E536: Graph Analytics
Homework 3
Due date: Monday, February 26. 11:59pm Eastern time
Total Points: 60

For all of the problems below you will be using the Cora data. The data is already available in PyTorch_Geometric. Take a look at the example codes shown in the class (already published on Canvas). Unless otherwise mentioned, please use the following default parameters.

- Number of hidden layers = 2
- Hidden layer dimension = 16
- Activation function = ReLU
- No dropout
- Output layer: use log_softmax
- Optimizer: Adam
- learning rate: 0.01
- Number of epochs: 100
- Training/Test split: as provided by the PyTorch Geometric (no need to change)

Q 1: Testing parameters in **GCN** [Total 40 points]

In this problem, you will experiment with different parameters as specified below and their effect on node classification on the Cora data. While testing a single parameter in GCN, set other parameters to their default values as mentioned above. For GCN, use GCNConv layers in PyTorch_Geometric.

1. parameter **hidden layer dimension:** 2, 4, 8, 16, 64, and 512. For each setting, train the GCN model for 100 epochs and compute the test accuracy.

   (a) Create a table with two rows and 6 columns (one column for each hidden layer dimension). In row 1 of the table, report test accuracies. Do the test accuracies increase with the hidden dimension? Why or why not? [5 + 3 points]

   (b) In row 2 of the same table, report the number of epochs needed to converge the model for each parameter considered. (a model is said to be converged when the training accuracy is not improving anymore. A rough estimation should be enough.). Does the model converge faster with increasing hidden dimension? Why or why not? [5 + 3 points]

2. parameter **number of layers:** 1, 2, 4, 8, and 16. For each setting, train the GCN model for 100 epochs and compute the test accuracy.

   (a) Create a table with two rows and 5 columns (one column for each layer). In row 1 of the table, report test accuracies. Do the test accuracies increase with increasing number of layers? Why or why not? [4 + 2 points]

(b) In row 2 of the same table, report the number of epochs needed to converge the model for each parameter considered. (a model is said to be converged when the training accuracy is not improving anymore). Does the model converge faster with increasing number of layers? Why or why not? [4 + 2 points]

3. parameter **learning rate:** 1, .5, .1, .01, and .0001. For each setting, train the GCN model for 100 epochs and compute the test accuracy.

   (a) Create a table with two rows and 5 columns (one column for learning rate). In row 1 of the table, report test accuracies. How do the test accuracies change with different learning rates? Explain your answer. [4 + 2 points]

   (b) In row 2 of the same table, report the number of epochs needed to converge the model for each parameter considered. How do the convergence rates change with different learning rates? Explain your answer. [4 + 2 points]

Thus, for Question 1, you notebook should generate three tables, one for each set of parameters.

Q 2: Compare GCN, GraphSAGE, and GAT [Total 20 points]

In this problem, you will compare three GNN methods with default parameters mentioned above. For GCN, use GCNConv layers in PyTorch_Geometric. For GraphSAGE, use SAGEConv and for GAT (Graph Attention Network), use GATConv. For each convolution, train the GNN model for 100 epochs and compute the test accuracy.

1. Create a table with two rows and 3 columns (one column for GNN). In row 1 of the table, report test accuracies. [10]

2. In row 2 of the same table, report the number of epochs needed to converge the model for each GNN. Which GNN converges slower than others? Why (just write your intuition)? [8 + 2 points]

Q 3: Bonus Question [Total 20 points]

Let X be an $n \times d$ matrix storing the current embedding of all vertices. You can collect this embedding from the last layer of GNN. In the first epoch, X will be input features of vertices. At each epoch of GNN training, create a k-nearest graph G_{knn} from X. You can use this code if you like: `https://github.com/rusty1s/pytorch_cluster/blob/master/torch_cluster/knn.py`. See this code to for an example on how to create a KNN graph: `https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/conv/edge_conv.html`. In this way, you have two graphs: the original graph and G_{knn}. Now, write a convolution layer called *DualConv* that will run GCN on both G_{knn} and the original graph G. This is an open ended question. Please attempt this question only if you would like to go deep into GNN models. We will not help you significantly to answer this question.

1. Write a function that will create G_{knn} from current embedding/feature matrix at the start of every epoch (note that this graph will change in different epochs). [5]

2. Write a convolution layer that will run GCN on both G_{knn} and the original graph G. Write the corresponding forward propagation functions. See the examples of the GCN layer in PyTorch Geometric. [5 points]

3. In the last layer before computing the loss function, compute the average of embeddings coming from both graphs to generate the final embedding. Compute your loss function with this average embedding. Note that this final embedding will be used to create G_{knn} in the next epoch. [5 points]

4. Next, create a GNN with two *DualConv* layers and train the network with default parameters mentioned above. Then, after 100 epochs (or after the convergence), report the test accuracy of the trained model. [5 points]

Submission Instructions

Please submit your code that can be executed. Submit all files needed to execute your notebook. Notebooks must be executable with your helper functions.

1. Please submit only one zipped folder "E536_LastName_HW3.zip" (or E536_LastName_HW3.tar.gz). This folder should contain all of your notebooks (one per question and named like HW-03-q02.ipynb).

2. In addition to the notebooks, submit your reports as a PDF file (named like HW-03-q02-report.pdf). Include the pdf files in the zipped folder. This helps in case we cannot execute your code.

3. Do not use any installation (pip-or-otherwise) code in your notebooks and no OS shell commands.

4. Do not use your google drive locations or other absolute directory locations in the notebooks.

And as usual:
You are welcome to discuss HW questions with others. I am always happy to help you. Please don't copy answers or code from the internet or from another person.

Associate Instructor: The AI responsible for this homework is Abhigya Agrawal. Please visit her office hours with questions.

Late policy: For every late day you will lose 20% of the assigned grade for this HW. That means, if you submit 5 days after the deadline, you will not get any point. Only exceptions are medical or family emergencies. In emergency, you can submit after the due date without any penalty. Please ask if you have questions.