Node Feature Kernels Increase Graph Convolutional Network Robustness

Mohamed El Amine Seddik
Changmin Wu
Johannes Lutzeyer
Michalis Vazirgiannis

Message-Passing Framework

- Graph Convolutional Network (GCN, [3]): $\sigma(\tilde{A}X\Theta)$.
- Input: graph structure $\tilde{A}$ and node features X.
- Procedure: *Aggregation* step $\tilde{A}X$ and *Update* step $X\Theta$.

Research Problem

Question 1: *What is the influence of the graph structure information and node feature information on each other in the GCN?*
Question 2: *How will the inference drawn from a GCN be impacted by graph structural noise and is there a way to enhance its robustness to such noise?*

Take-away Message

The message-passing step dilutes (or ***in the extreme case completely ignores****) information present in the* ***node features*** *if the underlying graph structure is noisy (or in the extreme case* ***completely random****). Adding a* ***node feature kernel*** *addresses this problem.*

- Replace trainable weight Θ with normally distributed matrix, i.e., $\sigma(\tilde{A}XW)$, where $W_{ij} \sim \mathcal{N}(0,1)$, $\tilde{A} \in \mathbb{R}^{n\times n}$, $X \in \mathbb{R}^{n\times p}$ and $W \in \mathbb{R}^{p\times d}$.
- Insight from this model: we study the spectral behaviour of Gram matrix $G = \frac{1}{d}\sigma(\tilde{A}XW)\sigma(W^\intercal X^\intercal \tilde{A}^\intercal)$, specifically the eigenvector of G corresponding to its largest eigenvalue (the *informative* eigenvector).

Why *Random*GCN?

- To enable a Random Matrix Theory (RMT) analysis and its powerful tools in the theoretical study of neural networks.
- Empirically, *Random*GCNs **can** achieve comparable results with the vanilla GCN, i.e., no training is needed for the update weights in high dimensions.

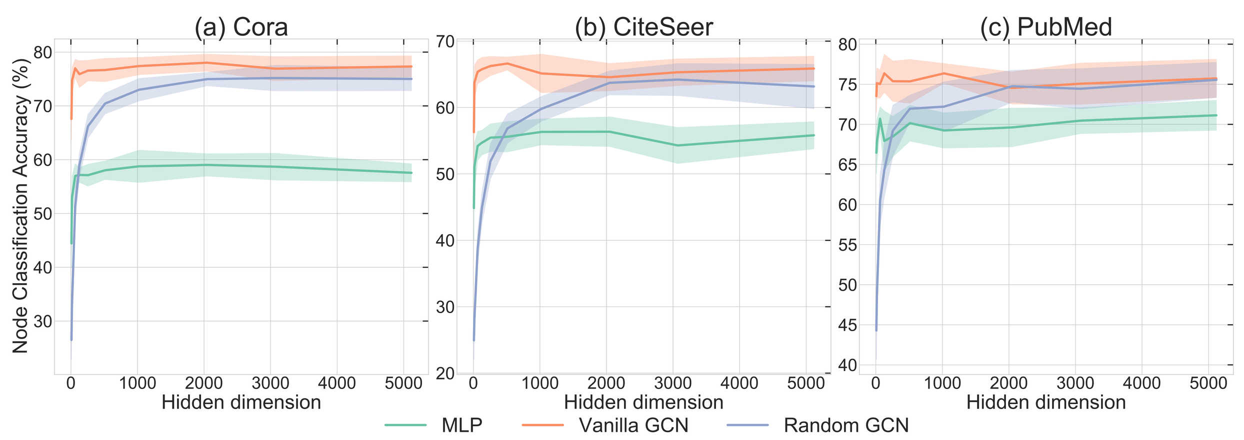

Figure 1: In high dimensions: GCN and *Random*GCN exhibit equivalent performance.

Assumptions

Data *Node features* follow a Gaussian Mixture Model,

$$x_i = (-1)^a \frac{\mu}{\sqrt{p}} + z_i \quad \text{with} \quad z_i \sim \mathcal{N}(\mathbf{0}, I_p/p).$$

Data *Graph Structure* follows a Stochastic Block Model (SBM),

$$\tilde{A} = \frac{1}{\sqrt{n}}(A - qq^\intercal) \quad \text{where} \quad A_{ij} \sim Ber(q_i q_j C_{ab}).$$

RMT *Growth Rate Assumptions* on the number of nodes, feature dimension, dimension of the random matrix W and edge probabilities.

RMT *Regularity Assumptions* on the activation function $\sigma(\cdot)$.

Theorem 1 (Informal)**.** *The extent to which the labels vector, that we are trying to predict, correlates with the informative eigenvector of the Gram matrix of our RandomGCN depends on the presence of cluster structure in the SBM.*

Theorem 2 (Main Corollary)**.** *When $\eta = 0$, let $\tilde{\boldsymbol{X}} = \tilde{\boldsymbol{A}}\boldsymbol{X}$ and $\bar{y}$ be the node labels, we have $|\bar{\boldsymbol{y}}^\intercal \hat{\boldsymbol{y}}|^2 \xrightarrow[n\to\infty]{} 0$, where $\hat{\boldsymbol{y}}$ is the eigenvector corresponding to the largest eigenvalue of $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\intercal$.*

Observation If a graph is **sufficiently** perturbed, then the GCN will fail to benefit from the node features **no matter how informative they are**.

Intuitive Explanation In a message-passing framework, node features are aggregated over graph neighbourhoods. When these neighbourhoods are random, we are aggregating random subsets of node features, thus destroying potential information.

Proposed Solution

This can be addressed by using the node feature information to directly inform the structure of the GCNs message passing scheme

$$\epsilon\hat{A} + (1-\epsilon)\tilde{K}.$$

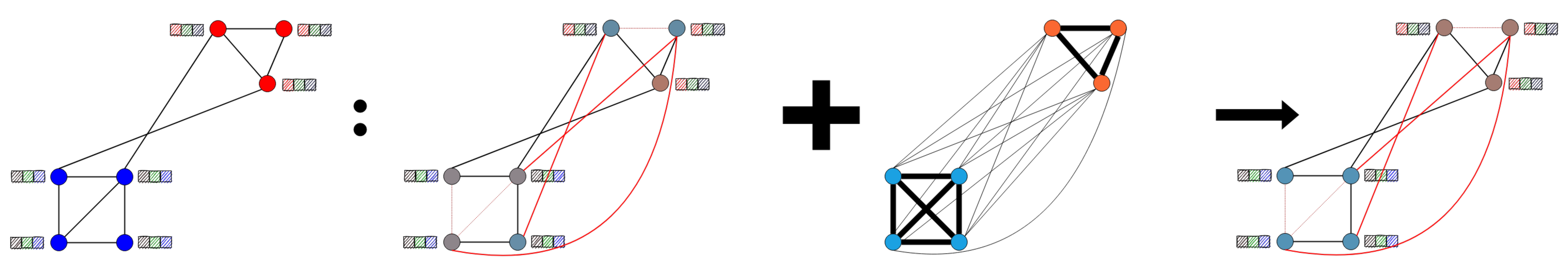

Figure 2: Adding a node feature kernel helps reconstruct meaningful neighbourhoods.

We evaluate the robustness of the GCN model on the **node classification task** with two structural perturbation schemes: ***edge deletion*** of ratio α and ***edge insertion*** of ratio β. Consistent with our theoretical analysis, we use a **single-layer** GCN model (with MLP readout). For simplicity we use the **linear kernel** $K_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$ and sparsify the dense kernel with the adjacency matrix $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$.

Stochastic Blockmodels (Community number: 2)

	(α, β)	SBM($p=0.25$, $q=0.25$) GCN	SBM($p=0.25$, $q=0.25$) GCN-k	SBM($p=0.275$, $q=0.25$) GCN	SBM($p=0.275$, $q=0.25$) GCN-k	SBM($p=0.225$, $q=0.25$) GCN	SBM($p=0.225$, $q=0.25$) GCN-k
Deletion	(0.0, 0.0)	50.53 ± 0.49	$\mathbf{66.36 \pm 0.81}$	$\mathbf{64.42 \pm 0.43}$	62.26 ± 1.04	$\mathbf{63.20 \pm 0.94}$	61.03 ± 1.08
	(0.2, 0.0)	51.03 ± 0.56	$\mathbf{65.44 \pm 1.07}$	58.63 ± 0.68	$\mathbf{71.57 \pm 1.42}$	$\mathbf{60.89 \pm 0.83}$	54.91 ± 1.00
	(0.5, 0.0)	49.29 ± 0.59	$\mathbf{64.14 \pm 1.01}$	60.76 ± 1.29	$\mathbf{68.80 \pm 2.04}$	58.41 ± 1.11	59.51 ± 2.47
Insertion	(0.0, 0.5)	50.57 ± 0.75	$\mathbf{68.57 \pm 1.25}$	60.49 ± 0.40	$\mathbf{68.20 \pm 1.38}$	58.82 ± 1.16	$\mathbf{63.54 \pm 0.97}$
	(0.0, 1.0)	49.19 ± 0.47	$\mathbf{59.31 \pm 0.58}$	53.67 ± 1.11	$\mathbf{66.57 \pm 1.73}$	54.87 ± 0.53	$\mathbf{60.84 \pm 0.75}$
Delet.+Insert.	(0.5, 0.5)	49.26 ± 0.59	$\mathbf{68.84 \pm 0.86}$	50.50 ± 0.37	$\mathbf{63.36 \pm 1.67}$	50.94 ± 0.86	$\mathbf{63.02 \pm 0.91}$
	(0.5, 1.0)	49.84 ± 0.69	$\mathbf{65.49 \pm 1.22}$	48.34 ± 0.22	$\mathbf{60.16 \pm 1.21}$	49.23 ± 0.45	$\mathbf{59.64 \pm 1.33}$

- ∗ The performance of the GCN degrades on SBMs with cluster structure (homophilic/heterophilic) as a result of *edge-deletion* and *edge-insertion* noise.
- ∗ Addition of the proposed kernel improves GCNs robustness against graph structural noise.

Citation/Co-purchase/Co-author graphs

	(α, β)	CoraFull GCN	CoraFull GCN-k	Photo GCN	Photo GCN-k	CS GCN	CS GCN-k
Deletion	(0.0, 0.0)	57.21 ± 0.84	56.88 ± 0.48	90.94 ± 0.49	90.09 ± 0.65	92.89 ± 0.41	92.63 ± 0.31
	(0.2, 0.0)	$\mathbf{57.25 \pm 0.67}$	55.56 ± 0.69	91.87 ± 0.40	92.19 ± 0.45	90.58 ± 0.48	90.89 ± 0.48
	(0.5, 0.0)	53.90 ± 0.70	54.62 ± 0.87	$\mathbf{91.10 \pm 0.40}$	87.97 ± 0.54	89.75 ± 0.60	$\mathbf{91.27 \pm 0.67}$
Insertion	(0.0, 0.5)	48.11 ± 0.89	$\mathbf{51.79 \pm 0.65}$	82.79 ± 1.43	84.18 ± 1.27	87.16 ± 0.65	$\mathbf{90.81 \pm 0.70}$
	(0.0, 1.0)	41.76 ± 1.03	$\mathbf{51.91 \pm 1.00}$	72.70 ± 6.40	$\mathbf{79.58 \pm 1.80}$	80.34 ± 0.80	$\mathbf{90.61 \pm 0.37}$
Delet.+Insert.	(0.5, 0.5)	34.70 ± 0.47	$\mathbf{46.50 \pm 0.61}$	69.70 ± 3.70	$\mathbf{74.65 \pm 2.36}$	73.75 ± 0.98	$\mathbf{87.28 \pm 0.72}$
	(0.5, 1.0)	27.50 ± 1.04	$\mathbf{43.04 \pm 0.77}$	61.13 ± 2.49	63.73 ± 5.04	66.26 ± 0.95	$\mathbf{87.51 \pm 0.58}$

- ∗ *Edge-insertion* noise seems to have a greater impact on real-world graphs.
- ∗ Node feature kernel can largely compensate the performance reduction caused by graph structural noise.

Deeper GCN model (4 layers)

	(α, β)	CS GCN	CS GCN-k ($\epsilon = 0.5$)	CS GCN-k ($\epsilon = 0.2$)	CS GCN-jk	CS GCNII	CS GCN-k-jk
Deletion	(0.0, 0.0)	88.44 ± 0.84	89.81 ± 0.52	91.64 ± 0.39	90.19 ± 0.59	$\mathbf{92.13 \pm 0.39}$	$\underline{91.73 \pm 0.26}$
	(0.2, 0.0)	89.19 ± 0.57	88.41 ± 0.53	$\underline{91.68 \pm 0.55}$	91.04 ± 0.65	91.56 ± 0.53	$\mathbf{91.89 \pm 0.77}$
	(0.5, 0.0)	86.68 ± 0.57	86.17 ± 1.06	88.91 ± 0.62	88.44 ± 0.69	$\underline{90.01 \pm 0.69}$	$\mathbf{91.43 \pm 0.60}$
Insertion	(0.0, 0.5)	70.94 ± 2.59	84.36 ± 1.19	88.84 ± 0.57	87.37 ± 0.66	$\underline{90.36 \pm 0.58}$	$\mathbf{92.66 \pm 0.49}$
	(0.0, 1.0)	35.84 ± 6.91	81.06 ± 3.94	88.27 ± 0.92	81.70 ± 0.63	$\underline{89.33 \pm 1.02}$	$\mathbf{91.42 \pm 0.48}$
Delet.+Insert.	(0.5, 0.5)	45.08 ± 4.82	76.27 ± 1.08	82.23 ± 1.08	73.08 ± 1.07	$\mathbf{88.66 \pm 0.70}$	$\underline{87.53 \pm 0.85}$
	(0.5, 1.0)	18.16 ± 3.86	53.12 ± 6.21	80.80 ± 0.99	63.84 ± 0.95	$\mathbf{88.77 \pm 0.89}$	$\underline{87.89 \pm 0.37}$

- ∗ Our proposed kernel performs better or on par with **Jumping Knowledge** [5] and **GCNII** [1] under all noise schemes.
- ∗ It can be further **combined with** JK to improve the performance.

We also observed similar behaviour for other GNNs (GIN[6], GraphSage[2] and GAT[4]). More experiments are on our paper.

[1] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li."Simple and deep graph convolutional networks", *International Conference on Machine Learning*, 2020.
[2] W. L. Hamilton, R. Ying & J. Leskovec, "Inductive Representatino Learning on Large Graphs," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025 – 1035, 2017.
[3] Thomas N. Kipf & M. Welling, "Semi-supervised classification with graph convolutional networks" *International Conference on Learning Representations*, 2017.
[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò & Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.
[5] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, S. Jegelka."Representation learning on graphs with jumping knowledge networks", *International Conference on Machine Learning*, 2018.
[6] K. Xu, W. Hu, J. Leskovec & S. Jegelka."How powerful are graph neural networks?", *International Conference on Learning Representations*, 2019.

PAPER

CODE