Solal Danan
solal.danan@polytechnique.edu

Lab session # 7
ALTEGRAD 2024

12/10/24

1 Question 1

For the task of determining the sum of a multiset of integers using the DeepSets architecture, the optimal parameters for the MLPs ϕ and ρ should yield the identity mapping, ensuring that the network computes the exact sum in a permutation-invariant manner.
The DeepSets model represents the function $f(X)$ as:

$$f(X) = \rho\left(\sum_{x_i \in X} \phi(x_i)\right),$$

where $X = \{x_1, x_2, \dots, x_M\}$ is the input multiset, $\phi : \mathbb{R} \to \mathbb{R}$ transforms each element of the set, and $\rho : \mathbb{R} \to \mathbb{R}$ transforms the aggregated result. For simplicity, assume each x_i is a scalar (an integer).

Transformation $\phi(x_i)$

Since we want to learn a summation, the ideal scenario is $\phi(x_i) = x_i$. Consider a simple linear layer for ϕ:

$$\phi(x_i) = W_\phi x_i + b_\phi.$$

Here, W_ϕ and b_ϕ are parameters of a single-layer MLP with no nonlinear activation. If $x_i \in \mathbb{R}^d$, we would have W_ϕ as a $d \times d$ matrix and b_ϕ as a d-dimensional vector. To preserve each dimension without change, we would set:

$$W_\phi = I_d \quad \text{(the } d \times d \text{ identity matrix)}, \quad b_\phi = \mathbf{0} \quad \text{(the } d\text{-dimensional zero vector)}.$$

Sum Aggregation

After applying ϕ to each element, we sum them:

$$z = \sum_{x_i \in X} \phi(x_i) = \sum_{i=1}^{M} x_i.$$

Transformation $\rho(z)$

The final transformation $\rho : \mathbb{R} \to \mathbb{R}$ should also be the identity to correctly map the summed value to the output. A single linear layer for ρ is:

$$\rho(z) = W_\rho z + b_\rho.$$

For vector inputs/outputs, we would similarly use:

$$W_\rho = I_d, \quad b_\rho = \mathbf{0}.$$

Summary

By setting the parameters to the identity transformation at both ϕ and ρ:

$$\phi(x_i) = x_i, \quad \rho(z) = z,$$

the DeepSets model will perform:

$$f(X) = \sum_{x_i \in X} x_i.$$

Thus, the optimal parameters are those that yield identity transformations, which ensures the final output is the exact sum of the input multiset.

2 Question 2

We aim to show that there exists a DeepSets model that can embed the two sets

$$X_1 = \{[1.2, -0.7]^T, [-0.8, 0.5]^T\} \quad \text{and} \quad X_2 = \{[0.2, -0.3]^T, [0.2, 0.1]^T\}$$

into different vectors. Note that these two sets have the same element-wise sum:

$$\sum_{x \in X_1} x = [0.4, -0.2]^T = \sum_{x \in X_2} x.$$

A purely linear transformation $\phi(x) = W_\phi x + b_\phi$ cannot distinguish these sets after summation, since a linear map on two identical sums will produce identical results. To differentiate the sets, we need a nonlinear ϕ.

Introducing Nonlinearity with ReLU

We define $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ as:

$$\phi(x) = \text{ReLU}(W_\phi x + b_\phi),$$

where $\text{ReLU}(z) = \max(0, z)$ is applied element-wise. Consider the choice:

$$W_\phi = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad b_\phi = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This configuration takes the first component of the input vector and copies it into both output coordinates, then applies ReLU to each component. Hence:

$$\phi([x_1, x_2]^T) = \begin{bmatrix} \max(0, x_1) \\ \max(0, x_1) \end{bmatrix}.$$

For X_1

Consider the elements of X_1:

$$[1.2, -0.7]^T \xrightarrow{\phi} \begin{bmatrix} \max(0, 1.2) \\ \max(0, 1.2) \end{bmatrix} = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}, \quad [-0.8, 0.5]^T \xrightarrow{\phi} \begin{bmatrix} \max(0, -0.8) \\ \max(0, -0.8) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

For X_2

Consider the elements of X_2:

$$[0.2, -0.3]^T \xrightarrow{\phi} \begin{bmatrix} \max(0, 0.2) \\ \max(0, 0.2) \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}, \quad [0.2, 0.1]^T \xrightarrow{\phi} \begin{bmatrix} \max(0, 0.2) \\ \max(0, 0.2) \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}.$$

We have:

$$\sum_{x \in X_1} \phi(x) = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}, \quad \sum_{x \in X_2} \phi(x) = \begin{bmatrix} 0.4 \\ 0.4 \end{bmatrix}.$$

These two resulting vectors are clearly distinct.

Applying ρ

Finally, choose $\rho : \mathbb{R}^2 \to \mathbb{R}$ as a simple linear transformation:

$$\rho(z) = W_\rho z, \quad \text{with } W_\rho = [1, 0].$$

Then:

$$\rho\left(\sum_{x \in X_1} \phi(x)\right) = [1, 0] \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix} = 1.2,$$

$$\rho\left(\sum_{x \in X_2} \phi(x)\right) = [1, 0] \begin{bmatrix} 0.4 \\ 0.4 \end{bmatrix} = 0.4.$$

The outputs for X_1 and X_2 are 1.2 and 0.4 respectively, demonstrating that by choosing a suitable nonlinear ϕ (in this case, using ReLU), we can embed the two sets into different final vectors.

3 Question 3

DeepSets could correspond to a submodule of a GNN architecture in a graph classification problem. Consider that a GNN often operates on sets of node features and edge features. If we treat the graph as a set of nodes with associated embeddings, then the process of aggregating node representations into a single graph-level representation can be viewed as a form of permutation-invariant aggregation over a set. Specifically, we can decompose a GNN's final readout function, which aggregates the node-level embeddings into a single graph-level vector, into a function of the form:

$$f(\{x_1, x_2, \dots, x_M\}) = \rho\left(\sum_{m=1}^{M} \phi(x_m)\right),$$

where ϕ and ρ are neural transformations (MLPs), and $\{x_1, \dots, x_M\}$ are the node embeddings of the graph. This is exactly the form of a DeepSets model. Thus, the node aggregation step in a GNN's readout phase can be seen as an instance of the DeepSets framework.
In other words, the final pooling layer of a GNN, which collects and combines the node-level representations into a single graph-level embedding, can be implemented as a DeepSets module, making DeepSets naturally applicable as a submodule within a GNN architecture for graph-level classification tasks.

4 Question 4

Consider an Erdős–Rényi random graph $G(n, p)$ with n nodes. In this model, each of the $\binom{n}{2}$ possible edges is included independently with probability p. Thus, the number of edges E in $G(n, p)$ is a binomial random variable with parameters $\binom{n}{2}$ and p.
More formally, let:

$$E = \sum_{1 \le i < j \le n} X_{ij},$$

where each X_{ij} is a Bernoulli random variable with parameter p, representing the presence (1) or absence (0) of the edge (i, j). Hence:

$$E \sim \text{Binomial}\left(\binom{n}{2}, p\right).$$

The expected value of a Binomial(m, p) random variable is mp and its variance is $mp(1-p)$. For our case:

$$\mathbb{E}[E] = \binom{n}{2}p = \frac{n(n-1)}{2}p,$$

$$\text{Var}(E) = \binom{n}{2}p(1-p) = \frac{n(n-1)}{2}p(1-p).$$

Since $\binom{15}{2} = \frac{15 \cdot 14}{2} = 105$, we have:

Case 1: $p = 0.2$

$$\mathbb{E}[E] = 105 \cdot 0.2 = 21,$$

$$\text{Var}(E) = 105 \cdot 0.2 \cdot (1 - 0.2) = 105 \cdot 0.2 \cdot 0.8 = 16.8.$$

Case 2: $p = 0.4$

$$\mathbb{E}[E] = 105 \cdot 0.4 = 42,$$

$$\text{Var}(E) = 105 \cdot 0.4 \cdot (1 - 0.4) = 105 \cdot 0.4 \cdot 0.6 = 25.2.$$

These results highlight how increasing the edge probability p not only increases the expected number of edges but also the variance.