Deep Learning and Applications

Homework 3

Due date: Apr. 20, 2020

1 Task Description

In class, we have learned several important graph analysis approaches, including node embedding methods (e.g., LINE, DeepWalk), graph neural networks (e.g., GCN, GAT) and graph visualization techniques (e.g., t-SNE, LargeVis).

To practice with those approaches, in this homework you are required to leverage them for some real applications, including graph embedding + graph visualization, and graph neural networks for node classification.

There are two sections in our homework:

- **Part 1: Node Embedding + Embedding Visualization.**
- **Part 2: Graph Neural Networks for Node Classification.**

You could find the details in the Colab file.

2 Instruction on Part 1

In this part, we would try node embedding method on a social network. The goal is to get familiar with the pipelines of node embedding and visualization. Specifically, you can use any existing package (e.g., GraphVite, PyTorch Big-Graph, Open NE) to implement node embedding training. We will evaluate the learned embeddings on node classification task, which tests the predictive power of embeddings. Besides, you are also expected to visualize the node embeddings in a 2D plot using existing tools (e.g., GraphVite, tSNE-CUDA, scikit-learn), and perform some qualitative analysis.

3 Instruction on Part 2

For this part, we will use the Cora dataset, which is widely used for evaluating graph neural networks. The dataset is available at the Google drive link. You could find several files there. Among those files, *net.txt* provides the edges between different nodes, and the three columns correspond to source nodes, target nodes and edge weights respectively. For *feature.txt*, it gives the features of nodes. For *label.txt*, it provides the node labels. *train.txt*, *valid.txt*, *test.txt* provide the training nodes, validation nodes and test nodes respectively. The goal is to train a model on the training nodes, and further apply the model for classifying test nodes. In this process, you may use the validation nodes for hyper-parameter tuning and early stopping.

For implementation, you may choose the Graph Convolutional Network (GCN) to implement, as it is widely used in many different applications. Below are some nice GitHub repositories you could refer to for the implementation: GCN by Kipf, GAT by Grattarola.

To better understand the model you implement, we will further do the following performance analysis:

- **Performance w.r.t. the number of layers.**
  GCN typically uses only a few layers for information propagation. Otherwise, they may suffer from the over-smoothing problem. To look into that, please draw a table to compare the performance of the GCN model with respect to the number of propagation layers, e.g., from 1 layer to 10 layers. Also, please write down your observation and the potential reason based on your thinking.

- **Performance w.r.t. the hidden dimension.**
  Another important hyperparameter for GCN is the dimension of hidden layers. In this part, you may try different dimensions for the hidden layers, e.g., 8, 16, 32, 64, 128. Also, please write down your observation and the potential reason based on your thinking.

4 Handing in Your Answer

For this homework, you could submit your answer by filling in the Google form.

5 Evaluation

For all the three parts of the homework, we will check your codes and your analysis. As long as your implementation is correct and your analysis makes sense, you will get all the points.