Graph Neural Networks

Naeemullah Khan

naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

February 1, 2023

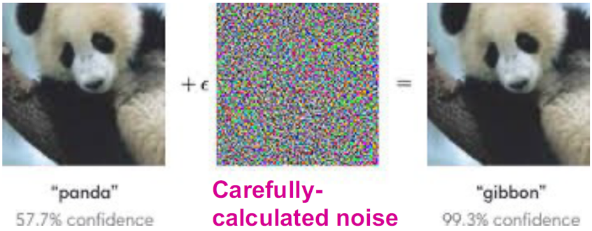
Recap

- Complex graphs can be successfully generated via sequential generation using deep learning.
- Two of the ways that we can do this by leveraging RNNs or Reinforcement Learning.
- In GraphRNN, we sequentially predict nodes and then edge connections for these nodes.
- In GCPN (RL based method), the model predicts potential links using node embeddings.

Deep Learning Performance

- Recent years have seen impressive performance of deep learning models in a variety of applications.
- For example, In computer vision, deep convolutional networks have achieved human-level performance on ImageNet (image category classification).
- But are these models ready to be deployed in real world?

- Deep convolutional neural networks are vulnerable to adversarial attacks.
- For example, Imperceptible noise changes the prediction.



Adversarial example

Adopted from Goodfellow et al. ICLR 2015

- Adversarial examples are also reported in natural language processing *[Jia Liang et al. EMNLP 2017]* and audio processing *[Carlini et al. 2018]* domains.

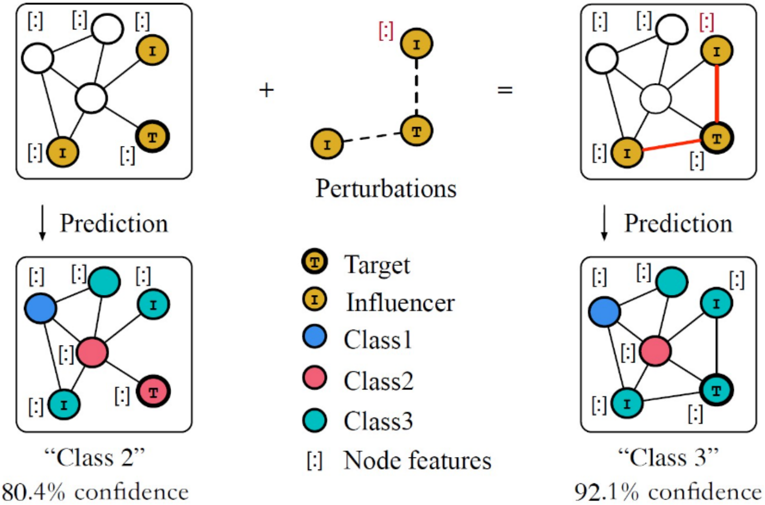
Implications

KAUST ACADEMY أكاديمية كاوست

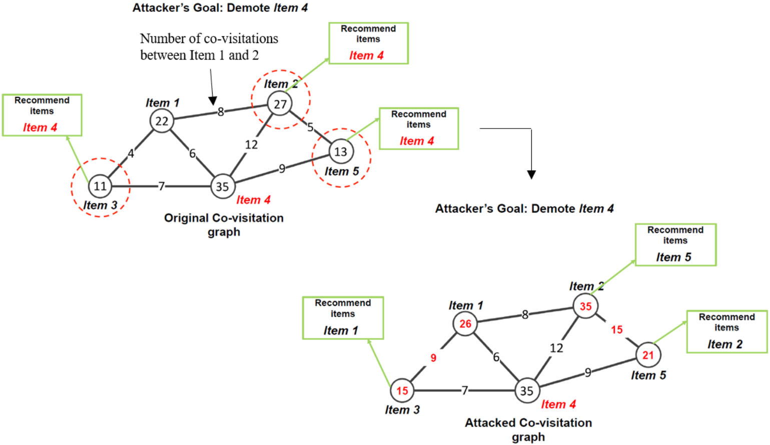
- The existence of adversarial examples prevents the reliable deployment of deep learning models to the real world.
  - Adversaries may try to actively hack the deep learning models.
  - The model performance can become much worse than we expect.
- Deep learning models are often not robust.
  - In fact, it is an active area of research to make these models robust against adversarial examples.

Graph Adversarial Learning

- Common applications of GNNs involve public platforms and monetary interests.
  - Recommender systems
  - Social networks
  - Search engines
- Adversaries have the incentive to manipulate input graphs and hack GNNs' predictions.



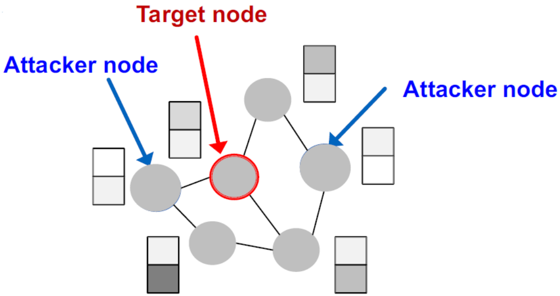
[:]
I
T
+
Perturbations
=
Prediction
Prediction
Target
Influencer
Class1
Class2
Class3
Node features
"Class 2"
80.4% confidence
"Class 3"
92.1% confidence



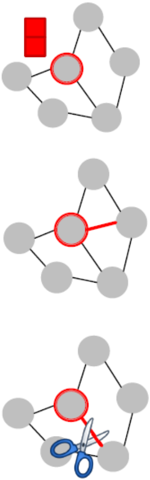
Attacker's Goal: Demote Item 4
Number of co-visitations between Item 1 and 2
Recommend items Item 4
Recommend items Item 4
Recommend items Item 4
Item 1
Item 2
Item 3
Item 4
Item 5
22
27
13
11
35
8
5
4
6
12
9
7
Original Co-visitation graph
Attacker's Goal: Demote Item 4
Recommend items Item 5
Recommend items Item 1
Recommend items Item 2
Item 1
Item 2
Item 3
Item 4
Item 5
26
35
21
15
35
8
15
9
6
12
9
7
Attacked Co-visitation graph

Attack Possibilities

- What are the attack possibilities in real world?
  - Target node $t \in V$: node whose label prediction we want to change.
  - Attacker nodes $S \subset V$: nodes the attacker can modify.

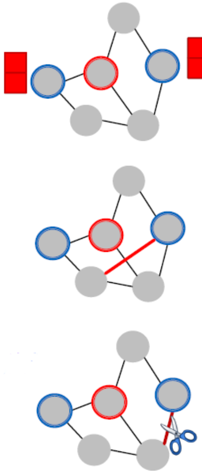


- **Direct Attack:** Attacker node is the target node: $S = \{t\}$
- Modify target node feature.
  - For example, change website content.
- Add connections to target.
  - For example, buy likes/followers.
- Remove connections from target.
  - For example, unfollow users.

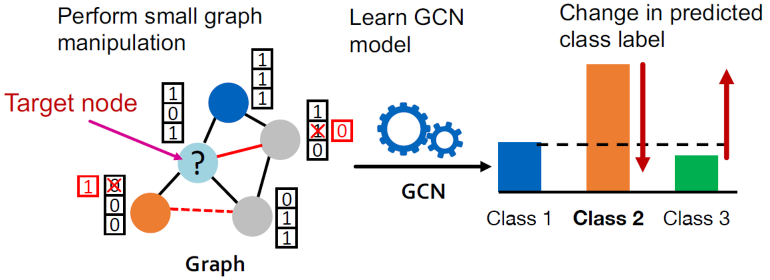


Indirect Attack

- **Indirect Attack:** The target node is not in the attacker nodes: $t \notin S$
- Modify attacker node features.
  - For example, hijack friends of targets.
- Add connections to attackers.
  - For example, create a link, link farm.
- Remove connections from attackers.
  - For example, delete undesirable link.



- **Objective for the attacker:** Maximize (change of target node label prediction) Subject to (graph manipulation is small).
- If graph manipulation is too large, it will easily be detected. Successful attacks should change the target prediction with "unnoticeably-small" graph manipulation.



- Original graph:
  - A: adjacency matrix, X: feature matrix
- Manipulated graph (aXer adding noise):
  - A': adjacency matrix, X': feature matrix
- Assumption: $(A, X) \approx (A', X')$
  - Graph manipulation is unnoticeably small i.e., preserves basic graph statistics (e.g,. degree distribution) and feature statistics.
  - Graph manipulation is either direct (changing the feature/connection of target nodes) or indirect.

Mathematical Formulation (cont.)

Overview of the attack framework

- Original adjacency matrix A, node features X, node labels Y.
- θ^*: Model parameter learned over A, X, Y.
  - c_v^*: class label of node v predicted by GCN with θ^*.
- An attacker has access to A, X, Y and the learning algorithm.
- The attacker modifies (A, X) into (A', X').
- $\theta^{*'}$: Model parameter learned over A', X', Y'.
  - $c_v^{*'}$: class label of node v predicted by GCN with $\theta^{*'}$.
- The goal of the attacker is to make $c_v^* \neq c_v^{*'}$.

Mathematical Formulation (cont.)

- Target Node: $v \in V$.
- GCN learned over the original graph

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{train}(\theta; A, X)$$

- GCN's original prediction on the target node:

$$c_v^* = \arg\max_{c} f_{\theta^*}(A, X)_{v,c}$$

Mathematical Formulation (cont.)

- GCN learned over the manipulated graph

$$\theta^{*'} = \arg\min_{\theta} \mathcal{L}_{train}(\theta; A', X')$$

- GCN's prediction on the target node:

$$c_v^{*'} = \arg\max_{c} f_{\theta^{*'}}(A', X')_{v,c}$$

- We want the prediction to change after the graph is manipulated:

$$c_v^* \neq c_v^{*'}$$

Change of prediction on target node v:

$$\Delta(v; A', X') =$$

$$\log f_{\theta^{*\prime}}(A', X')_{v, c_v^{*\prime}} - \log f_{\theta^{*\prime}}(A', X')_{v, c_v^{*}}$$

Predicted (log) probability of the newly-predicted class $c_v^{*\prime}$

Predicted (log) probability of the originally-predicted class c_v^{*}

Want to increase this term

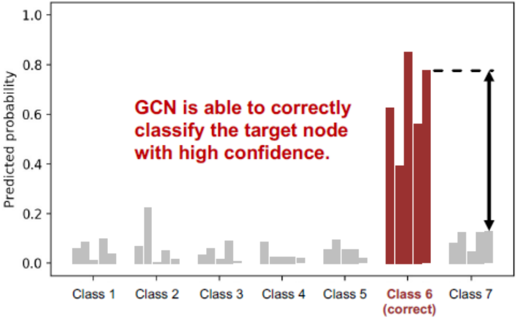
Want to decrease this term

Experiment

- **Setting:** Semi-supervised node classification with GCN.
- **Graph:** Paper citation network (2,800 nodes, 8,000 edges).
- **Attack type:** Edge modification (addition or deletion of edges).
- **Attack budget on node v:** $d_v + 2$ modifications (d_v: degree of node v).
  - Intuition: It is harder to attack a node with a larger degree.
- Model is trained and attacked 5 times using different random seeds.

Experiment (cont.)

Predicted probabilities of a target node v over 5 re-trainings (each bar represents a single trial)
(without graph manipulation, i.e., clean graph)

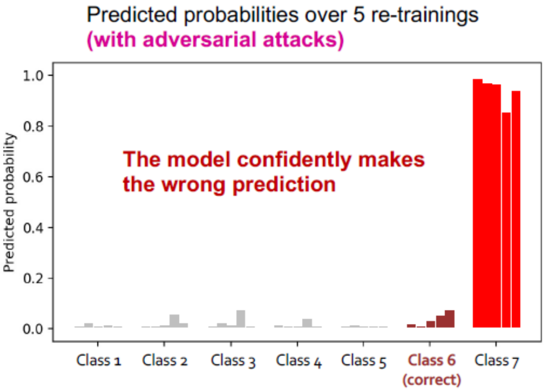


Classification margin
> 0: Correct classification
< 0: Incorrect classification

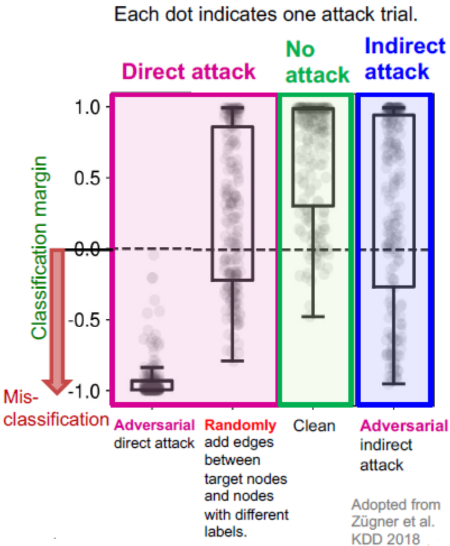
7-class classification

Experiment (cont.)

GCN's prediction after modifying 5 edges attached to the target node (direct adversarial attack).



- **Adversarial direct attack** is the strongest attack, significantly worsening GCN's performance (compared to no attack).
- **Random attack** is much weaker than adversarial attack.
- **Indirect attack** is more challenging than direct attack.



Adopted from Zügner et al. KDD 2018

Adversarial Defense on Graph

The defense methods can be categorized into three main categories.

- **Adversarial Training:** The model is made more resilient to adversarial attacks by training it on perturbed graphs as well as clean graphs during training.
- **Adversarial Detection:** These methods assume that the data has already been polluted and employ preprocessing methods to detect and remove/reduce the effect of attacks.
- **Robust Optimization:** These methods emply more robust loss functions to improve the robustness of models.

Summary

- Machine Learning models can be very senstive to very perturbations and can be tricked.
- GCN's prediction performance can be significantly harmed by adversarial attacks.
- GCN is not robust to adversarial attacks but it is somewhat robust to indirect attacks and random noise.
- Several defense methods exist which are used to improve the robustness of models.

References

These slides have been adapted from

- Jure Leskovec, Stanford CS224W: Machine Learning with Graphs
- Stephan Gunnemann, Graph Neural Networks: Adversarial Robustness
- Jintang Li,Spectra: Adversarial Learning on Graph Introduction