**Cora: 2 different splits**

Wenqian Cui, 2021/08/04

**Overview**

This is the report for running the 4 data augmentation methods on GCN and GAT using Cora’s 2 different split. First of all, the process of running the experiments encountered some unexpected issues, and it caused some delay. I’ll talk about the details in the later chapters.

**About the splits**

140/500/1000: This is the original split of Cora. 0~139 forms the training set, 140~639 forms the validation set and 1708~2707 forms the test set. (Note: 640~1707 nodes are never used)

1208/500/1000: According to the online material, they just expand the training set without changing the distribution of the validation/test set. Therefore, I just merge the 640~1707 nodes into the training set.

**Experiments & Results**

The experiment settings are the same as before, and the visualization of the results are included in the compressed file.

(Note: some experiments took the average of the 50 trials while others took the average of just 10 trials. The reason is that running the experiments on the larger training set cost too much time. Therefore, you can first take a look at the trend of the curve to decide our further experiments that need to be performed.)

**Unexpected issues**

1. Dropedge experiments take too much time to finish.

When performing dropedge experiments on GAT using the 1208-nodes larger dataset, it runs too slow. The reason is that dropedge randomly drops edges on every epoch, and that will cause too many computations. This is even worse for the case that the training set is larger, since the total number of edges are increased. In my previous experiment settings, the edges dropping operation for every epoch is pre-computed. Therefore, if I expect that the algorithm will run about 500 epochs, then I have to calculated 500 times on CPU in advance. For a single experiment, it may take hours to complete the process, and the total amount of experiments are about 30.

I think maybe there is a way to do all the computation during the running of the epochs and use the GPU resources to speed up the computations. I’ll try to find a way to implement that.

1. GAboL (hop=7) on GAT on larger training set cause the CUDA memory leak.

For this experiment, I ran it for 10 times and took the average. At first, I noticed that after running 1 time, CUDA memory leak error happened. I once thought that maybe it’s the redefinition of the model or the training data (maybe) cause the memory leak. So I checked the references of all the CUDA device variable to make sure there’s no unused variable. Then, I empty the CUDA memory cache after each time of running. As a result, it didn’t make much of a difference. The program fails when performing the 7th experiment.

Since this is not happening on smaller training set or the number of hops is smaller, I think maybe the problem is caused by the large number of the edges. For GAboL, it will add more edges when the number of hops increases. Therefore, I add the number of hops from 7 to 8, and it couldn’t perform a single run.

**Conclusion**: GAboL on GAT on larger training set experiments just need more CUDA memory to run (The CUDA memory on my local computer is 8Gb). Do you have any solutions to that?