Exercise Sheet #11

Due: 30.01.23 11h00

Roman Remme

General Regulations.

- Please hand in your solutions in groups of three people. A mix of attendees from Monday and Tuesday tutorials is fine.
- Your solutions to theoretical exercises can be either handwritten notes (scanned), or typeset using LaTeX. In case you hand in handwritten notes, please make sure that they are legible and not too blurred or low resolution, otherwise we might not correct your submission.
- For the practical exercises, the data and a skeleton for your jupyter notebook are available at `https://github.com/hci-unihd/mlph_sheet11`. Always provide the (commented) code as well as the output, and don't forget to explain/interpret the latter. Please hand in both the notebook (`.ipynb`), as well as an exported pdf.
- Submit all your files in the Übungsgruppenverwaltung, only once for your group of three.

1 Positional Encoding

In transformers, the features X are combined with the positional embeddings E of the tokens. We will consider the simplest case of self-attention where the attention weights are computed as the soft(arg)max over the "scores" K^TQ with $K = Q = X + E$ in case features and positional embeddings are combined by adding them up and $K = Q = \mathrm{cat}(X, E)$ in case they are concatenated. Let the positional embeddings $E \in \mathbb{R}^{p\times n}$ be defined as

$$E_{(2k),i} = \sin\left(2i \cdot \exp\left(-\frac{k \cdot \log(10000)}{p}\right)\right) \tag{1}$$

$$E_{(2k+1),i} = \cos\left(2i \cdot \exp\left(-\frac{k \cdot \log(10000)}{p}\right)\right)\,, \tag{2}$$

for $k \in \{0, 1, 2, \ldots, \frac{p}{2} - 1\}$.

(a) Expand the scores in X and E both for addition and concatenation. Discuss the different resulting terms. (2 pts)

(b) Implement the positional encoding as defined above and plot E^TE for 64 tokens with an embedding dimension of 256. What do you observe? (2 pts)

(c) Plot K^TQ for some random features (with the same variance as the positional embedding) for both addition and concatenation and discuss what you see. (2 pts)

(d) Assume that both the positional encodings and the features of all tokens lie in some (comparatively) low-dimensional subspace of the high-dimensional euclidean space they both are elements of. Argue why under these assumptions, adding the positional encodings will lead to similar results as concatenating them with the features. (2 pts)

2 Interpolating between Aggregation functions

In the design of neural architectures for graphs, oftentimes aggregation functions $\bigoplus$ are used either to combine incoming messages of a node to the node features of the next layer, or to combine features of all nodes to ones for the whole graph in a permutation invariant way. Simple choices such as the sum, min, mean and max are popular, however it's also possible to interpolate between those, which we will look into in this exercise.

(a) Let us define the generalized soft(arg)max aggregation as

$$\text{soft(arg)max\_agg}(v, \mathbf{m}; \lambda, \eta) = |\mathcal{N}(v)|^\eta \sum_{u \in \mathcal{N}(v)} \text{soft(arg)max}(\mathbf{m}; \lambda)_u \cdot \mathbf{m}_u \tag{3}$$

for some node v with incoming messages $\mathbf{m}$ ($\mathbf{m}_u$ is the message from neighboring node u in the neighborhood $\mathcal{N}(v)$ of v), temperature $\lambda \in (0, \infty)$ and $\eta \in [0, 1]$. Show one can obtain the mean max and sum aggregation functions as limits of this function. Hint: Consider the cases $\beta \to 0$, $\beta \to \infty$ and recall the results from Sheet 7. (3 pts)

(b) **Bonus:** Another option is the power mean aggregation. Using the same notation as in (a), it reads

$$\text{power\_mean\_agg}(v, \mathbf{m}; q, \eta) = |\mathcal{N}(v)|^\eta \cdot \left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{m}_u^q \right)^{\frac{1}{q}}, \tag{4}$$

where q is a scalar parameter in $(0, \infty)$. As in (a), show how mean max and sum can be obtained as limits of this generalized aggregation function. (2 pts)

(c) Determine how the generalized aggregation functions could be incorporated into a GCN and how you would deal with their extra parameters. (1 pts)

3 Observing Oversmoothing

(a) Train a Graph Convolutional Network (GCN) with two layers and the model and training hyperparameters from the given jupyter notebook on the provided[1] subset of the CellTypeGraph Benchmark. Repeat the same for 1, 5 and 20 layers and plot training and validation losses over the epochs. Discuss the performance of the different models. (2 pts)

(b) For each of your trained models, compute the standard deviation of the model outputs, averaged over all output channels and samples of the validation set. Repeat the same for newly initialized models of the same sizes and discuss the results. (2 pts)

(c) Look up at least two distinct possible avenues to combat over-smoothing in GCNs. Give the references and give a short description of the ideas. (4 pts)

[1] `https://github.com/hci-unihd/mlph_sheet11`, for the full dataset see `https://github.com/hci-unihd/celltype-graph-benchmark`