Scribe Note: Adversarial Attack on Graph Structured Data

Presenter: Faizan Ahmad, Scribe: Ji Gao

2019/3/15

1 Task

Attack graph models.

- Model types:
  - Inductive Graph Classification (Graph classification): For example, drug molecular classification.
  - Transductive Node Classification (Node classification): Classifying papers in a citation database.
- Means:
  - Add or Delete edges
  - Delete nodes via delete edges
- Restriction on the modification:
  - Explicit semantics(Oracle) will tell whether the modification is valid. Used in sythetic data.

$$\mathcal{I}(G, \tilde{G}, c) = \mathbb{I}(f^*(G, c) = f^*(\tilde{G}, c)), \tag{1}$$

  - Small modification. Number of edges is limited to N, and within a neighborhood graph where the distance between nodes are less than b.

$$\mathcal{I}(G, \tilde{G}, c) = \mathbb{I}(|(E - \tilde{E}) \cup (\tilde{E} - E)| < m) \cdot \mathbb{I}(\tilde{E} \subseteq \mathcal{N}(G, b))).$$

2 RL attack Overview

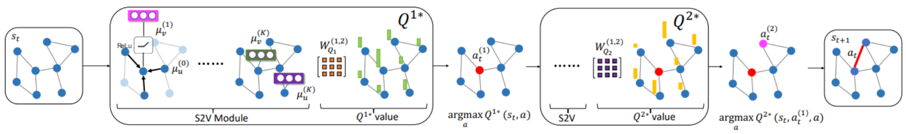


Figure 1. Illustration of applying hierarchical Q-function to propose adversarial attack solutions. Here adding a single edge a_t is decomposed into two decision steps $a_t^{(1)}$ and $a_t^{(2)}$, with two Q-functions Q^{1*} and Q^{2*}, respectively.

3 Algorithm

3.1 Black-box Hierarchical Reinforcement Learning

- **In one sentence:** Use Deep Q Learning to generate edge modification
- **RL setting:**
  - **State:** State is represented by $(\hat{G}_t, c)$, where $\hat{G}_t$is a partially modified graph and c is the original label.
  - **Action:** Action is add or remove an edge. Because the space is too large, the authors use hierarchical structure to model actions: Pick first node, and pick the second node, with two different Q function.
  - **Terminal State:** When m edge is modified, the process stops.
  - **Reward:** Non-zero reward is only received at the terminal state.
- **Value iteration:**

$$\begin{aligned} Q^{1*}(s_t, a_t^{(1)}) &= \max\nolimits_{a_t^{(2)}} Q^{2*}(s_t, a_t^{(1)}, a_t^{(2)}) \\ Q^{2*}(s_t, a_t^{(1)}, a_t^{(2)}) &= r\big(s_t, a_t = (a_t^{(1)}, a_t^{(2)})\big) + \\ &\quad \max\nolimits_{a_{t+1}^{(1)}} Q^{1*}(s_t, a_{t+1}^{(1)}). \end{aligned} \tag{2}$$

- Parameterization of Q: The authors find that time and previous modification is not useful in Q, therefore:
  - For Q_1, use a structure2vec weight μ_1

$$Q^{1*}(s_t, a_t^{(1)}) = W_{Q_1}^{(1)} \sigma\big(W_{Q_1}^{(2)\top}[\mu_{a_t^{(1)}}, \mu(s_t)]\big), \tag{3}$$

  - For Q_2, use with an extra consideration of the chosen node a

$$Q^{2*}(s_t, a_t^{(1)}, a_t^{(2)}) = W_{Q_2}^{(1)} \sigma\big(W_{Q_2}^{(2)\top}[\mu_{a_t^{(1)}}, \mu_{a_t^{(2)}}, \mu(s_t)]\big) \tag{4}$$

3.2 Other attacks

- **Random Sampling**: the simplest baseline.
- **Gradient based attack (white-box attack)**:

  Sort the gradient of all edge, greedily pick the edge with largest gradient.

$$\hat{G}_{t+1} = \begin{cases} (\hat{V}_t, \hat{E}_t \setminus (u_t, v_t)) : \frac{\partial \mathcal{L}}{\partial \alpha_{u_t, v_t}} < 0 \\ (\hat{V}_t, \hat{E}_t \cup \{(u_t, v_t)\}) : \frac{\partial \mathcal{L}}{\partial \alpha_{u_t, v_t}} > 0 \end{cases} \tag{5}$$

- **Genetic programming:**
  - **Fitness function:** $\mathcal{L}(f(\hat{G}_j^{(r)}, c), y)$, similar to reward.
  - **Selection**: Do a weighted sampling/greedy selection to select the 'breeding' population $\mathcal{P}_b^{(r)}$.
  - **Crossover**: After selection, randomly pick two candidates and mixing the edges together:

$$\hat{G}' = (V, (\hat{E}_1 \cap \hat{E}_2) \cup \mathrm{rp}(\hat{E}_1 \setminus \hat{E}_2) \cup \mathrm{rp}(\hat{E}_2 \setminus \hat{E}_1)). \tag{6}$$

  - **Mutation**: Pick a solution (u_t, v_t), have a certain probability to change it to either (u_t, v') or (u', v_t).

4 Experiment

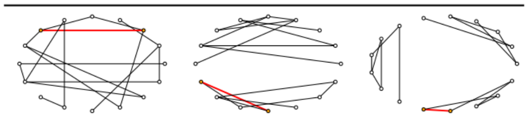
- Tasks:
  - **Graph Level Attack:**
    * Synthesize 15K graphs using Erdos-Renyi graph model
    * Predict number of connected components (1,2,3)
  - **Node Level Attack** Core, Citeseer, Pubmed and Finance dataset.
- Target: Structure2vec model
- Result on graph classification attack: Genetic programming give the best result in comparison.

Table 2. Attack graph classification algorithm. We report the 3-class classification accuracy of target model on the vanilla test set I and II, as well as adversarial samples generated. The upper half of the table reports the attack results on test set I, with different levels of access to the information of target classifier. The lower half reports the results of RBA setting on test set II where only *RandSampling* and *RL-S2V* can be used. K is the number of propagation steps used in GNN family models (see Eq (3)).

attack test set I		15-20 nodes				40-50 nodes				90-100 nodes			
Settings	Methods	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$
	(unattacked)	93.20%	98.20%	98.87%	99.07%	92.60%	96.20%	97.53%	97.93%	94.60%	97.47%	98.73%	98.20%
RBA	*RandSampling*	78.73%	92.27%	95.13%	97.67%	73.60%	78.60%	82.80%	85.73%	74.47%	74.13%	80.93%	82.80%
WBA	*GradArgmax*	69.47%	64.60%	95.80%	97.67%	73.93%	64.80%	70.53%	75.47%	72.00%	66.20%	67.80%	68.07%
PBA-C	*GeneticAlg*	39.87%	39.07%	65.33%	85.87%	59.53%	55.67%	53.70%	42.48%	65.47%	63.20%	61.60%	61.13%
PBA-D	*RL-S2V*	42.93%	41.93%	70.20%	91.27%	61.00%	59.20%	58.73%	49.47%	66.07%	64.07%	64.47%	64.67%
Restricted black-box attack on test set II													
	(unattacked)	94.67%	97.33%	98.67%	97.33%	94.67%	97.33%	98.67%	98.67%	96.67%	98.00%	99.33%	98.00%
RBA	*RandSampling*	78.00%	91.33%	94.00%	98.67%	75.33%	84.00%	86.00%	87.33%	69.33%	73.33%	76.00%	80.00%
RBA	*RL-S2V*	44.00%	40.00%	67.33%	92.00%	58.67%	60.00%	58.00%	44.67%	62.67%	62.00%	62.67%	61.33%

- Result on node classification attack: RL gets a result that closes to the exhuast result.

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%
Restricted black-box attack on test set II				
(unattacked)	72.60%	80.20%	80.40%	91.88%
RandSampling	68.00%	78.40%	79.00%	90.75%
RL-S2V	66.00%	75.00%	74.00%	89.10%
Exhaust	62.60%	70.80%	71.00%	88.88%



(a) pred = 2 (b) pred = 1 (c) pred = 2