[Openreview 2020] SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks [paper]

Node/Graph Tasks: node classification

Training Type: joint training

Pretext task data: node features
The pretext task here is to optimize the generated adjacency matrix by denoising node features through a GNN-based encoder. The generated adjacency matrix is used for downstream classification task.

Initial short summary here
The need for a clean graph structure impedes the applicability of GNNs to domains where one has access to a noisy structure. This paper address this limitation by developing a model that learns both the GNN parameters as well as an adjacency matrix simultaneously by supplement the classification task with a self-supervised task that helps learn a high-quality adjacency matrix. The self-supervision task masks some input features and trains a separate GNN aiming at updating the adjacency matrix such that it can recover the masked features.

Firstly, a graph generator function to generate the adjacency matrix $\tilde{A}$ is specified. The first method FP treats every entry in $\tilde{A}$ as a parameter and directly optimizes its n^2 parameters. The second method MLP-kNN considers a mapping function $\mathrm{kNN}(\mathrm{MLP}(X))$, where MLP updates the original node features $X \in \mathbb{R}^{n\times f}$ to $X' \in \mathbb{R}^{n\times f'}$ and kNN produces a sparse matrix by selecting top k similar nodes for each node and connect them with the current node.

Secondly, a adjacency processor is specified to normalize and symmetrize the generated adjacency matrix $\tilde{A}$ to $\tilde{A}$:

$$A = D^{-\frac{1}{2}} \frac{P(\tilde{A}) + P(\tilde{A})^{\top}}{2} D^{-\frac{1}{2}} \tag{19}$$

where in MLP-kNN method, P is an element-wise ReLU function ensuring every element in A is positive and in FP method, P is an element-wise ELU function and then add a value of 1 avoiding the gradient flow problem.

The third procedure is simply a two-layer GCN model that takes the node features X and the generated adjacency A as input and outputs the probability distribution of the predefined classes for each node.

The last procedure is to train a GNN-based encoder that takes noisy node features and the normalized adjacency matrix as input and output the updated node features with the same dimension. The de-noising procedure is realized by minimizing:

$$\mathcal{L}' = L(X_{idx}, \mathrm{GNN}(\tilde{X}, A; \theta_{\mathrm{GNN}})_{idx}) \tag{20}$$

where idx represent dimensions of node features to which we have added noise. $\tilde{X}$ is the noisy feature matrix that is generated either by randomly zeroing some dimensions of X in the case that the input features are binary or by adding independent Gaussian noises in the case that the input features are continuous numbers. The total loss $\mathcal{L} = \mathcal{L}_C + \lambda \mathcal{L}_{DAE}$ where $\mathcal{L}_C$ represent the classification loss in the third procedure and $\mathcal{L}_{DAE}$ is the denoising autoencoder loss in the last procedure. Bibtex: