This exam has nine questions. The first four are intended to have very short answers. All questions vary in difficulty and topic, so I recommend searching around for those you're more familiar with first. I don't expect many people will achieve a perfect score, so keep that in mind if you're struggling. I recommend putting some thoughts down for each question at least, as we will be giving partial credit. ***Write your UNI and name on the* TOP RIGHT *of every page.***

Problem 1 (2 points) Given a distribution p_θ over strings over a vocabulary $\mathcal{V}$ that includes an `EOS` token, describe the condition under which the greedy decoding generation process terminates. (The answer should be a single sentence.)

Solution:

The greedy decoding process terminates when the EOS token is the most likely token under the model's conditional distribution at the current step.

- **Full credit:** EOS is most likely
- **Partial credit:** EOS is generated (without specifying likelihood condition)
- No additional points for mentioning maximum sequence length termination

Problem 2 (2 points) I have two possible prompts for asking for the location of Columbia University.

a. `Where is Columbia University located?`

and

b. `Columbia University is located in`

I have a pretrained, not posttrained language model. Which (a or b) prompt is most likely to lead to the model generating the correct answer? Just provide the answer, no commentary.

Solution:

There are likely more examples in the training set that are of the format "Columbia University is located in New York" than "Where is Columbia University located? New York". The question/answer formatting of information is less common.

- **Full credit:** b

Problem 3 (2 points) Consider performing DPO on a preference dataset

$$\{(x, y_c, y_r)_i\}_{i=1}^k \tag{1}$$

where preferences come from a model $\phi(\hat{y}, x)$ compared to running Simple Reward Learning with a rollout count of 2 on $\{x\}_{i=1}^{k}$ (the same x question set), where preferences also come from $\phi(\hat{y}, x)$.

What is the fundamental difference between these two strategies? The answer should be a single sentence.

Solution:

DPO directly optimizes the policy using a fixed dataset of preference pairs (y_c, y_r), giving a static distribution of responses, whereas SRL generates new outputs from the current model at each training step and uses their rewards to update the policy dynamically.

Problem 4 (3 points) I have roughly 1000 examples of problems I'm trying to solve. I'm testing providing them as few-shot examples to my language model. At first, the model's performance improves with more examples, but then my python script crashes when I use 100 or more.

What went wrong, and what should I do? The answer should be roughly two sentences.

Solution:

The script crashes because the model's input exceeds its context window when too many few-shot examples are included. To fix this, you can reduce the number of examples per prompt, use a model with a larger context window or memory mechanism, or alternatively fine-tune the model on your examples to avoid exceeding the context limit.

Problem 5 (6 points) The feed-forward layers in Transformers often have larger hidden dimensionality than the ambient dimensionality d of the model. Here we'll see one reason why.

Part a. Consider the first step of a feed-forward layer—a linear transformation. Let $A \in \mathbb{R}^{d \times d}$, and an input $X \in \mathbb{R}^{d \times b}$, where b is a batch size. Then the linear transformation is

$$AX \tag{2}$$

Assuming 32-bit floating-point numbers, what are the number of floating-point operations, and the number of byte accesses of this function? **Write these and the arithmetic intensity.**

Solution:

Part a. Assume 32-bit floats (4 bytes).

$$\text{FLOPs} = d^2 b \quad \text{(plus lower-order terms)}$$

$$\text{Bytes (no write)} = 4(d^2 + db) \quad \text{(read } A \text{ and } X\text{)}$$

$$\text{Bytes (with write)} = 4(d^2 + db + db) \quad \text{(also write } AX\text{)}$$

$$\text{Arithmetic intensity (no write)} = \frac{d^2 b}{4(d^2 + db)}, \qquad \text{with write: } \frac{d^2 b}{4(d^2 + 2db)}$$

Grading note: Full credit is given whether write accesses are included or not.

Part b. Now compute the same for $C \in \mathbb{R}^{5d \times d}$, and as before, $X \in \mathbb{R}^{d \times b}$. The layer is thus

$$CX \tag{3}$$

As before, 32-bit floating-point numbers, what are the number of floating-point operations, and the number of byte accesses of this function? **Write these and the arithmetic intensity.**

Solution:

Part b. Assume 32-bit floats (4 bytes).

$$\text{FLOPs} = 5d^2 b \quad \text{(plus lower-order terms)}$$

$$\text{Bytes (no write)} = 4(5d^2 + db) \quad \text{(read } C \text{ and } X\text{)}$$

$$\text{Bytes (with write)} = 4(5d^2 + db + 5db) \quad \text{(also write } CX\text{)}$$

$$\text{Arithmetic intensity (no write)} = \frac{5d^2 b}{4(5d^2 + db)}, \qquad \text{with write: } \frac{5d^2 b}{4(5d^2 + 6db)}$$

Grading note: Full credit is given whether write accesses are included or not.

Problem 6 (4 points) Recall Low-rank Adaptation, in which we finetune a neural network by learning only a low-rank update to each matrix, like:

$$\tilde{A} \leftarrow A + uv^\top \tag{4}$$

for $A \in \mathbb{R}^{m \times n}$, and $u, v \in \mathbb{R}^d$.

Let $\|u\| = \|v\| = 1$. Now, consider a vector z, with $\|z\| = 1$. Show that $\|Az - \tilde{A}z\| \leq 1$.

Solution:

$$Az - \tilde{A}z = Az - (A + uv^T)z = Az - Az - uv^T z = -uv^T z$$

$$\|Az - \tilde{A}z\| = \| - uv^T z\| = \|uv^T z\| = \|u\|v^T z = v^T z$$

Since both v and z are unit-norm, then $v \cdot z <= 1$.

- **Full credit:** Correctly reduced the problem to $\|uv^T z\|$, then correctly showed that it is $<= 1$.
- **Partial credit:** At least some linear algebra mistakes.
- **No credit:** Answer given was not close to solution.

Problem 7 (4 points) Imagine you are given a large corpus of web text, $\mathcal{D} = \{D_1, D_2, ..., D_N\}$ where each document $D_i = w_1 ... w_{T_{D_i}}, w_j \in \mathcal{V}$. You want to use this data to create a pretraining corpus, so you will develop a model-based quality filter. To train the filter, you label texts from Wikipedia, Books, and similar sources as high-quality documents. You label texts randomly sampled from CommonCrawl as low-quality documents [1]. After fitting the model, you look at the distribution of predictions on the original high-quality sources, original low-quality sources, and your data pool, $\mathcal{D}$. But uh oh, you poorly labeled your data files and don't know which corpus is which. You do know that the distributions of model outputs on the three data sets looks like the following (Figure 1):

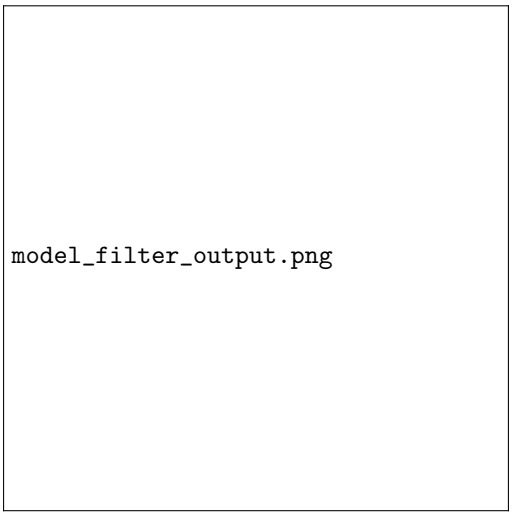


Figure 1: Distribution of model outputs on three samples of 1,000 documents from three different sources. For the purposes of this problem, imagine that the frequency is measured in 1,000's of documents.

What data sets do "Data A", "Data B", and "Data C" likely represent? Why do you think so based on the distributions?

[1] This is a replicated setup for the PALM quality filter, and the plot below shows real outputs of the filter.

Solution:

Data A represents the original high-quality data, Data B represents the original low-quality data, and Data C represents samples from our data pool. Because the classifier was trained on the original high and low-quality sources, we expect these distributions to be highly concentrated around 0 and 1 respectively. Importantly, this does not necessarily mean that the original low quality sources are all low quality; as we discussed in lecture, CommonCrawl may contain some high quality documents, but because we use them as the negative class in training the classifier, we don't expect the filter to label any of these documents as high quality. While Data C is similarly a random sample of web text, it was not explicitly included in training, and so we expect to see a concentration around 0 but more distributed predictions from 0 to 1.

Correctly assigning Data A, B, and C was worth a total of **1 point**. Even if solutions did not explicitly use the original high/low quality sources and data pool, we gave credit for assignments we felt were sufficiently close to these datasets. The explanations for Data A and Data B were worth **1 point** total. Explanations generally received credit if they discussed that Data A represents primarily high quality predictions and Data B represents primarily low quality predictions. The explanation for Data C was worth **2 points**. Full credit was only received if the solution sufficiently distinguished Data B and Data C. As both are really random web texts and contain a mix of document quality, the primary difference is that the original low-quality sources were included in training. Strictly explaining that Data C contains a mix of document quality or is a more uniform/variable distribution is not completely sufficient to distinguish Data B and C, so these solutions received a small penalty.

Problem 8 (6 points) Consider self-attention, in which we have an input set $h_1^{(\ell)}, \ldots, h_n^{(\ell)}$, and we define our output set as

$$h_i^{(\ell+1)} = \sum_{j=1}^{n} \alpha_{ij} W_V h_j^{(\ell)} \tag{5}$$

$$\alpha_{ij} = \frac{\exp((W_Q h_i^{(\ell)})^\top (W_K h_j^{(\ell)}))}{\sum_{j'=1}^{n} \exp((W_Q h_i^{(\ell)})^\top (W_K h_{j'}^{(\ell)}))} \tag{6}$$

$$\text{For } i \in \{1, \ldots, n\} \tag{7}$$

Assume that $W_Q = W_K = K_V = \beta I$, that is, they're all the identity matrix multiplied by some potentially large constant β. Further assume that all inputs $h_i^{(\ell)}$ are unit norm.

Show that for an arbitrary $h_i^{(\ell+1)}$ output, we have that $\alpha_{ii} \geq \frac{1}{n}$. (Note that this is α_{ii}, that is, the same index, not α_{ij} for potentially differing i and j.)

Solution:

We start with

$$\alpha_{ii} = \frac{\exp((W_Q h_i^{(\ell)})^\top (W_K h_i^{(\ell)}))}{\sum_{j'=1}^{n} \exp((W_Q h_i^{(\ell)})^\top (W_K h_{j'}^{(\ell)}))} \tag{8}$$

$$= \frac{\exp((\beta h_i^{(\ell)})^\top (\beta h_i^{(\ell)}))}{\sum_{j'=1}^{n} \exp((\beta h_i^{(\ell)})^\top (\beta h_{j'}^{(\ell)}))} \qquad \text{(because } W_Q = Q_k = \beta I\text{)} \tag{9}$$

$$= \frac{\exp(\beta^2 \|h_i^{(\ell)}\|_2^2)}{\sum_{j'=1}^{n} \exp((\beta h_i^{(\ell)})^\top (\beta h_{j'}^{(\ell)}))} \tag{10}$$

$$= \frac{\exp(\beta^2)}{\sum_{j'=1}^{n} \exp((\beta h_i^{(\ell)})^\top (\beta h_{j'}^{(\ell)}))} \qquad \text{(because } h_i^{(\ell)} \text{ is unit norm)} \tag{11}$$

$$= \frac{\exp(\beta^2)}{\sum_{j'=1}^{n} \exp(\beta^2 h_i^{(\ell)\top} h_{j'}^{(\ell)})} \tag{12}$$

Now, we use the fact that all $h_i^{(\ell)}$ are unit norm, so $h_i^{(\ell)\top} h_j^{(\ell)} \leq 1$ (in order to receive credit, you have to include this reasoning), so we return to the last equation and continue with:

$$\alpha_{ii} = \frac{\exp(\beta^2)}{\sum_{j'=1}^{n} \exp(\beta^2 h_i^{(\ell)\top} h_{j'}^{(\ell)})} \tag{13}$$

$$\leq \frac{\exp(\beta^2)}{\sum_{j'=1}^{n} \exp(\beta^2)} \tag{14}$$

$$= \frac{\exp(\beta^2)}{n \exp(\beta^2)} \tag{15}$$

$$= \frac{1}{n} \tag{16}$$

(Mentioning β being a large value leads to convergence is not a correct reasoning).

Problem 9 (6 points) Consider a Transformer language model, in which for a sequence $w_{1:n} \in \mathcal{V}^*$, we have independent input embeddings E and softmax embeddings U as follows:

$$h_1^{(0)}, \ldots, h_n^{(0)} = (E_{w_1} + p_1), \ldots, (E_{w_n} + p_n) \tag{17}$$

$$h_1^{(L)}, \ldots, h_n^{(L)} = \text{Transformer}(h_1^{(0)}, \ldots, h_n^{(0)}) \tag{18}$$

$$p_\theta(w_t \mid w_{<t}) = \text{softmax}(U h_{t-1}^{(L)}) \tag{19}$$

Recall that the Transformer is a sequence of (L) Transformer blocks, assuming a prenorm Transformer (the style we've mainly talked about in this class.) Imagine that we take a single loss $-\log p_\theta(w_t \mid w_{<t})$, and compute the gradient for every parameter in the Transformer. Imagine further that—oh no!—the contribution to the gradient from every self-attention sublayer and every feed-forward sublayer is exactly 0.

What is the gradient of this loss with respect to the input embedding $E_{w_{t-1}}$?

Solution:

The loss is $\mathcal{L} = -\log p_t$, where $p_t = p_\theta(w_t \mid w_{<t})_{w_t}$ is the probability corresponding to token w_t (note that this is a scalar). Let y be the one-hot vector for w_t, and $z = Uh_{t-1}^{(L)}$.

First note that we can decompose the gradient (by chain-rule) into:

$$\frac{\partial \mathcal{L}}{\partial E_{w_{t-1}}} = \frac{\partial \mathcal{L}}{\partial p_t}\frac{\partial p_t}{\partial z}\frac{\partial z}{h_{t-1}^{(L)}}\frac{\partial h_{t-1}^{(L)}}{\partial h_{t-1}^{(0)}}\frac{\partial h_{t-1}^{(0)}}{E_{w_t}}$$

Computing each part separately:

$$\frac{\partial \mathcal{L}}{\partial p_t} = -\frac{1}{p_t} \tag{1}$$

$$\frac{\partial p_t}{\partial z} = p_t(y - p) \qquad \text{because } \frac{\partial p_t}{\partial z_i} = p_t(\delta_{t,i} - p_i) \tag{2}$$

$$\frac{\partial z}{h_{t-1}^{(L)}} = U \tag{3}$$

$$\frac{\partial h_{t-1}^{(L)}}{\partial h_{t-1}^{(0)}} = I \tag{4}$$

$$\frac{\partial h_{t-1}^{(0)}}{E_{w_{t-1}}} = I \tag{5}$$

where (4) is because for all $l \in \{2, \ldots, L\}$,

$$\frac{\partial h_{t-1}^{(l)}}{\partial h_{t-1}^{(l-1)}} = \frac{\partial}{\partial h_{t-1}^{(l-1)}}\left[f\big(\mathrm{LN}(z^{(l-1)})\big) + h_{t-1}^{(l-1)}\right] = 0 + I = I$$

where $z^{(l-1)}$ is the intermediate output of the transformer (directly after attention), LN is layer-norm, and f is the feed-forward layer.

Putting this all together, we get

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial E_{w_{t-1}}} &= -\frac{1}{p_t} \cdot p_t(y - p) \cdot U \cdot I \cdot I \\ &= (p - y)U\end{aligned}$$

Common mistakes:

- Assumes zero-gradient contribution from attention and feed-forward means $h_{t-1}^{(L)} = E_{w_{t-1}} + p_{t-1}$