Exercise Sheet 2

Due date: Tuesday, May 10th, 23:59

- The content of your master branch at the time of the deadline is your submission.
- You should use the following Python packages in this exercise:
  - PyTorch
  - NetworkX
  - Numpy
  - Scikit-Learn
  - argparse

Exercise 1 (GCN Pre-Processing) 3 points

Implement a method that computes the normalized adjacency matrix $\tilde{A}$ for a given graph G. Recall that $\tilde{A} \in \mathbb{R}^{|V| \times |V|}$ is defined such that

$$\tilde{A}_{i,j} = \begin{cases} \frac{1}{\sqrt{d_i d_j}} & \text{if } v_i v_j \in E \text{ or } i = j \\ 0 & \text{otherwise} \end{cases}$$

with $d_i = \deg(v_i) + 1$.

Hints:

- To combine multiple graphs into one batch you have to pad the adjacency and feature matrices with zeros to a uniform size across the whole dataset. The matrices can then be stacked along the batch dimension into 3D-tensors. The `data_utils.py` script provides the methods `get_padded_adjacency`, `get_padded_node_labels` and `get_padded_node_attributes` for this task. The stacked dataset can then be split into batches along the batch dimension. You can use the `TensorDataset` and `Dataloader` classes from PyTorch to help with batching and shuffling.

Exercise 2 (GCN Layer) 6 points

Implement the GCN Layer as a PyTorch module.

Reminder GCN: A GCN layer computes the function $H^\ell = \sigma(\tilde{A} \cdot H^{\ell-1} \cdot W^\ell)$ with the following components:

- $\tilde{A} \in \mathbb{R}^{|V| \times |V|}$ is the normalized adjacency matrix of the input graph.
- The vertex embedding $H^\ell \in \mathbb{R}^{|V| \times d^{(\ell)}}$ is the output of the ℓ-th layer. $H^0 \in \mathbb{R}^{|V| \times d^{(0)}}$ are the original node features.
- $W^\ell \in \mathbb{R}^{d^{(\ell-1)} \times d^{(\ell)}}$ is the trainable weight matrix of the ℓ-th layer. The weights are initialized randomly.
- σ is the ReLU activation function.

Hint: Remember that the first dimension of $\tilde{A}$ and H^ℓ is the batch dimension, which is omitted in the definition above. The update equation must be performed in parallel for each graph in the batch. You may find `torch.bmm` helpful to implement this correctly.

Exercise 3 (Graph-Level GCN) 6 points

Use your GCN Layer to implement a GCN for graph classification as a PyTorch module. Your neural network should have the following components:

- 5 GCN layers with ReLU activation and hidden dimension $d^{(\ell)} = 64$.
- A sum pooling layer which sums the node embeddings in H^5 element-wise for each graph.
- A classification MLP with one hidden layer of dimension 64 and a linear output layer.

Exercise 4 (Node-Level GCN) 6 points

Implement a GCN for node classification as a PyTorch module. The network should have the following components:

- 3 GCN layers with ReLU activation and dimension $d^{(\ell)} = 64$.
- A classification MLP which consists of a single a linear output layer.

Hints

- It is common to simply use the same hidden dimension $d^{(\ell)}$ in all layers to reduce hyperparameters. The dimension of the node features $d^{(0)}$ depends on the dataset and has to be passed to the constructor of the neural network. The same thing holds for the number of classes which is the output dimension of the classifier MLP.
- If you train with categorical cross-entropy in PyTorch you do **not** have to apply the Softmax function to the output of your network. This happens internally in the loss function. The final layer of a network should simply be linear with no activation function.

Exercise 5 (Evaluation Graph Classification) 7 points

Test your GCN for graph classification on the datasets NCI1 and ENZYMES. Use cross-entropy to train your network. For training, you can use an optimizer of your choice (i.e. Adam, RMSProp,...). Choose hyperparameters like the learning rate and number of epochs empirically. We do not require a systematic grid search for the optimal hyperparameters.

Perform 10-fold cross validation on each dataset. Please report the mean accuracy and standard deviation for the train and test data in your readme. Your GCN should achieve at least the following accuracy on each dataset (-5 points it does not):

- NCI1: 75%
- ENZYMES: 55%

Hints

- For NCI1, the node features are the one-hot vectors of the node labels.
- The ENZYMES dataset has additional node attributes in the form of real-valued vectors. Concatenate these vectors with the one-hot encoding of the node label to obtain the full input features. For stability, you may find it helpful to normalize the node attributes in the l2-norm before appending them to the node label.
- If you have problems with overfitting, add dropout layers in suitable places.

Exercise 6 (Evaluation Node Classification) 7 points

Test your GCN for node classification on the datasets Cora und Citeseer. Both datasets are citation networks that model the citations between scientific publications. Every node represents a publication and two nodes share an edge if one publication cites the other. (The direction of the citation is not modeled.) The node attributes are vectors that count the occurrence of certain key words in the publication. The node class that you predict is the scientific field of the publication. (The node classes are stored as 'node labels' in our datasets.)

Use the cross-entropy loss over all nodes to train the network. The provided datasets are already split into train and test data. We do not require you to perform cross validation in this exercise. Repeat the training and testing 10 times on both datasets and report the mean accuracy and standard deviation for the train and test data in your readme.

Your GCN should achieve at least the following accuracy on each dataset (-5 points it does not):

- Cora: 75%
- Citeseer: 70%

Hints

- The training and test datasets each consist of one large graph each. You should not split the training graph into smaller batches. Use the full training graph in every training step. The batch size is therefore simply 1 in this experiment.
- If you have problems with overfitting, add dropout layers in suitable places.

Exercise 7 (Code Quality, Comments and Presentation) 15 points

Clean your code and add useful comments. Your repository must contain a `README.md` file which provides the following information:

- A brief description of the structure of your repository
- How to run every executable script. For each script, all command line options must be fully specified
- The accuracy and standard deviation achieved for every dataset for both node and graph classification.

Prepare a short presentation ($\sim$5 min) for your group meeting (held in the week after the submission deadline). It should briefly provide the following information:

- What you implemented and how the work was split
- The results you obtained
- A brief interpretation and discussion of the results