive power and over-smoothing in GCNII

- Spectral analysis of GCNII:

$$\left(\sum_{\ell=0}^{K} \theta'_{\ell} \tilde{\mathbf{P}}^{\ell}\right) \cdot \mathbf{x}$$

where θ'_l are arbitrary coefficients

  - Since θ'_l is arbitrary, we can choose coefficients in such a way that input feature and graph structure information is kept even if K goes to infinity
  - This is what β_l does in GCNII, and we preserve information by decaying it with the equation $\beta_l = \log(\frac{\lambda}{l} + 1)$

Future Directions & Discussion

- Addition of GCNII with attention
- Analyzing GCNII with ReLU and other activation functions in more depth
- Only outperforms state-of-the-art in semi-supervised learning by a small margin even with a much larger number of layers