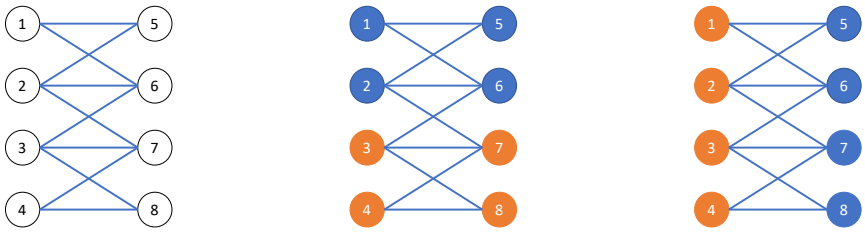
A APPENDIX

A.1 CONCLUSIONS

We addressed two fundamental weaknesses of existing GNNs: Failing to act as universal learners by not generalizing to heterophilic graphs and making use of large number of propagation steps. We developed a novel GPR-GNN architecture which combines adaptive generalized PageRank (GPR) scheme with GNNs. We theoretically showed that our method does not only mitigates feature over-smoothing but also works on highly diverse node label patterns. We also tested GPR-GNNs on both homophilic and heterophilic node label patterns, and proposed a novel synthetic benchmark datasets generated by the contextual stochastic block model. Our experiments on real-world benchmark datasets showed clear performance gains of GPR-GNN over the state-of-the-art methods. Moreover, we showed that GPR-GNN has desirable interpretability properties which could be of independent interest.

A.2 DETAILED DISCUSSION ON PREVENTING OVER-SMOOTHING.

As mentioned in Section 4, another method – APPNP – can also provably prevents over-smoothing Klicpera et al. (2018). The authors of this study use the fact that the PPR propagation will converge to $\mathbf{\Pi}_{\text{ppr}}\mathbf{H}^{(0)}$, where $\mathbf{\Pi}_{\text{ppr}} = \alpha(\mathbf{I}_n - (1-\alpha)\tilde{\mathbf{A}}_{\text{sym}})^{-1}$ is independent on the node label information provided in the training data. Each row of $\mathbf{\Pi}_{\text{ppr}}\mathbf{H}^{(0)}$ still depends on $\mathbf{H}^{(0)}$ and thus APPNP will not suffer from the over-smoothing effect. However, since $\mathbf{\Pi}_{\text{ppr}}$ is independent of the label information, it can cause undesired consequences that we discuss in what follows.



(a) A graph G. (b) One example of label assignment. (c) Another example of label assignment.

Figure 5: A simple example demonstrating how GPR-GNN escapes over-smoothing.

Let us consider a simple example shown in Figure 5 involving a connected and undirected graph $G = (V, E)$ (Figure 5 (a)). Consider two different node label assignments shown in Figure 5 (b) and Figure 5 (c). Obviously, the graph topologies depicted in Figure 5 (b) and (c) are identical and the only difference is the class label assignment. In Figure 5 (b), the graph is homophilic and hence the optimal graph filter should emphasize the low-frequency part of the graph signal. In contrast, in Figure 5 (c), the graph is heterophilic as the graph is bipartite with respect to the labels. Hence, the optimal graph filter should emphasize the high-frequency part of the graph signal. This example illustrates that the optimal graph filter should depend on both the graph topology and the node label information. Recall that the equivalent graph filter that APPNP uses in the asymptotic regime is $\mathbf{\Pi}_{\text{ppr}}$ which is independent on the node label information. Also, Theorem 4.1 established that APPNP intrinsically utilizes a low-pass filter. In contrast, GPR-GNN learns the GPR weights guided by the node label information which allows it to account for both cases (homophilic and heterophilic) shown.

A.3 PROOF OF THEOREM 4.1

We first state the formal version of Theorem 4.1.

Theorem A.1 (Formal version of Theorem 4.1)**.** *Assume the graph G is connected. Let $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ be the eigenvalues of $\tilde{\mathbf{A}}_{sym}$. If $\gamma_k \geq 0\ \forall k \in \{0, 1, ..., K\}$, $\sum_{k=0}^{K} \gamma_k = 1$ and $\exists k' > 0$ such that $\gamma_{k'} > 0$, then $|g_{\gamma,K}(\lambda_i)/g_{\gamma,K}(\lambda_1)| < |\lambda_i/\lambda_1|\ \forall i \geq 2$. Also, if $\gamma_k = (-\alpha)^k, \alpha \in (0, 1)$ and $K \to \infty$, then $|\lim_{K\to\infty} g_{\gamma,K}(\lambda_i)/\lim_{K\to\infty} g_{\gamma,K}(\lambda_1)| > |\lambda_i/\lambda_1|\ \forall i \geq 2$.*

Note that $|g_{\gamma,K}(\lambda_i)/g_{\gamma,K}(\lambda_1)| < |\lambda_i/\lambda_1|\ \forall i \geq 2$ implies that after applying the graph filter $g_{\gamma,K}$, the lowest frequency component (correspond to λ_1) further dominates. Hence $g_{\gamma,K}$ acts like a low pass filter in this case. In contrast, $|\lim_{K\to\infty} g_{\gamma,K}(\lambda_i)/\lim_{K\to\infty} g_{\gamma,K}(\lambda_1)| > |\lambda_i/\lambda_1|\ \forall i \geq 2$ implies that after applying the graph filter, the lowest frequency component (correspond to λ_1) no longer dominates. This correspond to the high pass filter case.

Proof. We start with the low pass filter result. From basic spectral analysis (Von Luxburg, 2007) we know that $\lambda_1 = 1$ and $|\lambda_i| < 1, \forall i \geq 2$. One can also find the analysis in the proof of our Lemma A.2 in the Supplement. Then by assumption we know that

$$g_{\gamma,K}(\lambda_1) = \sum_{k=0}^{K} \gamma_k = 1.$$

Hence, proving Theorem A.1 is equivalent to show

$$|g_{\gamma,K}(\lambda_i)| < |\lambda_i|\ \forall i \geq 2.$$

This is obvious since $g_{\gamma,K}(\lambda) = \sum_{k=0}^{K} \gamma_k \lambda^k$ is nothing but the polynomial of order K with all coefficient non-negative. It is easy to check that $\forall k \geq 1,\ |\lambda|^k < |\lambda|, \forall |\lambda| < 1$. Combine with the fact that all γ_k are non-negative we have

$$|g_{\gamma,K}(\lambda_i)| \leq \sum_{k=0}^{K} \gamma_k |\lambda^k| = \sum_{k=0}^{K} \gamma_k |\lambda|^k \overset{\text{(a)}}{\leq} \sum_{k=0}^{K} \gamma_k |\lambda| = |\lambda|.$$

Finally, note that the only possibility that the inequality (a) holds is $\gamma_k = \delta_{0,K}$ since $\forall k \geq 1,\ |\lambda|^k < |\lambda|, \forall |\lambda| < 1$. However, by assumption $\sum_{k=0}^{K} \gamma_k = 1$ and $\exists k' > 0$ such that $\gamma_{k'} > 0$ we know that this is impossible. Hence (a) is a strict inequality $<$. Together we complete the proof for low pass filtering part.

For the high pass filter result, it is not hard to see that

$$\lim_{K\to\infty} g_{\gamma,K}(\lambda) = \lim_{K\to\infty} \sum_{k=0}^{K} \gamma_k \lambda^k = \lim_{K\to\infty} \sum_{k=0}^{K} (-\alpha\lambda)^k = \frac{1}{1+\alpha\lambda},$$

where the last step is due to the fact that $\alpha \in (0, 1)$ and thus $\lim_{K\to\infty}(-\alpha\lambda)^K = 0, \forall |\lambda| \leq 1$. Thus we have

$$\left| \frac{\lim_{K\to\infty} g_{\gamma,K}(\lambda_i)}{\lim_{K\to\infty} g_{\gamma,K}(\lambda_1)} \right| = \left| \frac{1+\alpha}{1+\alpha\lambda_i} \right| \overset{\text{(b)}}{>} 1 \overset{\text{(c)}}{>} |\lambda_i|\ \forall i \geq 2.$$

Both strict inequalities (b) and (c) are from the fact that $|\lambda_i| < 1, \forall i \geq 2$. Notably, $\sup_{\lambda\in[1,-1)} \frac{1}{1+\alpha\lambda}$ happens at the boundary $\lambda = -1$, which corresponds the the bipartite graph. It further shows that the graph filter with respect to the choice $\gamma_k = (-\alpha)^k$ emphasizes high frequency components and thus it is indeed acting as a high pass filter. □

A.4 Proof of Theorem 4.2

We start by introducing some additional notation, lemmas and definition before we proceed to the formal statement of Theorem 4.2. The label matrix is denoted by $\mathbf{Y} \in \mathbb{R}^{n\times C}$, where each row is a one-hot vector. We use $\mathbf{1}[\boldsymbol{\beta}] \in \mathbb{R}^C$ to denote the argmax of the vector $\boldsymbol{\beta} \in \mathbb{R}^C$: we have $\mathbf{1}[\boldsymbol{\beta}]_i = 1$ if and only if $\boldsymbol{\beta}_i = \max(\boldsymbol{\beta})$ (ties are broken evenly), and $\mathbf{1}[\boldsymbol{\beta}]_i = 0$ otherwise. Let us replace the softmax$(\cdot)$ with softmax$_\eta(\cdot)$, where we let softmax$_\eta(\boldsymbol{\beta})_i = e^{\eta\boldsymbol{\beta}_i}/(\sum_j e^{\eta\boldsymbol{\beta}_j})$ stand for the softmax with a smooth parameter $\eta > 0$. Note that for $\eta = 1$ we recover the standard softmax. With a slight abuse of notation, for the vector $\boldsymbol{\beta}$ we write $\exp(\boldsymbol{\beta})$ to denote element-wise exponentiation. We use $\langle \cdot, \cdot \rangle$ to denote the standard Euclidean inner product. Also we use L for the cross entropy loss where

$$L = \sum_{i\in V} -\log(\left\langle \hat{\mathbf{P}}_{i:}, \mathbf{Y}_{i:} \right\rangle).$$

Lemma A.2. *Assume that the nodes in an undirected and connected graph G have one of C labels. Then, for k large enough, we have*

$$\mathbf{H}^{(k)}_{:j} = \boldsymbol{\beta}_j \boldsymbol{\pi} + o_k(1)\ \forall j \in [C],\ \textit{where}\ \boldsymbol{\pi}_i = \frac{\sqrt{\tilde{\mathbf{D}}_{ii}}}{\sqrt{\sum_{v\in V}\tilde{\mathbf{D}}_{vv}}}\ \textit{and}\ \boldsymbol{\beta}^T = \boldsymbol{\pi}^T\mathbf{H}^{(0)}. \tag{2}$$

For any $\mathbf{H}^{(0)}$ and large enough $k \leq K$, if the label prediction is dominated by $\mathbf{H}^{(k)}$, all nodes will have a representation proportional to $\gamma_k\beta$. Hence, we will arrive at the same label for all nodes. This is what we refer to as the over-smoothing phenomenon.

Definition A.3 (The over-smoothing phenomenon)**.** *First, recall that $\mathbf{Z} = \sum_k \gamma_k \mathbf{H}^{(k)}$. If over-smoothing occurs in the GPR-GNN for K large enough, we have $\mathbf{Z}_{:j} = c_0\boldsymbol{\beta}_j\boldsymbol{\pi},\ \forall j \in [C]$ for some $c_0 > 0$ if $\gamma_k > 0$ and $\mathbf{Z}_{:j} = -c_0\boldsymbol{\beta}_j\boldsymbol{\pi},\ \forall j \in [C]$ for some $c_0 > 0$ if $\gamma_k < 0$.*

Lemma A.4. *Let $L = \sum_{i\in\mathcal{T}} L_i = \sum_{i\in\mathcal{T}} -log(\left\langle \hat{\mathbf{P}}_{i:}, \mathbf{Y}_{i:} \right\rangle)$ be the cross entropy loss and let $\mathcal{T}$ be the training set. Under the same assumption as given in Lemma A.2, the gradient of γ_k for k large enough is $\frac{\partial L}{\partial \gamma_k} = \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left\langle \hat{\mathbf{P}}_{i:} - \mathbf{Y}_{i:}, \boldsymbol{\beta} \right\rangle + o_k(1)$.*

Lemma A.5. *For any real vector $\boldsymbol{\beta} \in \mathbb{R}^C$ and $\eta > 0$ large enough, we have $softmax_\eta(\boldsymbol{\beta}) = \mathbf{1}[\boldsymbol{\beta}] + o_\eta(1)$.*

Now we are ready to state the formal version of Theorem 4.2.

Theorem A.6 (Formal version of Theorem 4.2)**.** *Under the same assumptions as those listed in Lemma A.2, if the training set contains nodes from each class, then the GPR-GNN method can always avoid over-smoothing. More specifically, for k, η large enough we have*

$$\frac{\partial L}{\partial \gamma_k} = \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left(\max_{j\in[C]} \boldsymbol{\beta}_j - \boldsymbol{\beta}_{\mathbf{1}[\mathbf{Y}_{i:}]} \right) + o_k(1) + o_\eta(1),\ \textit{when}\ \gamma_k > 0. \tag{3}$$

$$\frac{\partial L}{\partial \gamma_k} = \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left(\min_{j\in[C]} \boldsymbol{\beta}_j - \boldsymbol{\beta}_{\mathbf{1}[\mathbf{Y}_{i:}]} \right) + o_k(1) + o_\eta(1),\ \textit{when}\ \gamma_k < 0. \tag{4}$$

Note that when $\gamma_k > 0$, (3) ≥ 0 when ignoring the $o(1)$ term. The equality is achieved if and only if $\max_{j\in[C]} \boldsymbol{\beta}_j = \boldsymbol{\beta}_{\mathbf{1}[\mathbf{Y}_{i:}]}$. This means that over-smoothing results in a prediction that perfectly aligns with the ground truth label in the training set. However, if our training set contains at least one node from each class then the equality can never be attained. Thus, the gradient of γ_k will always be positive when $\gamma_k > 0$. Similarly when $\gamma_k < 0$, (4) ≤ 0 when ignoring the $o(1)$ term. The equality is achieved if and only if $\min_{j\in[C]} \boldsymbol{\beta}_j = \boldsymbol{\beta}_{\mathbf{1}[\mathbf{Y}_{i:}]}$. By the same reason we know that under the assumption on training set the equality can never be attained. Thus, the gradient of γ_k will always be negative when $\gamma_k < 0$. Finally, it is not hard to check that the gradient is bounded in magnitude. Together we have shown that the gradient of γ_k and γ_k are of the same sign. This directly implies that $|\gamma_k|$ will approach to 0 until we escape from over-smoothing when we use a decreasing learning rate for the optimizer (i.e. SGD).

Proof. First, let us assume the over-smoothing takes place and the $\gamma_k > 0$ for the dominate term. By Definition A.3, we know that $\mathbf{Z}_{:j} = c_0\boldsymbol{\beta}_j\boldsymbol{\pi},\ \forall j \in [C]$ for some $c_0 > 0$ and K sufficiently large. By Lemma A.4 we have

$$\frac{\partial L}{\partial \gamma_k} = \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left\langle \frac{e^{\eta\mathbf{Z}_{i:}}}{\sum_{j\in[C]} e^{\eta\mathbf{Z}_{ij}}} - \mathbf{Y}_{i:}, \boldsymbol{\beta} \right\rangle + o_k(1) \tag{5}$$

$$= \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left\langle \frac{e^{\eta c_0\boldsymbol{\pi}_i\boldsymbol{\beta}}}{\sum_{j\in[C]} e^{\eta c_0\boldsymbol{\pi}_i\boldsymbol{\beta}_j}} - \mathbf{Y}_{i:}, \boldsymbol{\beta} \right\rangle + o_k(1), \tag{6}$$

where the last step follows from Definition A.3. Next, by Lemma A.5, we may approximate the softmax_η by the true argmax for $\eta > 0$ large enough according to

$$\sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \langle \mathbf{1}[c_0\boldsymbol{\pi}_i\boldsymbol{\beta}] - \mathbf{Y}_{i:}, \boldsymbol{\beta}\rangle + o_k(1) + o_\eta(1) \tag{7}$$

$$= \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \langle \mathbf{1}[\boldsymbol{\beta}] - \mathbf{Y}_{i:}, \boldsymbol{\beta}\rangle + o_k(1) + o_\eta(1) \tag{8}$$

$$= \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left(\max_{j\in[C]} \boldsymbol{\beta}_j - \boldsymbol{\beta}_{\mathbf{1}[\mathbf{Y}_{i:}]} \right) + o_k(1) + o_\eta(1). \tag{9}$$

The first equality is due to the fact that $c_0 > 0$ and $\boldsymbol{\pi}_i > 0$. Recall that by Lemma A.2, $\boldsymbol{\pi}_i = \frac{\sqrt{\tilde{\mathbf{D}}_{ii}}}{\sqrt{\sum_{v\in V}\tilde{\mathbf{D}}_{vv}}}$. Since we have a self- loop for each node, $\tilde{\mathbf{D}}_{ii} > 0$ and thus $\boldsymbol{\pi}_i > 0$. For the case $\gamma_k < 0$, the same analysis still valid until (7). Hence we have

$$\sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \langle \mathbf{1}[-c_0\boldsymbol{\pi}_i\boldsymbol{\beta}] - \mathbf{Y}_{i:}, \boldsymbol{\beta}\rangle + o_k(1) + o_\eta(1) \tag{10}$$

$$= \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \langle \mathbf{1}[-\boldsymbol{\beta}] - \mathbf{Y}_{i:}, \boldsymbol{\beta}\rangle + o_k(1) + o_\eta(1) \tag{11}$$

$$= \sum_{i\in\mathcal{T}} \eta\boldsymbol{\pi}_i \left(\min_{j\in[C]} \boldsymbol{\beta}_j - \boldsymbol{\beta}_{\mathbf{1}[\mathbf{Y}_{i:}]} \right) + o_k(1) + o_\eta(1). \tag{12}$$

Together we complete the proof.

□

A.5 cSBM DETAILS

The cSBM adds Gaussian random vectors as node features on top of the classical SBM. For simplicity, we assume $C = 2$ equally sized communities with node labels v_i in $\{+1, -1\}$. Each node i is associate with a f dimensional Gaussian vector $b_i = \sqrt{\frac{\mu}{n}} v_i u + \frac{Z_i}{\sqrt{f}}$ where n is the number of nodes, $u \sim N(0, I/f)$ and $Z_i \in \mathbf{R}^f$ has independent standard normal entries. The (undirected) graph in cSBM is described by the adjacency matrix $\mathbf{A}$ defined as

$$\mathbf{P}\left(\mathbf{A}_{ij} = 1\right) = \begin{cases} \frac{d+\lambda\sqrt{d}}{n} & \text{if } v_i v_j > 0 \\ \frac{d-\lambda\sqrt{d}}{n} & \text{otherwise} \end{cases}.$$

Similar to the classical SBM, given the node labels the edges are independent. The symbol d stands for the average degree of the graph. Also, recall that μ and λ control the information strength carried by the node features and the graph structure respectively.

One reason for using the cSBM to generate synthetic data is that the information-theoretic limit of the model is already characterized in Deshpande et al. (2018). This result is summarized below.

Theorem A.7 (Informal main result in Deshpande et al. (2018))**.** *Assume that $n, f \to \infty$, $\frac{n}{f} \to \xi$ and $d \to \infty$. Then there exists an estimator $\hat{v}$ such that $\liminf_{n\to\infty} \frac{|\langle \hat{v}, v\rangle|}{n}$ is bounded away from 0 if and only if $\lambda^2 + \frac{\mu^2}{\xi} > 1$.*

In our experiment, we set $n = 5000, f = 2000$ and thus have $\xi = 2.5$. We vary μ and λ along the arc $\lambda^2 + \mu^2/\xi = 1 + \epsilon$ for some $\epsilon > 0$ to ensure that we are in the achievable parameter regime. We also choose $\epsilon = 3.25$ for all our experiment.

A.6 PROOF OF LEMMA A.2

Note that the proof of Lemma A.2 reduces to a standard analysis of random walks on graph. We include it for completeness and refer the interested readers to the tutorial Von Luxburg (2007).

We start by showing that the symmetric graph Laplacian

$$\tilde{\mathbf{L}}_{\text{sym}} = \mathbf{I} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} = \mathbf{I} - \tilde{\mathbf{A}}_{\text{sym}} \tag{13}$$

is positive semi-definite. Let u be any real vector of unit norm and $f = \tilde{\mathbf{D}}^{-1/2}u$, then we have

$$u^T\tilde{\mathbf{L}}_{\text{sym}}u = u^Tu - u^T\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}u = \sum_{i=1}^{n} u_i^2 - \sum_{i,j=1}^{n} f_i f_j \tilde{\mathbf{A}}_{ij} \tag{14}$$

$$= \sum_{i=1}^{n} \tilde{\mathbf{D}}_{ii} f_i^2 - \sum_{i,j=1}^{n} f_i f_j \tilde{\mathbf{A}}_{ij} = \frac{1}{2}\Big(\sum_{i=1}^{n} \tilde{\mathbf{D}}_{ii} f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j \tilde{\mathbf{A}}_{ij} + \sum_{j=1}^{n} \tilde{\mathbf{D}}_{jj} f_j^2\Big) \tag{15}$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} \tilde{\mathbf{A}}_{ij}(f_i - f_j)^2, \tag{16}$$

where the last step follows from the definition of the degree.

Next we show that 0 is indeed an eigenvalue of $\tilde{\mathbf{L}}_{\text{sym}}$ associated with the unit eigenvector $\boldsymbol{\pi}$ where $\boldsymbol{\pi} = \frac{\sqrt{\tilde{\mathbf{D}}_{ii}}}{\sqrt{\sum_v \tilde{\mathbf{D}}_{vv}}}$.

Let $\mathbb{1}$ be the all one vector. Then, a direct calculation reveals that

$$\tilde{\mathbf{L}}_{\text{sym}}\boldsymbol{\pi} = \boldsymbol{\pi} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\boldsymbol{\pi} = \boldsymbol{\pi} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{D}}^{1/2}\mathbb{1} \times \frac{1}{\sqrt{\sum_v \tilde{\mathbf{D}}_{vv}}} \tag{17}$$

$$= \boldsymbol{\pi} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\mathbb{1} \times \frac{1}{\sqrt{\sum_v \tilde{\mathbf{D}}_{vv}}} = \boldsymbol{\pi} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{D}}\mathbb{1} \times \frac{1}{\sqrt{\sum_v \tilde{\mathbf{D}}_{vv}}} \tag{18}$$

$$= \boldsymbol{\pi} - \tilde{\mathbf{D}}^{1/2}\mathbb{1} \times \frac{1}{\sqrt{\sum_v \tilde{\mathbf{D}}_{vv}}} = \boldsymbol{\pi} - \boldsymbol{\pi} = 0. \tag{19}$$

Combining this result with the positive semi-definite property of the Laplacian shows that 0 is indeed the smallest eigenvalue of $\tilde{\mathbf{L}}_{\text{sym}}$ associated with the eigenvector $\boldsymbol{\pi}$. Moreover, from (16) and the assumption that the graph is connected, it is not hard to see that the multiplicity of the eigenvalue 0 is exactly 1 (See Proposition 2 and 4 in Von Luxburg (2007) for more detail). Finally, from (13) it is obvious that the the largest eigenvalue of $\tilde{\mathbf{A}}_{\text{sym}}$ is 1, which correspond to the eigenvector $\boldsymbol{\pi}$. Hence all other eigenvalues of $\tilde{\mathbf{A}}_{\text{sym}}$ $1 > \lambda_2 \geq ... \geq \lambda_n$.

Next, we prove that $|\lambda_n| < 1$. This can also be shown directly from (16). Note that

$$u^T\tilde{\mathbf{L}}_{\text{sym}}u = \frac{1}{2}\sum_{i,j=1}^{n} \tilde{\mathbf{A}}_{ij}(f_i - f_j)^2 \tag{20}$$

$$\leq \sum_{i,j=1}^{n} \tilde{\mathbf{A}}_{ij}(f_i^2 + f_j^2) = 2\sum_{i,j=1}^{n} \tilde{\mathbf{A}}_{ij} f_i^2 = 2\sum_{i,j=1}^{n} \tilde{\mathbf{A}}_{ij}\frac{u_i^2}{\tilde{\mathbf{D}}_{ii}} \tag{21}$$

$$= 2\sum_{i=1}^{n} \frac{u_i^2}{\tilde{\mathbf{D}}_{ii}} \sum_{j=1}^{n} \tilde{\mathbf{A}}_{ij} = 2\sum_{i=1}^{n} \frac{u_i^2}{\tilde{\mathbf{D}}_{ii}}\tilde{\mathbf{D}}_{ii} = 2\sum_{i=1}^{n} u_i^2 = 2. \tag{22}$$

The inequality follows from an application of the Cauchy-Schwartz inequality. Consequently, the largest eigenvalue of $\tilde{\mathbf{L}}_{\text{sym}}$ is bounded by 2 which means that $|\lambda_n| \leq 1$. Note that equality holds if and only if the underlying graph is bipartite. However, this is impossible in our setting since we have added a self loop to each node. Hence $|\lambda_n| < 1$. This means

$$\lim_{k\to\infty} \tilde{\mathbf{A}}_{\text{sym}}^k = \boldsymbol{\pi}\boldsymbol{\pi}^T. \tag{23}$$

Hence, for any $\mathbf{H}^{(0)}$ we have

$$\lim_{k\to\infty} \tilde{\mathbf{A}}_{\text{sym}}^k \mathbf{H}^{(0)} = \boldsymbol{\pi}\boldsymbol{\pi}^T\mathbf{H}^{(0)} = \boldsymbol{\pi}\boldsymbol{\beta}^T. \tag{24}$$

Note that this can also be written with the $o_k(1)$ term as

$$\tilde{\mathbf{A}}_{\text{sym}}^k \mathbf{H}^{(0)} = \boldsymbol{\pi}\boldsymbol{\beta}^T + o_k(1). \tag{25}$$

This completes the proof.

A.7 PROOF OF LEMMA A.4

Recall that our loss function equals

$$L = \sum_{i \in \mathcal{T}} L_i = \sum_{i \in \mathcal{T}} - \log(\frac{e^{\eta \langle \mathbf{Z}_{i:}, \mathbf{Y}_{i:} \rangle}}{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}}). \tag{26}$$

Then by taking the partial derivative of the loss function with respect to $\gamma_{k'}$ we have

$$\frac{\partial L}{\partial \gamma_{k'}} = \frac{\partial}{\partial \gamma_{k'}} \sum_{i \in \mathcal{T}} (\log(\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}) - \langle \eta \mathbf{Z}_{i:}, \mathbf{Y}_{i:} \rangle). \tag{27}$$

Next, recall that for GPR-GNN we also have $\mathbf{Z} = \sum_{k=0}^{K} \gamma_k \mathbf{H}^{(k)}$. Plugging this expression into the previous formula and applying the chain rule we obtain

$$\frac{\partial}{\partial \gamma_{k'}} \sum_{i \in \mathcal{T}} (\log(\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}) - \langle \eta \mathbf{Z}_{i:}, \mathbf{Y}_{i:} \rangle) = \sum_{i \in \mathcal{T}} (\frac{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}} \frac{\partial \eta \mathbf{Z}_{im}}{\partial \gamma_{k'}}}{\sum_{m=1}^{C} e^{\mathbf{Z}_{im}}} - \left\langle \eta \mathbf{H}_{i:}^{(k')}, \mathbf{Y}_{i:} \right\rangle) \tag{28}$$

$$= \sum_{i \in \mathcal{T}} (\frac{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}} \eta \mathbf{H}_{im}^{(k')}}{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}} - \left\langle \eta \mathbf{H}_{i:}^{(k')}, \mathbf{Y}_{i:} \right\rangle) \tag{29}$$

Settin $k' = k$ for large enough k, it follows from Lemma A.2 that

$$\frac{\partial L}{\partial \gamma_k} = \sum_{i \in \mathcal{T}} \eta (\frac{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}} \mathbf{H}_{im}^{(k)}}{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}} - \left\langle \mathbf{H}_{i:}^{(k)}, \mathbf{Y}_{i:} \right\rangle) \tag{30}$$

$$= \sum_{i \in \mathcal{T}} \eta (\frac{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}} (\boldsymbol{\pi}_i \boldsymbol{\beta}_m + o_k(1))}{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}} - \langle \boldsymbol{\pi}_i \boldsymbol{\beta} + o_k(1), \mathbf{Y}_{i:} \rangle) \tag{31}$$

$$= \sum_{i \in \mathcal{T}} \boldsymbol{\pi}_i \eta (\frac{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}} \boldsymbol{\beta}_m}{\sum_{m=1}^{C} e^{\eta \mathbf{Z}_{im}}} - \langle \boldsymbol{\beta}, \mathbf{Y}_{i:} \rangle) + o_k(1) \tag{32}$$

$$= \sum_{i \in \mathcal{T}} \boldsymbol{\pi}_i \eta (\sum_{m=1}^{C} \hat{\mathbf{P}}_{im} \boldsymbol{\beta}_m - \langle \boldsymbol{\beta}, \mathbf{Y}_{i:} \rangle) + o_k(1) = \sum_{i \in \mathcal{T}} \eta \boldsymbol{\pi}_i \left\langle \hat{\mathbf{P}}_{i:} - \mathbf{Y}_{i:}, \boldsymbol{\beta} \right\rangle + o_k(1). \tag{33}$$

Note that in (32) and (33) we used the definition of the soft prediction $\hat{\mathbf{P}} = \text{softmax}_\eta(\mathbf{Z})$. This completes the proof.

A.8 PROOF OF LEMMA A.5

Let $\hat{\beta} = \max(\boldsymbol{\beta})$. Then by the definition of softmax_η for $\eta > 0$ we have

$$\text{softmax}_\eta(\boldsymbol{\beta}) = \frac{e^{\eta \boldsymbol{\beta}}}{\sum_{m=1}^{C} e^{\eta \boldsymbol{\beta}_m}} = \frac{e^{-\eta(\hat{\beta} - \boldsymbol{\beta})}}{\sum_{m=1}^{C} e^{-\eta(\hat{\beta} - \boldsymbol{\beta}_m)}}. \tag{34}$$

Note that $\hat{\beta} - \boldsymbol{\beta}_m > 0$ when $\boldsymbol{\beta}_m \neq \hat{\beta}$ and $\hat{\beta} - \boldsymbol{\beta}_m = 0$ when $\boldsymbol{\beta}_m = \hat{\beta}$. Without loss of generality we assume that there are p maxima in $\boldsymbol{\beta}$, where $1 \leq p \leq C$, and let $\mathcal{P}$ denote the set of indices of those maxima. Then, taking the limit $\eta \to \infty$ we have

$$\lim_{\eta \to \infty} \text{softmax}_\eta(\boldsymbol{\beta})_j = \lim_{\eta \to \infty} \frac{e^{-\eta(\hat{\beta} - \boldsymbol{\beta}_j)}}{\sum_{m \notin \mathcal{P}} e^{-\eta(\hat{\beta} - \boldsymbol{\beta}_m)} + p} = \begin{cases} 0, & \text{if } \boldsymbol{\beta}_j \neq \hat{\beta} \\ \frac{1}{p}, & \text{otherwise.} \end{cases} \tag{35}$$

This implies that for $\eta > 0$ large enough one has

$$\text{softmax}_\eta(\boldsymbol{\beta}) = \mathbf{1}[\boldsymbol{\beta}] + o_\eta(1). \tag{36}$$

The above result completes the proof.

Table 3: The values of the homophily measure for cSBM datasets.

ϕ	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	1
$\mathcal{H}(G)$	0.001	0.002	0.009	0.029	0.077	0.189	0.419	0.688	0.809

A.9 Additional Experimental details

All experiments are performed on a Linux Machine with 48 cores, 376GB of RAM, and a NVIDIA Tesla P100 GPU with 12GB of GPU memory. For the training set, we ensure that number of nodes from each class is approximately the same an keep the total number of training nodes close to $2.5\%/60\%$. For the validation set, we randomly sample $2.5\%/20\%$ of the nodes and place the remaining ones into the test set.

For all baseline models, we directly use the implementation available in the Pytorch Geometric library Fey & Lenssen (2019).We use early stopping 200 and a maximum number of epochs equal to 1000 for both real benchmark dataset and our cSBM synthetic datasets. All models use the Adam optimizer Kingma & Ba (2014). Note that the early stopping criteria is exactly the same as in Pytorch Geometric – when the epoch is greater than half of the maximum epoch, we check if the current validation loss is lower than the average over the past 200 epochs. If it is not lower, we stop the training process.

For GCN, we use 2 GCN layers with 64 hidden units. For GAT, we use 2 GAT convolutional layers, where the first layer has 8 attention heads and each head has 8 hidden units; the second layer has 1 attention head and 64 hidden units. For GCN-Cheby, we use 2 steps propagation for each layer with 32 hidden units. Note that the number of equivalent hidden units for each layer is64 for this case. For JK-Net, we use the GCN-based model with 2 layers and 16 hidden units in each layer. As for the layer aggregation part, we use a LSTM with 16 channels and 4 layers. For the MLP, we choose a 2-layer fully connected network with 64 hidden units. For APPNP we use the same 2-layer MLP with 10 steps of propagation. Besides the GPR-GNN, we fix the dropout rate for the NN part to be 0.5 as APPNP and optimize the dropout rate for the GPR part among $\{0, 0.5, 0.7\}$. For Geom-GCN, we choose the datasets already tested in the paper were the method was first described (Pei et al., 2019).

The heterophilic datasets used in (Pei et al., 2019). The graphs Chameleon, Actor, Squirrel, Texas and Cornell in their original form are directed graphs (see the github repository of (Pei et al., 2019)). Since the usual setting for semi-supervised node classifications involves undirected graph, we transformed the graphs into undirected to test them on all previously described benchmark methods. We keep the input graph directed for Geom-GCN as the method uses a fixed preprocessing scheme that was unfortunately not made public by the authors. Our homophily measure values $\mathcal{H}(G)$ in Table 1 are all based on undirected graphs and hence the numbers are different from those reported in (Pei et al., 2019).

A.10 Additional Experimental results

Table 4: Results for cSBM, sparse splitting. Bold values indicate the best obtained result and while bold, underlined values indicate results within a 95% confidence interval with respect to the best result.

	$\phi=-1$	$\phi=-0.75$	$\phi=-0.5$	$\phi=-0.25$	$\phi=0$	$\phi=0.25$	$\phi=0.5$	$\phi=0.75$	$\phi=1$
GPRGNN	**97.19±0.16**	**95.54±0.15**	**81.54±0.73**	60.65±0.31	**62.16±0.23**	**68.83±0.28**	**89.31±0.16**	**96.98±0.08**	**96.71±0.13**
GPRGNN(random)	88.39±3.31	88.54±3.01	66.91±2.93	56.35±0.98	58.09±0.71	64.01±1.39	81.93±1.68	94.59±0.29	93.69±1.04
APPNP	49.57±0.11	52.45±0.27	56.32±0.40	59.55±0.48	61.21±0.23	68.41±0.30	85.66±0.22	94.37±0.09	90.02±0.16
MLP	49.88±0.10	53.40±0.34	57.14±0.41	60.55±0.41	**62.15±0.33**	61.26±0.21	57.91±0.35	53.36±0.32	49.92±0.11
GCN	55.24±0.35	61.04±0.39	56.40±0.39	52.23±0.24	54.43±0.32	67.23±0.29	84.56±0.20	90.19±0.14	78.67±0.19
GAT	53.97±0.32	57.18±0.45	53.39±0.34	51.23±0.19	53.26±0.27	64.45±0.36	81.94±0.34	88.45±0.26	78.06±0.30
JKNet	51.70±0.39	55.83±0.75	52.67±0.51	50.27±0.15	52.02±0.35	65.67±0.44	86.35±0.19	95.13±0.09	90.32±0.17
GCN-Cheby	61.44±0.51	73.91±0.75	71.96±0.6	**63.96±0.43**	59.70±0.34	64.00±0.38	72.34±0.63	73.56±0.65	60.88±0.58

Table 5: Results for cSBM, dense splitting. Bold values indicate the best results found while bold, underlined values indicate results within a 95% confidence interval with respect to the best result.

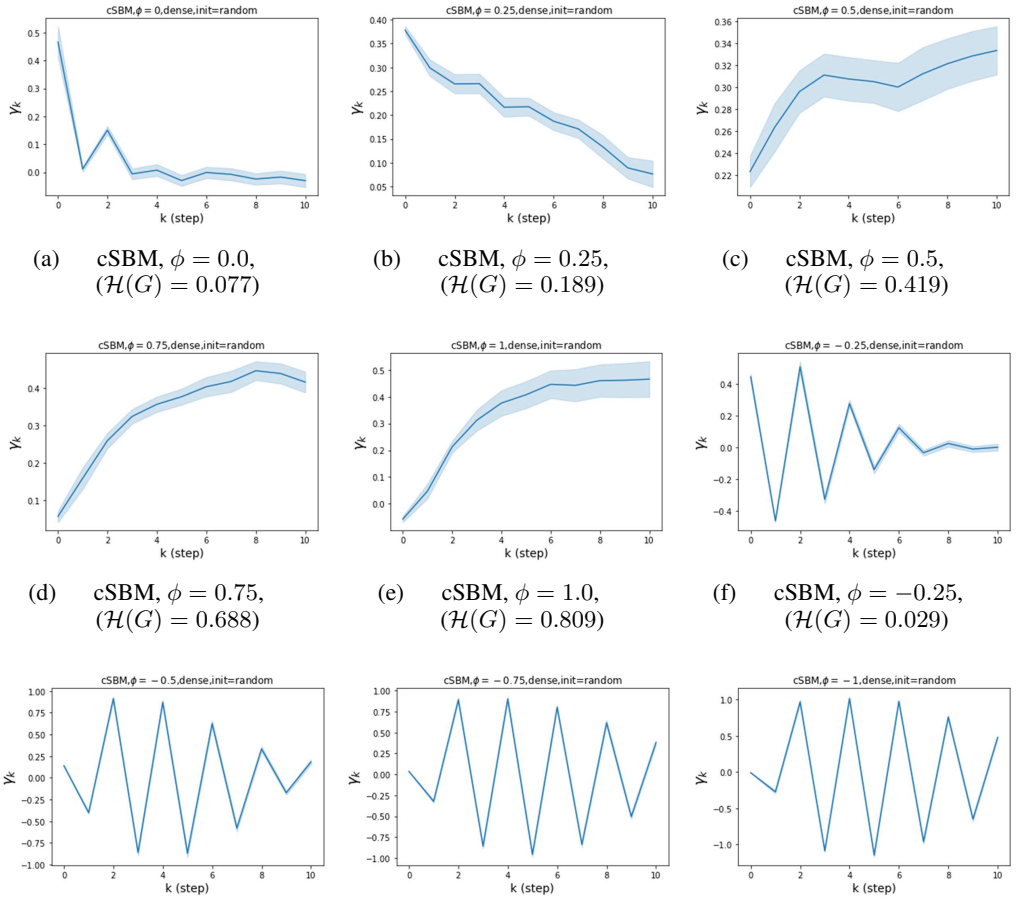
	$\phi = -1$	$\phi = -0.75$	$\phi = -0.5$	$\phi = -0.25$	$\phi = 0$	$\phi = 0.25$	$\phi = 0.5$	$\phi = 0.75$	$\phi = 1$
GPRGNN	**98.83±0.06**	**98.19±0.08**	**94.23±0.14**	**86.06±0.20**	**82.22±0.20**	**86.48±0.20**	**94.34±0.13**	**98.46±0.08**	**98.84±0.06**
GPRGNN(random)	98.75±0.05	98.08±0.08	94.22±0.14	86.06±0.20	**81.57±0.23**	**86.36±0.20**	94.09±0.14	**98.38±0.08**	98.77±0.07
APPNP	48.94±0.29	63.87±0.29	73.30±0.26	79.30±0.20	82.41±0.23	86.47±0.18	94.20±0.14	97.96±0.10	98.53±0.08
MLP	49.79±0.29	66.69±0.27	75.36±0.26	80.30±0.24	82.19±0.24	80.88±0.22	76.07±0.24	66.61±0.25	49.65±0.29
GCN	78.50±0.28	83.68±0.22	75.98±0.25	59.98±0.25	64.09±0.26	81.89±0.19	93.91±0.12	97.78±0.08	96.29±0.11
GAT	82.39±0.41	80.37±0.22	71.01±0.26	57.68±0.29	62.95±0.28	80.61±0.24	93.26±0.14	97.99±0.08	98.40±0.09
JKNet	96.11±0.37	95.33±0.25	87.98±0.56	59.61±0.49	63.28±0.10	80.23±0.36	93.28±0.15	98.33±0.07	98.22±0.07
GCN-Cheby	90.94±0.16	94.82±0.13	91.83±0.17	85.18±0.21	80.80±0.25	85.28±0.21	92.70±0.16	95.06±0.13	90.34±0.18

Table 6: Results on homophilic real-world benchmark datasets tested in (Pei et al., 2019), dense splitting: Mean accuracy ($\%$) $\pm$ 95% confidence interval. Boldface values indicate the best results found while boldface, underlined values indicates results within the confidence interval with respect to the best result.

	Cora	Citeseer	PubMed
GPRGNN	**88.65±0.28**	80.01±0.28	**89.18±0.15**
APPNP	88.1±0.23	**80.5±0.26**	**89.15±0.13**
MLP	76.44±0.3	76.25±0.28	86.43±0.13
GCN	86.87±0.25	79.28±0.25	86.97±0.12
GAT	87.52±0.24	**80.56±0.31**	86.64±0.11
JKNet	86.97±0.27	77.69±0.35	87.38±0.13
GCN-Cheby	86.46±0.26	78.66±0.26	88.2±0.09
GeomGCN	85.4±0.26	76.42±0.37	88.51±0.08

Table 7: Additional experiments illustrating that GPR-GNN escapes over-smoothing. We initialize the GPR weights $\gamma_k = \delta_{kK}$ as described in Section 5. We report the mean accuracy at Epoch 0 and after training (Final epoch). The over-smoothing ratio indicates how many time out of the 100 runs that GPR-GNN started with lead to the same label for all nodes. For an illustration of how GPR weights change over different epochs, please check Figure 8.

	Accuracy at epoch $0(\%)$	Accuracy at the final epoch$(\%)$	Over-smoothing ratio$(\%)$
Cora	12.75	88.25	84
Computers	9.41	85.93	89
Squirrel	19.87	52.06	97
Texas	21.05	90.05	100



(a) cSBM, $\phi = 0.0$, ($\mathcal{H}(G) = 0.077$)

(b) cSBM, $\phi = 0.25$, ($\mathcal{H}(G) = 0.189$)

(c) cSBM, $\phi = 0.5$, ($\mathcal{H}(G) = 0.419$)

(d) cSBM, $\phi = 0.75$, ($\mathcal{H}(G) = 0.688$)

(e) cSBM, $\phi = 1.0$, ($\mathcal{H}(G) = 0.809$)

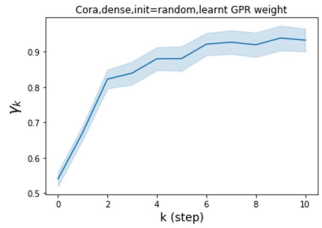
(f) cSBM, $\phi = -0.25$, ($\mathcal{H}(G) = 0.029$)

(g) cSBM, $\phi = -0.5$, ($\mathcal{H}(G) = 0.009$)

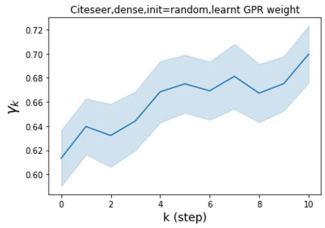
(h) cSBM, $\phi = -0.75$, ($\mathcal{H}(G) = 0.002$)

(i) cSBM, $\phi = -1.0$, ($\mathcal{H}(G) = 0.001$)

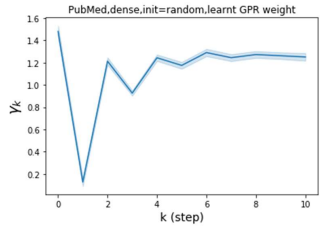
Figure 6: Figures (a)-(i) show the learned GPR weights by GPR-GNN with random initialization on cSBM, dense splitting. The shaded region indicates a 95% confidence interval.



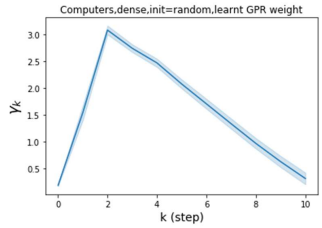
(a) Cora, $(\mathcal{H}(G) = 0.656)$



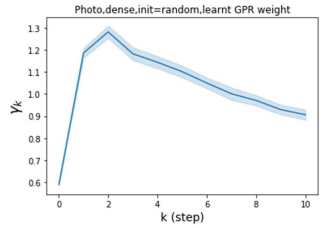
(b) Citeseer, $(\mathcal{H}(G) = 0.578)$



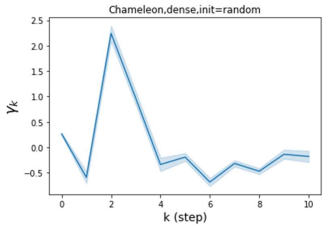
(c) PubMed, $(\mathcal{H}(G) = 0.644)$



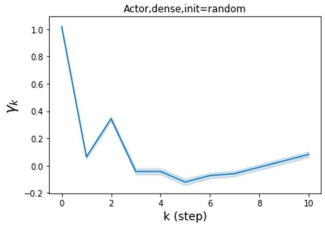
(d) Computers, $(\mathcal{H}(G) = 0.272)$



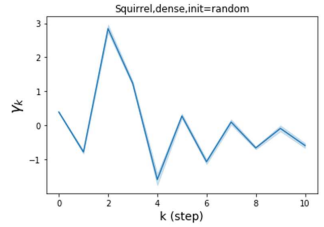
(e) Photo, $(\mathcal{H}(G) = 0.459)$



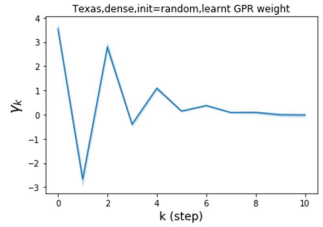
(f) Chameleon, $(\mathcal{H}(G) = 0.024)$



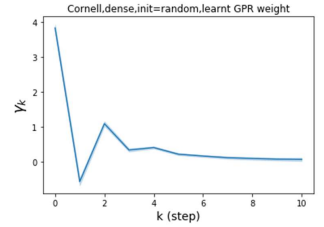
(g) Actor, $(\mathcal{H}(G) = 0.008)$



(h) Squirrel, $(\mathcal{H}(G) = 0.055)$

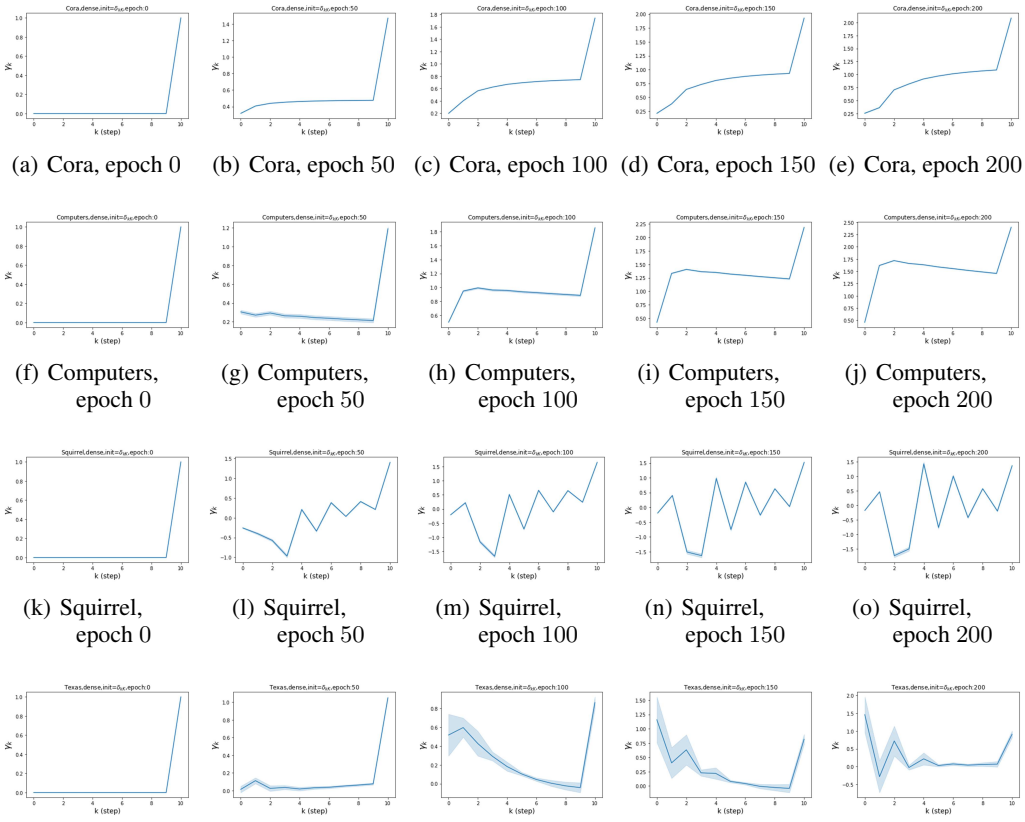


(i) Texas, $(\mathcal{H}(G) = 0.016)$



(j) Cornell, $(\mathcal{H}(G) = 0.137)$

Figure 7: Figures (a)-(j) show the learned GPR weights by GPR-GNN with random initialization on various benchmark datasets, dense splitting. The shaded region indicates a 95% confidence interval. Note that the learned GPR weights are all positive for every homophilic dataset. There is at least one negative learned GPR weight for every heterophilic dataset.



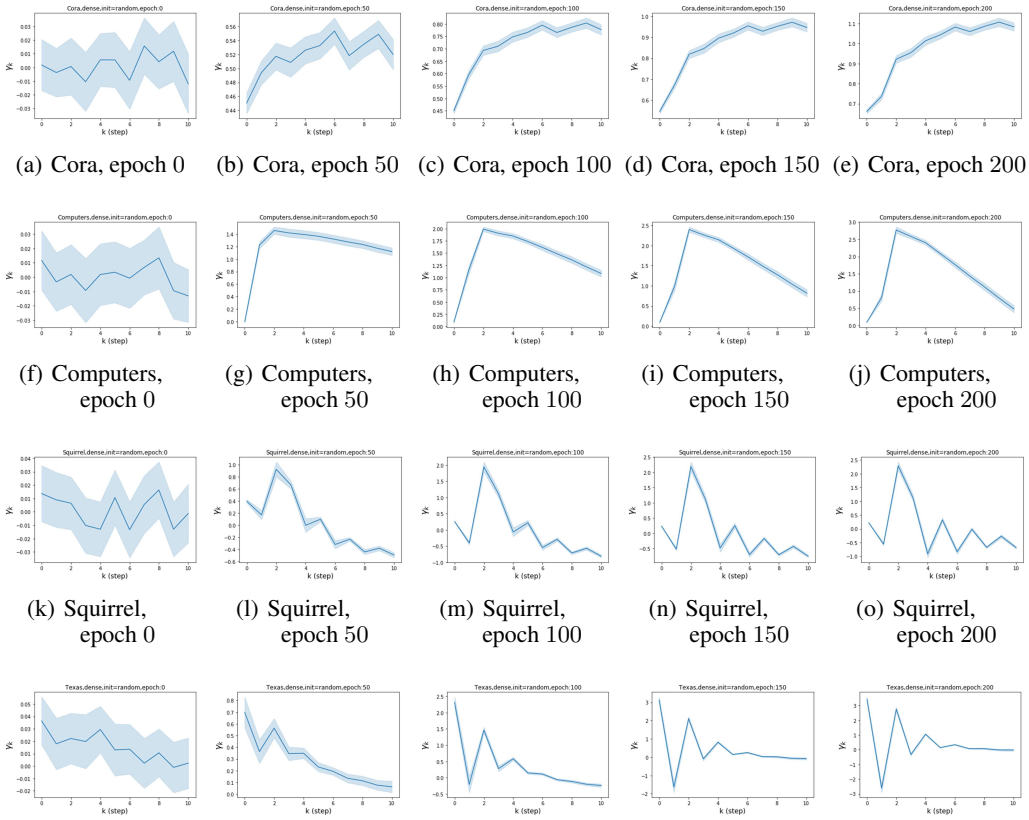
(a) Cora, epoch 0 (b) Cora, epoch 50 (c) Cora, epoch 100 (d) Cora, epoch 150 (e) Cora, epoch 200

(f) Computers, epoch 0 (g) Computers, epoch 50 (h) Computers, epoch 100 (i) Computers, epoch 150 (j) Computers, epoch 200

(k) Squirrel, epoch 0 (l) Squirrel, epoch 50 (m) Squirrel, epoch 100 (n) Squirrel, epoch 150 (o) Squirrel, epoch 200

(p) Texas, epoch 0 (q) Texas, epoch 50 (r) Texas, epoch 100 (s) Texas, epoch 150 (t) Texas, epoch 200

Figure 8: Learned GPR weights by GPR-GNN with initialization $\gamma_k = \delta_{kK}$ (last step) on various benchmark datasets, dense splitting. The shaded region indicates a 95% confidence interval. Also, please check Table 7. Note that the GPR weights $\{\gamma_k\}_{k=0}^{K}$ are identical to $\{-\gamma_k\}_{k=0}^{K}$ in terms of graph filtering.



(a) Cora, epoch 0 (b) Cora, epoch 50 (c) Cora, epoch 100 (d) Cora, epoch 150 (e) Cora, epoch 200

(f) Computers, epoch 0 (g) Computers, epoch 50 (h) Computers, epoch 100 (i) Computers, epoch 150 (j) Computers, epoch 200

(k) Squirrel, epoch 0 (l) Squirrel, epoch 50 (m) Squirrel, epoch 100 (n) Squirrel, epoch 150 (o) Squirrel, epoch 200

(p) Texas, epoch 0 (q) Texas, epoch 50 (r) Texas, epoch 100 (s) Texas, epoch 150 (t) Texas, epoch 200

Figure 9: The dynamics of learning GPR weights with random initialization on various benchmark datasets, dense splitting. The shaded region indicates a 95% confidence interval.