KDD Cup 2020: Graph Adversarial Attack and Defense

Solution of the team **u1234x1234**, Dmitrii Tsybulevskii, u1234x1234@gmail.com

Introduction

I had no prior practical experience/knowledge on the problem of adversarial attacks and defence on the graph data. The solution is mostly ad-hoc and does not use any specific algorithms targeted to this kind of problem. The solution does not take into account any research from the field of adversarial attacks and defence on graph structured data.

Why used very simple models:
During the competition, I have read a few dozens of papers on the topic of graph adversarial attack / defence, but I've not used any ideas from them, due to the following reasons:

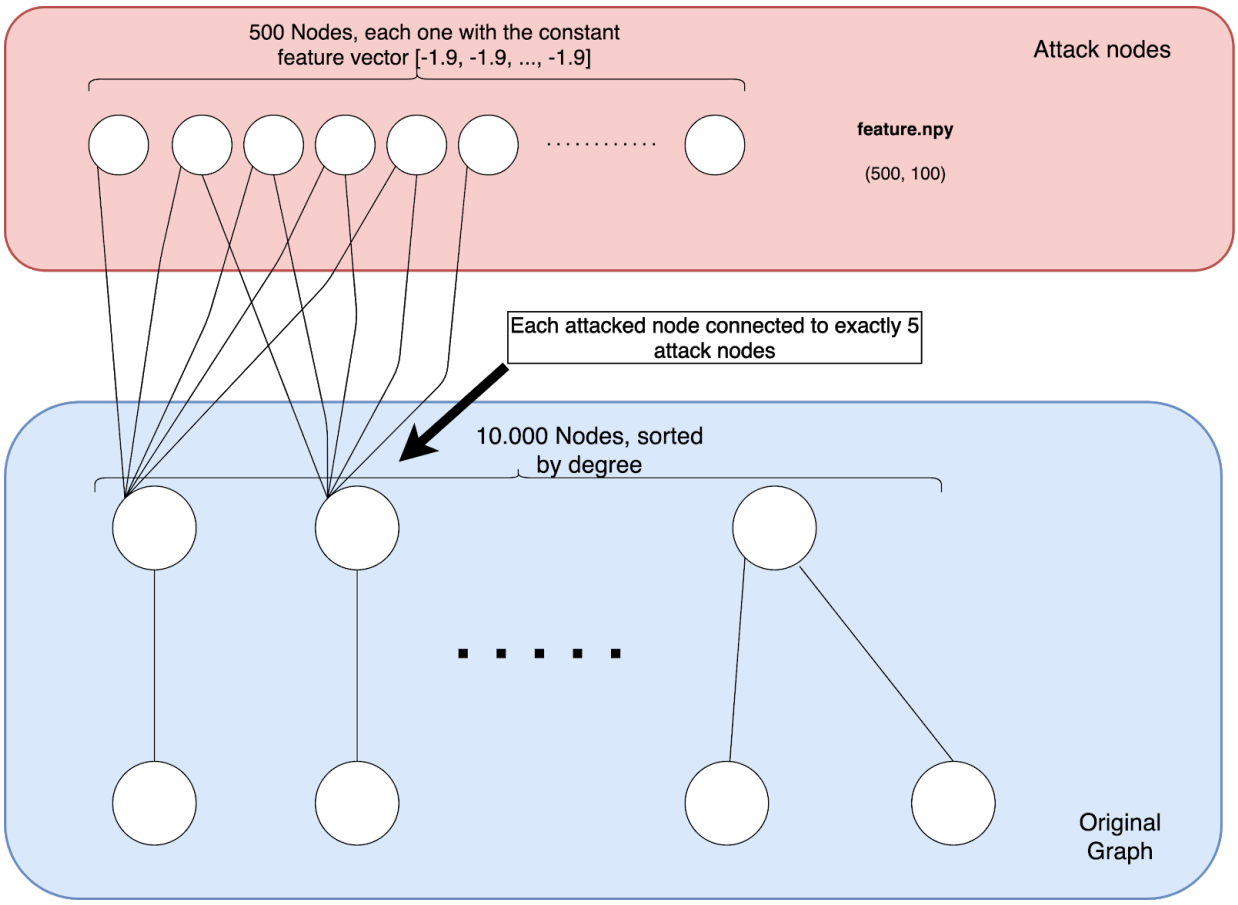
1. There are no standard datasets / baselines on the problem. Different papers use different evaluation protocols. Mostly it differs on the question what is an unnoticeable change: Adding/Removing/Modification of (edge)|(nodes)|(node features).
2. As a result of the previous point: no ready to use open source implementations of the attack / defence algorithm were found that exactly address the formulation presented in this competition.
3. Lack of time to implement more elaborated algorithms from scratch.

The final solution consists of the two parts: attacker and defender. According to the evaluation metric we need to maximize the accuracy of the defender algorithm under the assumption of different attackers from other teams, and provide the attacking subgraph that minimizes accuracies of classifiers provided by other teams.

Attacker

We need to reduce the overall node classification accuracy (non-targeted attack). Basic assumption: It is easier to trick defenders on the nodes with the small number of neighbours (degree of the node). Let's focus on them. Algorithm works as follows:

1. Create the sorted sequence of the nodes of the original graph, where the ordering is a degree of the node:
   [(node_1, degree_1), (node_2, degree_2), …, (node_n, degree_n)],
   Where **degree_i <= degree_i+1**
2. Take the first 10.000 nodes of this sequence.
3. Connect each node from this list of 10.000 nodes to the 5 random attackers nodes.
4. The feature vector of the each attacker node is constant: [-1.9, -1.9, …, -1.9]



We are restricted to adding 50.000 edges. There's a tradeoff: we could connect 50.000 different original nodes to the one of the attacking nodes, or 25.000 to the pair, 10.000 to 5, and so on. The numbers 10.000 and 5 was selected by the validation: drop in accuracy on adding different kinds of attacking nodes.

Defender

The intuition for a solution based on the following points:

- We do not know beforehand what attackers will be used by the other competitors
- Some teams will produce very simple (random) attackers that will have little to no effect on the original graph
- The final score will benefits from the solution that performs well even on the unmodified graph
- Let's just focus on a robust node classifier that works **without considering whether the original graph has been changed at all.**
- Performance constraints: The defender must be fast enough to make inference on the graph under 10 seconds. It restricts us from the very complex models / ensembles.

The original dataset was splitted into two parts: training set (90%) / validation set (10%). During the model selection, the model was trained on the training set and then evaluated on the validation set. Metric: multiclass classification accuracy. Different graph convolutional networks / hyperparameters were considered. The one that maximizes validation performance was chosen as the final model. The chosen configuration was retrained on the full dataset (100% of data).

Model configuration / hyperparameters:

Stack of the graph convolutional layers

Basic block: **Simplifying Graph Convolution layer [2]** with number of hops=4
Number of layers**: 3**
Number of hidden nodes per layer: **[140, 120, 100]**
Optimizer: **AdamW** with learning rate **0.01**
Normalization of input features**: BatchNorm**
Normalization between Graph Convolutional layers**: LayerNorm**
Activation between Graph Convolutional layers**: tanh**
Weight decay: **0**
Loss function: Multi-class classification **cross entropy with softmax**
Number of epochs: 600

As a alternative to the best performing **Simplifying Graph Convolution layer [2]** the following graph convolutional layers were considered:

- Graph convolution, GCN [3]
- Topology Adaptive Graph Convolutional layer [4]
- Graph Attention Network [5]
- GraphSAGE [6]
- Graph Isomorphism Network layer [7]

- The chebyshev spectral graph convolutional operator [8]
- Attention-based Graph Neural Network layer [9]

Used libraries:

- Deep Graph Library, DGL [1] with pytorch backend (GPU acceleration) for the node classification training
- Scipy, sparse matrix manipulations

Training time: 5 minutes on a single GPU, GTX 1080 TI

Future directions of work

Defender

According to the leaderboard, the defender solution has mean accuracy 0.676 ± 0.014, i.e most of the score stems from the classification capabilities of the model. Node classifier submitted as a final solution does not use any explicit adversarial defence mechanism. So it makes sense to add the one.

Attacker

The current solution uses only information about the degree of the attacked node. The natural direction for improvements is to take into account more information such as: features vector of the node, label of the node, features/ labels of the adjacent (or k-neighbour) nodes. Instead of hand-crafted algorithms use some kind of directed search in the space of adjacency matrix configuration / feature vectors of attacker nodes.

References

1. Wang, Minjie and Yu, Lingfan and Zheng, Da and Gan, Quan and Gai, Yu and Ye, Zihao and Li, Mufei and Zhou, Jinjing and Huang, Qi and Ma, Chao and Huang, Ziyue and Guo, Qipeng and Zhang, Hao and Lin, Haibin and Zhao, Junbo and Li, Jinyang and Smola, Alexander J and Zhang, Zheng. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019, https://arxiv.org/abs/1909.01315, https://github.com/dmlc/dgl

2. Wu, Felix, Zhang, Tianyi, Souza Jr, Amauri Holanda de,Fifty, Christopher, Yu, Tao, and Weinberger, Kilian Q. Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153, 2019. https://arxiv.org/abs/1902.07153

3. Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
4. Du, Jian, et al. "Topology adaptive graph convolutional networks." *arXiv preprint arXiv:1710.10370* (2017).
5. Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.
6. Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).
7. Xu, Keyulu, et al. "How powerful are graph neural networks?." *arXiv preprint arXiv:1810.00826* (2018).
8. Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems*. 2016.
9. Thekumparampil, Kiran K., et al. "Attention-based graph neural network for semi-supervised learning." *arXiv preprint arXiv:1803.03735* (2018).