Layer-wise Adaptive Graph Convolution Networks Using Generalized Pagerank

Kishan Wimalawarne   KISHANWN@GMAIL.COM
University of Tokyo, Hongo, Japan

Taiji Suzuki   TAIJI@MIST.I.U-TOKYO.AC.JP
University of Tokyo, Hongo, Japan
RIKEN, Nihombashi, Tokyo, Japan

Abstract

We investigate adaptive layer-wise graph convolution in deep GCN models. We propose AdaGPR to learn generalized Pageranks at each layer of a GCNII network to induce adaptive convolution. We show that the generalization bound for AdaGPR is bounded by a polynomial of the eigenvalue spectrum of the normalized adjacency matrix in the order of the number of generalized Pagerank coefficients. By analysing the generalization bounds we show that oversmoothing depends on both the convolutions by the higher orders of the normalized adjacency matrix and the depth of the model. We performed evaluations on node-classification using benchmark real data and show that AdaGPR provides improved accuracies compared to existing graph convolution networks while demonstrating robustness against oversmoothing. Further, we demonstrate that analysis of coefficients of layer-wise generalized Pageranks allows us to qualitatively understand convolution at each layer enabling model interpretations.

Keywords: GCN, adaptive, generalized pagerank, generalization bounds

1. Introduction

In recent years Graph Convolution Networks (GCN) have gained increased recognition as a versatile tool to learn from graphs. Graph convolution networks use the graph topological structures among the data to extract nonlinear features to perform learning tasks. Many recent advances in graph convolution networks have produced state of the art performances in applications such as social influence prediction (Li and Goldwasser, 2019), relationship modelling (Schlichtkrull et al., 2018), and recommendation systems (Ying et al., 2018)

Despite the promising capabilities and many novel approaches, GCN still faces several limitations that hinders its full potential in learning with graphs. A well known limitation with GCN is oversmoothing (Oono and Suzuki, 2020a), where stacking of multiple convolution layers leads to drop in performance. Oversmoothing is prominent with a model like the Vanila GCN (Kipf and Welling, 2017), since multiple convolutions by global graph data lead to generalized features that lack the ability learn from labelled data. Recently, many approaches have been proposed to mitigate the effect of oversmoothing. Some of these methods include simple data processing such as data normalization by Pair-Norms (Zhao and Akoglu, 2020) and random removal of edges using dropedges (Rong et al., 2020).

Many other methods use more complex methods such as random walks as employed in Scatter-GCN (Min et al., 2020) and skipping layers as with JKNet (Xu et al., 2018). A notable recent development is GCNII (Chen et al., 2020), which uses scaled residual layers and addition of the initial layer to each convolution layer. GCNII has reported strong robustness against oversmoothing, however, it often requires a deep network to gain a considerable high accuracy.

Another limitation that we identify with GCN is the lack of adaptability of graph convolution at each layer. Most GCN models apply the same graph convolution method to each layer of a deep network (Kipf and Welling, 2017; Chen et al., 2020). This not only cause oversmoothing but it may also lead to redundant memory usages and computations. Furthermore, most GCN models do not provide a systematic approach to understand and interpret graph convolutions applied at each layer of a deep model. In practice, to design a optimal GCN model it is desirable to know the suitable graph convolution method to apply as well as the amount of convolution to be applied at each layer depending on the data and the learning task. Recently proposed GPR-GNN (Chien et al., 2021) learns a generalized Pagerank within the APPNP model (Klicpera et al., 2019) to perform adaptive graph convolution. However, GPR-GNN is a shallow network and does not consider graph convolution in multiple layers, hence, it may not be efficient as a deep GCN model.

In this paper, we investigate adaptive convolution in deep graph convolution networks. In contrast to the widely adapted view of applying the same graph convolution method at each layer, we propose that graph convolution should be different for each layer. In our view, graph convolution should be adaptive in a layer-wise manner where the GCN model should be able to learn how to apply graph convolution depending on the network architecture, nature of the data, and the learning task. We propose AdaGPR to apply adaptive generalized Pageranks at each layer of a GCNII model by learning to predict the coefficients of generalized Pageranks using sparse solvers. We also give a new generalization error analysis of AdaGPR in which the Rademacher complexity is given as a polynomial of the eigenvalue spectrum of the normalized adjacency matrix. This bound reflects the mixing information effect, more specifically, the oversmoothing effect, and thus yields a better generalization error bound for a graph with a large node degree. We conduct evaluations on node-classification and show that AdaGPR provides better accuracy compared to state of the art GCN methods. As a further advantage of our method, we demonstrate that analysis of the coefficients of layer-wise generalized Pagerank allows us to quantitatively understand layer-wise convolution leading to semi-interpretable GCN models.

2. Review

We start by defining notations used in this paper. Let $G = (V, E)$ a graph with nodes $v_i \in V$, $i = 1, \ldots, N$ and edges $(v_i, v_j) \in E$. Let $X \in \mathbb{R}^{N \times q}$ represents a feature matrix with each row representing q features. Let $Y \in \mathbb{R}^{N \times c}$ represents labels of the N nodes with each consisting of c classes. The adjacency matrix of G is represented as $A \in \mathbb{R}^{N \times N}$, and the self-loops added adjacency matrix is $\hat{A} = A + I_N$, where $I_N \in \mathbb{R}^{N \times N}$ is a identity matrix. We denote the diagonal degree matrix of $\hat{A}$ by $\hat{D}_{ij} = \sum_k \hat{A}_{ik} \delta_{ij}$, then the normalized adjacency matrix is $\tilde{A} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$.

The most simple graph convolution network (also known as the Vanilla GCN) was proposed in Kipf and Welling (2017), where each layer of a multilayer network is multiplied by the normalized graph adjacency matrix before applying a nonlinear activation function. A 2-layer Vanilla GCN is given as

$$Z = \text{softmax}\big(\tilde{A}\text{ReLU}(\tilde{A}XW_0)W_1\big),$$

where $W_0 \in \mathbb{R}^{q \times h}$ and $W_1 \in \mathbb{R}^{h \times c}$ are learning weights with h hidden units. It is well observed that the Vanilla GCN model is highly susceptible to oversmoothing with the increase of depth (Oono and Suzuki, 2020a; Chen et al., 2020).

Recently, many methods that have been proposed to overcome oversmoothing (Zhao and Akoglu, 2020; Chen et al., 2020). One of the successful methods robust agasint oversmoothing with the increase of convolution layers is GCNII (Chen et al., 2020). It proposes to multiply convolution at each layer and add the initial layers with sclaing. The resulting $l + 1$th convolution layer of GCNII is given as

$$H^{(l+1)} = \sigma\Big(\big((1 - \alpha_l)\tilde{A}H^{(l)} + \alpha_l H^{(0)}\big)\big((1 - \beta_l)I_N + \beta_l W^{(l)}\big)\Big), \tag{1}$$

where $\sigma(\cdot)$ is the ReLU operator, $H^{(0)} = \sigma(XW^{(0)})$ is the output from initial layer, $W^{(0)}$ and $W^{(l)}$ are weight matrices, and $\alpha_l \in [0, 1]$ and $\beta_l \in [0, 1]$ are user-defined parameters.

Another approach that resaerhers have adaptoed to overcome oversmoothness in graph convolution models is to use the personalized Pagerank (Brin and Page, 1998) instead of the convolution by the adjacency matrix. PPNP and APPNP (Klicpera et al., 2019) are tow methods that uses the personalized Pagerank convolution to obtain improved accuracy for node classification. A computationally feasible methods that avoid matrix inversion of the personalized Pagerank is the generalized Pagerank (GPR) (Li et al., 2019), which is defined with K powers of the normalized adjacency matrix with coefficients $\mu = [\mu_0, \ldots, \mu_{K-1}] \in [0, 1]^K$ as

$$\text{GPR}(\mu) := \sum_{k=0}^{K-1} \mu_k \tilde{A}^k. \tag{2}$$

The advantage of using GPR is the ability to learn the coefficients μ from the data (Li et al., 2019). Adaptive learning of GPR is used by GPRGNN (Chien et al., 2021) given by the following model,

$$P = \text{softmax}(Z), \ Z = \sum_{k=0}^{K-1} \mu_k H^{(k)}, \ H^{(k)} = \tilde{A}H^{(k-1)}, \ H^{(0)}_{i:} = f_\theta(X_{i:}), \tag{3}$$

where θ represents learning parameters of a multilayer network and μ is learned using message passing.

There are several limitations in above models. Both personalized Pagerank based models and GPR-GNN apply only a single convolution by variants of the Pagerank on the learned representation prior to the output layer. Further, these models do not apply any learning weights and nonlinear activation functions after convolution, hence, they do not create deep GCN models. On the other hand, the GCNII allows us to develop deep models that are robust against the oversmoothing, however, it relies solely on convolutions by the normalized adjacency matrix lacking any adaptive convolution or any benefits offered by the Pagerank.

3. Proposed Method

We propose adaptive layer-wise graph convolution for deep graph conventional models. Our approach is simple, where we propose to apply a generalised Pagerank at each layer of the GCNII and learn coefficients of generalised Pageranks.

As in GCNII, we use a initial layer $H^{(0)} = \sigma(XW^{(0)})$ without any graph convolution using learning weights $W^{(0)} \in \mathbb{R}^{N \times h}$, where h is the number of hidden units. Given L layers of graph convolutions, we replace the convolution by $\tilde{A}$ at layer l of (1) with the generalized Pagerank (2) using K orders of $\tilde{A}$ and coefficients $\mu^{(l)} = (\mu_0^{(l)}, \ldots, \mu_{K-1}^{(l)}) \in [0, 1]^K$. Additionally, we impose the constraint $\sum_{k=0}^{K-1} \mu_k^{(l)} = 1$. In order to make generalized Pagerank adaptive for each layer, the model needs to learn coefficients $\mu^{(l)} \in \mathbb{R}^K$, $l = 1, \ldots, L$ by using separate learning weights $v^{(l)} \in \mathbb{R}^K$, $l = 1, \ldots, L$, respectively. Furthermore, we provide flexibility to apply a suitable activation function $g(\cdot)$ on $v^{(l)}$ in order to obtain specific properties such as sparseness. We call the new graph convolution network AdaGPR, where its $(l+1)$th layer is defined as

$$H^{(l+1)} = \sigma\left(\left((1-\alpha_l)\left(\sum_{k=0}^{K-1} \mu_k^{(l)} \tilde{A}^k\right)H^{(l)} + \alpha_l H^{(0)}\right)\left((1-\beta_l)I_N + \beta_l W^{(l)}\right)\right), \quad \mu^{(l)} = g(v^{(l)}),$$
(4)

where $W^{(l)} \in \mathbb{R}^{h \times h}$, $l = 1, \ldots, L-1$ and $W^{(L)} \in \mathbb{R}^{h \times c}$. Similarly to GCNII, parameters α_l and β_l need to be specified by the user or tuned as hyperparamters. As with GCNII (Chen et al., 2020), we also specify a predefined $\alpha := \alpha_l \in (0, 1)$ for all layers and decaying $\beta_l = \log(\lambda/l + 1) \approx \lambda/l$ where λ is a predefined parameter.

The main advantage with AdaGPR compared to conventional graph convolution methods and GCNII is that it can learn how to apply convolution at each layer. It is obvious that when $\mu_0^{(l)} = 1.0$ or $\mu_1^{(l)} = 1.0$ for all $l = 1, \ldots, L$ AdaGPR is equivalent to a multilayer residual network or GCNII, respectively. Again, notice that AdaGPR has a generalized Pagerank at each layer with aggregations and nonlinear activations compared to APPNP and GPR-GNN. To our knowledge AdaGPR is the first graph convolution model to apply layer-wise adaptive Pagerank in a deep graph convolution model.

We point out that AdaGPR has more learning parameters and hyperparamters than GCNII. In practice, we have found that we need to consider K as a hyperparameter that needs to be selected during the training phase. The increased number of hyperparameters is a limitation of the proposed method. We can also use a different K for each layer, however, that may be impractical due to the large combinations of GPRs we may have to consider. Depending on the learning problem, we may also have to apply a separate weight decay for $v^{(l)}$.

3.1. Learning Sparse Solutions for GPR

There are several ways to learn $\mu^{(l)}$ of (4) such that $\sum_{k=0}^{K-1} \mu_k^{(l)} = 1$. One of the simplest methods is to use the Softmax, however, the resulting $\mu^{(l)}$ may not be sparse which would not give us the desired interpretable results. Variants of Softmax (Martins and Astudillo, 2016) such as sphericalmax and sum-normalization may lead to the same limitation of spraseness in addition to the difficulty of implementing the restriction $\sum_{k=0}^{K-1} \mu_k^{(l)} \neq 0$.

Another approach is message passing as used in GPR-GNN (Chien et al., 2021), however, it can be computationally expensive to implement message passing in a deep GCN model such as our proposed method.

We adopt the recently developed sparse activation function Sparsemax (Martins and Astudillo, 2016) for the task of predicting each $\mu^{(l)}$. Without loss of generality we restate $\mu^{(l)}$ belonging to a $(K-1)$-dimensional simplex $\Delta^{K-1} := \{\mu^{(l)} \in \mathbb{R}^K | \mathbf{1}^\top \mu^{(l)} = 1, \mu^{(l)} \leq \mathbf{0}\}$, then Sparsemax is the solution of

$$\text{sparsemax}(z^{(l)}) = \underset{\mu^{(l)} \in \Delta^{K-1}}{\operatorname{argmin}} \|\mu^{(l)} - z^{(l)}\|^2. \tag{5}$$

The closed-form solution of (5) is given by $\text{sparsemax}_i(z) = [z_i - \tau(z)]_+$ (Martins and Astudillo, 2016), where $\tau(z) = \frac{(\sum_{j \in k(z)} z_{(j)}) - 1}{k(z)}$ with $k(z) := \max\{k \in [K] | 1 + kz_{(k)} > \sum_{j<k} z_{(j)}\}$ given sorted $z_{(1)} \geq z_{(2)} \geq \ldots \geq z_{(K)}$. By empirical evaluations, we found that we can obtain better solutions for AdaGPR by using $\exp(\mu^{(l)})$ instead of $\mu^{(l)}$, which resembles a sparse version of softmax. Our implementations of AdaGPR use the Pytorch code for sparsemax associate with the paper (Martins and Astudillo, 2016)[1].

4. Theoretical Analysis

We give a new generalization error bound for the proposed method. Unlike existing bounds, our bound fully incorporates the information of the spectrum of the normalized adjacency matrix $\tilde{A}$ and thus can take the effect of oversmoothing into account.

We analyse generalization bounds under transductive settings (El-Yaniv and Pechyony, 2009; Oono and Suzuki, 2020b) for semi-supervised node classification. We recall that $X \in \mathbb{R}^{N \times q}$ is the feature matrix of N nodes with an associated graph $G = (V, E)$ and consider a 1-class labeled output $Y \in \mathbb{R}^{N \times 1}$. Let us consider the sets $\mathcal{X}$ and $\mathcal{Y}$ such that $X \subseteq \mathcal{X}$, $Y \subseteq \mathcal{Y}$ and $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Let us consider D_{train} and D_{test} as the training and test sets, respectively. Samples are drawn without replacement from D_{train} and D_{test} such that $D_{\text{train}} \cup D_{\text{test}} = V$ and $D_{\text{train}} \cap D_{\text{test}} = \emptyset$. Given $M := |D_{\text{train}}|$ and $U := |D_{\text{test}}|$, we define $Q := 1/M + 1/U$. Let $\mathcal{F} \subset \{\mathcal{X} \to \mathcal{Y}\}$ be the hypothesis for the transductive learning for AdaGPR. For a predictor $h : \mathcal{X} \to \mathcal{Y}$, $h \in \mathcal{F}$ and a loss function $l(\cdot, \cdot)$ (e.g., sigmoid, sigmoid cross entropy), we denote the training error by $R(h) = \frac{1}{M} \sum_{n \in V_{\text{train}}} l(h(x_n), y_n)$ and test error by $\hat{R}(h) = \frac{1}{U} \sum_{n \in V_{\text{test}}} l(h(x_n), y_n)$. Using a well-known result from El-Yaniv and Pechyony (2009), for a given hypothesis class $\mathcal{F}$ we state the generalization bounds based on transductive Rademacher complexity $\mathcal{R}(\mathcal{F}, p)$ with $p \in [0, 0.5]$ and $S := \frac{2(M+U)\min(M,U)}{(2(M+U)-1)(2\min(M,U)-1)}$ and probability $1 - \delta$ as[2]

$$R(h) \leq \hat{R}(h) + \mathcal{R}(\mathcal{F}, p_0) + c_0 Q \sqrt{\min(M, U)} + \sqrt{\frac{SQ}{2} \log \frac{1}{\delta}}, \tag{6}$$

where

$$\mathcal{R}(\mathcal{V}, p) = Q \mathbb{E}_\epsilon \left[\sup_{v \in \mathcal{V}} \langle \epsilon, v \rangle \right],$$

---

1. https://github.com/KrisKorrel/sparsemax-pytorch
2. Here, by abuse of notation, we regard $\mathcal{F}$ as a subset of $\mathbb{R}^N$ by the identity $\{f(X) \mid f \in \mathcal{F}\} \subset \mathbb{R}^N$ although it is a set of functions from $\mathcal{X}$ to $\mathcal{Y}$.

where $\epsilon = (\epsilon_1, \ldots, \epsilon_N)$ is a sequence of i.i.d. Rademacher variables with distribution $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = p$ and $\mathbb{P}(\epsilon_i = 0) = 1 - 2p$ and c_0 is a constant. Following Oono and Suzuki (2020b), the generalization error bound holds for the special case of $p = p_0 = MU/(M+U)^2$.

For the ease of analysis, we consider unscaled weight in (4) with $\beta_l = 1.0$ and a single $\alpha_l = \alpha \in (0, 1)$ for all layers. We consider a predefined $\mu^{(l)} \in [0,1]^K$ with $\sum_{k=0}^{K-1} \mu_k^{(l)} = 1$ for each layer l to construct layer-wise a GPR as $\tilde{A}(\mu^{(l)}) := \sum_{k=0}^{K-1} \mu_k^{(l)} \tilde{A}^k$. Let us define $C_0, \ldots, C_L \in \mathbb{N}_+$ with $C_0 = q$, $C_1 = \cdots = C_{L-1} = h$ and $C_L = 1$ to represent the dimensions of hidden layers and the output of AdaGPR. We define the hypothesis class for AdaGPR fr semi-supervised node-classification as

$$\mathcal{F} = \Big\{ X \mapsto f^{(L)} \circ \cdots \circ f^{(1)}(X) \ \Big| \ f^{(l)}(\cdot) = \sigma\Big(((1-\alpha)\tilde{A}(\mu^{(l)})(\cdot) + \alpha\sigma(XW^{(0)}))W^{(l)}\Big),$$
$$\|W_{\cdot c}^{(l)}\|_1 \le B^{(l)} \text{ for all } c \in [C_{l+1}] \Big\}, \tag{7}$$

where $W^{(l)} \in \mathbb{R}^{C_l \times C_{l+1}}$ $l = 0, \ldots, L$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a 1-Lipschitz function such that $\sigma(0) = 0$ with bounded output[3] as $|\sigma(\cdot)| \le R$, and $B^{(l)}$ $l = 0, \ldots, L$ are constants. We point out that $\sigma(\cdot)$ can be a ReLU (with output clipping) or a sigmoid function.

Analysing the Rademacher complexity allows up to obtain a data dependent bounds for our proposed model. Theorem 1 gives the Rademacher complexity for the AdaGPR.

Theorem 1 *Given the hypothesis class $\mathcal{F}$, the Rademacher complexity of the AdaGPR is bounded by*

$$\mathcal{R}(\mathcal{F}, p_0) \le QC' \Bigg\{ \sqrt{\frac{2MU}{(M+U)^2}} B^{(0)} \alpha \Bigg[\sum_{l=1}^{L} (1-\alpha)^l 2^l \prod_{j=0}^{l-1} B^{(L-j)} \Big(\sum_{i=1}^{N} \sum_{k=0}^{K-1} \mu_k^{(L-j)} |\lambda_i|^k \Big) \|X\|_{\mathrm{F}} \Bigg]$$
$$+ \sum_{l=1}^{L} (1-\alpha)^{l+1} 2^l \times \prod_{j=0}^{l} \Bigg[B^{(L-j)} \Big(\sum_{i=1}^{N} \sum_{k=0}^{K-1} \mu_k^{(L-j)} |\lambda_i|^k \Big) \Bigg] D \Bigg\}, \tag{8}$$

where λ_i is the ith largest eigenvalue of $\tilde{A}$, $D = \sqrt{N}R$, and C' is a universal constant.

We extend the hypothesis class (7) to derive the hypothesis class for GCNII by setting $\mu_1^{(l)} = 1.0$, $l = 1, \ldots, L$ and obtain the Rademacher complexity for GCNII given in the Corollary 1.

Corollary 1 *The Rademacher complexity of the GCNII is bounded as*

$$\mathcal{R}(\mathcal{F}, p_0) \le QC' \Bigg\{ \sqrt{\frac{2MU}{(M+U)^2}} B^{(0)} \alpha \sum_{l=1}^{L} 2^l (1-\alpha)^l \prod_{j=0}^{l-1} B^{(L-j)} \Big(\sum_{i=1}^{N} |\lambda_i| \Big) \|X\|_{\mathrm{F}} +$$
$$\sum_{l=1}^{L} (1-\alpha)^{l+1} 2^l \prod_{j=0}^{l} \Bigg[B^{(L-j)} \Big(\sum_{i=1}^{N} |\lambda_i| \Big) \Bigg] D \Bigg\}, \tag{9}$$

where λ_i is the ith largest eigenvalue of $\tilde{A}$, $D = \sqrt{N}R$, and C' is a universal constant.

---

3. This is just a technical condition to ensure the input to each layer is bounded.

Properties	Cora	Citeseer	Pubmed	Chameleon	Cornell	Texas	Wisconsin
Classes	7	4	3	4	5	5	5
Nodes	2708	3327	19717	2277	183	183	251
Edges	5429	4732	44338	36101	295	309	499
Features	1433	3703	500	2325	1703	1703	1703

Table 1: Properties of datasets used for node-classification

The proof is given in Appendix A. We notice that the bounds (8) and (9) are characterized by the spectrum of $\tilde{A}$. It shows that the mixing speed of information by node aggregations at each layer affects the model complexity. The use of the normalized adjacency matrix results in a eigenvalue spectrum of $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq -1$ and as k increases the summations of the eigenvalue spectrum with the higher powers shrink quickly. With a large k, the Rademacher complexity may become small which coincides with the intuition that the oversmoothing effect makes the model "simpler" and gives smaller generalization gap. Our bound successfully characterizes such an effect through the spectrum information which represents how fast the node features are mixed by aggregation. On the other hand, multiple applications of node aggregations would induce strong oversmoothing and results in underfitting (while the generalization gap is small). Because of it, the model complexity is uniformly bounded even there are many additional multiple node aggregation terms. Our proposed method automatically finds the appropriate weight that fits the data well. It is also important to notice that the deeper layers have a strong influence on the overall generalization bound due to recursive summations, which is coming from the input injection ($H^{(0)}$) to every layer. This indicates that the recursive multiplications of the spectral components in deeper layers induce stronger bias (although it simultaneously yields smaller generalization gap). Hence, in order to have less oversmoothing and to have a small overall bias less graph convolutions are preferred at deep layers. This observation agrees with the experimental results (Table 4).

The characterization by the spectrum is beneficial especially for large node-degree graphs. Indeed, a PAC-Bayesian bound for GCNs given by Liao et al. (2020) includes $d^{k/2}$ instead of $|\lambda_i|^k$ where d is the maximum node degree of the graph. Such a bound becomes loose for large degree d. However, a graph with large degree likely to have small spectrum λ_i (because it can "mix" the information rapidly) and thus our bound gives a tighter bound, which is contrary to the existing bound. Other bounds (e.g., by Oono and Suzuki (2020b)) are merely characterized by the spectral norm of the weight matrices, but our bound is characterized by not only the spectral norm B but also the spectrum of the node aggregation.

5. Experiments

In this section we discuss node classification experiments that we carried out to evaluate AdaGPR. Additionally, we discuss the behaviour of layer-wise sparse solutions of generalised Pagerank coefficients to understand the adaptive behaviour of AdaGPR.

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 (hidden 64)	82.2	82.6	84.2	84.6	85.4	**85.5**
	GCNII* (hidden 64)	80.2	82.3	82.8	83.5	84.9	**85.3**
	AdaGPR (hidden 32, GPR coeffs. 4)	83.8	84.5	84.8	85.0	**85.0**	85.0
Citeseer	GCN	**70.8**	67.7	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	**72.6**	71.0
	Incep(Drop)	-	**72.7**	71.4	72.5	72.6	71.0
	GCNII (hidden 256)	68.2	68.9	70.6	72.9	**73.4**	73.4
	GCNII* (hidden 256)	66.1	67.9	70.6	72.0	**73.2**	73.1
	AdaGPR (hidden 64, GPR coeffs. 16)	59.9	68.6	73.2	**73.5**	73.4	73.1
Pubmed	GCN	**79.0**	76.5	60.1	40.9	22.4	35.5
	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**	78.5
	IncepGCN	-	77.7	77.9	74.9	OOM	OOM
	IncepGCN(Drop)	-	**79.5**	78.6	79.0	OOM	OOM
	GCNII (hidden 256)	78.2	78.8	79.3	**80.2**	79.8	79.7
	GCNII* (hidden 256)	77.7	78.2	78.8	**80.3**	79.8	80.1
	AdaGPR (hidden 128, GPR coeffs. 4)	78.3	78.8	79.4	**79.6**	79.3	OOM

Table 2: Accuracy for semi-supervised node classification

5.1. Setup

We performed semi-supervised and fully-supervised node classification. Datasets and their properties used in our experiments are listed in Table 1. Since our method stems from GCNII, we used a similar experimental setting as in Chen et al. (2020) and borrowed their reported results for baseline methods. In addition to the hyperparameters α_l, λ_l, weight decays, and dropout rates common with GCNII, the number of GPR coefficients K and in some cases (semi-supervised learning) weight decay for learning weights $v^{(l)}$ in (4) are considered as hyperparameters. We tuned hyperparameters based on the loss over the validation sets. The optimization method for all experiments is Adam with learning rate of 0.01. We use the publicly available processed data provided by Chen et al. (2020). Further, we use code from Chen et al. (2020) to assist our implementations. The data and Pytorch based implementation of AdaGPR is available at https://github.com/kishanwn/adaGPR. We carried out experiments on NVidia V100-PCIE-16GB GPUs hosted on Intel Xeon Gold 6136 processor servers.

5.2. Semi-Supervised Node Classification

Method	Dataset						
	Cora	Citeseer	Pubmed	Chameleon	Cornell	Texas	Wisconsin
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**	80.05	60.31	56.76	57.58	58.24
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 (64)	57.30 (4)	56.49 (32)	48.82 (8)
JKNet(Drop)	87.46 (16)	75.96 (8)	89.45 (64)	62.08 (64)	61.08 (4)	57.30 (32)	50.59 (8)
IncepGCN(Drop)	86.86 (8)	76.83 (8)	89.18	61.71 (4)	61.62 (16)	57.84 (8)	50.20 (8)
GPR-GNN	88.16	77.39	85.8	63.22	78.37	77.30	81.57
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)
AdaGPR	88.19 (64,3)	77.25 (64,4)	90.23 (4,3)	**64.71 (2,3)**	**82.70 (4,2)**	**81.08 (4,4)**	**83.53 (16,3)**

Table 3: Accuracy for fully-supervised node classification

We used the commonly used citation datasets Cora, Citeseer, and Pubmed to evaluate performance of semi-supervised node classification. These datasets are split based on the commonly used the setting in Yang et al. (2016) that results in training sets with 20 nodes per each class, test sets with 500 nodes, and validation sets with 1000 nodes. The number of coefficients of the GPR is considered a hyperparameter and selected from $(2, 3, 4, 8, 16)$. We used the same hyperparameter ranges as in Chen et al. (2020) for λ, and dropout rates from $(0.1, \ldots, 0.9)$. We fixed $\alpha = 0.1$ following Chen et al. (2020).

We used separate weight decay rates for different learning weights in AdaGPR; $WD_1 \in (1.0, 0.1, 0.01, \ldots, 0.0001)$ for $W^{(0)}$, $WD_2 = 0.0001$ for $W^{(l)}$, $l = 1, \ldots, L$, and $WD_3 \in (1.0, 0.1, 0.01)$ for $v^{(l)}$ $l = 1, \ldots, L$. WD_1 and WD_3 are selected from hyperparameter tuning (see Section 1 of the supplementary materials section for details). We borrowed results for baseline methods Vanilla GCN (Kipf and Welling, 2017), JKNet (Xu et al., 2018), IncepGCN (Rong et al., 2020), and GCNII (Chen et al., 2020) from Chen et al. (2020).

The Table 2 shows that classification accuracies for Cora using AdaGPR did not outperform the accuracy produced by GCNII. However, AdaGPR has produced better performances for shallow networks with layers ranging from 2 to 16 compared to GCNII. AdaGPR achieved a slightly improved accuracy for Citeseer compared to GCNII. The noteworthy observation is that AdaGPR provides the best accuracy of 73.5 with 16 layers and 32 hidden units compared to the GCNII which used 32 layers and 256 hidden units. AdaGPR obtained a slightly lower accuracy for Pubmed compared to GCNII. The stable accuracies with the increase in depth for all datasets show robustness against oversmoothing of AdaGPR.

5.3. Fully-Supervised Node Classification

We experimented with fully-supervised node classification using the standards baseline graph datasets of Cora, Citeseer, Pubmed, Chameleon, Cornell, Texas, and Wisconsin. As suggested in Pei et al. (2020), all these datasets were randomly split into training, validation and testing sets consisting of nodes by each class with percentages of 60%, 20%, and 20%, respectively. We ran experiments over 10 different random splits as used in Chen et al. (2020). For fair comparisons with Chen et al. (2020) we used the 64 hidden units for all methods. Hyperparameter sets for dropout rates, and K are same as fully-supervised learning. Similar to Chen et al. (2020), we used a single weight decay selected from the set $(0.001, 0.0005, \ldots, 1e{-}6)$, $\alpha \in (0.1, \cdots, 0.9)$, and $\lambda \in (0.5, 1.0, 1.5)$.

The mean accuracy for node classification of AdaGPR and baseline methods (borrowed from Chen et al. (2020)) are shown in the Table 3. These baseline methods are Vanilla GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2017), Geom-GCN (Pei et al., 2020), APPNP (Klicpera et al., 2019), JKNet (Xu et al., 2018), IncepGCN (Rong et al., 2020), and GCNII (Chen et al., 2020). We also experimented with GPR-GNN whose results are included in Table 3. In addition to accuracy of AdaGPR, we show the number of layers and number of GPR coefficients (K) in brackets that were selected from the hyperparameter tuning.

Form Table 3 we can see that AdaGPR has obtained comparable accuracies compared to GCNII for Cora, Citeseer, and Pubmed. Chameleon dataset has a similar number of nodes as with Cora and Citeseer (Table 1), however, it has a larger number of edges compared to Cora and Citeseer. This indicates that Chameleon has a dense adjacency matrix compared to Cora and Citeseer, which may lead to faster oversmoothing with multiple convolutions. This observation is reflected in AdaGPR model with 2 layers and 3 coefficients giving the best accuracy for Chameleon. Notice that GCNII also has used a smaller network (8 layers) for Chemeleon compared to other datasets. Further, it is noteworthy that GPR-GNN which is another shallow model has gained a accuracy comparable to AdaGPR for Chemeleon.

There is a significant high accuracy for the three small scale datasets of Cornell, Texas, and Wisconsin with AdaGPR compared to all the baseline methods. Again, we can see that the increased performance with AdaGPR are achieved for Cornell and Texas with less number of convolution layers compared to GCNII. These observations provide evidence that adaptive GPR can perform model compression while enhancing prediction accuracy.

5.4. Layer-wise GPR Adaptation

We can quantitatively understand the amount of convolution by different orders of the normalized adjacency matrix at each layer by analysing the coefficients of each generalized Pagerank. In order to demsntrate layer-wise adaptation, we show coefficients of each generalized Pagerank at each layer for Cornell in Table 4. Notice the clear lawer-wise adaptation where only the first two layers apply graph convolutions with gradual decrease of the GPR from shallow layers to deeper layers and the last two layers of the trained model have no graph convolution.

Layers	GPR Coeff.	
	0	1
1	0.5150	0.4849
2	0.8581	0.1418
3	1	0
4	1	0

Table 4: GPR coefficients of Cornell

6. Conclusions

We proposed the AdaGPR to perform layer-wise adaptive graph convolution using generalized Pageranks within GCNII models. We provide generalization bounds to analyse the relationship between eigenvalue spectrum of a graph and the depth of the network and its effect on oversmoothing. We evaluate our proposed method using benchmark node-classification datasets to show performance improvements compared to other GCN models. By analysing coefficients of the generalized Pagerank in the trained models, we confirm that adaptive behaviour of graph convolution in each layer.

Acknowledgement

References

Radoslaw Adamczak. A note on the hanson-wright inequality for random vectors with dependencies. *Electronic Communications in Probability*, 20:1–13, 2015.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 1998. ISSN 0169-7552. WWW.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*. PMLR, 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *ICML*, 2021.

Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its applications. *J. Artif. Int. Res.*, 35(1):193–234, June 2009. ISSN 1076-9757.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, ICLR '17, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR 2019*, 2019.

Chang Li and Dan Goldwasser. Encoding social information with graph convolutional networks for Political perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Pan Li, Eli Chien, and Olgica Milenkovic. Optimizing generalized pagerank methods for seed-expansion community detection. *NeurIPS*, 32, 2019. ISSN 1049-5258.

Renjie Liao, Raquel Urtasun, and Richard S. Zemel. A pac-bayesian approach to generalization bounds for graph neural networks. In *NeurIPS*, 2020.

André F. T. Martins and Ramón F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, ICML'16, 2016.

Yimeng Min, Frederik Wenkel, and Guy Wolf. Scattering GCN: overcoming oversmoothness in graph convolutional networks. *CoRR*, abs/2003.08414, 2020.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR 2020*, 2020a.

Kenta Oono and Taiji Suzuki. Optimization and generalization analysis of transduction through gradient boosting and application to multi-scale graph neural networks. In *NeurIPS 2020*, 2020b.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR 2020*, ICLR'20, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR 2020*, 2020.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham, 2018. Springer International Publishing.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *6th ICLR*, 2017.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, volume 80, pages 5453–5462, 10–15 Jul 2018.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. ICML'16, page 40–48, 2016.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD '18*, KDD '18, 2018.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *ICLR*, 2020.

Appendix A. Proofs of Generalization Bounds

In this section we provide detailed proofs of Theorems given in the Section 4. The following transductive Rademacher complexity is defined in El-Yaniv and Pechyony (2009).

Definition 1 *Given $p \in [0, 0.5]$ and $\mathcal{V} \subset \mathbb{R}^N$, the transductive Rademacher complexity is defined as*

$$\mathcal{R}(\mathcal{V}, p) = Q \mathbb{E}_\epsilon \left[\sup_{v \in \mathcal{V}} \langle \epsilon, v \rangle \right],$$

where $Q = \frac{1}{M} + \frac{1}{N}$ and $\epsilon = (\epsilon_1, \ldots, \epsilon_N)$ is a sequence of i.i.d. Rademacher variables with distribution $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = p$ and $\mathbb{P}(\epsilon_i = 0) = 1 - 2p$.

Below we restate the symmetric Rademacher complexity (Oono and Suzuki, 2020b), a variant of the above tranductive Rademacher complexity.

Definition 2 *Given $p \in [0, 0.5]$ and $\mathcal{V} \subset \mathbb{R}^N$, the symmetric transductive Rademacher complexity is defined as*

$$\bar{\mathcal{R}}(\mathcal{V}, p) = Q \mathbb{E}_\epsilon \left[\sup_{v \in \mathcal{V}} |\langle \epsilon, v \rangle| \right],$$

where $Q = \frac{1}{M} + \frac{1}{N}$ and $\epsilon = (\epsilon_1, \ldots, \epsilon_N)$ is a sequence of i.i.d. Rademacher variables with distribution $\mathbb{P}(\epsilon_i = 1) = \mathbb{P}(\epsilon_i = -1) = p$ and $\mathbb{P}(\epsilon_i = 0) = 1 - 2p$.

In (Oono and Suzuki, 2020b), it has been shown that $\mathcal{R}(\mathcal{V}, p) \leq \bar{\mathcal{R}}(\mathcal{V}, p)$.

Below we provide the proof for the Theorem 1.

Proof of Theorem 1. We use the symmetric Rademacher complexity

$$\bar{\mathcal{R}}(\mathcal{F}, p) = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\sup_{v \in \mathcal{F}} |\langle \boldsymbol{\epsilon}, v \rangle| \right], \tag{10}$$

which upper bounds the Rademacher complexity $\mathcal{R}(\mathcal{F}, p)$ in (9) as $\mathcal{R}(\mathcal{F}, p) \leq \bar{\mathcal{R}}(\mathcal{F}, p)$. We give the bound for a general p. The assertion can be obtained by substituting $p \leftarrow p_0$.

In the rest of the proof we abbreviate row s of any matrix Z by $Z_s := Z[s, :]$, columns c by $Z_{\cdot c} := X[:, c]$, and an element by $Z_{sc} := Z[s, c]$. For the convenience of analysis we break the hypothesis class $\mathcal{F}$ in (10) into different components and define

$$\mathcal{H}^{(0)} = \left\{ \sum_{c=1}^{C_0} X_{\cdot c} w_c^{(0)} \| \|w_c^{(0)}\|_1 \leq B^{(0)} \right\},$$
$$\tilde{\mathcal{H}}^{(0)} = \sigma \circ \mathcal{H}^{(0)},$$

$$\mathcal{H}^{(l+1)} = \left\{ \sum_{c=1}^{C_{l+1}} ((1-\alpha)[\tilde{A}(\mu^{(l)})Z]_{\cdot c} + \alpha H_{\cdot c}) w_c^{(l)} | Z_{\cdot c} \in \tilde{\mathcal{H}}^{(l)}, H_{\cdot c} \in \tilde{\mathcal{H}}^{(0)}, \|w^{(l)}\|_1 \leq B^{(l)} \right\},$$
$$\tilde{\mathcal{H}}^{(l+1)} = \sigma \circ \mathcal{H}^{(l)} \ l = 1, \ldots, L.$$

Now, for a given layer $l + 1$, we have

$$Q^{-1}\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(l+1)}, p) = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{\|w^{(l)}\|_1 \leq B^{(l)}, Z_{\cdot c} \in \mathcal{H}^{(l)}, H_{\cdot c} \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^{N} \epsilon_n \sum_{c=1}^{C_{l+1}} ((1-\alpha)[\tilde{A}(\mu^{(l)})Z]_{nc} + \alpha H_{nc})w_c^{(l)})\right|\right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{\|w^{(l)}\|_1 \leq B^{(l)}, Z_{\cdot c} \in \mathcal{H}^{(l)}, H_{\cdot c} \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{c=1}^{C_{l+1}} \sum_{n=1}^{N} \epsilon_n ((1-\alpha)[A(\mu^{(l)})Z]_{nc} + \alpha H_{nc})w_c^{(l)})\right|\right]$$

$$= B^{(l)}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{Z \in \mathcal{H}^{(l)}, H \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^{N} \epsilon_n ((1-\alpha)[\tilde{A}(\mu^{(l)})Z]_n + \alpha H_n)\right|\right]$$

$$= B^{(l)}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{Z \in \mathcal{H}^{(l)}, H \in \mathcal{H}^{(0)}} \left|(1-\alpha)\sum_{n=1}^{N} \epsilon_n [\tilde{A}(\mu^{(l)})Z]_n + \alpha \sum_{n=1}^{N} \sigma_n H_n)\right|\right]$$

$$= B^{(l)}(1-\alpha)\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{Z \in \mathcal{H}^{(l)}} \left|\sum_{n=1}^{N} \epsilon_n [\tilde{A}(\mu^{(l)})Z]_n\right|\right] + \alpha B^{(l)}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{H \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^{N} \epsilon_n H_n\right|\right]. \tag{11}$$

Let $\epsilon' = (\epsilon_1', \ldots, \epsilon_N')$ be a random variable that is independent to and has the identical distribution as ϵ. Then, we have that

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{Z \in \mathcal{H}^{(l)}} \left|\sum_{n=1}^{N} \epsilon_n [\tilde{A}(\mu^{(l)})Z]_n\right|\right] = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{Z \in \mathcal{H}^{(l)}} \left|\sum_{n=1}^{N} \epsilon_n [\tilde{A}(\mu^{(l)})\frac{1}{2p}\mathbb{E}_{\epsilon'}[\epsilon'\epsilon'^\top]Z]_n\right|\right] \quad (\because \mathbb{E}_{\epsilon'}[\epsilon'\epsilon'^\top] = 2pI)$$

$$\leq \frac{1}{2p}\mathbb{E}_{\epsilon,\epsilon'}\left[\sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon^\top \tilde{A}(\mu^{(l)})\epsilon'\epsilon'^\top Z\right|\right] \leq \frac{1}{2p}\mathbb{E}_{\epsilon,\epsilon'}\left[\left|\epsilon^\top \tilde{A}(\mu^{(l)})\epsilon'\right| \sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right]$$

$$= \frac{1}{2p}\mathbb{E}_{\epsilon'}\left[\mathbb{E}_{\epsilon}\left[\left|\epsilon^\top \tilde{A}(\mu^{(l)})\epsilon'\right|\right] \sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right]$$

$$\leq \frac{1}{2p}\mathbb{E}_{\epsilon'}\left[\sqrt{\mathbb{E}_{\epsilon}\left[\left(\epsilon^\top \tilde{A}(\mu^{(l)})\epsilon'\right)^2\right]} \sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right] = \frac{1}{2p}\mathbb{E}_{\epsilon'}\left[\sqrt{\mathbb{E}_{\epsilon}\left[\epsilon'^\top \tilde{A}(\mu^{(l)})\epsilon\epsilon^\top \tilde{A}(\mu^{(l)})\epsilon'\right]} \sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right]$$

$$= \mathbb{E}_{\epsilon'}\left[\sqrt{\epsilon'^\top \tilde{A}(\mu^{(l)})^2\epsilon'} \sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right],$$

where we used $\mathbb{E}_{\epsilon}[\epsilon\epsilon^\top] = 2pI$ in the last equation. Here, by the Hanson-Wright concentration inequality (see, for example, Theorem 2.5 of (Adamczak, 2015)) implies that

$$\mathbb{P}[|\epsilon'^\top \tilde{A}(\mu^{(l)})^2\epsilon' - \mathbb{E}_{\epsilon'}[\epsilon'^\top \tilde{A}(\mu^{(l)})^2\epsilon']| \geq c(\sqrt{2p}\|\tilde{A}(\mu^{(l)})^2\|_F\sqrt{t} + \|\tilde{A}(\mu^{(l)})^2\|t)] \leq \exp(-t) \quad (t > 0),$$

with a universal constant c, where $\|A\|_F = \sqrt{\text{Tr}[AA^\top]}$[4]. Moreover, Talagrand's concentration inequality yields

$$\mathbb{P}\left[\left|\sup_{Z \in \mathcal{H}^{(l)}} \epsilon'^\top Z\right| \geq c'\left(\mathbb{E}_{\epsilon'}\left[\sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right] + \sqrt{Nt \sup_{Z \in \mathcal{H}^{(l)}} \sum_{n=1}^{N} Z_n^2/N + t \sup_{Z \in \mathcal{H}^{(l)}} \|Z\|_\infty}\right)\right] \leq e^{-t} \quad (t > 0),$$

---

4. There also exists a uniform type Hanson-Wright inequality.

where $c' > 0$ is a universal constant. Then, by noticing that $\mathbb{E}_{\epsilon'}[\epsilon'^\top A \epsilon'] = 2p \text{Tr}[A]$, these inequalities yield

$$\mathbb{E}_{\epsilon'}\left[\sqrt{\epsilon'^\top \tilde{A}(\mu^{(l)})^2 \epsilon'} \sup_{Z \in \mathcal{H}^{(l)}} \left|\epsilon'^\top Z\right|\right]$$
$$\leq \int \sqrt{2p\text{Tr}[\tilde{A}(\mu^{(l)})^2] + c(\sqrt{2p}\|\tilde{A}(\mu^{(l)})^2\|_F \sqrt{t} + \|\tilde{A}(\mu^{(l)})^2\|t)}$$
$$c'(|\mathbb{E}_{\epsilon'} \sup_{Z \in \mathcal{H}^{(l)}} \epsilon'^\top Z| + \sqrt{t}\|\mathcal{H}^{(l)}\|_2 + t\|\mathcal{H}^{(l)}\|_\infty)2\exp(-t)\mathrm{d}t$$

where we define $\|\mathcal{H}^{(l)}\|_* := \sup_{Z \in \mathcal{H}^{(l)}} \|Z\|_*$ for $* = 2$ and ∞. The right hand side can be further bounded as $C\sqrt{\text{Tr}[\tilde{A}(\mu^{(l)})^2]} \left(\mathbb{E}_{\epsilon'}\left[\sup_{Z \in \mathcal{H}^{(l)}} |\epsilon'^\top Z|\right] + \|\mathcal{H}^{(l)}\|_2\right)$ for a universal constant C, where we used $2p \leq 1$.

Since the output is bounded by the assumption on the activation function, we have $\sup_{Z \in \mathcal{H}^{(l)}} \|Z\|_2 \leq \sqrt{N}R =: D$. Now substituting the above result back to (11), we have

$$Q^{-1}\mathcal{R}(\tilde{\mathcal{H}}^{(l+1)}, p)$$
$$\leq B^{(l)}(1-\alpha)\left[C\sqrt{\text{Tr}[\tilde{A}(\mu^{(l)})^2]}\left(\mathbb{E}_{\epsilon} \sup_{Z \in \mathcal{H}^{(l)}} \left|\sum_{n=1}^N \epsilon_n Z_n\right| + D\right)\right] + \alpha B^{(l)}\mathbb{E}_{\epsilon}\left[\sup_{H \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^N \epsilon_n H_n\right|\right]$$
$$\leq B^{(l)}(1-\alpha)\left[C\sqrt{\sum_{i=1}^N \left(\sum_{k=0}^{K-1} \mu_k^{(l)} \lambda_i^k\right)^2}\left(\mathbb{E}_{\epsilon} \sup_{Z \in \mathcal{H}^{(l)}} \left|\sum_{n=1}^N \epsilon_n Z_n\right| + D\right)\right]$$
$$+ \alpha B^{(l)}\mathbb{E}_{\epsilon}\left[\sup_{H \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^N \epsilon_n H_n\right|\right]$$
$$= CB^{(l)}(1-\alpha)\left[\left(\sum_{i=1}^N \sum_{k=0}^{K-1} \mu_k^{(l)}|\lambda_i|^k\right)\left(Q^{-1}\mathcal{R}(\mathcal{H}^{(l)}, p) + D\right)\right] + \alpha B^{(l)}\mathbb{E}_{\epsilon}\left[\sup_{H \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^N \epsilon_n H_n\right|\right],$$
$$(12)$$

where λ_t is the t-th eigenvlaue of $\tilde{A}$ and we have used the used $\sqrt{\sum_{i=1}^N (\sum_{k=0}^{K-1} \mu_k^{(l)} \lambda_i^k)^2} \leq \sum_{i=1}^N \sqrt{(\sum_{k=0}^{K-1} \mu_k^{(l)} \lambda_i^k)^2} = \sum_{i=1}^N |\sum_{k=0}^{K-1} \mu_k^{(l)} \lambda_i^k| \leq \sum_{i=1}^N \sum_{k=0}^{K-1} \mu_k^{(l)}|\lambda_i|^k$.

Since σ is 1-Lipschitz using the contraction property from Proposition 10 of (Oono and Suzuki, 2020b), we have
$$\bar{\mathcal{R}}(\mathcal{H}^{(l+1)}, p) \leq 2\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(l+1)}, p),$$

leading to reduction in (12) to

$$Q^{-1}\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(l+1)}, p) \leq C2B^{(l)}(1-\alpha)\left[\left(\sum_{i=1}^N \sum_{k=0}^{K-1} \mu_k^{(l)}|\lambda_i|^k\right)\left(Q^{-1}\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(l)}, p) + D\right)\right]$$
$$+ \alpha B^{(l)}\mathbb{E}_{\epsilon}\left[\sup_{H \in \tilde{\mathcal{H}}^{(0)}} \left|\sum_{n=1}^N \epsilon_n H_n\right|\right]. \quad (13)$$

Given that $\mathcal{F} = \tilde{\mathcal{H}}^{(L)}$, the final reduction using (13) leads to

$$Q^{-1}\bar{\mathcal{R}}(\mathcal{F},p) \leq C'\alpha \sum_{l=1}^{L}(1-\alpha)^l 2^l \prod_{j=0}^{l-1}\left\{B^{(L-j)}\left(\sum_{i=1}^{N}\sum_{k=0}^{K-1}\mu_k^{(L-j)}|\lambda_i|^k\right)\right\}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{H\in\tilde{\mathcal{H}}^{(0)}}\left|\sum_{n=1}^{N}\epsilon_n H_n\right|\right]$$
$$+ C'(1-\alpha)\sum_{l=1}^{L}(1-\alpha)^l 2^l \prod_{j=0}^{l}B^{(L-j)}\left(\sum_{i=1}^{N}\sum_{k=0}^{K-1}\mu_k^{(L-j)}|\lambda_i|^k\right)D. \quad (14)$$

By construction of $\mathcal{F}$, we know that $\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(0)},p) = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{H\in\tilde{\mathcal{H}}^{(0)}}\left|\sum_{n=1}^{N}\epsilon_n H_n\right|\right]$, and we have that

$$\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(0)},p) \leq 2\bar{\mathcal{R}}(\mathcal{H}^{(0)},p) = 2\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{w\in\mathbb{R}^{C_0}:\|w\|_1\leq B^{(0)}}\left|\sum_{n=1}^{N}\sum_{c=1}^{C_0}\epsilon_n X_{nc}w_c\right|\right]$$
$$= 2B^{(0)}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\max_{c\in[C_0]}\left|\sum_{n=1}^{N}\epsilon_n X_{nc}\right|\right] \leq 2B^{(0)}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\left\|\sum_{n=1}^{N}\epsilon_n X_{n\cdot}\right\|_2\right]$$
$$\leq 2B^{(0)}\sqrt{\mathbb{E}_{\boldsymbol{\epsilon}}\sum_{c=1}^{C_0}\left(\sum_{n=1}^{N}\epsilon_n X_{nc}\right)^2} \quad (\because \text{ Jensen Inequality})$$
$$= 2B^{(0)}\sqrt{\mathbb{E}_{\boldsymbol{\epsilon}}\sum_{c=1}^{C_0}\sum_{n,m=1}^{N}\epsilon_n\epsilon_m X_{nc}X_{mc}} = 2B^{(0)}\sqrt{\sum_{c=1}^{C_0}\sum_{m=1}^{N}2p(X_{mc})^2}$$
$$= 2B^{(0)}\sqrt{2p}\|X\|_{\mathrm{F}}.$$

Given that $p = p_0 = \frac{MU}{(M+U)^2}$, we have

$$\bar{\mathcal{R}}(\tilde{\mathcal{H}}^{(0)},p_0) \leq 2B^{(0)}\sqrt{\frac{2MU}{(M+U)^2}}\|X\|_{\mathrm{F}}. \quad (15)$$

By combining (15) with (14), the resulting final Rademacher complexity bound is given by

$$Q^{-1}\mathcal{R}(\mathcal{F},p_0) \leq Q^{-1}\bar{\mathcal{R}}(\mathcal{F},p_0)$$
$$\leq C'\left\{\sqrt{\frac{2MU}{(M+U)^2}}B^{(0)}\alpha\sum_{l=1}^{L}(1-\alpha)^l 2^l \prod_{j=0}^{l-1}\left\{B^{(L-j)}\left(\sum_{i=1}^{N}\sum_{k=0}^{K-1}\mu_k^{(L-j)}|\lambda_i|^k\right)\right\}\|X\|_{\mathrm{F}}\right.$$
$$\left. + (1-\alpha)\left[\sum_{l=1}^{L}(1-\alpha)^l 2^l \prod_{j=0}^{l}B^{(L-j)}\left(\sum_{i=1}^{N}\sum_{k=0}^{K-1}\mu_k^{(L-j)}|\lambda_i|^k\right)\right]D\right\}, \quad (16)$$

by redefining the universal constant C' if necessary.

GCNII *Proof of Corollary 1.* By replacing the generalized Pagerank $\tilde{A}(\mu)$ with the normalized adjacency matrix $\tilde{A}$, which is equivalent to setting $\mu_1^{(l)} = 1$ and rest of the elements in $\mu^{(l)}$ to zero, we obtain the desired result.