zakaria echchair
zakaria.echchair@polytechnique.edu

Lab session # 7

ALTEGRAD 2022

12/21/22

1 Question 1

As we notice in Figure 1, if the MLP is the identity so we will have at the output of this layer exactly the input vector, then the sum aggregator will sum up the elements of the vector and also the last MLP layer will be the identity, so its input and output are the same which is the sum of the vector elements. and the bais is 0.

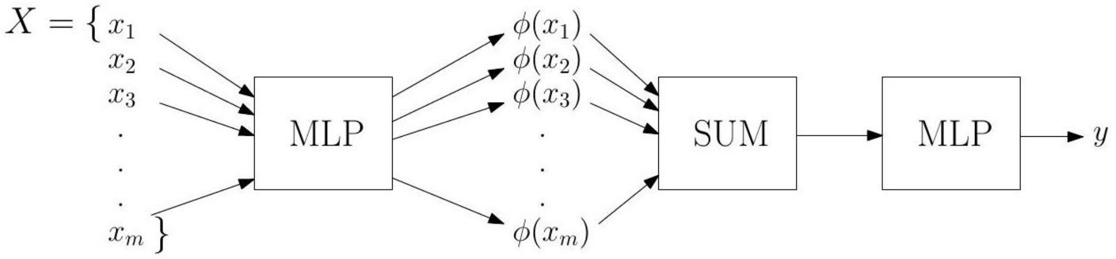

Figure 1: the architecture

2 Question 2

A DeepSets model is a neural network architecture that is designed to process sets of inputs and output a fixed-length representation of the set. This is achieved by first encoding each element of the set separately, and then aggregating the encoded elements using a permutation-invariant function, such as a sum or a mean.
To show that there exists a DeepSets model that can embed the two sets X_1 and X_2 into different vectors, we can use a simple example. Consider a DeepSets model with two encoding layers, each with two units, and a final aggregation layer with one unit. The model can be represented by the following equations:

$$x_{1,encoded} = \mathrm{ReLU}(W_1 \cdot x_1 + b_1)$$
$$x_{2,encoded} = \mathrm{ReLU}(W_1 \cdot x_2 + b_1)$$
$$x_{1,aggregated} = \sum x_{1,encoded}$$
$$x_{2,aggregated} = \sum x_{2,encoded}$$
$$output = \mathrm{ReLU}(W_2 \cdot [x_{1,aggregated}, x_{2,aggregated}] + b_2)$$

In this model, the input x_1 and x_2 are the two sets X_1 and X_2, respectively. The encoding layers apply a ReLU activation function to the dot product of the inputs and the weight matrix W_1, plus the bias vector b_1. The aggregation layer sums the encoded elements of each set. Finally, the output layer applies a ReLU activation function to the concatenation of the aggregated sets and the weight matrix W_2, plus the bias vector b_2.
By choosing appropriate values for the weight matrices and bias vectors, it is possible to map the two sets X_1 and X_2 to different vectors in the output layer. For example, if we set the weight matrices and bias vectors as follows:

$$W_1 = \begin{bmatrix} 1 & 0\,0 & 1 \end{bmatrix} \quad b_1 = \begin{bmatrix} 0\,0 \end{bmatrix} \, W_2 = \begin{bmatrix} 1 & -1 \end{bmatrix} \quad b_2 = \begin{bmatrix} 0 \end{bmatrix}$$

then the output for X_1 will be $[1.2 + (-0.7) + (-0.8) + 0.5] = [0.2]$ and the output for X_2 will be $[0.2 + (-0.3) + 0.2 + 0.1] = [0.2]$. As a result, the two sets are mapped to different vectors in the output layer, as desired.

3 Question 3

An end-to-end learning approach for classifying sets of graphs could involve the following steps:

”DeepSets” is an end-to-end learning approach that can be used to classify sets of graphs. This approach involves several steps, including preprocessing, encoding, aggregation, classification, and training.

In the preprocessing step, each graph in the set is transformed into a suitable input representation for the model. This could involve converting the graph into a matrix representation, such as an adjacency matrix or a Laplacian matrix, or extracting structural features from the graph, such as degree centrality or shortest path lengths.
The individual graphs in the set are then encoded separately using a suitable neural network architecture in the encoding step. This could involve using a graph convolutional network (GCN) to encode the graph structure, or using a recurrent neural network (RNN) to process the graph node-by-node or edge-by-edge.
The encoded graphs are aggregated into a single fixed-length representation of the set in the aggregation step. This could involve using a permutation-invariant function, such as a sum or a mean, or using a more complex function, such as a self-attention mechanism.
The aggregated representation of the set is fed into a final classification layer in the classification step to predict the class label for the set. This could involve using a fully-connected layer with a softmax activation function, or using a more complex architecture, such as a support vector machine (SVM) or a multi-layer perceptron (MLP).
Finally, the entire model is trained end-to-end using a suitable loss function and an optimization algorithm in the training step. This could involve using cross-entropy loss for classification tasks and stochastic gradient descent as the optimization algorithm.
There are many possible variations and approaches that could be employed in this end-to-end learning approach, depending on the specific requirements and constraints of the task.

4 Question 4

It is important to consider the ordering of the protein sequence in this task, as the specific arrangement of amino acids in the sequence can affect the protein's structure and function.
To take the protein sequence ordering into account in the model, one possibility could be to incorporate additional information about the protein sequence into the graph representation. For example, this could involve adding edge features to the graph to indicate the relative positions of amino acids in the sequence or using a graph representation that explicitly encodes the sequence order (such as a linear chain graph).
It may also be worthwhile to consider modifying the readout function to incorporate sequence information. For example, the readout function could be designed to take into account the relative positions of nodes in the graph, or it could use additional information about the sequence order as input. In this context, Graph convolutional layers operate on graph-structured data and could be used to compute a weighted sum of the feature vectors of a node's neighbors, where the weights are learned during training. The weights could be used to emphasize the contribution of certain neighbors based on their relative positions in the graph. For example, the weights could be higher for neighbors that are closer in the sequence order, or for neighbors that are connected by edges with certain label attributes (e.g., ”previous,” ”next,” etc.).

References