CSE 151B Deep Learning
UCSD, Spring 2025

Homework 4

Due 11:59 p.m, June 3rd, 2025

Please submit your written solutions as a single PDF file using the provided LaTeX template on the course website. (https://sites.google.com/view/cse151b-251b). You can use Overleaf (https://www.overleaf.com/) to compile LaTex files online. For problems requiring code, additionally submit all necessary code by running '`collect_submission`' script to create one zip file. Code must run and produce the results reported in the PDF for full credit.

1 Attention Mechanism [6 Points]

Multi-head self-attention is the core modeling component of Transformers. Recall that attention can be viewed as an operation on a query vector $q \in \mathbb{R}^d$, value vectors $\{v_1, \cdots, v_n\}$ where $v_i \in \mathbb{R}^d$ and key vectors $\{k_1, \cdots, k_n\}$, $k_i \in \mathbb{R}^d$ as

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

where $\boldsymbol{\alpha} := \{\alpha_1, \cdots, \alpha_n\}$ are the "attention weights".

Problem A [2 points]: One advantage of attention is that it is particularly easy to "copy" a value vector to the output. In this problem, we'll explore why this is the case.

i. [0.5 points]: Explain why $\boldsymbol{\alpha}$ can be interpreted as a categorical probability distribution.

ii. [0.5 points]: The distribution $\boldsymbol{\alpha}$ is typically relatively "diffuse"; the probability mass is spread out between many different α_i. However, this is not always the case. Describe (in one sentence) under what conditions the categorical distribution $\boldsymbol{\alpha}$ puts almost all of its weight on some α_j (i.e. $\alpha_j \gg \sum_{i \neq j} \alpha_i$). What must be true about the query q and/or the keys $\{k_1, \cdots, k_n\}$?

iii. [0.5 points]: Under the conditions you gave in (ii), **describe** the output c.

iv. [0.5 points]: **Explain** (in two sentences or fewer) what your answer to (ii) and (iii) means intuitively.

Problem B [4 points]: Single-headed attention has drawbacks. Consider a set of key vectors $\{k_1, \cdots, k_n\}$ that are now randomly sampled, $k_i \sim N(\mu_i, \Sigma_i)$, where the means $\mu_i \in \mathbb{R}^d$ are known, but the covariances Σ_i are unknown. Further, assume that the means μ_i are all orthogonal: $\mu_i^\top \mu_j = 0$ if $i \neq j$ and unit norm $\|\mu_i\| = 1$.

i. [2 points]: Assume that the covariance matrices are $\Sigma_i = \alpha I, \forall i \in \{1, 2, \cdots, n\}$, for vanishingly small α. Design a query q in terms of the μ_i such that $c \approx \frac{1}{2}(v_a + v_b)$ and provide a brief argument as to why it works. ***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.*

ii. [2 points]: Although single-headed attention is robust against small perturbations in the keys, certain larger perturbations may pose a signficant issue. In some cases, one key vector k_a may be larger or smaller in norm than the others, while still pointing in the same direction as μ_a. As an example, let us consider a covariance for item a as $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$

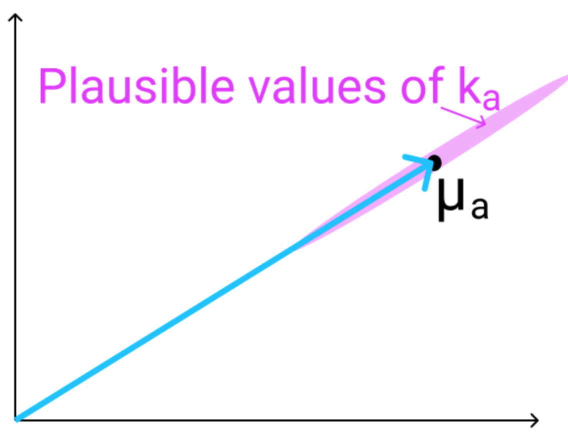

Figure 1: The vector μ_a, with the range of possible values of k_a shown in red. k_a points in roughly the same direction as μ_a , but may have larger or smaller magnitude.

for vanishingly small α (as shown in Figure 1). This causes k_a to point in roughly the same direction as μ_a, but with large variances in magnitude.

Further, let $\Sigma_i = \alpha I$ for all $i \neq a$. When you sample $\{k_1, \cdots, k_n\}$ multiple times, and use the q vector that you defined in part i., what do you expect the vector c will look like qualitatively for different samples? Think about how it differs from part (i) and how c's variance would be affected.

2 Transformers [4 Points]

In this problem you will implement a Transformer model and use them to perform machine translation. We will be implementing a one-layer Transformer encoder which can encode a sequence of inputs and produce a final output of possibility of tokens in target language. You can refer to the original paper. In `models` folder, you will see the file `Transformer.py`. You will implement the functions in the `TransformerTranslator` Jupyter notebook.

Problem A [2 points]: Recall that a Transformer does not include any posi- tional information about the order in which the words in the sentence occur. Therefore, we need to append a positional encoding token at each po- sition. We will use BERT embedding. Replace the embeddings with pre-trained word embeddings such as word2vec or GloVe; Many of them are available on Hugging Face. Experiment with different embeddings and report the impact in the notebook.

Problem B [2 points]: Train the transformer architecture on the dataset with the default hyperparameters – you should get a descent perplexity. Then perform hyperparameter tuning and include the improved results in a report explaining what you have tried. Do NOT just increase the number of epochs as this is too trivial.