Assignment Overview

Assignment 3 contains two parts:

1. **Attention exploration (14 pts):** What kinds of operations can self-attention easily implement? Why should we use fancier things like multi-headed self-attention? This section will use some mathematical investigations to illuminate a few of the motivations of self-attention and Transformer networks.
2. **Transformer from Scratch (16 points):** This section will ask you to code and answer short questions about a Decoder-Only Transformer and training for French-to-English translation. Note that training and evaluation will take about 30mins on a Colab T4 GPU, so please plan your time accordingly.

   (Note: You can certainly find the code online or have it completed easily via ChatGPT, but we encourage you to try coding it yourself for a better learning experience. If you are stuck, please come to office hours and if that's not possible, ask ChatGPT for "hints" instead of the full answer so that you can truly learn!)

Submission Instructions

You will submit this assignment on GradeScope as two submissions:

1. For the **written assignment**, ensure it is typeset in LaTeX, each answer is on a different page, and that the pages are tagged correctly on Gradescope.
2. For the **coding assignment**: Complete the coding and short QA blocks in `a3.ipynb` and upload to Gradescope.

Attention for In-context Learning (14 pts)

Multi-headed self-attention is the core modeling component of Transformers. Through these written questions, we'll explore how attention allows a language model to perform a simple in-context learning task, and motivate why multi-headed self-attention can be preferable to single-headed self-attention.

A running example. Suppose you are playing rock–paper–scissors against your friend, and you want to use a language model to predict what your friend will do next given their past moves to win the game:

rock	*paper*	*rock*	____
1	2	3	4

Everyone's habits are their own, so we can assume that the language model does not have knowledge about your friend's habit. It must be inferred from this sequence: learning in context. One strategy is to get and output the action that followed previous *rock* (*paper*). *How could attention implement this strategy*? [1]

Problem 1 (2 points) Copying *paper* in attention

Recall that attention can be viewed as an operation on a query $q \in \mathbb{R}^d$, a set of key vectors $\{k_1, \ldots, k_n\}, k_i \in \mathbb{R}^d$, and a set of value vectors $\{v_1, \ldots, v_n\}, v_i \in \mathbb{R}^d$, specified as follows:

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}, \tag{1}$$

$$c = \sum_{i=1}^{n} v_i \alpha_i, \tag{2}$$

where α_i are frequently called the "attention weights", and the output $c \in \mathbb{R}^d$ is a correspondingly weighted average over the value vectors.

In our example, q would be your friend's latest action *rock*, and each past time step i contributes a key k_i and a value v_i. The key k_i identifies the action taken at step i but is also position-dependent, so two steps with the same action still have distinct keys (e.g., $k_1 = k^{(1)}_{\text{rock}}$, $k_2 = k^{(2)}_{\text{paper}}$, $k_3 = k^{(3)}_{\text{rock}}$, with $k_1 \neq k_3$). The value v_i encodes the action itself (e.g., $v_2 = v_{\text{paper}}$). The goal of our model is to output *paper*. We'll first show that it's particularly simple for attention to "copy" the value vector of *paper* to the output c.

Describe (in one sentence) what properties of the inputs to the attention operation would result in the output c being approximately equal to v_2. Specifically, what must be true about the query q, the values $\{v_1, v_2, v_3\}$ and/or the keys $\{k_1, k_2, k_3\}$?

Problem 2 (4 points) Averaging the two

Two rounds later, the game reads:

rock	*paper*	*rock*	*scissors*	*rock*	____
1	2	3	4	5	6

Now two actions follow a rock: move 2 (*paper*) and move 4 (*scissors*). Copying just one of them discards evidence. A more reasonable answer is predicting each with probability

[1] This assignment is inspired by https://arxiv.org/pdf/2402.14735 that theoretically prove how attention can perform a in-context learning task. Check it out if you are interested!

$\frac{1}{2}$ attending *equally* to both and outputting $\frac{1}{2}(v_2 + v_4)$. We will now show attention supports such averaging.

Assuming all the key vectors $\{k_1, \ldots, k_n\}$ $(n = 5)$ are orthonormal, that is, $k_i \perp k_j$ for all $i \neq j$ and $\|k_i\| = 1$ for all i. Give an expression for our query vector $q = q_{\text{rock}}$ (the last *rock* action) such that the output c is approximately equal to the average of $v_2 = v_{paper}$ and $v_4 = v_{scissor}$, that is, $\frac{1}{2}(v_2 + v_4)$.[2] Note that you can reference the corresponding key vector of v_2 and v_4 as k_2 and k_4.

Problem 3 (5 points) Drawbacks of single-headed attention

In the previous part, we saw how it was *possible* for a single-headed attention to focus equally on two values. The same concept could easily be extended to any subset of values. In this question we'll see why it's not a *practical* solution.

Now suppose the key vectors $\{k_1, \ldots, k_n\} \subset \mathbb{R}^d$ $(n = 5,\ d \geq n)$ are randomly sampled as $k_i = \mu_i + \epsilon_i$, where the vectors $\mu_i \in \mathbb{R}^d$ are fixed and known to you, and $\epsilon_i \in \mathbb{R}^d$ is a random noise vector, unknown to you and independent across keys. Further, assume that the μ_i are orthonormal: $\mu_i^\top \mu_j = 0$ if $i \neq j$, and $\|\mu_i\| = 1$.

(a) (2 points) Assume that the noise in each key is negligible, i.e., $\|\epsilon_i\| \approx 0$ for all i. Design a query q in terms of the μ_i such that, as before, $c \approx \frac{1}{2}(v_2 + v_4)$, and provide a brief argument as to why it works.

(b) (3 points) Though single-headed attention is resistant to small noise in the keys, some types of larger noise may pose a bigger issue. Specifically, in some cases, the key vector k_2 may be larger or smaller in norm than the others, while still pointing in the same direction as μ_2. As an example, suppose that $\|\epsilon_i\| \approx 0$ for all $i \neq 2$, but $\epsilon_2 = \delta\mu_2$ for some random scalar $\delta > -1$ with large variance. See figure 1 for a visual illustration.

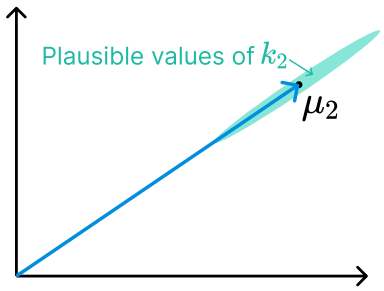

Figure 1: The vector μ_2 (shown here in 2D as an example), with the range of possible values of k_2 shown in green. k_2 points in the same direction as μ_2, but may have larger or smaller magnitude.

When you sample $\{k_1, \ldots, k_n\}$ multiple times, and use the q vector that you defined in part (a), what qualitatively do you expect the vector c will look like for different samples?

Problem 4 (3 points) Benefits of multi-headed attention

Now we'll see some of the power of multi-headed attention. We'll consider a simple version of multi-headed attention which is identical to single-headed self-attention as

[2] Hint: while the softmax function will never *exactly* average the two vectors, you can get close by using a large scalar multiple in the expression.

we've presented it in this homework, except that we define two query vectors, q_1 and q_2, for the last action *rock*. Running single-headed attention with each query gives two outputs: c_1 from q_1 and c_2 from q_2. The final output of the multi-headed attention is their average, $c = \frac{1}{2}(c_1 + c_2)$.

We'll use the same setup as in Problem 3: the key vectors $\{k_1, \ldots, k_n\} \subset \mathbb{R}^d$ ($n = 5$, $d \geq n$) are randomly sampled as $k_i = \mu_i + \epsilon_i$, where you know the μ_i but not the random noise ϵ_i, and the noise is independently drawn across keys. As before, the μ_i are orthonormal: $\mu_i^\top \mu_j = 0$ if $i \neq j$, and $\|\mu_i\| = 1$.

(a) (1 point) Suppose the noise is negligible, so $\|\epsilon_i\| \approx 0$ for every i. Design q_1 and q_2 such that $c \approx \frac{1}{2}(v_2 + v_4)$. (Note: do not use $q_1 = q_2 =$ the same q you designed for Problem 3.)

(b) (2 points) Now take the noise from Problem 3: $\|\epsilon_i\| \approx 0$ for every $i \neq 2$, but $\epsilon_2 = \delta\mu_2$ for some random scalar $\delta > -1$ with large variance. When you sample $\{k_1, \ldots, k_n\}$ multiple times and use the q_1 and q_2 you designed in part (a), what do you expect the output c to look like for different samples? Briefly explain why.