Node Feature Kernels Increase Graph Convolutional Network Robustness

Mohamed El Amine Seddik   Changmin Wu
Johannes F. Lutzeyer   Michalis Vazirgiannis

l'X ÉCOLE POLYTECHNIQUE
IP PARIS

The Random GCN

Our work focuses on the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$\sigma(\tilde{A}X\Theta) \quad \text{with } \Theta \text{ trainable.}$$

The Random GCN

Our work focuses on the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$\sigma(\tilde{A}X\Theta) \quad \text{with } \Theta \text{ trainable.}$$

To enable a random matrix theory analysis, we propose the *Random*GCN

$$\sigma(\tilde{A}XW) \quad \text{with} \quad W_{ij} \sim \mathcal{N}(0,1).$$

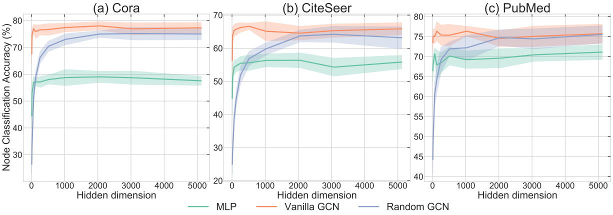
The Random GCN

Our work focuses on the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$\sigma(\tilde{A}X\Theta) \quad \text{with } \Theta \text{ trainable.}$$

To enable a random matrix theory analysis, we propose the *Random*GCN

$$\sigma(\tilde{A}XW) \quad \text{with} \quad W_{ij} \sim \mathcal{N}(0,1).$$



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

The Random GCN

Our work focuses on the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$\sigma(\tilde{A}X\Theta) \quad \text{with } \Theta \text{ trainable.}$$

To enable a random matrix theory analysis, we propose the *Random*GCN

$$\sigma(\tilde{A}XW) \quad \text{with} \quad W_{ij} \sim \mathcal{N}(0,1).$$

We gain insight on the behaviour of this model by studying its Gram matrix,

$$G = \frac{1}{d}\sigma(\tilde{A}XW)\sigma(W^\intercal X^\intercal \tilde{A}^\intercal).$$

The Random GCN

Our work focuses on the Graph Convolutional Network (GCN, Kipf and Welling, 2017)

$$\sigma(\tilde{A}X\Theta) \quad \text{with } \Theta \text{ trainable.}$$

To enable a random matrix theory analysis, we propose the *Random*GCN

$$\sigma(\tilde{A}XW) \quad \text{with} \quad W_{ij} \sim \mathcal{N}(0,1).$$

We gain insight on the behaviour of this model by studying its Gram matrix,

$$G = \frac{1}{d}\sigma(\tilde{A}XW)\sigma(W^\intercal X^\intercal \tilde{A}^\intercal).$$

We characterise the eigenvector of G corresponding to its largest eigenvalue (the *informative* eigenvector).

Theoretical Result

Assumptions:

- *Node features* follow a Gaussian Mixture Model.
- *Graph Structure* follows a Stochastic Block Model (SBM).
- *Growth Rate Assumptions* on the number of nodes, feature dimension, dimension of the random matrix W and edge probabilities.
- *Regularity Assumptions* on the activation function $\sigma(\cdot)$.

Theoretical Result

Assumptions:

- *Node features* follow a Gaussian Mixture Model.
- *Graph Structure* follows a Stochastic Block Model (SBM).
- *Growth Rate Assumptions* on the number of nodes, feature dimension, dimension of the random matrix W and edge probabilities.
- *Regularity Assumptions* on the activation function $\sigma(\cdot)$.

(Informal) Theorem

The extent to which the labels vector, that we are trying to predict, correlates with the informative eigenvector of the Gram matrix of our *Random*GCN depends on the presence of cluster structure in the SBM.

Intuition & Proposed Solution

Our Observation

If the graph is sufficiently perturbed then the GCN fails to benefit from the node features no matter how informative they are.

Intuition & Proposed Solution

Our Observation

If the graph is sufficiently perturbed then the GCN fails to benefit from the node features no matter how informative they are.

Intuition: Node features are aggregated over neighbourhoods. If these neighbourhoods are random, then we smooth random subsets of node features.

Intuition & Proposed Solution

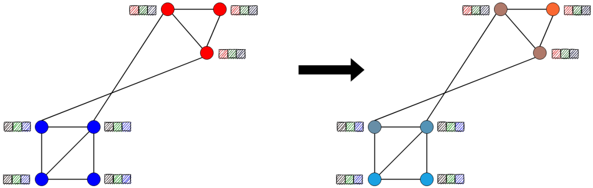
Our Observation

If the graph is sufficiently perturbed then the GCN fails to benefit from the node features no matter how informative they are.

Intuition: Node features are aggregated over neighbourhoods. If these neighbourhoods are random, then we smooth random subsets of node features.

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

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



Intuition & Proposed Solution

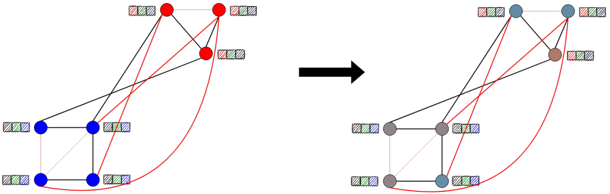
Our Observation

If the graph is sufficiently perturbed then the GCN fails to benefit from the node features no matter how informative they are.

Intuition: Node features are aggregated over neighbourhoods. If these neighbourhoods are random, then we smooth random subsets of node features.

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

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



Intuition & Proposed Solution

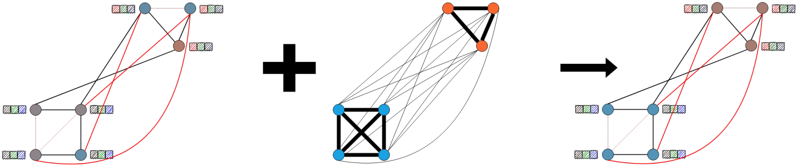
Our Observation

If the graph is sufficiently perturbed then the GCN fails to benefit from the node features no matter how informative they are.

Intuition: Node features are aggregated over neighbourhoods. If these neighbourhoods are random, then we smooth random subsets of node features.

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

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



Experiments: Stochastic Blockmodels

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- Single-layer model: (GCN + MLP)

Experiments: Stochastic Blockmodels

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- Single-layer model: (GCN + MLP)

- 2-community SBMs with no community structure, weakly homophilic communities and weakly heterophilic communities.

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

Experiments: Stochastic Blockmodels

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- Single-layer model: (GCN + MLP)

• 2-community SBMs with no community structure, weakly homophilic communities and weakly heterophilic communities.

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

• The addition of the **node feature kernel improves the GCN's robustness** against edge-deletion and edge insertion noise.

Experiments: Real-World Datasets

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- One-layers model: (GCN + MLP)

• Experiments on citation, co-purchase and co-authorship graphs.

		CoraFull		Photo		CS	
	(α, β)	GCN	GCN-k	GCN	GCN-k	GCN	GCN-k
	(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
Deletion	(0.2, 0.0)	**57.25 ± 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	**91.10 ± 0.40**	87.97 ± 0.54	89.75 ± 0.60	**91.27 ± 0.67**
Insertion	(0.0, 0.5)	48.11 ± 0.89	**51.79 ± 0.65**	82.79 ± 1.43	84.18 ± 1.27	87.16 ± 0.65	**90.81 ± 0.70**
	(0.0, 1.0)	41.76 ± 1.03	**51.91 ± 1.00**	72.70 ± 6.40	**79.58 ± 1.80**	80.34 ± 0.80	**90.61 ± 0.37**
Delet.+Insert.	(0.5, 0.5)	34.70 ± 0.47	**46.50 ± 0.61**	69.70 ± 3.70	**74.65 ± 2.36**	73.75 ± 0.98	**87.28 ± 0.72**
	(0.5, 1.0)	27.50 ± 1.04	**43.04 ± 0.77**	61.13 ± 2.49	63.73 ± 5.04	66.26 ± 0.95	**87.51 ± 0.58**

Experiments: Real-World Datasets

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- One-layers model: (GCN + MLP)

• Experiments on citation, co-purchase and co-authorship graphs.

		CoraFull		Photo		CS	
	(α, β)	GCN	GCN-k	GCN	GCN-k	GCN	GCN-k
	(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
Deletion	(0.2, 0.0)	**57.25 ± 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	**91.10 ± 0.40**	87.97 ± 0.54	89.75 ± 0.60	**91.27 ± 0.67**
Insertion	(0.0, 0.5)	48.11 ± 0.89	**51.79 ± 0.65**	82.79 ± 1.43	84.18 ± 1.27	87.16 ± 0.65	**90.81 ± 0.70**
	(0.0, 1.0)	41.76 ± 1.03	**51.91 ± 1.00**	72.70 ± 6.40	**79.58 ± 1.80**	80.34 ± 0.80	**90.61 ± 0.37**
Delet.+Insert.	(0.5, 0.5)	34.70 ± 0.47	**46.50 ± 0.61**	69.70 ± 3.70	**74.65 ± 2.36**	73.75 ± 0.98	**87.28 ± 0.72**
	(0.5, 1.0)	27.50 ± 1.04	**43.04 ± 0.77**	61.13 ± 2.49	63.73 ± 5.04	66.26 ± 0.95	**87.51 ± 0.58**

• On real-world datasets insertion noise seems to have a greater impact, which can largely be **compensated by the node feature kernel**.

Experiments: Real-World Datasets

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- One-layers model: (GCN + MLP)

- Baselines: Jumping Knowledge (Xu et al., 2018), GCNII (Chen et al., 2020)
- 4-layer GCN model

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

Experiments: Real-World Datasets

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- One-layers model: (GCN + MLP)

- Baselines: Jumping Knowledge (Xu et al., 2018), GCNII (Chen et al., 2020)
- 4-layer GCN model

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

- For better performance our kernel can be combined with JK.

Experiments: Real-World Datasets

- Node Classification
- Two structural perturbation schemes: *edge deletion* of ratio α, *edge insertion* of ratio β.
- Kernel choice: $K_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$
- Scalability issue with dense kernel: sparsification $\boldsymbol{K} \circ \hat{\boldsymbol{A}}$
- One-layers model: (GCN + MLP)

We also observed similar behaviour for other GNNs (GIN (Xu et al., 2019), GraphSage (Hamilton et al., 2017) and GAT (Veličković et al., 2018)).

Thank you for your attention!

@melaseddik @cmwu8 @JLutzeyer @mvazirg

https://github.com/ChangminWu/RobustGCN

Please visit us at our virtual poster to discuss :)

References

M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li."Simple and deep graph convolutional networks", *International Conference on Machine Learning (ICML),* 2020.

W. L. Hamilton, R. Ying & J. Leskovec, "Inductive Representation Learning on Large Graphs," *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS),* pp. 1025 – 1035, 2017.

Thomas N. Kipf & M. Welling, "Semi-supervised classification with graph convolutional networks" *International Conference on Learning Representations (ICLR)*, 2017.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò & Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations (ICLR),* 2018.

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 (ICML),* 2018.

K. Xu, W. Hu, J. Leskovec & S. Jegelka."How powerful are graph neural networks?", *International Conference on Learning Representations (ICLR)*, 2019.