Exploring PyTorch Geometric: a geometric deep learning library built on top of PyTorch

Zejian Li
30/11/2022
Motivation

• Graphs provide a rich framework for many types of problems.
• What happens when graphs and deep learning are combined?
  • The AlphaFold and AlphaFold2 developed by DeepMind solved protein structure prediction for structural biology
  • drug discovery, social networks, and fake news detection, ...
• Open-source libraries for graph neural networks.
  • PyG: PyTorch Geometric (PyTorch)
  • DGL: Deep Graph Library (PyTorch, TensorFlow)
  • TF-GNNs: TensorFlow Graph Neural Networks (TensorFlow)
  • Spektral (Keras, TensorFlow)
The goal of the work

• Explore the functionality and connections between the various modules of PyTorch Geometric and delve into some of the important principles or modules such as MEMORY-EFFICIENT AGGREGATIONS.

• Use PyTorch Geometric to complete basic steps, such as building datasets, customising GNN networks, training GNN models, etc.

• Use PyTorch Geometric for two advanced applications.
  • Recommender Systems with GNNs in PyG.
  • Experiment with PyTorch Geometric in the area of model optimization and compression, e.g. Compressing Deep Graph Neural Networks via Adversarial Knowledge Distillation. [1]
Compressing Deep Graph Neural Networks

• Deep and complicated GNNs significantly outperform shallow models on large-scale graphs, implying the great expressive power of over-parameterized GNNs. (Figure 1)

• The overstacked architecture of deep graph models makes it difficult to deploy and rapidly test on mobile or embedded systems.

• Knowledge distillation via a teacher-student architecture turns out to be an effective technique.

![Graph Neural Networks Diagram](source: [1])

Figure 1: Node classification accuracy v.s. graph size. Each bubble's area is proportional to the number of parameters of a model. Model name with * means the variant. The statistics are collected from OGB leaderboards.
Compressing Deep Graph Neural Networks

• Existing work

  • Existing work force the student network to mimic the teacher network with **hand-crafted distance functions**, of which the **optimal formulation** is hard to determine [2]. Even worse, the performance of the student trained this way is always suboptimal because it is difficult to learn the exact distribution from the teacher [3, 4].

  • The predefined and fixed distance is unfit to measure the distribution discrepancy of teacher and student representations in different feature spaces.
Compressing Deep Graph Neural Networks

- GraphAKD is the first to introduce adversarial training to knowledge distillation in graph domains [1].
- By adversarially training a discriminator and a generator, GraphAKD is able to transfer both inter-node and inter-class correlations from a complicated teacher GNN to a compact student GNN [1].

Figure 2: Illustration of the proposed adversarial knowledge distillation framework GraphAKD.

source: [1]
Compressing Deep Graph Neural Networks

• Experiments:
  • Datasets: Performing node classification on some of the widely-used datasets.
  • Five research questions:
    • RQ1: How does GraphAKD perform on node-level classification?
    • RQ2: How does GraphAKD perform on graph-level classification?
    • RQ3: How efficient are the student GNNs trained by GraphAKD?
    • RQ4: How do different components affect the performance of GraphAKD?
    • RQ5: Do student GNNs learn better node representations when equipped with GraphAKD?

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Nodes</th>
<th>#Edges</th>
<th>#Feat.</th>
<th>Data Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora [3, 34]</td>
<td>2,708</td>
<td>5,429</td>
<td>1,433</td>
<td>140/500/1K</td>
</tr>
<tr>
<td>CiteSeer [40]</td>
<td>3,327</td>
<td>4,732</td>
<td>3,703</td>
<td>120/500/1K</td>
</tr>
<tr>
<td>PubMed [35]</td>
<td>19,717</td>
<td>44,338</td>
<td>500</td>
<td>60/500/1K</td>
</tr>
<tr>
<td>Flickr [33, 58]</td>
<td>89,250</td>
<td>899,756</td>
<td>500</td>
<td>44K/22K/22K</td>
</tr>
<tr>
<td>Arxiv [23]</td>
<td>169,343</td>
<td>1,166,243</td>
<td>128</td>
<td>90K/29K/48K</td>
</tr>
<tr>
<td>Yelp [58]</td>
<td>716,847</td>
<td>13,954,819</td>
<td>300</td>
<td>537K/107K/71K</td>
</tr>
<tr>
<td>Products [23]</td>
<td>2,449,029</td>
<td>61,859,140</td>
<td>100</td>
<td>196K/39K/2M</td>
</tr>
</tbody>
</table>
Current work progress

• Reading the source code of PyTorch Geometric.
• Using PyTorch Geometric to build datasets and create GNN layers.
Work plan for the next stage

• Complete basic steps including building datasets, customising GNN networks, training GNN models, etc.

• Use PyTorch Geometric for 2 more advanced applications.

• Writing the final report.
References


