Self-supervision on Dynamic Graphs

Yusong Ye[004757800]]

April 2022

1 Problem and Goal

1.1 Background and Problem

Graph Neural Network becomes extremely popular recently. It works well in handling graph related data by learning the node representations. However, in real-world scenario, graph structure may not be fully available. Therefore, there are some research trying to figure out a way to learn the latent graph structure. Traditional ways to do this including using similarity graph and link nodes with similar attributes, leveraging domain knowledge to infer the graph structure. Recent literature focus on using graph neural networks to do the latent graph structure learning using trainable graph generator[3].

We want to consider a case where not only the graph structure is unavailable but also the graph structure is changing according to time. In the case, the learning of the latent graph structure becomes more difficult but valuable. In the real-world setting, most social, citation networks are dynamic.

1.2 Goal

Examine the applicability of GNNs and GNN based graph generator to node classification/regression problems where the graph structure is dynamic and not readily available.

2 Data Plan

The first step is to evaluate the model based on synthetic dataset. We can use this opinion migration generator provided by Zijie [2](https://github.com/ZijieH/CG-ODE/tree/main/data/social). Our Training process need access to part of graph structure, node label, and node feature. Any real-world dynamic dataset with these three elements can be applied to our model.

3 Solution

3.1 SLAPS

SLAPS [1] (Simultaneous Learning of Adjacency and GNN Parameters with Self-supervision) proposed a supplement task to the training of graph generator with a self-supervised objective to increase the amount of supervision in learning a structure. The goal of SLAPS is to learn latent graph representation so that the performance on downstream tasks like node classification can be improved. It is composed of four main components: generator,adjacency processor,classifier, and self-supervision.

3.1.1 Generator

The Generator is a function $\mathrm{G} : \mathbb{R}^{n \times f} \rightarrow \mathbb{R}^{n \times n}$ with parameters θ_G which takes the node features $\boldsymbol{X} \in \mathbb{R}^{n \times f}$ as input and produces a matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ as output. There are mainly three kinds of generators:

- Full Parameterization(FP): $\tilde{\boldsymbol{A}} = \mathrm{G}_{FP}\left(\boldsymbol{X}; \boldsymbol{\theta}_{\mathrm{G}}\right) = \boldsymbol{\theta}_{\mathrm{G}}$, which means the generator ignores the input node features and produces a matrix $\tilde{\boldsymbol{A}}$ as output.
- MLP-kNN: $\tilde{\boldsymbol{A}} = \mathrm{G}_{MLP}\left(\boldsymbol{X}; \boldsymbol{\theta}_{\mathrm{G}}\right) = kNN\left(MLP\left(\boldsymbol{X}\right)\right)$. MLP : $\mathbb{R}^{n \times f} \rightarrow \mathbb{R}^{n \times f'}$ is an MLP $\mathbb{R}^{n \times f} \rightarrow \mathbb{R}^{n \times f'}$ that produces a matrix with updated node representations $\boldsymbol{X}'$ and kNN $\mathbb{R}^{n \times f'} \rightarrow \mathbb{R}^{n \times n}$ produces a sparse matrix where each node has up to k neighbors. Final output can be written as $\widetilde{\boldsymbol{A}}_{ij} = \phi\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)$, where $\boldsymbol{z}_i$ represents the i-th row of the output of MLP. Here the metric function ϕ is simply the cosine similarity.

3.1.2 Adjacency Processor

The Adjacency Processor is a function $\mathrm{G} : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$ which forces the adjacency matrix to be positive, symmetric and normalized. We use $\boldsymbol{A} = \frac{1}{2}\boldsymbol{D}^{-\frac{1}{2}}\left(\mathrm{P}(\tilde{\boldsymbol{A}}) + \mathrm{P}(\tilde{\boldsymbol{A}})^T\right)\boldsymbol{D}^{-\frac{1}{2}}$, where P is a function with a non-negative range applied element-wise on its input. D is the degree matrix of $\frac{1}{2}\left(\mathrm{P}(\tilde{\boldsymbol{A}}) + \mathrm{P}(\tilde{\boldsymbol{A}})^T\right)$ and guarantee the normalization.

3.1.3 classifier

The classifier is a function $\mathrm{GNN_c} : \mathbb{R}^{n \times f} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times |C|}$. C represents classes and |C| represents the number of classes. Its parameter $\theta_{\mathrm{GNN_c}} = \{\boldsymbol{W}^1, \boldsymbol{W}^2\}$ generated by a two layers GNC(Graph Convolutional Network). The function is defined as $\mathrm{GNN}_c(\boldsymbol{A}, \boldsymbol{X}; \theta_{\mathrm{GNN_c}}) = \boldsymbol{A}\mathbb{RLU}(\boldsymbol{AXW^{(1)}})\boldsymbol{W}^2$. The training loss L_C for the classification task is computed by take the softmax of the logitis to produce a probability distribution for each node and then computing the cross-entropy loss.

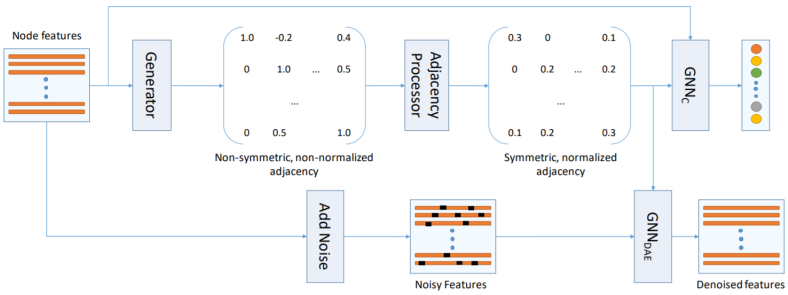

Figure 1: Overview of SLAPS. At the top, a generator receives the node features and produces a non-symmetric, non-normalized adjacency having (possibly) both positive and negative values (Section 4.1). The adjacency processor makes the values positive, symmetrizes and normalizes the adjacency (Section 4.2). The resulting adjacency and the node features go into $\mathsf{GNN_C}$ which predicts the node classes (Section 4.3). At the bottom, some noise is added to the node features. The resulting noisy features and the generated adjacency go into $\mathsf{GNN_{DAE}}$ which then denoises the features (Section 4.5).

Figure 1: Overview of SLAP [1]

3.1.4 self-supervision

Self-supervised learning (SSL) is a method of machine learning that learns from unlabeled sample data by artificially initializing network weights. The self-supervised task of SLAPs work is based on denoising autoencoder. It adds noisy feature to node feature, simultaneously train two GNN model with and without noisy feature and compare the results.

3.2 Our Approach

3.2.1 Optimize SLAPS

We want to continue examining the applicability of GNNs to classification problems where a graph structure is not readily available. The previous framework proposed by SLAPS is very interesting but We find the following drawbacks of this approach:

- Lack of interpret-ability: Different features have different meanings, we cannot interpret the meaning behind the random masking.
- Easy to attack: If we add perturbation to the feature matrix, this denoising method is very vulnerable.
- Limitation on large-scale dataset: For small matrix, it is easy to generate random mask, for large matrix like recommender system dataset, generate large mask each epoch is time-consuming.

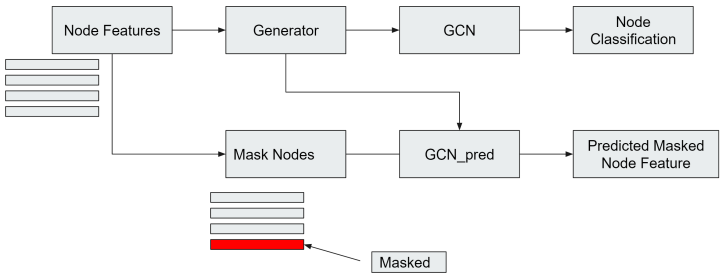

Figure 2: Framework of Node-SLAPS

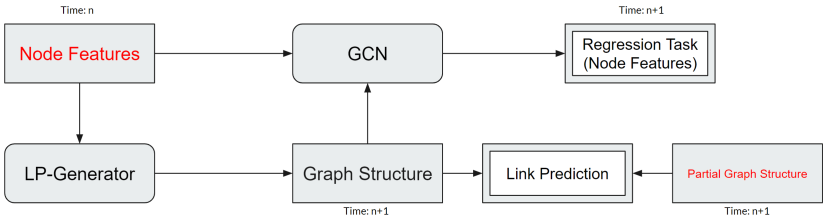

Figure 3: Framework of our proposed model

Therefore, we propose a new models: Node-SLAPS to alleviate the above problems.

We first propose Node-SLAPS. Instead of adding random noise, Node-SLAPS mask out entire node features and use a GCN to recover the masked node features. We can easily interpret this process: at the training stage, we mask out the nodes that we think are more important and use a auxiliary GCN to learn these features. We have access to all nodes' feature and use a generator to generate structure information, use this generated structure we train a GCN to do node classification and another GCN to predict masked features at the same time.

3.2.2 Dynamic Graph

We also want to examining the applicability of GNNs on classification/regression problems on dynamic dataset. Given a opinion migration dataset, the node representing the users and node feature is the user's location. Our goal is to predict the user's location at time N+1 given the user's location from time N. Suppose that only very few positive and negative edges are available at different time span is available, the framework is the following.

For the graph generation part, we want to use a GNN based model to do the graph generation instead the traditional way(knn-base etc).

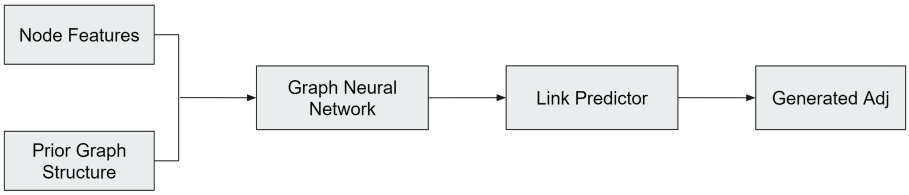

Figure 4: Link Prediction Generator

4 Evaluation Plan

We evaluation our model based the performance on downstream tasks like node classification and regression. We can compare the result with the classic GNN mdoels on dynamic graphs. For the mutant of SLAPS, we compare the performance with SLAPS.

References

[1] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. SLAPS: self-supervision improves structure learning for graph neural networks. *CoRR*, abs/2102.05034, 2021.

[2] Zijie Huang, Yizhou Sun, and Wei Wang. Coupled graph ODE for learning interacting system dynamics. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 705–715. ACM, 2021.

[3] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *CoRR*, abs/2103.03036, 2021.