A Study of Adversarial Attacks and Defense on Graph Convolution Networks

Vishal Peter
Department of Computer Science and Automation
Indian Institute of Science, Bangalore
vishalpeter@iisc.ac.in

Abstract

Deep Neural Networks(DNNs) find a strong presence today in various applications such as image classification, text generation, audio recognition, and graph data analysis. But, recent studies have shown that DNNs are vulnerable to adversarial attacks. While there are several works studying such attacks on DNNs on domains like images and natural language processing – similar studies on graph deep learning models, for instance, Graph Convolutional Networks(GCN) are still at its infancy. However, in domains where they are likely to be used, e.g. the web, adversaries are common. In this term paper, I present a study of [7] - which proposes an adversarial attack model for GCN and [6], which provides a defense strategy against such attacks. Furthermore, I have implemented[1] the attack and the defense model presented in the papers and analysed their effectiveness on different graph datasets.

1 Paper I : *Adversarial Attacks on Neural Networks for Graph Data*

[7] is one of the first works studying adversarial attacks on deep learning models for graphs. The paper provides an algorithm, NETTACK, which performs small, unnoticeable perturbations on graph data to drastically reduce classification accuracy for the attacked nodes. Before detailing the attack model, I first define graph convolution operation and different types of attacks.

1.1 Preliminaries

1.1.1 Graph Convolution

Let, $G = (A, X)$ be an attributed graph where $A \in \{0,1\}^{N*N}$ is the adjacency matrix and $X \in \{0,1\}^{N*D}$ represents the nodes' features. Let the set of nodes be $\mathcal{V} = \{1, 2...N\}$. Given a subset $\mathcal{V}_L \subseteq \mathcal{V}$ of labeled nodes, with class labels from $\mathcal{C} = \{1, 2...c_K\}$, the goal of node classification is to learn a function $g : \mathcal{V} \rightarrow \mathcal{C}$ which maps each node $v \in \mathcal{V}$ to one class in $\mathcal{C}$.

NETTACK targets node classification task via graph convolution. In graph convolution, node features at hidden layer $l + 1$ is defined as

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of the (undirected) input graph G after adding self-loops via the identity matrix I_N . $W^{(l)}$ is the trainable weight matrix of layer l, $\tilde{D}_{ii} = \Sigma_j \tilde{A}_{ij}$, and $\sigma(\cdot)$ is an activation function (usually ReLU). In the first layer we have $H^{(0)} = X$, i.e. the nodes' features

[1] available at https://github.com/viz27/Attack_on_Graphs

Done towards credit requirement for course E0 270, Machine Learning 2020, Bangalore.

used as input. Following the authors of [2], GCNs with a single hidden layer is described as:

$$Z = f_\theta(A, X) = softmax(\hat{A}\sigma(\hat{A}XW^{(1)})W^{(2)}) \tag{2}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. The output Z_{vc} denotes the probability of assigning node v to class c. θ denotes the set of all learnable parameters, i.e. $\theta = \{W^{(1)}, W^{(2)}\}$. The optimal parameters θ^* are then learned in a semi-supervised fashion by minimizing cross-entropy on the output of the labeled samples $\mathcal{V}_\mathcal{L}$, i.e. minimizing

$$L(\theta; A, X) = -\sum_{v \in \mathcal{V}_L} lnZ_{v,c_v} \quad , \; Z = f_\theta(A, X) \tag{3}$$

where c_v is the given label of v from the training set.

1.1.2 Adversarial Attack Terminologies

Poisoning Attack: Poisoning attack tries to affect the performance of the model by making adversarial perturbations to the training dataset. In this case, after the attacker changes the data, the model is retrained. Most existing works in attacking graphs are on poisoning attacks as node classification task for graph data is commonly performed in the transductive learning setting, ie. where labelled and unlabelled nodes are trained together.
Evasion Attack: Evasion attack only changes the testing data, which does not require to retrain the model. Here, the parameters of the trained model are assumed to be fixed.

1.1.3 Unnoticeable Perturbations

Every attack which modifies the graph must ensure that the changes are imperceptible. In image data, this is verifiable visually and by using simple constraints. But in the graph setting, this is harder as the graph structure is discrete preventing the use of infinitesimally small changes and sufficiently large graphs are not suitable for visual inspection. Consequently, there is no clear discussion in existing research about how to set the adversarial cost for attacks on graph data so far.

A simple strategy at making unnoticeable perturbations, adopted by [7], is to limit the number of allowed changes by a small budget Δ:

$$\Sigma_u \Sigma_i |X_{ui} - X^{'}_{ui}| + \Sigma_{u<v} |A_{uv} - A^{'}_{uv}| \leq \Delta \tag{4}$$

where $X^{'}$ and $A^{'}$ refers to the node features and adjacency matrix after perturbation. In the paper, such an approach at imperceptibility is used, while additionally ensuring that the degree distribution of the graph does not change much(via statistical two-sample test for power-law distributions[1]) and that some feature statistics are preserved(based on feature co-occurrence among nodes).

1.2 Attack Model

The adversarial attack problem that NETTACK tries to solve can be formally stated as

Problem 1 *Given a graph $G = (A, X)$ and a target node v_0. Let c_{old} denote the class for v_0 based on the graph G (predicted or ground truth). Determine*

$$\underset{(A', X')}{argmax} \; \underset{c \neq c_{old}}{max} \; lnZ^*_{v_0,c} - lnZ^*_{v_0,c_{old}} \tag{5}$$

Subject to $Z^* = f_{\theta^*}(A', X') \;\; with \;\; \theta^* = \underset{\theta}{argmin}\, L(\theta; A', X')$ and A', X' satisfies imperceptibility constraints.

That is, NETTACK aims to find a perturbed graph $G^{'}$ that classifies v_0 as $c \neq c_{old}$ and has maximal 'distance' (in terms of log-probabilities/logits) to c_{old}. Problem 1 defines *poisoning attack* and as a simpler variant, one can also consider an *evasion attack* assuming the parameters are static and learned based on the old graph, $\theta^* = \underset{\theta}{argmin}\, L(\theta; A, X)$.

As solving problem 1 is highly challenging, a sequential approach is proposed in the paper, where first a *surrogate model* is attacked, thus, leading to an attacked graph which is subsequently used to train the GCN model.

Surrogate model. To obtain a tractable surrogate model that still captures the idea of graph convolutions, a linearization of the model from Eq. 2 is performed. Since the goal is to maximize the difference in the log-probabilities of the target node, the instance-dependent normalization induced by the softmax can also be ignored, leading to

$$Z^{'} = \hat{A}\hat{A}XW^{(1)}W^{(2)} = \hat{A}^2XW \tag{6}$$

NETTACK finds optimal $A^{'}, X^{'}$ which maximizes loss for target node on the surrogate model using a greedy approximation scheme. At each iteration, the perturbation(edge or feature, satisfying imperceptibility constraints) leading to maximum increase in loss for target node is made. The process is continued until the budget Δ is reached. The algorithm also handles efficient computation of scores associated with perturbations and checks to assert compliance to imperceptibility constraints.

2 Paper II : *Adversarial Examples on Graph Data: Deep Insights into Attack and Defense*

In the paper [6], a new attack method, as well as a defense technique against attacks, is proposed. For attack, [6] shows that the discreteness problem of graph data could be resolved by introducing integrated gradients(similar to [5]) which could accurately reflect the effect of perturbing certain features or edges. The paper also presents a defense approach which inspects the graph and recovers potential adversarial perturbations based on the observation that the adversarially manipulated graph for the targeted attack differs from normal graphs statistically on some metrics. In this term paper, I will be covering the defense method proposed in the paper.

2.1 Insights into Attacking Strategies

The paper makes the following important observations on graphs subjected to adversarial attack.

i Perturbing edges is more effective at attacking a node than modifying features.

ii Nodes with more neighbours are difficult to attack compared to those with fewer neighbours.

iii An adversarial attack tends to connect target node to nodes with different features and labels or disconnect target node from nodes with similar features and labels

An explanation for these observations comes from the fact that the GCN models strongly rely on the graph structure and local aggregations during the convolution operation. Attacking edges are more effective since adding or removing one edge affects all the values in a node's feature vector during the aggregation. In contrast, modifying one feature only affects a single value in the feature vector and such perturbation can easily get masked by neighbours a node having higher degree. Hence, attack models typically try to make perturbations which increases dissimilarity between two connected nodes.

2.2 Defense Model

Based on these observations, a very simple defense model, GCNJaccard, is proposed in the paper. GCNJaccard uses a pre-processing based approach. The input graph is first analysed to find Jaccard similarity between nodes connected edges in the graph. Jaccard similarity between two nodes u and v with binary features is given as

$$J_{u,v} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{7}$$

where M_{11} is the number of features where u and v both have value 1, M_{01} is the number of features having value 0 for u but 1 for v and M_{10} is the number of features having value 1 for u but 0 for v. GCNJaccard then removes those edges which connect nodes with similarity score below a threshold, as such edges are potentially added by attackers. Although a clean graph may contain a small number of such edges, their influence while learning on the graph is usually negligible. In some cases, removing these edges even improves classification accuracy on clean graphs as the operation can be viewed as outlier elimination. The algorithm is computationally efficient as well as it needs only one pass over the edges in the graph, leading to $O(|E|)$ complexity, where $|E|$ denotes the number of edges in the graph.

Table 1: Jaccard score threshold's impact on clean graph

	Citeseer		Cora	
Threshold	Edges Removed	Accuracy	Edges Removed	Accuracy
0.00	0	0.7198	0	0.8234
0.01	96	0.7216	548	0.8234
0.03	483	0.7186	1015	0.8104
0.04	817	0.7180	1203	0.7953
0.05	1085	0.7079	1561	0.7741
0.10	2284	0.6949	3253	0.7133

Table 2: Accuracy readings after running `NETTACK` and `GCNJaccard`

Method	NETTACK	GCNJaccard(threshold)					
		0.01	0.02	0.03	0.04	0.05	0.10
Cora	**0.10**	0.30	0.52	0.68	**0.70**	0.65	0.78
Citeseer	**0.15**	0.30	0.30	0.40	0.45	**0.55**	0.62

3 Experiments

Two experiments have been performed by applying `NETTACK` and `GCNJaccard` on two common graph datasets, Citeseer and Cora. The algorithms were implemented by using their pytorch implementation available at deeprobust[3] library.

3.1 Experiment 1 : Jaccard score threshold's impact on clean graph

Before applying `GCNJaccard` on graph data, the Jaccard score threshold needs to be set. `GCNJaccard` then removes all edges connecting nodes with similarity score below the threshold. The results of running `GCNJaccard` with varying thresholds on clean graphs is given in Table 1. As seen, for small threshold values(<0.05) applying `GCNJaccard` doesn't affect the overall accuracy of the GCN model.

3.2 Experiment 2 : Effectiveness of `NETTACK` and `GCNJaccard`

To study the effectiveness of `NETTACK`, a subset of nodes with varying margin of classification(after clean graph is trained on GCN) was selected. Specifically, 40 nodes were selected from among the unlabelled nodes, 10 nodes with the highest margin, 10 nodes with the lowest margin and remaining 20 nodes randomly. `NETTACK` was run targeting each of these nodes one at a time(poisoning attack). Classification accuracy(for the 40 nodes) is recorded after the perturbed graph is retrained on GCN. Then, `GCNJaccard` was run on the perturbed graph to study how effective as a defense it is against `NETTACK`. Results are given in Table 2. As seen, `NETTACK` proves to be extremely effective at triggering GCN misclassification on attacked nodes. Furthermore, as the Jaccard score threshold value increases, `GCNJaccard` gives better performance on attacked nodes. But a very large threshold(>0.1) would mean reduced performance in the absence of attacks. Considering these trade-offs, a threshold value of 0.04 for Citeseer and 0.05 for Cora seems to provide reasonable defense against attacks without compromising accuracy on graphs that haven't been attacked.

4 Areas for improvement

In [7], an imperceptibility metric maintaining degree distribution of graphs and some feature statistics were used. However, in [6], using a different metric(Jaccard similarity), most of these perturbations could be detected. This issue can be found in many existing attacks(as explained in [4]) and can be avoided by considering various imperceptibility metrics during the attack stage.

References

[1] Alessandro Bessi. *Two samples test for discrete power-law distributions*. 2015.

[2] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017.

[3] Yaxin Li et al. *DeepRobust: A PyTorch Library for Adversarial Attacks and Defenses*. 2020.

[4] Lichao Sun et al. *Adversarial Attack and Defense on Graph Data: A Survey*. 2018.

[5] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017.

[6] Huijun Wu et al. *The Vulnerabilities of Graph Convolutional Networks: Stronger Attacks and Defensive Techniques*. 2019.

[7] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. *Adversarial Attacks on Neural Networks for Graph Data*. 2018.