EE-559 Deep learning – Practice 6, Students' questions

A question is denoted by **Q**; the corresponding answer is denoted by **A**. The questions-related exercises are marked by their numbers in Practice_6.pdf and Practice_6.ipynb documents.

Graph Neural Networks

Q: What is the purpose of the 'message()' method in Pytorch Geometric and how is it linked to message passing?

A: The message() method of the MessagePassing class computes the message that a node i sends to another node j during the message passing phase. When all messages to j from its neighbours are computed, these are aggregated, for example by summing them, using the method aggregate(). The purpose of aggregation is to combine all the messages from the neighbours into a message of fixed size, regardless of the number of neighbours. The resulting message is then used to update the hidden state of node j according to the method update(). These methods are tightly linked to the theoretical framework of message passing described in Bishop's Deep Learning book (slightly different from the original formulation of Gilmer, 2017), where the function Aggregate includes both the computation of the i-th message (method message()) and the aggregation of all neighbours messages (method aggregate()).

Q: Why iterating over the DataLoader object gives different outputs in Pytorch and Pytorch Geometric?

A: Pytorch Geometric is designed to work with graphs, for which a predictive model typically needs several inputs: the adjacency matrix of the graph, the node-level features, the edgelevel features and the graph-level features. Since these inputs have different shapes, they cannot be combined into a single tensor X_train (or X_val, or X_test) as you have instead seen in the previous labs working with 1D/2D/3D datasets. For this reason, the datasets used in Pytorch Geometric – and for extension the DataLoaders built from these datasets – are usually organised into Data objects, which are similar to dictionaries: given a Data object data, you can access different objects inside it as data.x (node features), data.edge_index (adjacency matrix in COO format), data.y (labels), etc. Iterating over a data loader built in this way will therefore return a single Data object instead of two tensors X_train, y_train.

Q: If two nodes in a graph are connected does it mean they are similar?

A: No, it only means that there is a certain relationship or link between them. For example, in a social network graph a connection between two users might mean friendship or following.

Q: What does a grid graph represent?

A: A grid graph is simply a graph where nodes are placed on a grid and connected to nodes that are horizontally, vertically and potentially diagonally adjacent. The grid graph in the notebook of Practice 6, used to demonstrate the functioning of message passing for a colouring task, was chosen just for its simplicity. In practical applications however, grid graphs are widely used for example for navigation problems in robotics or for fluid dynamics simulations.

Q: In 6.3, for the conv1 layer, what is num_node_features? Where can I get its value?

A: num_node_features represents the number of features per node in the data. You can get its value from the dataset using dataset.num_node_features. To make the GCN model more flexible, you can modify the __init__ method to accept num_node_features as a parameter instead of accessing it directly from the dataset. This allows the model to work with different datasets without modification.