Adversarial Attacks on Graph Structured Data

Author: Hanjun Dai

Georgia Tech College of Computing

Presenter: Faizan Ahmad
https://qdata.github.io/deep2Read

Outline

Introduction

- What are Adversarial Attacks?
- Success of Adversarial Attacks
  - Images
  - Text
  - **Graphs?** Adversaries in graph-based domains.
- Adversarial Attacks on Graph & Node Classification by changing graph structured (edges).

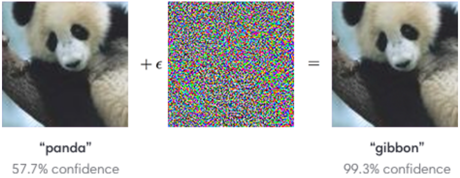

Figure: Adversarial Attack Example. Carefully crafted image is added to the input to make the model misclassify it.

Basic Idea

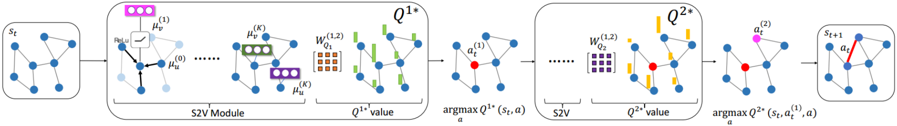

Figure: Small edge perturbations of the graph structure lead to misclassification of the target

- Edge modification achieved by
  1. Reinforcement Learning (Q-learning)
  2. Random Modifications
  3. Gradient based attack
  4. Genetic Algorithms

Why Attacks on Graphs?

- Large body of work on adversarial attacks for Images and Text
  - DeepFool (Moosavi et al. [1])
  - Adversarial Examples in Physical World (Kurakin et al. [2])
  - Hotflip (Ebrahimi et al. [3])
  - Plenty more..
- No work on adversarial attacks for graphs
- Many new challenges
  - Discrete Domain
  - Network Effects

- Input Graphs $G_1, G_2, ..., G_n \in G$
- $G = (V, E)$
  - V is the set of nodes
  - E is the set of edges
  - Nodes and edges can have featured denoted by $x(v) \in R^{D_f}$ and $w(e) \in R^{D_e}$ respectively
- Graph classification (inductive)
- Node classification $c_i \in V$ (transductive)
- GNN family models as:

$$u_v^{(k)} = h^{(k)}(\{w(u,v), x(u), u_u^{(k-1)}\}_{u \in N(v)}, x(v), u_v^{k-1})$$

Related Work

- Graph Neural Networks
  - Node Embeddings/Classification [4]
  - Graph Classification [5]
- Adversarial Attacks
  - Evasion Attacks [1]
  - Poisoning Attacks [6]
- Adversarial Attacks on Graphs
  - Using Greedy Approximation (Last Time)
  - Via Reinforcement Learning **(This Work)**

- Original Graph $G = (A, X)$
- Perturbed Graph $\hat{G} = (\hat{A}, X)$
- Optimization problem becomes:

$$max_{\hat{G}} f(\hat{G}, c) \neq y$$
$$s.t. \hat{G} = g(f, (G, c, y))$$
$$I(G, \hat{G}, c) = 1$$

- I is equivalence estimator (how similar are two graphs)

$$I(G, \hat{G}, c) = |(E - \hat{E})U(\hat{E} - E)| < m$$
$$\hat{E} \subseteq N(G, b)$$

- Attack is modeled as a markov decision process
  - **Action** a**:** Add or delete edges
  - **State** s**:** Modified graph
  - **Reward** r**:** Whether the classifier is fooled
    - -1 if no, 1 if yes
- Reward can be discrete, or continuous.
- Sample trajectory: $(s_1, a_1, r_1, ..., s_m, a_m, r_m, s_{m+1})$
- Q-learning for optimization

- Bellman optimality equation to pick the best action using Q function

$$Q^*(s_t, a_t) = r(s_t, a_t) + \lambda max_{a^*} Q^*(s_{t+1}, a^*)$$

- $Q^*(s_t, a_t) =$ Immediate Reward + Expected Future Reward
- Implicitly suggest **greedy policy**
- For efficiency, decompose into two Q functions

$$\mathrm{Q}^{1*}(s_t, a_t^{(1)}) = max_{a_t^{(2)}} Q^{2*}(s_t, a_t^{(1)}, a_t^{(2)})$$

$$\mathrm{Q}^{2*}(s_t, a_t^{(1)}, a_t^{(2)}) = r(s_t, a_t \leftarrow (a_t^{(1)}, a_t^{(2)}) + max_{a_{t+1}^{(1)}} Q^{1*}(s_t, a_{t+1}^{(1)})$$

- Final Q function to learn

$$max_\theta \sum_{i=0}^{N} Q^*(a_t|s_t;\theta)[r(\hat{G},c)]$$

- How to learn? Use GNNs

$$Q^{1*}(s_t, a_t^{(1)}) = W_{Q_1}^{(1)} \sigma(W_{Q_1}^{(2)}[u_{a_t^{(1)}}, u(s_t)])$$

$$Q^{2*}(s_t, a_t^{(1)}, a_t^{(2)}) = W_{Q_2}^{(1)} \sigma(W_{Q_2}^{(2)}[u_{a_t^{(1)}}, u_{a_t^{(2)}}, u(s_t)])$$

Attack

Gradient Based White Box

- Use a formulation of GNNs that allow gradients computations

$$\begin{aligned}\mu_v^{(k)} = h^{(k)}\Big(\quad & \{\alpha_{u,v}\left[w(u,v), x(u), \mu_u^{(k-1)}\right]\}_{u \in \mathcal{N}(v)} \cup \\ & \{\alpha_{u',v}\left[w(u',v), x(u'), \mu_{u'}^{(k-1)}\right]\}_{u' \notin \mathcal{N}(v)}, \\ & x(v), \mu_v^{(k-1)}\Big), k \in \{1, 2, \ldots, K\} \qquad (17)\end{aligned}$$

- Find gradients for each edge and do gradient ascent

$$\frac{\partial \mathcal{L}}{\partial \alpha_{u,v}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\mu_k}^{\top} \cdot \frac{\partial \mu_k}{\partial \alpha_{u,v}}.$$

Attack

Gradient Based White Box

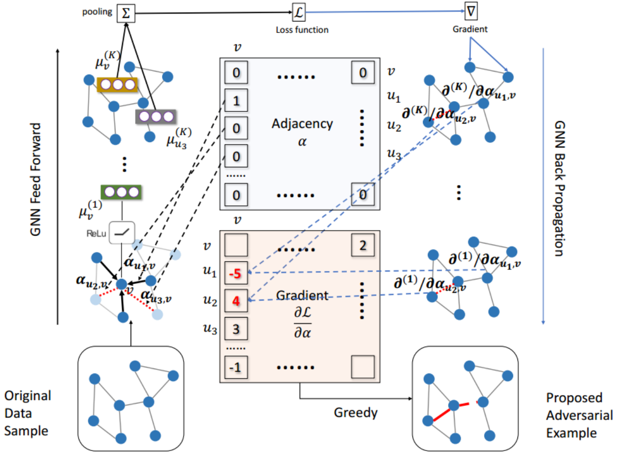

Figure 2. Illustration of graph structure gradient attack. This white-box attack adds/deletes the edges with maximum gradient (with respect to α) magnitudes.

Attack

Genetic Algorithms

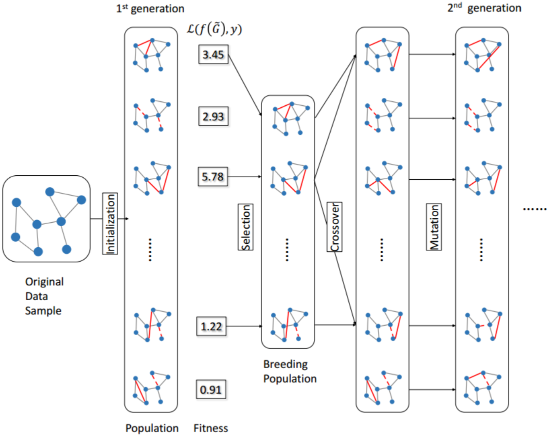

Figure 3. Illustration of attack using genetic algorithm. The population evolves with selection, crossover and mutation operations. Fitness is measured by the loss function.

- Two tasks
  - Graph Level Attack
    - Create 15K graphs using Erdos-Renyi graph model
    - Predict number of connected components (1,2,3)
  - Node Level Attack
    - Citation networks, pubmed, finance

Table 3. Statistics of the graphs used for node classification.

Dataset	Nodes	Edges	Classes	Train/Test I/Test II
Citeseer	3,327	4,732	6	120/1,000/500
Cora	2,708	5,429	7	140/1,000/500
Pubmed	19,717	44,338	3	60/1,000/500
Finance	2,382,980	8,101,757	2	317,041/812/800

Figure: Datasets used

- Multiple attack modes
  - White Box Attack (WBA)
  - Practical Black Box Attack (PBA)
    - Only label is available PBA-D
    - Confidence score is available PBA-C
  - Restrict BA (RBA)

Evaluation and Results

Results

attack test set I		15-20 nodes			
Settings	Methods	$K = 2$	$K = 3$	$K = 4$	$K = 5$
	(unattacked)	93.20%	98.20%	98.87%	99.07%
RBA	*RandSampling*	78.73%	92.27%	95.13%	97.67%
WBA	*GradArgmax*	69.47%	64.60%	95.80%	97.67%
PBA-C	*GeneticAlg*	39.87%	39.07%	65.33%	85.87%
PBA-D	*RL-S2V*	42.93%	41.93%	70.20%	91.27%

Figure: Results for attacks on **graph** classification

Method	Citeseer	Cora	Pubmed	Finance
(unattacked)	71.60%	81.00%	79.90%	88.67%
RBA, *RandSampling*	67.60%	78.50%	79.00%	87.44%
WBA, *GradArgmax*	63.00%	71.30%	72.4%	86.33%
PBA-C, *GeneticAlg*	63.70%	71.20%	72.30%	85.96%
PBA-D, *RL-S2V*	62.70%	71.20%	72.80%	85.43%
Exhaust	62.50%	70.70%	71.80%	85.22%

Figure: Results for attacks on **node** classification

Evaluation and Results

Attacks Visualization

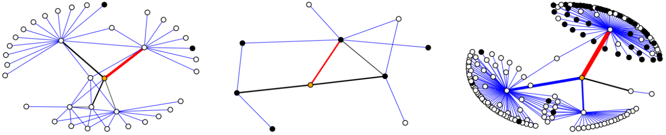

Figure 6. Attack solutions proposed by *GradArgmax* on node classification problem. Attacked node is colored orange. Nodes from the same class as the attacked node are marked black, otherwise white. Target classifier is GCN with $K = 2$.

Figure: Attacks proposed by gradient based method

Table 5. Results after adversarial training by random edge drop.

Method	Citeseer	Cora	Pubmed	Finance
(unattacked)	71.30%	81.70%	79.50%	88.55%
RBA, *RandSampling*	67.70%	79.20%	78.20%	87.44%
WBA, *GradArgmax*	63.90%	72.50%	72.40%	87.32%
PBA-C, *GeneticAlg*	64.60%	72.60%	72.50%	86.45%
PBA-D, *RL-S2V*	63.90%	72.80%	72.90%	85.80%

Figure: Results after adversarial training

Takeaways

- Genetic algorithms work well on discrete domains
- **Models trained on large real world datasets are very still hard to attack**
- Simple adversarial training methods don't help
- Structure can be enough to mount adversarial attacks (no feature modification in nodes)

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, ACM, 2014.

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in neural information processing systems*, pp. 1885–1893, 2016.