Professorship of Data Mining and Analytics
Department of Informatics
Technical University of Munich

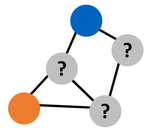
Adversarial Attacks on Graph Neural Networks via Meta Learning

Daniel Zügner, Stephan Günnemann

Summary

- ❑ Graph neural networks (GNNs) can be successfully poisoned by only small changes to the graph.
- ❑ We use techniques from meta-learning for adversarial attacks on graph neural networks.
- ❑ Our attacks can lead to graph neural networks performing worse in node classification than a linear classifier that treats all samples independently.

Semi-supervised node classification



- ❑ Given an (attributed) graph and a small number of labeled nodes, predict the labels of the remaining unlabeled nodes.
- ❑ Deep neural networks excel at this task. But are they also robust?
- ❑ Robustness is critical for graph learning methods as there are plenty of adversaries where they are deployed (e.g., the Web).

Problem formulation

Goal: Given a limited budget of perturbations Δ, insert/delete edges so that training on the resulting graph leads to weaker classification performance.

This corresponds to solving the following bilevel problem:

$$\max_{\hat{G} \in \Phi(G)} \mathcal{L}_{test}\left(f_{\theta^*}(\hat{G})\right) \quad s.t.\ \theta^* = \arg\min_{\theta} \mathcal{L}_{train}\left(f_{\theta}(\hat{G})\right)$$

f_θ: graph neural network with parameters θ
$\mathcal{L}_{train}$, $\mathcal{L}_{test}$: loss on labeled (training) or unlabeled (test) nodes
$\Phi(G)$: set of graphs under admissible (unnoticeable) perturbations

In our paper we provide a proof of concept that our method can also modify the node attributes to attack GNNs.

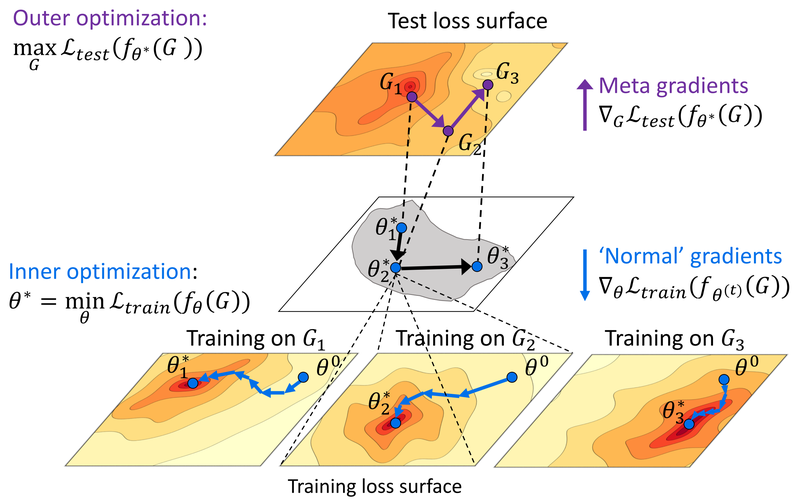
Challenges

- ❑ The number of possible edge perturbations is in $O(N^{2\Delta})$
- ❑ The data (i.e. graph structure) is discrete, which means that gradient-based optimization is not directly applicable.
- ❑ Solving the bilevel optimization problem above involves training a graph neural network in the inner problem.

Idea

- ❑ Perform poisoning adversarial attacks on graph neural networks by backpropagating through the training phase to obtain meta-gradients.
- ❑ Approximate the classification loss on the unlabeled nodes $\mathcal{L}_{test}$ via self-training.
- ❑ Relax the discreteness of the graph structure in order to obtain (meta-)gradients but perform discrete perturbations (actions).

Schematic view



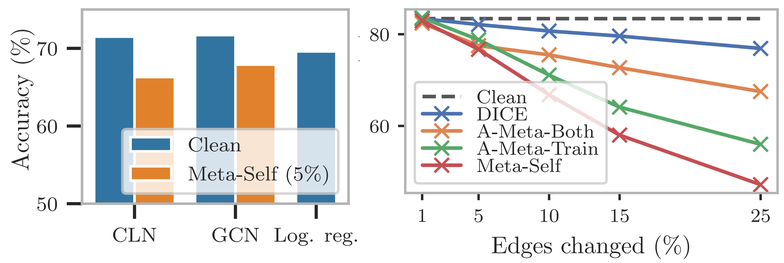
Algorithm

- ❑ Train a two-layer graph convolutional network with linear activation function for T steps using stochastic gradient descent with momentum.
- ❑ Predict the class labels of the unlabeled nodes and consider these as the true labels, i.e. $\mathcal{L}_{self} \approx \mathcal{L}_{test}$ (*self-training*).
- ❑ Unroll the training procedure to compute the meta-gradients:

$$\nabla_G^{meta} := \nabla_G \mathcal{L}_{self}\left(f_{\theta^T}(G)\right) \in \mathbb{R}^{N \times N}$$

- ❑ More scalable alternative: heuristic meta-gradients. Idea: average the inner gradients of the loss w.r.t. the graph structure during training.
- ❑ Select the edge to insert/delete with the largest meta-gradient while enforcing unnoticeability constraints $\Phi(G)$ on the degree distribution.

Results



Left: GNNs perform worse on Citeseer after 5% edge changes than logistic regression ignoring the graph altogether.

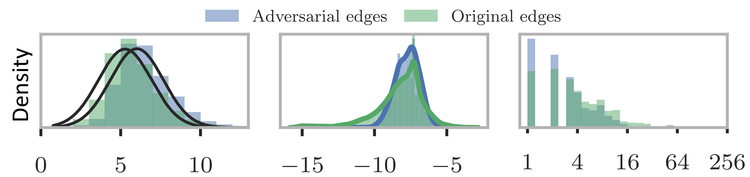
Right: Our meta-gradient-based attacks lead to the strongest decrease in classification performance (dataset: Cora-ML).

	Cora			Citeseer			Avg.
Attack	**GCN**	**CLN**	**DeepWalk**	**GCN**	**CLN**	**DeepWalk**	**rank**
Clean	16.6 ± 0.3	17.3 ± 0.3	20.3 ± 1.0	28.5 ± 0.9	28.3 ± 0.9	34.8 ± 1.4	5.2
DICE	18.0 ± 0.4	18.0 ± 0.2	22.8 ± 0.3	28.9 ± 0.3	29.1 ± 0.3	39.1 ± 0.4	4.0
Nettack*	-	-	-	31.9 ± 0.3	30.2 ± 0.4	**41.2 ± 0.4**	-
Meta-heuristic	21.8 ± 0.9	**20.5 ± 0.3**	25.0 ± 0.6	31.9 ± 0.7	30.1 ± 0.5	32.7 ± 0.5	3.4
Meta-gradient	**24.5 ± 1.0**	20.3 ± 0.4	**28.1 ± 0.6**	**34.6 ± 0.7**	**32.2 ± 0.6**	34.6 ± 0.7	**2.2**
Meta w/ Oracle	21.0 ± 0.5	21.6 ± 0.3	27.8 ± 0.7	34.2 ± 0.9	32.9 ± 0.6	36.1 ± 0.7	2.0

* Did not finish within three days on Cora-ML and PolBlogs

Table: Misclassification rate after training on graphs with 5% modified edges. Our attacks even transfer to unsupervised embeddings.

Analysis of attacks



Adversarially inserted edges cannot be distinguished from normal edges based on typical properties

Classification accuracy

Predict on \ **Trained** on	Original graph	Perturbed graph
Original graph	0.85	0.52
Perturbed graph	0.83	0.49

Once trained on the poisoned data, GNNs also fail on the clean data!

github.com/danielzuegner/gnn-meta-attack