A Note on Over-Smoothing for Graph Neural Networks

Chen Cai [1] **Yusu Wang** [1]

Abstract

Graph Neural Networks (GNNs) have achieved a lot of success on graph-structured data. However, it is observed that the performance of graph neural networks does not improve as the number of layers increases. This effect, known as over-smoothing [1], has been analyzed mostly in linear cases. In this paper, we build upon previous results (Oono & Suzuki, 2019) to further analyze the over-smoothing effect in the general graph neural network architecture. We show when the weight matrix satisfies the conditions determined by the spectrum of augmented normalized Laplacian, the Dirichlet energy of embeddings will converge to zero, resulting in the loss of discriminative power. Using Dirichlet energy to measure "expressiveness" of embedding is conceptually clean; it leads to simpler proofs than (Oono & Suzuki, 2019) and can handle more non-linearities.

1. Introduction

Graph neural networks (GNNs) are a family of neural networks that can learn from graph-structured data. Starting with the success of Graph Convolutional Network (GCN) (Kipf & Welling, 2016) in achieving state-of-the-art performance on semi-supervised classification, several variants of GNNs have been developed for this task, including GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), SGC (Wu et al., 2019), CGCNN (Xie & Grossman, 2018) and GMNN (Qu et al., 2019) to name a few most recent ones. See (Gurukar et al., 2019; Wu et al., 2020; Zhou et al., 2018) for survey.

However, a key issue with GNNs is their depth limitations. It has been observed that deeply stacking the layers often results in significantly worse performance for GNNs, such as GCN and GAT. This drop is associated with many factors, including the vanishing gradients during back-propagation, overfitting due to the increasing number of parameters, as well as the phenomenon called over-smoothing. (Li et al., 2018) was the first to call attention to the over-smoothing problem. Having shown that the graph convolution is a type of Laplacian smoothing, they proved that after repeatedly applying Laplacian smoothing many times, the features of the nodes in the (connected) graph would converge to similar values. Later, several others have alluded to the same problem. (Li et al., 2019; Luan et al., 2019; Zhao & Akoglu, 2019)

The goal of this paper is to extend some analysis of GNN in the ICLR 2020 spotlight paper (Oono & Suzuki, 2019) on the expressive power of GNNs for node classification. To the best of our knowledge, (Oono & Suzuki, 2019) is the first paper extending the analysis of over-smoothing in linear GNNs to the nonlinear ones. However, only ReLU is handled. It is noted by the authors that extension to other non-linearities such as Sigmoid and Leaky ReLU is far from trivial.

In this paper, we propose a simple technique to analyze the embedding when the number of layers goes to infinity. The analysis is based on tracking the Dirichlet energy of node embeddings across layers. Our contributions are the following:

- Using Dirichlet energy to measure expressiveness of embeddings is conceptually clean. Besides being able to recover the results in (Oono & Suzuki, 2019), our analysis can be easily applied to Leaky ReLU. In the special case of regular graphs, our proof can be extended to the most common nonlinearities. The proof is easy to follow and requires only elementary linear algebra. We discuss key differences between our proof and proofs in (Oono & Suzuki, 2019) as well as the benefits of introducing Dirichlet energy in Section 4.

- Second, we perform extensive experiments on a variety of graphs to study the effect of basic edge operations on the Dirichlet energy. We find in many cases dropping edges and increasing the weights of edges (to a high value) can increase the Dirichlet energy.

[1]Department of Computer Science, Ohio State University, Ohio, USA. Correspondence to: Chen Cai <cai.507@osu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

[1]Strictly speaking, over-smoothing is a misnomer. As we will show, what is decreasing is $tr(X^T\tilde{\Delta}X)$, not the real smoothness $\frac{tr(X^T\tilde{\Delta}X)}{||X||_2^2}$ of graph signal X.

2. Notation

Let $\mathbb{N}_+$ be the set of positive integers. We define $A \in \mathbb{R}^{N\times N}$ to be the adjacency matrix and D to be the degree matrix of graph G. Let $\tilde{A} := A + I_N, \tilde{D} := D + I_N$ be the adjacent and degree matrix of graph G augmented with self- loops. We define the augmented normalized Laplacian of G by $\tilde{\Delta} := I_N - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and set $P := I_N - \tilde{\Delta} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Let $L, C \in \mathbb{N}_+$ be the layer and channel sizes.

We define a GCN associated with G by $\mathbf{f} = \mathbf{f}_L \circ \ldots \circ \mathbf{f}_1$ where $\mathbf{f}_l : \mathbb{R}^{N\times C_l} \to \mathbb{R}^{N\times C_{l+1}}$ is defined by $\mathbf{f}_l(X) = \text{MLP}_l(PX)$. Here $\text{MLP}_l(X) := \sigma(\cdots\sigma(\sigma(X)W_{l1})W_{l2}\cdots W_{lH_l})$ where σ is an element-wise nonlinear function. Note that weight matrices $W_{l\cdot}$ are not necessarily square. We consider the embeddings $X^{(l+1)} := \mathbf{f}_l(X^{(l)})$ with initial value $X^{(0)}$. We are interested in the asymptotic behavior of the output $X^{(L)}$ of GCN as $L \to \infty$.

We state the following lemma without a proof.

Lemma 2.1. Eigenvalues of $\tilde{\Delta} \in [0, 2)$. Eigenvalues of $P = I_N - \tilde{\Delta} \in (-1, 1]$.

3. Main Result

The main idea of the proof is to track the Dirichlet energy of node embeddings w.r.t. the (augmented) normalized Laplacian at different layers. With some assumptions on the weight matrix of GCN, we can prove that Dirichlet energy decreases exponentially with respect to the number of layers. Intuitively, the Dirichlet energy of a function measures the "smoothness" of a function of unit norm, and eigenvectors of the normalized Laplacian are minimizers of the Dirichlet energy.

Definition 3.1. Dirichlet energy $E(f)$ of scalar function $f \in \mathbb{R}^{N\times 1}$ on the graph G is defined as

$$E(f) = f^T\tilde{\Delta}f = \frac{1}{2}\Sigma w_{ij}\left(\frac{f_i}{\sqrt{1+d_i}} - \frac{f_j}{\sqrt{1+d_j}}\right)^2.$$

For vector field $X_{N\times c} = [x_1, ..., x_N]^T, x_i \in \mathbb{R}^{1\times c}$, Dirichlet energy is defined as

$$E(X) = tr(X^T\tilde{\Delta}X) = \frac{1}{2}\Sigma w_{ij}||\frac{x_i}{\sqrt{1+d_i}} - \frac{x_j}{\sqrt{1+d_j}}||_2^2.$$

Without loss of generality, each layer of GCN can be represented as $\mathbf{f}_l(X) = \sigma(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H \text{ times}}PX)W_{l1})W_{l2}\cdots)W_{lH_l})$

Next we will analyze the effects of P, W_l, σ on the Dirichlet energy one by one.

Lemma 3.1. $E(PX) \le (1-\lambda)^2E(X)$ where λ is the smallest non-zero eigenvalue of $\tilde{\Delta}$.

Proof. Let us denote the eigenvalues of $\tilde{\Delta}$ by $\lambda_1, \lambda_2, ..., \lambda_N$, and the associated eigenvectors of length 1 by $v_1, ..., v_n$. Suppose $f = \Sigma c_i v_i$ where $c_i \in \mathbb{R}$.

$$E(f) = f^T\tilde{\Delta}f = f^T\Sigma c_i\lambda_i v_i = \Sigma c_i^2\lambda_i \tag{1}$$

Therefore,

$$\begin{aligned} E(Pf) &= f^T(I_N - \tilde{\Delta})^T\tilde{\Delta}(I_N - \tilde{\Delta})f \\ &= f^T(I_N - \tilde{\Delta})\tilde{\Delta}(I_N - \tilde{\Delta})f \\ &= \Sigma c_i^2\lambda_i(1-\lambda_i)^2 \\ &\le (1-\lambda)^2E(f) \end{aligned} \tag{2}$$

Extending the above argument from the scaler field to vector field finishes the proof for $E(PX) \le (1-\lambda)^2E(X)$. □

Lemma 3.2. $E(XW) \le ||W^T||_2^2E(X)$

Proof. By definition,

$$E(XW) = \Sigma_{(i,j)\in E}w_{ij}||\frac{1}{\sqrt{1+d_i}}x_iW - \frac{1}{\sqrt{1+d_j}}x_jW||_2^2$$

where $X_{n\times c} = [x_1, ..., x_n]^T, x_i \in \mathbb{R}^{1\times c}, W \in \mathbb{R}^{c\times c'}$. Since for each term

$$\begin{aligned} &||\frac{1}{\sqrt{1+d_i}}x_iW - \frac{1}{\sqrt{1+d_j}}x_jW||_2^2 \le \\ &||\frac{1}{\sqrt{1+d_i}}x_i - \frac{1}{\sqrt{1+d_j}}x_j||_2^2||W^T||_2^2, \end{aligned} \tag{3}$$

we get

$$E(XW) \le E(W)||W^T||_2^2 \tag{4}$$

□

Remark: Since $||A||_2 = \sigma_{max}(A)$ where $\sigma_{max}(A)$ represents the largest singular value of matrix A. Our result in Lemma 2 is essentially the same as the Lemma 2[2] of (Oono & Suzuki, 2019). Note that our proof can handle weight matrix not only of dimension $d \times d$ but also of dimension $d \times d'$ while the paper (Oono & Suzuki, 2019) assumes the embedding dimension to be fixed across layers. See detailed discussion at section 4.

Remark: The proof itself doesn't leverage the structure of graph. In particular, only the fact of Laplacian is p.s.d matrix is needed in the proof. See an alternative proof in the appendix. This also makes sense because W operates on the graph feature space and should be oblivious to the particular graph structure.

[2]For any $X \in \mathbb{R}^{N\times C}$, we have $d_{\mathcal{M}}(XW_{lh}) \le s_{lh}d_{\mathcal{M}}(X)$ where s_{lh} is the maximum singular value of W_{lh}.

Lemma 3.3. $E(\sigma(X)) \leq E(X)$ when σ is ReLU or Leaky-ReLU.

Proof. We first prove it holds for scalar field f and then extend it to vector field X. $E(f) = \Sigma_{(i,j)\in E} w_{i,j}(\frac{f_i}{\sqrt{1+d_i}} - \frac{f_j}{\sqrt{1+d_j}})^2$ where $w_{i,j} \geq 0$. And $\forall c_1, c_2 \in \mathbb{R}_+, a, b \in \mathbb{R}$

$$\begin{aligned} |c_1 a - c_2 b| &\geq |\sigma(c_1 a) - \sigma(c_2 b)| \\ &= |c_1 \sigma(a) - c_2 \sigma(b)| \end{aligned} \tag{5}$$

The first inequality holds for all σ whose Lipschitz constant is no more than 1, including ReLU, Leaky-ReLU, Tanh, Sigmoid, etc. The second equality holds because for ReLu and Leaky-Relu, $\sigma(cx) = c\sigma(x), \forall c \in \mathbb{R}_+, x \in \mathbb{R}$.

Therefore, by replacing c_1, c_2, a, b with $\frac{1}{\sqrt{1+d_i}}, \frac{1}{\sqrt{1+d_j}}, f_i, f_j$, we can see $E(\sigma(f)) \leq E(f)$ holds for ReLU and Leaky-ReLU. Extending the above argument to vector field completes the proof. $\square$

Remark: For regular graphs, the above conclusion can be extended to more non-linearities such as ReLU, Leaky-ReLU, Tanh, and Sigmoid.

Remark: The proof hinges on the simple fact that for ReLU and Leaky-ReLU, $\sigma(ca) = \sigma(c)a$ where $c \in \mathbb{R}_+, a \in \mathbb{R}$. For other activation functions, as long as $c_1 a = c_2 b$ and $c_1\sigma(a) \neq c_2\sigma(b)$ (easy to find examples for Sigmoid, Tanh [3], etc since there are no strong restrictions on a, b, c_1, c_2. [4]), we can not guarantee $E(\sigma(X)) \leq E(X)$.

Combining the above three lemmas, and denote the square of maximum singular value of W_{lh}^T by s_{lh} and set $s_l := \prod_{h=1}^{H_l} s_{lh}$. Also let $\bar{\lambda} := (1-\lambda)^2$. With those parameters, we arrive at the main theorem.

Theorem 3.4. For any $l \in \mathbb{N}_+$, we have $E(\mathbf{f}_l(X)) \leq s_l \bar{\lambda} E(X)$

See proof in the appendix A.

Corollary 3.4.1. Let $s := \sup_{l\in\mathbb{N}_+} s_l$. We have $E(X^{(l)}) \leq O((s\bar{\lambda})^l)$. In particular, $E(X^{(l)})$ exponentially converges to 0 when $s\bar{\lambda} < 1$.

Our result shares great similarity with the paper (Oono & Suzuki, 2019). The bounds are similar but our result handles more general cases. As noted in (Oono & Suzuki, 2019), eigengap plays an important role. The analysis of Erdos-Renyi graph $G_{N,p}$ (or any other graphs that have large eigengaps) when $\frac{\log N}{Np} = o(1)$ in the paper (Oono & Suzuki, 2019) can also be directly applied to our case.

[3]Sigmoid: Sigmoid$(x) = \frac{1}{1+e^{-x}}$. Tanh: Tanh$(x) = \frac{e^{2x}-1}{e^{2x}+1}$.

[4]For example, $c_1 = 1, x = 2, c_2 = 2, y = 1$.

4. Key Differences

The key quantity paper (Oono & Suzuki, 2019) looks at is the $d_\mathcal{M}(X)$ where $\mathcal{M}$ is a subspace of $\mathbb{R}^{N\times C}$, defined as $\mathcal{M} := U \otimes \mathbb{R}^C = \left\{\sum_{m=1}^M e_m \otimes w_m | w_m \in \mathbb{R}^C\right\}$, where $(e_m)_{m\in[M]}$ is orthonormal basis of null space U of a normalized graph Laplacian $\tilde{\Delta}$. The original definition of $d_\mathcal{M}(X)$ is defined for the case of the same embedding dimension across layers. It needs to be modified to handle the case of varying dimensions. One way to achieve this is to define $\mathcal{M} = U \otimes \mathbb{R}^C, \mathcal{M}' = U \otimes \mathbb{R}^{C'}$, respectively. The lemma 2 of paper (Oono & Suzuki, 2019) then can be modified from $d_\mathcal{M}(XW) \leq s d_\mathcal{M}(X)$ $(W \in \mathbb{R}^{c\times c})$ to the following:

$$d_{\mathcal{M}'}(XW) \leq s d_\mathcal{M}(X) \tag{6}$$

where s is the singular value of W [5].

As for the nonlinearity, (Oono & Suzuki, 2019) mentions that their analysis is limited to graph neural networks with the ReLU activation function because they implicitly use the property that ReLU is a projection onto the cone $\{X > 0\}$ (see Appendix A, Lemma 3 in (Oono & Suzuki, 2019) for details). This fact enables the ReLU function to get along with the nonnegativity of eigenvectors associated with the largest eigenvalues (Perron-Frobenius theorem). Therefore, the authors mentioned that it may not be easy to extend their results to other activation functions such as the sigmoid function or Leaky ReLU.

In contrast, the proof of Lemma 3.3 becomes trivial once we write out the Dirichlet energy as the sum of multiple terms for each of which the effect of nonlinearity can be easily analyzed.

5. Experiments

To investigate how basic edges operations, removing edges, and increasing edge weight[6], affect Dirichlet energy and over-smoothing, we perform extensive experiments on both common benchmarks (Cora and CiteSeer) and synthetic graphs. See appendix B for more details on datasets.

In particular, given a graph, we will compute its eigenvalues before and after randomly dropping/increasing weights of a certain percent $(10\% - 90\%)$ of edges. This is shown in the first/third column for each figure. In the second/fourth column, we generate three signals $x, Px(P'x), P^2x(P'^2x)$, where $x = \Sigma_i^T c_i v_i$ where v_i is the first T eigenvectors corresponding to lowest T eigenvalues of normalized Laplacian and c_i is uniform random number between 0 and 1. In other words, x is a mix of lower eigenvectors. We then compute the Dirichlet energy of three signals both

[5]Here with slight abuse of notation, $W \in \mathbb{R}^{c\times c'}$

[6]In this paper, we only consider the case where the edge weight is increased to very high (from 1 to 10000 in all experiments).

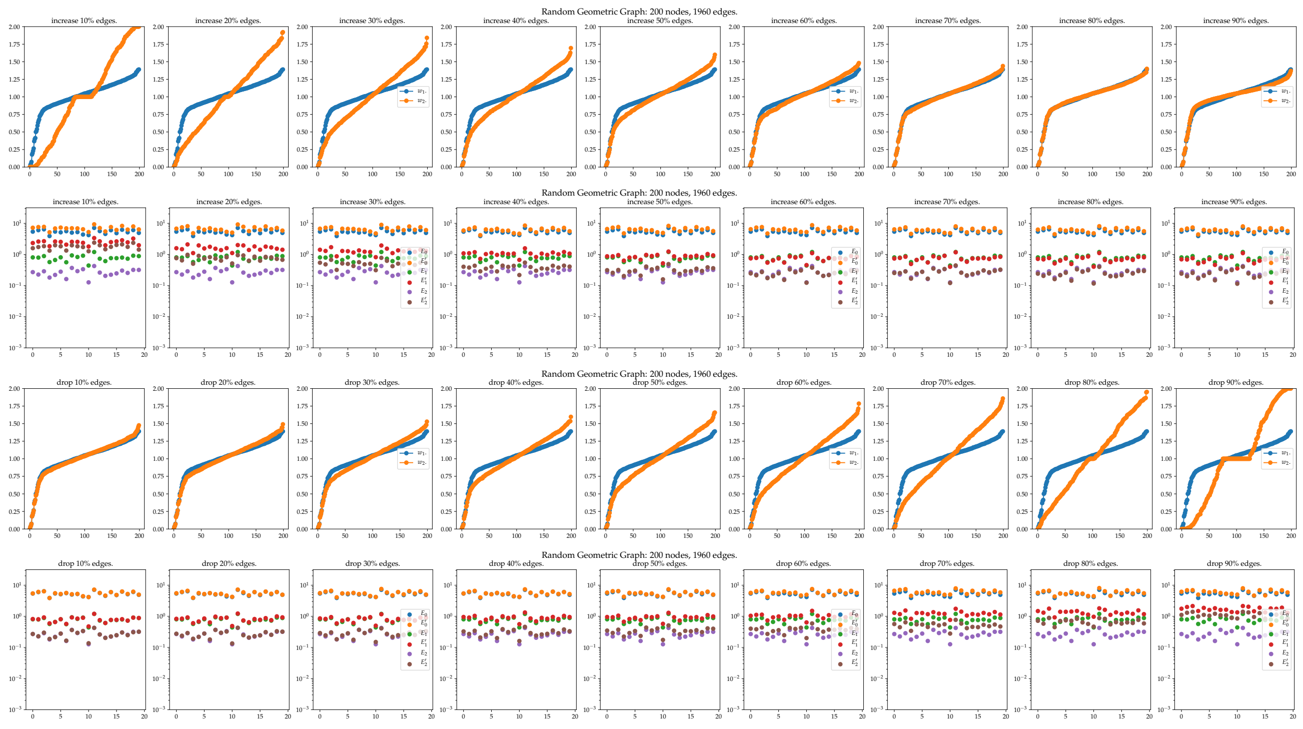

Figure 1. The effects of dropping edges and increasing edge weights on eigenvalues / Dirichilet energy for Random Geometric graph.

for original graph (E_0, E_1, E_2) and graph with edges removed/reweighted (E_0', E_1', E_2'). The same experiments are repeated 20 times and 120 data points are shown in the scatter plot.

We make the following observations.

- First, for nearly all graphs and ratios (except for some cases of Cora and CiteSeer), dropping edges increases Dirichlet energy for $x, Px(P'x), P^2x(P'^2x)$. This coincides with the observation in DropEdge (Rong et al., 2019) that dropping edges help relive over-smoothing.

- Second, in most cases, the effect of increasing the weight of edge (from 1 to 10000) and dropping edges is " dual" to each other, i.e., increasing weights of a few edges to a very high value is similar to dropping a lot of edges in terms of eigenvalue and Dirichlet energy. Intuitively, we can think of increasing the weight of an edge u, v to infinity as contracting node u and v into a supernode. For the planar graph and its dual graph, edge deletion in one graph corresponds to the contraction in the other graph and vice versa. We hypothesize that randomly increasing the weight of a few edges to a high value will also help to relieve over-smoothing. We leave the systematic verification as future work.

6. Conclusion

We provide an alternative proof of graph neural networks exponentially loosing expressive power. Being able to achieve the same bound as the paper (Oono & Suzuki, 2019), our simple proof also handles Leaky ReLU. We also empirically explore the effect of basic edge operations on the Dirichlet energy.

Some interesting future directions are: 1) The key challenge of analyzing the over-smoothing effect lies in the non-linearity. How to extend our strategy to more general graph learning such as other nonlinearities, normalization strategy (Zhao & Akoglu, 2019), graphs with both node and edge features and attention mechanism (Veličković et al., 2017) remains largely open. 2) The assumption on the norm of weight function of GNNs is crucial (may also be too strong) in our proof. Understanding how learning plays a role in resisting the over-smoothing effect is interesting. 3) Preserving Dirichlet energy for combinatorial Laplacian is well studied in the context of graph sparsification. Novel techniques in (Lee & Sun, 2017; Spielman & Srivastava, 2011; Spielman & Teng, 2004; 2011) may be applicable. Also, Dirichlet energy itself is easy to compute and can serve as a useful quantity to monitor during the training of graph networks for practitioners. Finally, analyzing the real over-smoothing effect, i.e., the Rayleigh quotient $\frac{tr(X^T \tilde{\Delta} X)}{||X||_2^2}$, for deep GNNs is still an open and important question.

A. Missing Proof

To show that Lemma 3.2 is not using any particular graph structure, we present an alternative proof of Lemma 3.2, show simply use the generic matrix inequality.

Lemma A.1. $E(XW) \leq ||W^T||_2^2 E(X)$

Proof. Expand $E(XW)$ in matrix form,

$$\begin{aligned} E(XW) &= tr(W^T X^T \tilde{\Delta} XW) \\ &= tr(X^T \tilde{\Delta} XWW^T) \\ &\leq tr(X^T \tilde{\Delta} X) \sigma_{max}(WW^T) \\ &= E(X) ||W^T||_2^2 \end{aligned}$$

Note σ_{max} denotes the largest eigenvalue and $\|A\|_2 = \sqrt{\lambda_{\max}(A^* A)} = \sigma_{\max}(A)$. □

Theorem A.2. For any $l \in \mathbb{N}_+$, we have $E(\mathbf{f}_l(X)) \leq s_l \bar{\lambda} E(X)$

Proof. By Lemma 3.1-3.3,

$$\begin{aligned} E(\mathbf{f}_l(X)) &= E(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H \text{ times}} PX)W_{l1})W_{l2}\cdots W_{lH_l})) \\ &\leq E(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H-1 \text{ times}} PX)W_{l1})W_{l2}\cdots)W_{lH_l}) \\ &\leq s_{lH_l-1} E(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H-1 \text{ times}} PX)W_{l1})W_{l2}\cdots)W_{lH_l-1})) \\ &\dots \\ &\leq \left(\prod_{h=1}^{H_l} s_{lh}\right) E(PX) \\ &\leq s_l \bar{\lambda} E(PX) \\ &\leq s_l \bar{\lambda} E(X) \end{aligned}$$

□

B. Experiments

We perform experiments on both synthetic graphs and real graphs benchmarks. The threshold T for the number of lower eigenvectors is set to be 20 for synthetic graphs. For Cora and Citeseer, it is set to be 400 and 600 respectively (due to a large number of nearly zero eigenvalues). The code is available on Github. [7] The details of each graph are listed as follows:

- Random graph $G(200, 0.05)$.
- Random geometric graph on the plane. There are 200 nodes uniformly at random in the unit cube. Two nodes are joined by an edge if the distance between the nodes is at most 0.2.
- Stochastic Block Model with 2 blocks. It consists of two blocks where each block has 100 nodes. The edge probability within the block is 0.1 and edge probability between blocks is 0.01.
- Stochastic Block Model with 4 blocks. It consists of four blocks where each block has 50 nodes. The edge probability within the block is $0.1, 0.2, 0.3, 0.4$. The edge probability between blocks is 0.08.
- Barabasi-Albert Graph. A graph of n nodes is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree. We set n, m to be 200 and 4.
- Cora is a citation graph where 2708 nodes are documents and 5278 edges are citation links.
- Citeseer is a citation graph where 3327 nodes are documents and 4552 edges are citation links.

[7]https://github.com/Chen-Cai-OSU/GNN-Over-Smoothing

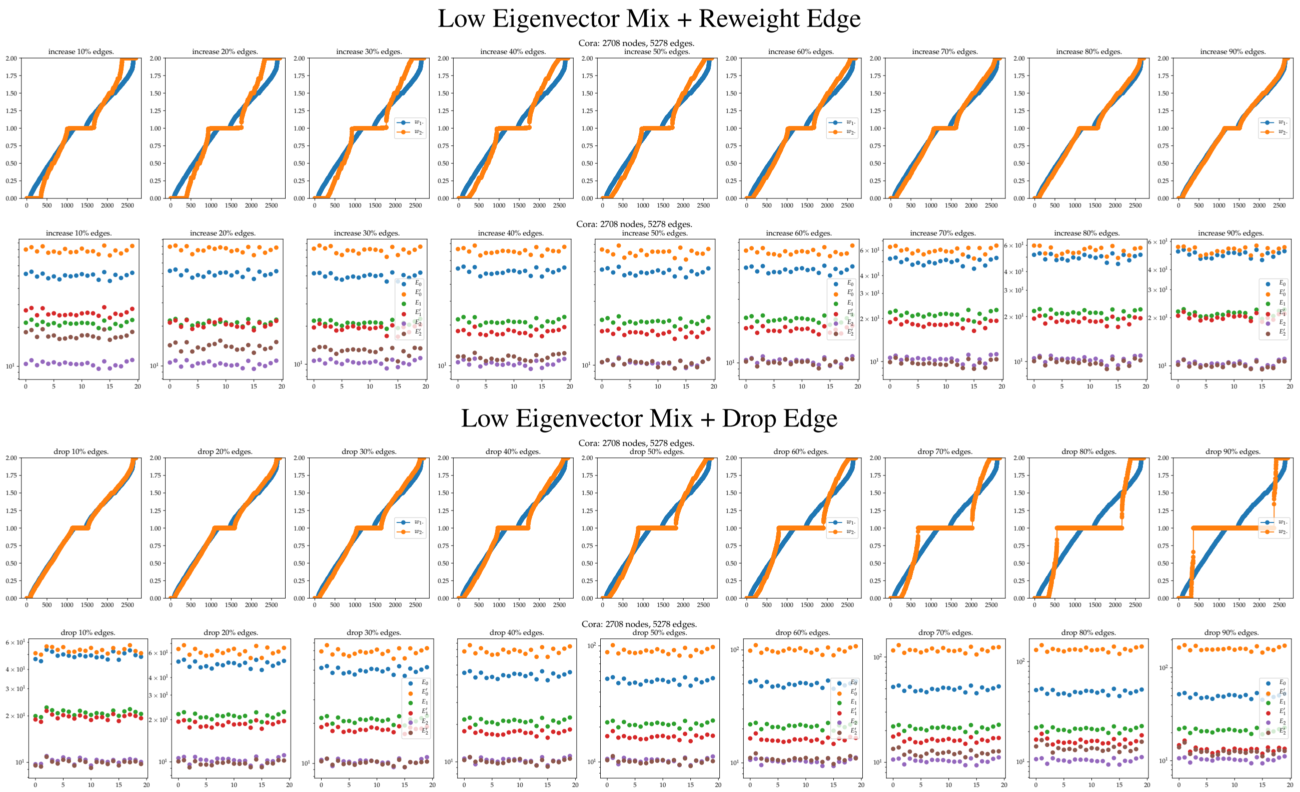

Figure 2. Cora.

References

Gurukar, S., Vijayan, P., Srinivasan, A., Bajaj, G., Cai, C., Keymanesh, M., Kumar, S., Maneriker, P., Mitra, A., Patel, V., et al. Network representation learning: Consolidation and renewed bearing. *arXiv preprint arXiv:1905.00987*, 2019.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Lee, Y. T. and Sun, H. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th annual acm sigact symposium on theory of computing*, pp. 678–687, 2017.

Li, G., Müller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? *arXiv preprint arXiv:1904.03751*, 2019.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Luan, S., Zhao, M., Chang, X.-W., and Precup, D. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 10943–10953, 2019.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint cs.LG/1905.10947*, 2019.

Qu, M., Bengio, Y., and Tang, J. Gmnn: Graph markov neural networks. *arXiv preprint arXiv:1905.06214*, 2019.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Spielman, D. A. and Srivastava, N. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6): 1913–1926, 2011.

Spielman, D. A. and Teng, S.-H. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth*

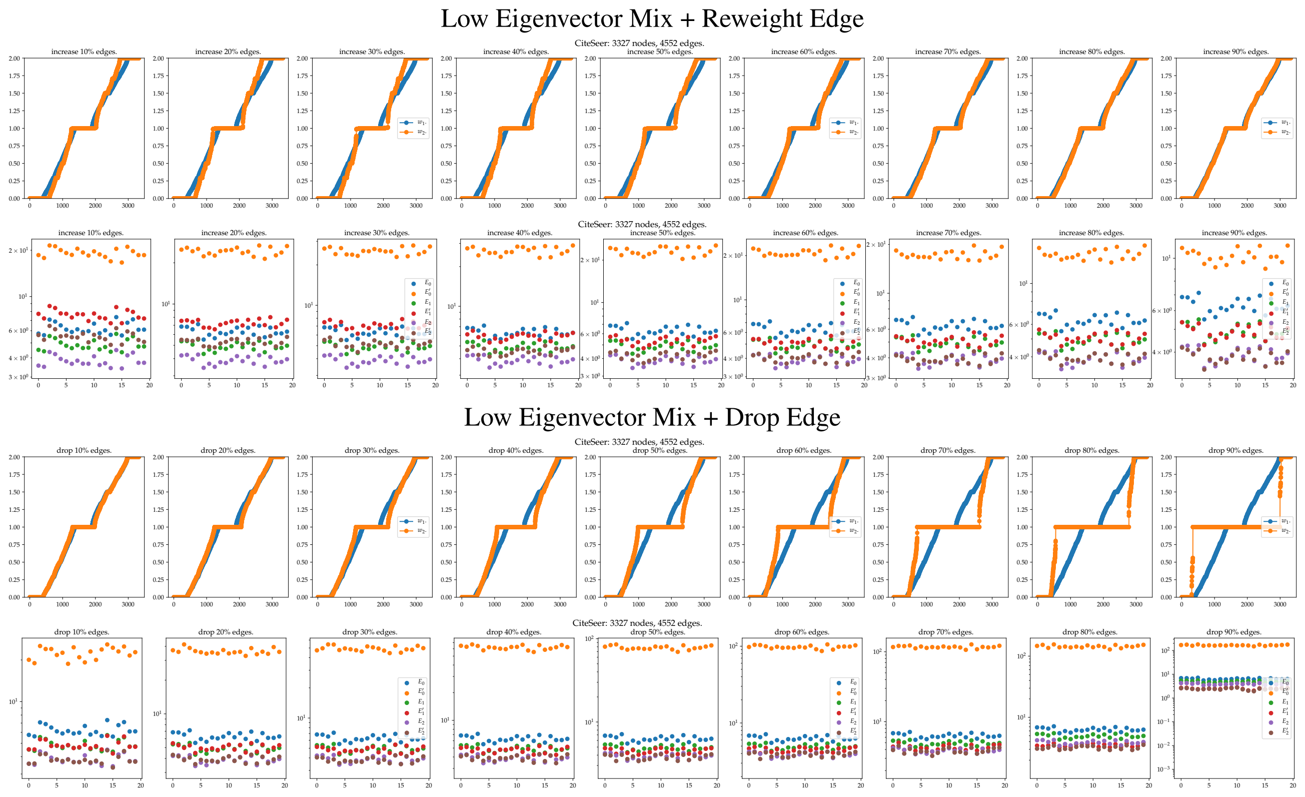

Figure 3. Citeseer.

annual ACM symposium on Theory of computing, pp. 81–90, 2004.

Spielman, D. A. and Teng, S.-H. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wu, F., Zhang, T., Souza Jr, A. H. d., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Xie, T. and Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120 (14):145301, 2018.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

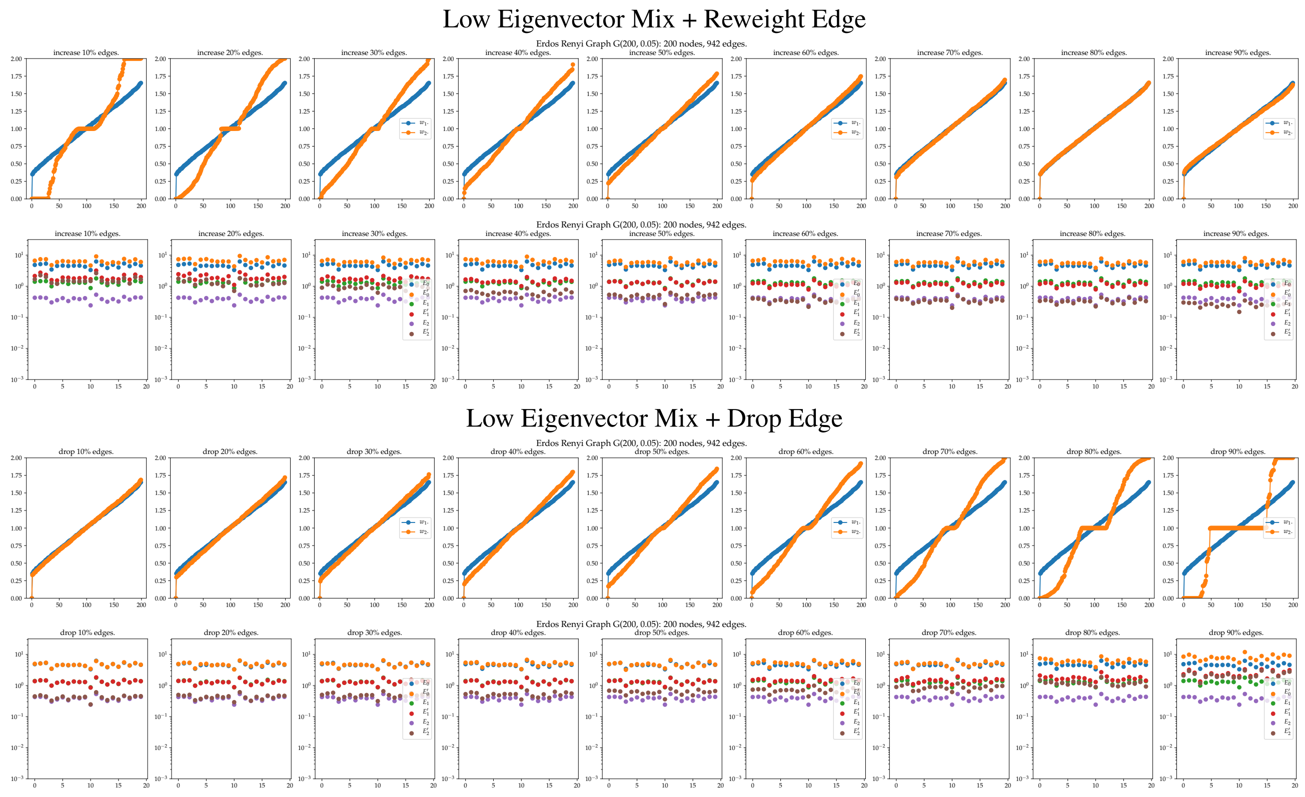

Figure 4. Random Graph.

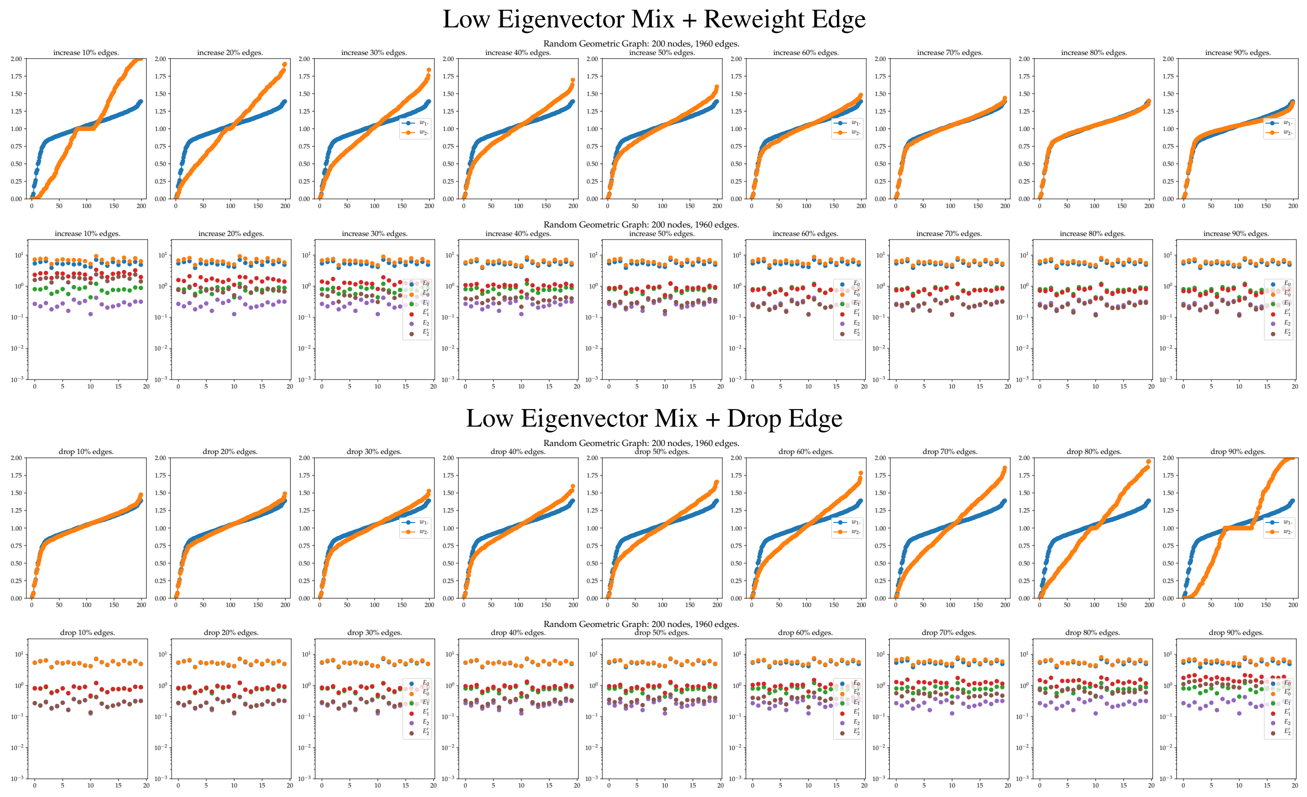

Figure 5. Random Geometric Graph.

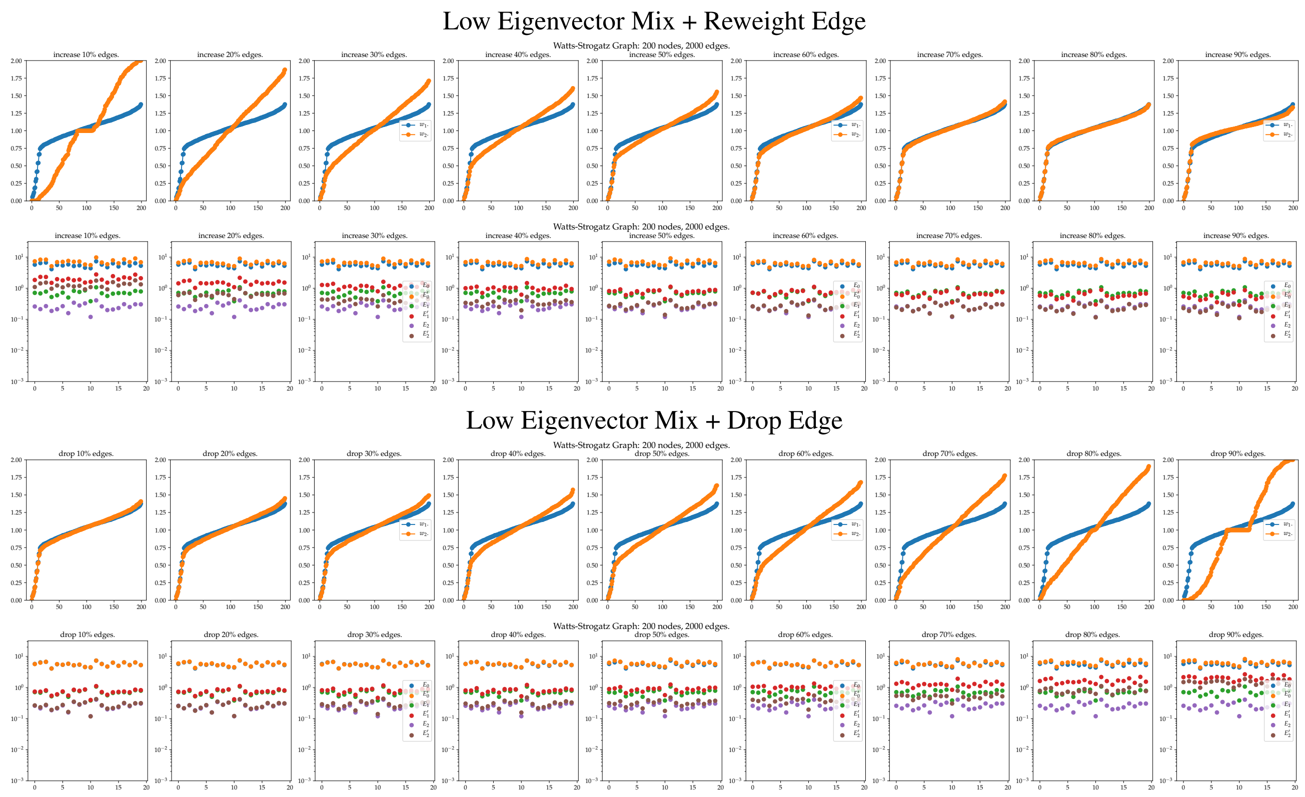

Figure 6. Watts-Strogatz Graph.

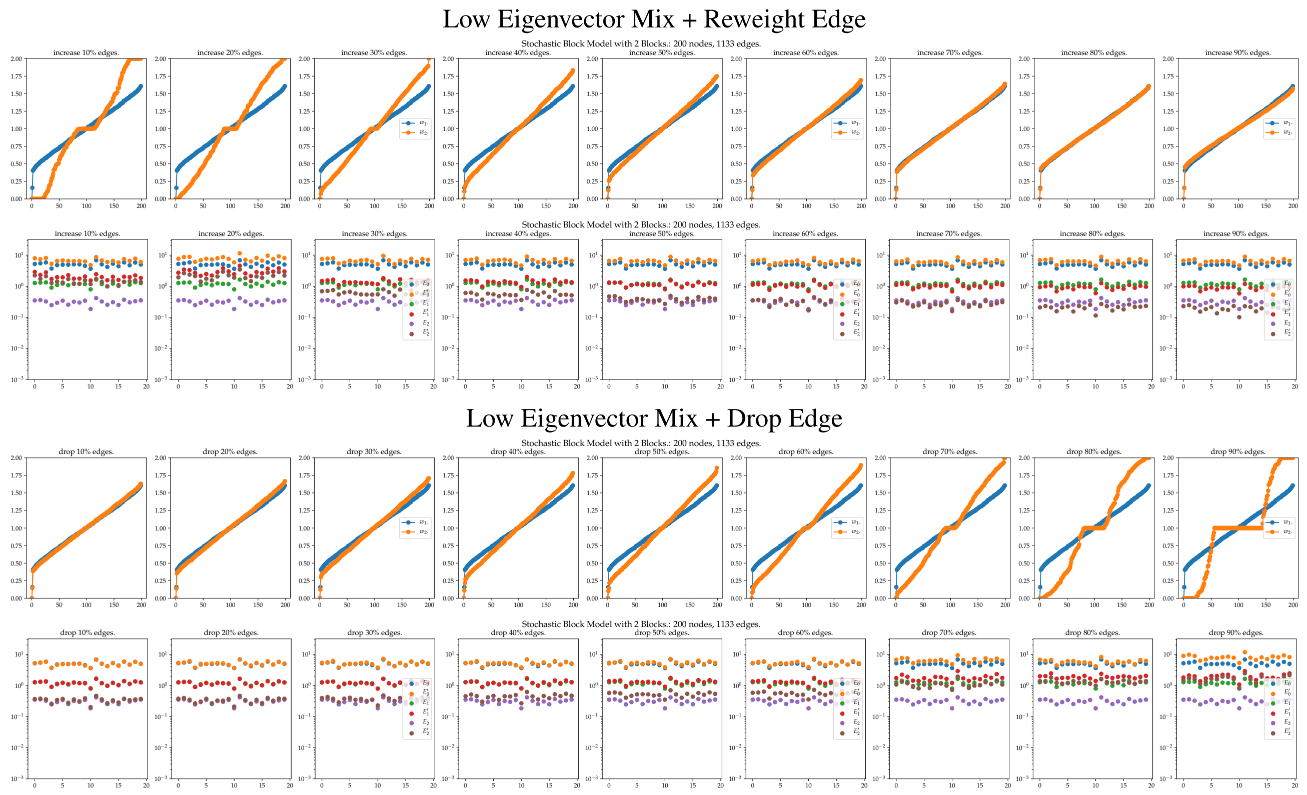

Figure 7. Stochastic Block Model with 2 Blocks.

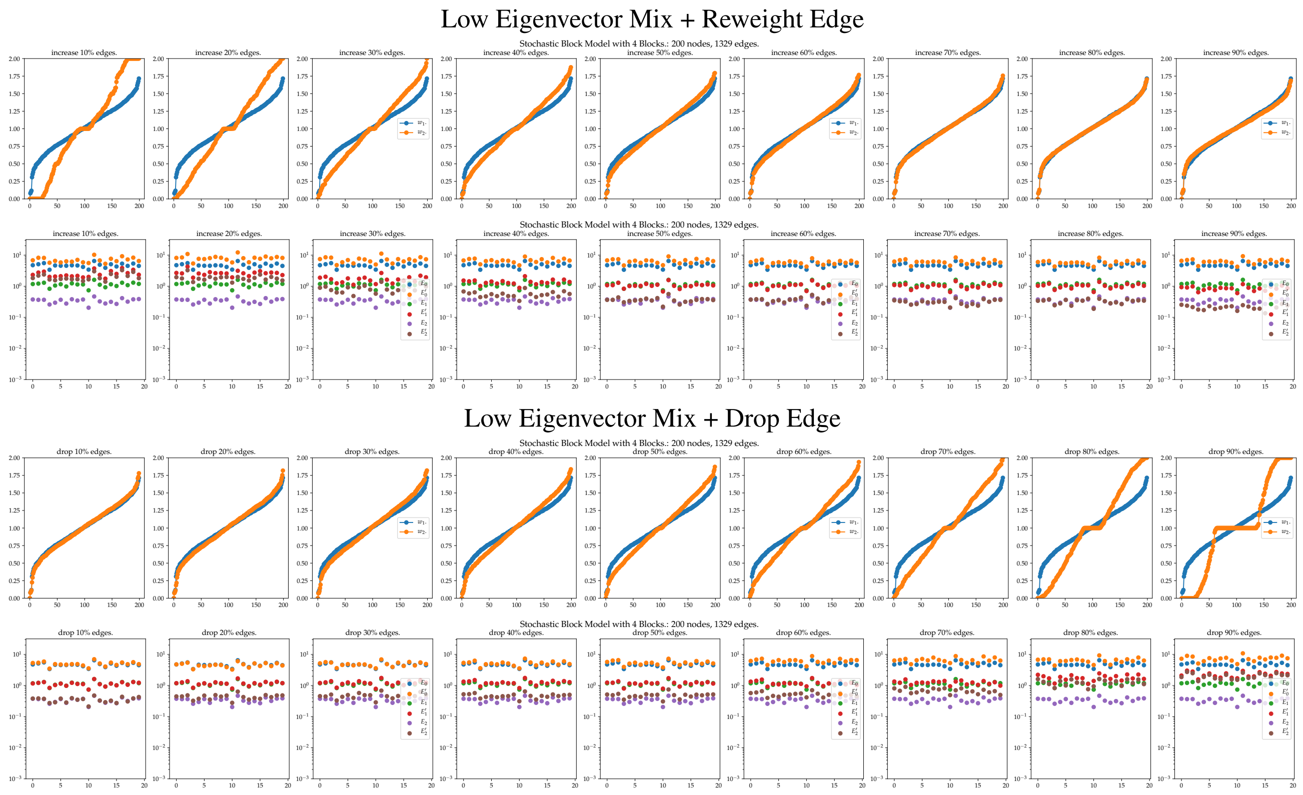

Figure 8. Stochastic Block Model with 4 Blocks.

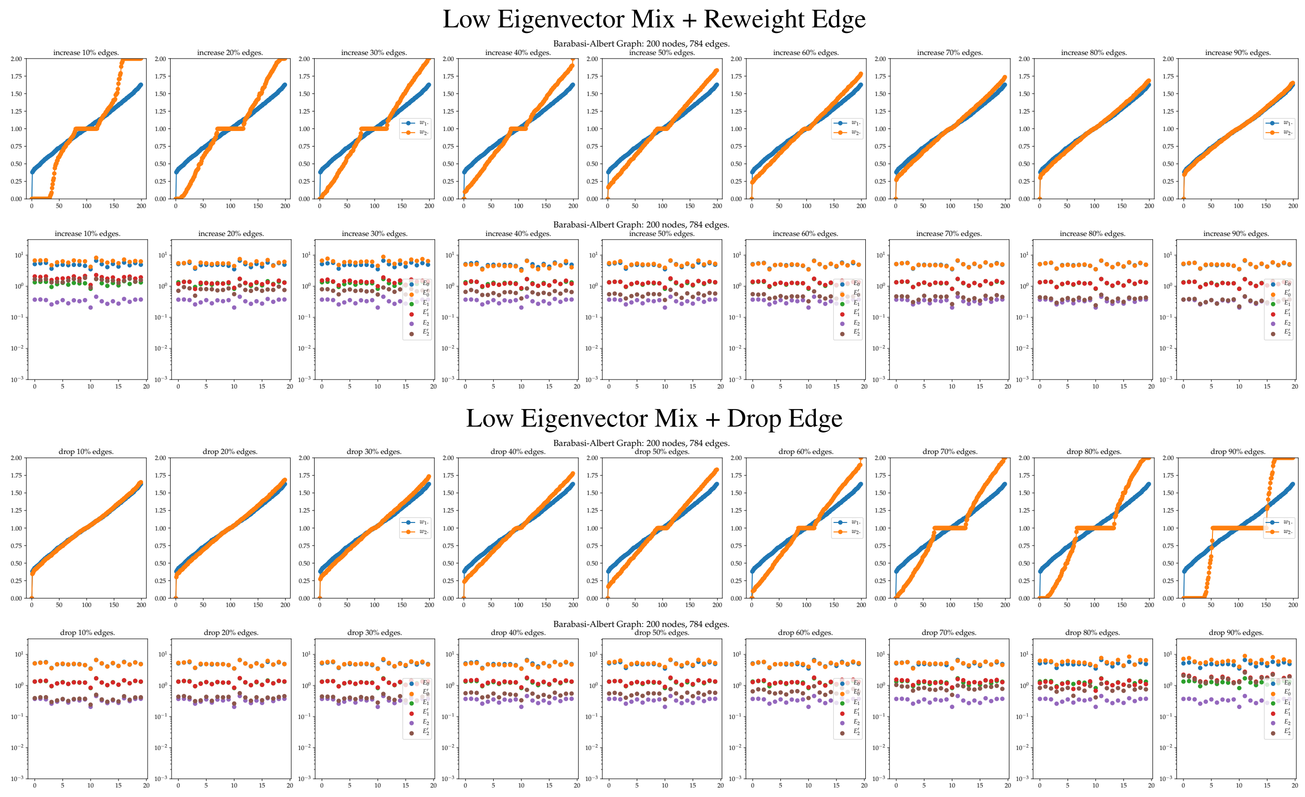

Figure 9. Barabasi-Albert Graph.