Technical University of Denmark

Advanced machine learning. Module 3.

Exercise 2. Graphs neural networks

Contents

Exercise A Invariant aggregation functions

In a graph neural network, when aggregating information from neighbors or when aggregating information from all nodes to form a graph-level prediction, it is important that the aggregation function does not depend on the order of the inputs.

Question A.1: Which of the following functions on $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ are permutation invariant:

1. Sum: $g\left(\sum_{n=1}^{N} f(\boldsymbol{x}_n)\right) = g\big(f(\boldsymbol{x}_1) + f(\boldsymbol{x}_2) + \cdots + f(\boldsymbol{x}_N)\big)$
2. Product: $g\left(\prod_{n=1}^{N} f(\boldsymbol{x}_n)\right) = g\big(f(\boldsymbol{x}_1) \cdot f(\boldsymbol{x}_2) \cdot \cdots \cdot f(\boldsymbol{x}_N)\big)$
3. Maximum: $g\left(\max_{n=1}^{N} f(\boldsymbol{x}_n)\right) = g\Big(\max\left(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)\right)\Big)$
4. Concatenation: $g\left(\operatorname*{concat.}_{n=1}^{N} f(\boldsymbol{x}_n)\right) = g\Big(\big[f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)\big]\Big)$

for arbitrary functions $f(\cdot)$ and $g(\cdot)$.

A

Exercise B Simple graph neural networks

Consider a GNN defined as

$$\text{AGGREGATE}: m_{\mathcal{N}(u)}^{(k)} = \sum_{v \in \mathcal{N}(u)} h_v^{(k)}$$

$$\text{UPDATE}: h_u^{(k+1)} = \frac{m_{\mathcal{N}(u)}^{(k)}}{\sqrt{\sum_{v \in \mathcal{V}} \left(m_{\mathcal{N}(v)}^{(k)}\right)^2}}$$

where the node representations are scalar, and initialized randomly.

Question B.1: Assuming that a large number of update rounds is computed, what will the node representations converge to?

Hint:

Consider the basic GNN, where each round consists of the following update:

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\boldsymbol{W}_{\text{self}}^{(k)} \boldsymbol{h}_u^{(k-1)} + \boldsymbol{W}_{\text{neigh.}}^{(k)} \sum_{v \in \mathcal{N}(u)} \boldsymbol{h}_v^{(k-1)} + \boldsymbol{b}^{(k)}\right)$$

Question B.2: If a graph contains $|\mathcal{V}| = 10$, the dimension of the node representation is $D = 32$ (i.e. $\boldsymbol{h}_u^{(k)} \in \mathbb{R}^{32}$), the GNN performs 5 message passing rounds, and weight matrices are not shared between rounds, what is the total number of parameters in the GNN?

B

Exercise C Programming exercise

In this exercise you will work with a graph neural network for graph-level classification implemented in the script `gnn_graph_classification.py`.

We will use the *MUTAG dataset* introduced by Debnath et al.: a collection of nitroaromatic compounds (molecular graphs) and the task is to predict their mutagenicity on Salmonella typhimurium (graph-level binary classification). Vertices represent atoms and edges represent bonds, and the 7 discrete node labels represent the atom type (one-hot encoded). There are a total of 188 graphs in the dataset.

Question C.1: Examine and run the code for loading the graph data.

- Extract a single batch from the training loader using the code `data_batch = next(iter(train_loader))`.
- The variable `data_batch` will then contain the following important variables which you should examine to make sure you understand:

  `data_batch.x`: Node features

  `data_batch.edge_index`: Edges

  `data_batch.batch`: Index of which graph in the batch each node belongs to.

Question C.2: Examine and run the code that defines the graph neural network `SimpleGNN`.

- Based on the components defined in the `__init__` function and the computations carried out in the `forward` function, sketch a diagram of the graph neural network architecture.
- What are the AGGREGATE and UPDATE functions that are implemented?
- Where and how are residual connections used?
- The messages are aggregated using a sum. To do this, the code uses the function `torch.index_add`. Make sure you understand this function, and look up its documentation if necessary. The same function is used to compute the graph level aggregation.
- What are the dimensions and purpose of the inputs and the output of the forward function?

Question C.3: Examine and run the remaining code to fit the GNN. Make sure you understand the following:

- Which loss function, optimizer, and learning rate are used?
- What does the learning rate scheduler do?
- How is the training/validation loss and accuracy computed.

After having fitted the GNN, examine the two generated plots. Does the model seem to overfit or underfit?

Question C.4: Modify the code to achieve the best possible validation loss. *Do not change the training/validation split, and do no look at the test set.* You might consider the following modifications:

- Change the model hyperparameters (the state dimension and number of message passing rounds)
- Change optimizer hyperparameters (learning rate schedule and number of epochs).
- Regularize by adding weight decay or dropout layers.
- Change the model architecture, for example by introducing a GRU update.

Question C.5: Using the provided code, save your test set predictions in a file `test_predictions.pt`, and hand it in on DTU Learn. I will compute the test loss on your predictions and lowest loss will be honored as the class winner.