Learning to Pre-train Graph Neural Networks

Yuanfu Lu\textsuperscript{1,2}, Xunqiang Jiang\textsuperscript{1}, Yuan Fang\textsuperscript{3}, Chuan Shi\textsuperscript{1}

\textsuperscript{1}Beijing University of Posts and Telecommunications
\textsuperscript{2}WeChat Search Application Department, Tencent Inc. China
\textsuperscript{3}Singapore Management University
\textsuperscript{4}Peng Cheng Laboratory, Shenzhen, China
**Background**

**GNNs**

- **node-level representation**
  
  \[
  h_v^l = \Psi(\psi; A, \mathcal{X}, Z)^l \\
  = \text{UPDATE}(h_v^{l-1}, \\
  \text{AGGREGATE}((h_v^{l-1}, h_u^{l-1}, z_{uv}) : u \in \mathcal{N}_v))
  \]

- **graph-level representation**
  
  \[
  h_g = \Omega(\omega; H^l) = \text{READOUT}(\{h_v^l | v \in \mathcal{V}\})
  \]

**Pre-train GNNs**

- \(\theta_0\) is pre-trained without accommodating the adaptation in fine-tuning

\[
\theta_0 = \arg \min_\theta \mathcal{L}^{pre}(f_\theta; \mathcal{D}^{pre})
\]

\[
\theta_1 = \theta_0 - \eta \nabla_{\theta_0} \mathcal{L}^{fine}(f_{\theta_0}; \mathcal{D}^{tr})
\]
Motivation

learn how to pre-train
Motivation

Pre-train a GNN model over a graph $\mathcal{G} \in \mathcal{D}^p$

- sample sub-structures $\mathcal{D}^f_t$ for training
  
  *(the training data of a simulated downstream task)*

- mimic the evaluation on testing sub-structures $\mathcal{D}^e_t$

$$\theta_0 = \arg \min_\theta \sum_{G \in \mathcal{D}^p} \mathcal{L}^{pre}(f_{\theta} - \alpha \nabla_{\theta} \mathcal{L}^{pre}(f_{\theta}; \mathcal{D}^{tr}_t); \mathcal{D}^{te}_t)$$

the fine-tuned parameters

*(in a similar manner as the fine-tuning step on the downstream task)*
L2P-GNN

(a) An Example of Graph

\( G = \{ \mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Z} \} \)

(b) Task Construction

\( \mathcal{T}_G = \{ \mathcal{T}_G^1, \ldots, \mathcal{T}_G^k \} \)

Child Task \( \mathcal{T}_G^c \)

Support Set \( S_G^c \)

Query Set \( Q_G^c \)

\( \theta = \{ \psi, \omega \} \)

Adaptation on support set

\( \theta' = \{ \psi', \omega' \} \)

Optimization on query set

(c) Dual Adaptation in Self-supervised Base Model

Node-level Aggregation \( \Psi(\psi; \mathcal{A}, \mathcal{X}, \mathcal{Z}) \)

Node-level loss on support / query set

Graph-level Pooling \( \Omega(\omega; \mathcal{H}) \)

Graph-level loss on support / query set

Node-level adaptation \( \psi' \leftarrow \psi \)

Graph-level adaptation \( \omega' \leftarrow \omega \)
**L2P-GNN**

Task Construction

- the pre-training data \( \mathcal{P}_e = \{G_1, G_2, \ldots, G_N\} \)
- A task involving a graph \( T_G = (S_G, Q_G) \)
- **gradient descent** w.r.t. the loss on \( S_G \)
- **optimize** the performance on \( Q_G \)
- simulating the **training and testing** in the fine-tuning step
Self-supervised Base Model

node-level aggregation

\[ \mathcal{L}^{\text{node}}(\psi; S_G^c) = \sum_{(u,v) \in S_G^c} - \ln(\sigma(h_u^\top h_v)) - \ln(\sigma(-h_u^\top h_v')) \]

graph-level pooling

\[ \mathcal{L}^{\text{graph}}(\omega; S_G) = \sum_{c=1}^{k} - \log(\sigma(h_{S_G^c}^\top h_G)) - \log(\sigma(-h_{S_G^c}^\top h_{G'})) \]

\[ \mathcal{L}_{\mathcal{T}_G}(\theta; S_G) = \mathcal{L}^{\text{graph}}(\omega; S_G) + \frac{1}{k} \sum_{c=1}^{k} \mathcal{L}^{\text{node}}(\psi; S_G^c) \]
L2P-GNN

Dual Adaptation

\[ \psi' = \psi - \alpha \frac{\partial \sum_{c=1}^{k} \mathcal{L}_{\text{node}}(\psi; S_g)}{\partial \psi} \]

\[ \omega' = \omega - \beta \frac{\partial \mathcal{L}_{\text{graph}}(\omega; S_g)}{\partial \omega} \]

\[ \theta \leftarrow \theta - \gamma \frac{\partial \sum_{g \in D_{\text{pre}}} \mathcal{L}_{T_g}(\theta'; Q_g)}{\partial \theta} \]

(c) Dual Adaptation in Self-supervised Base Model
Experiments

Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Biology</th>
<th>PreDBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>#subgraphs</td>
<td>394,925</td>
<td>1,054,309</td>
</tr>
<tr>
<td>#labels</td>
<td>40</td>
<td>6</td>
</tr>
<tr>
<td>#subgraphs for pre-training</td>
<td>306,925</td>
<td>794,862</td>
</tr>
<tr>
<td>#subgraphs for fine-tuning</td>
<td>88,000</td>
<td>299,447</td>
</tr>
</tbody>
</table>

Baselines

- EdgePred to predict the connectivity of node pairs
- DGI to maximize mutual information across the graph’s patch representations
- ContextPred to explore graph structures
- AttrMasking to learn the regularities of node/edge attributes

GNN Architectures

- GCN, GraphSAGE, GAT, GIN
### Performance Comparison

Table 2: Experimental results (mean ± std in percent) of different pre-training strategies w.r.t. various GNN architectures. The improvements are relative to the respective GNN without pre-training.

<table>
<thead>
<tr>
<th>Model</th>
<th>Biology</th>
<th>PreDBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GCN</td>
<td>GraphSAGE</td>
</tr>
<tr>
<td>No pre-train</td>
<td>63.22±1.06</td>
<td>65.72±1.23</td>
</tr>
<tr>
<td>EdgePred</td>
<td>64.72±1.06</td>
<td>67.39±1.54</td>
</tr>
<tr>
<td>DGI</td>
<td>64.33±1.14</td>
<td>66.69±0.88</td>
</tr>
<tr>
<td>ContextPred</td>
<td>64.56±1.36</td>
<td>66.31±0.94</td>
</tr>
<tr>
<td>AttrMasking</td>
<td>64.35±1.23</td>
<td>64.32±0.78</td>
</tr>
<tr>
<td>L2P-GNN (Improv.)</td>
<td><strong>66.48±1.59</strong></td>
<td><strong>69.89±1.63</strong></td>
</tr>
</tbody>
</table>

- 6.27% and 3.52% improvements compared to the best baseline
- 8.19% and 7.88% gains relative to non-pretrained models
- **negative transfer** harms the generalization of the pre-trained GNNs (e.g., EdgePred and AttrMasking strategies w.r.t. GAT)
Comparative Analysis

whether L2P-GNN narrows the gap between pre-training and fine-tuning?

- Comparation of the pre-trained GNN model before and after fine-tuning
- Centered Kernel Alignment (CKA) similarity between the parameters
  - Smaller similarity, larger changes of model parameters
- Changes in loss and performance (delta loss and RUC-AUC/Micro-F1)
  - Smaller change, more easily achieve the optimal point
Model Analysis

Ablation Study
- L2P-GNN-Node with only node-level adaptation
- L2P-GNN-Graph with only graph-level adaptation

Parameter Analysis
- the number of node- and graph-level adaptation steps \((s, t)\)
- the dimension of node representations
THANKS

2020.12.19

Codes and datasets:  https://github.com/rootlu/L2P-GNN