Week 5

Problem 1 (0 points) *Topics: Algorithmic Complexity, GEMM*

Matrix A is $m \times (C - m)$, and matrix B is $(C - m) \times 2m$. For an arbitrary scalar C, $C \in \mathbb{R}$, we will determine the optimal value of m in terms of C that optimizes (i.e., minimizes) the arithmetic complexity of the matrix multiplication AB. What is the range of possible values that m may hold? What is the optimal value or set of optimal values of m?

Solution:

Because the matrix dimensions must be positive and non-zero, m must be greater than 0 and less than C, so that $c - m > 1$. Therefore, the range of m values is $0 < m < C$.
The arithmetic intensity of the operation AB is:

$$\frac{C_{\text{math}}}{C_{\text{mem}}} = \frac{mnk}{mn + mk + kn} \tag{1}$$

$$= \frac{m(2m)(C - m)}{m(2m) + m(C - m) + (C - m)(2m)} \tag{2}$$

$$= \frac{-2m^3 + 2Cm^2}{m^2 + 3Cm} \tag{3}$$

$$= \frac{-2m^2 + 2Cm}{-m + 3C} \tag{4}$$

So, we calculate the derivative of this operation with respect to m:

$$\frac{d}{dm}\frac{-2m^2 + 2Cm}{-m + 3C} = \frac{(-m + 3C)(-4m + 2C) - (-1)(-2m^2 + 2Cm)}{(-m + 3C)^2} \tag{5}$$

$$= \frac{2m^2 - 12Cm + 6C^2}{(-m + 3C)^2} \tag{6}$$

$$\tag{7}$$

From here, note that the arithmetic intensity is roughly symmetric in the range $0 < m < C$ and reaches a maximum around $m \approx C/2$. Therefore, the optimal value of m is either 1 or $C - 1$. The value of the arithmetic intensity at $m = 1$ is slight lower than at $m = C - 1$, and so the optimal value is 1.

Problem 2 (0 points) *Topics: Algorithmic Complexity, Single vs. Multi-head attention*

Assume we are implementing both a single and multi-headed attention. For the single-headed attention, our Q, K, and V matrices are $\in \mathbb{R}^{d \times d}$, but for the multi-headed attention, the $Q^(\ell)$, K^ℓ, and V^ℓ matrices are $\in \mathbb{R}^{d \times d/k}$ for $\ell \in 1, ..., k$. Ignoring position embeddings and considering only one layer, compare the number of FLOPs required to pass a sequence of length T embeddings through a single-headed self-attention and a multi-headed attention layer. Additionally, how does the relative operation cost depend on the number of heads (k) You may use $C_{softmax}(x)$ to represent the cost in FLOPs of calculating the softmax on a vector of dimension x and may ignore additions in calculation of FLOPs.

Solution:

For self-attention, we do the following computations:

1. Compute q_i by multiplying Q by x_i
2. Compute k_j for all terms in the sequence by multiplying K by $x_j, j \in \{1, ..., T\}$
3. Compute dot product of all q_i with all k_j
4. Softmax over $q_i \cdot k_j$ products to get attention scores, $a_{ij}, f \in \{1, ..., T\}$
5. Compute v_j for all terms by multiplying V by $x_j, j \in \{1, ..., T\}$
6. Finally, compute h_i by multiplying each a_{ij} by each v_j

So for each step, the number of FLOPs required are as follows:

1. Computing q_j's, $\{Q \cdot x_i, i \in \{1, ..., T\}\}$: $\mathbf{T} * (\mathbf{d^2})$
2. Computing k_j's, $\{K \cdot x_j, j \in \{1, ..., T\}\}$: $\mathbf{T} * (\mathbf{d^2})$
3. Computing raw attention scores between all tokens, $\{a_{ij} = q_i \cdot k_j, i, j \in \{1, ..., T\}\}$: $\mathbf{T^2} * \mathbf{d}$
4. Normalizing attention scores for each token: $T * \mathbf{C_{softmax}}(\mathbf{T})$
5. Computing v_j's, $\{V \cdot x_j, j \in \{1, ..., T\}\}$: $\mathbf{T} * (\mathbf{d^2})$
6. Computing attention-weighted representation for each token, $\{a_{ij} \cdot v_j, i, j \in \{1, ..., T\}\}$: $\mathbf{T^2} * \mathbf{d}$

This results in the total number of FLOPs being:

$$C_{self} = Td^2 + Td^2 + T^2d + T * C_{softmax}(T) + Td^2 + T^2d \tag{8}$$
$$= 2T^2d + 3Td^2 + TC_{softmax}(T) \tag{9}$$

For multi-head attention, all Q, K, and V projections (1, 2, and 5) become k separate multiplications requiring $\frac{T}{k}d^2$ operations. Similarly, for each head, computing similarity (3) and the attention weighted representations (6) require $\frac{T^2}{k}d$ operations. Furthermore, each softmax is conducted over k vectors of size T, requiring $k * T * C_{softmax}(T)$ operations (4). So, multi-head attention requires

$$C_{MHA} = k\frac{T}{h}d^2 + k\frac{T}{k}d^2 + k\frac{T^2}{k}d + kTC_{softmax}(T) + k\frac{T}{k}d^2 + k\frac{T^2}{k}d \tag{10}$$
$$= 2T^2d + 3Td^2 + TkC_{softmax}(T) \tag{11}$$

Therefore, the only difference between single and multi-headed attention is a factor of k applied to the cost of the softmax function, and there is little significant added cost in terms of total operations to applying the softmax function.

Week 6

Problem 3 (0 points) *Topics: Transformer, Encoder-Decoder Architecture*

You have implemented a transformer from scratch, but the course staff tells you that there is an issue with your implementation. After discussing, you figure out that it is something to do with the architecture of your model (shown below in Figure 1).

a) What is wrong with this implementation of the transformer? (Note: the question continues on the next page and will give away the answer. If you want the practice, try to think of it yourself first!)

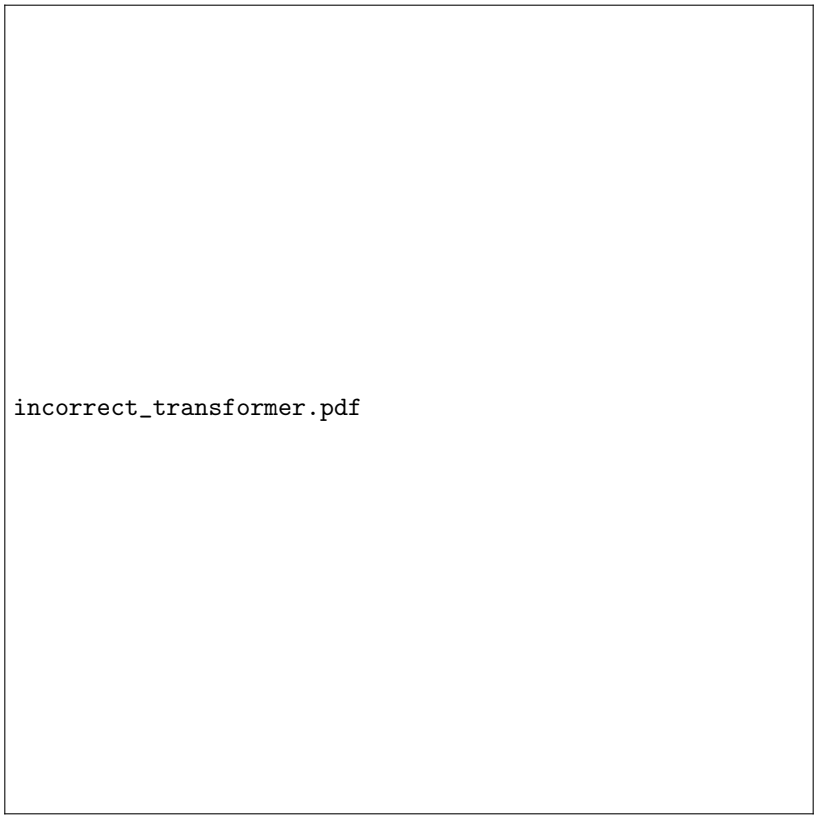


Figure 1: Architecture diagram of an incorrectly implemented transformer.

Solution:

The masked attention in the decoder comes after the cross-attention. As we will see in the rest of the question, this is problematic for both training and inference.

b) You figure out that you have swapped the cross-attention and the masked attention blocks in the decoder. To see why this is an issue, assume a simplified encoder where the encoder output of an input sequence $\mathbf{x} = x_1, x_2, ..., x_{T_e}$ is given by $\theta_e x_i$ for each token. Also assume a simplified decoder ignoring position embeddings and assuming single-head attention. You have the input to the decoder block given by $E_d y_j$ for the target sequence $\mathbf{y} = y_1, y_2, ..., y_{T_d}$. What is the output of the cross-attention block in this incorrectly implemented transformer? Why might this be a problem for the rest of the decoder block to predict the next token?

Solution:

This is a problem because after the cross-attention block of the incorrect transformer, all information about the original context is effectively lost. The residual connections will ensure that later components of the decoder recover this information, but token interactions are now also affected by encoder outputs and the original embeddings are diluted by the encoder outputs. Formally, the output of the cross-attention layer (we will use h) would be:

$$q_i = W_Q E_d y_i, k_j = W_K \theta_e x_j, v_j = W_V \theta_e x_j \tag{12}$$

$$a_{ij} = \text{softmax}(q_i k_j) \tag{13}$$

$$h_{y_t} = \sum_{k=1}^{T_e} a_{ij} v_j \tag{14}$$

Notice that v_j is dependent only on $\mathbf{x}$, while the target sequence context, $\mathbf{y}$, is only used for computing the attention weights. Therefore, the output will strictly be a linear combination of the encoder outputs.

Problem 4 (0 points) *Topics: Attention, entropy*

You have two tokens, w_a and w_b, where $a \neq b$ and $a, b \in [1, ..., T]$. Recall again the single-headed self-attention mechanism given $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$ for all tokens $w_i \in w_{[1,...,T]}$.

$$a_{ij} = \text{softmax}(q_i k_j) \tag{15}$$

$$h_i = \sum_{k=1}^{T} a_{ij} v_j \tag{16}$$

$$\tag{17}$$

a) Assume that the dot products of $\mathbf{q}_a$ with all other tokens' associated keys, $\mathbf{k}_i$, can be treated as a random variable drawn from a Normal distribution, $\mathcal{N}(\mu_a, \sigma_a^2)$. That is:

$$[\mathbf{q}_a K]_i \sim \mathcal{N}(\mu_a, \sigma_a^2) \tag{18}$$

Consider what intuitively you think may happen to the attention vector as σ_a becomes very small or very large? What about as μ_a becomes very small or very large? Finally, what about as d becomes very small or very large? Why?

Solution:

As σ_a becomes becomes smaller, all dot products are more similar and the normalized attention weights should all be similar. So, smaller σ_a leads to a "flatter" attention vector, while larger σ_a is more likely to place more weight on specific tokens. In contrast, there is no change based on μ_a alone, as what matters is the values of each dot product compared to others rather than their magnitude alone. Finally, as d grows larger, the attention distribution will (increasingly slowly) become flatter, as the probability mass will be distributed across more tokens.

b) To formalize this intuition, here is an approximation [1] of the entropy of $softmax(\mathbf{x})$ for $\mathbf{x}_i \sim \mathcal{N}(\mu, \sigma^2), \mathbf{x} \in \mathbb{R}^t$ (each of t elements is drawn independently).

$$H\left[softmax(x)\right] = \ln(t-1) - \frac{\sigma^2}{2} \tag{19}$$

Now assume that $\mathbf{q}_b$ is defined by $\mathbf{q}_b = c \cdot \mathbf{q}_a$. Calculate the derivative of the entropy with respect to this constant c. As the magnitude of the query grows larger (e.g., increasing c), what happens to the entropy? [2]

Solution:

If the dot products of $\mathbf{q}_a$ are normally distributed, they will also be normally distributed for $\mathbf{q}_b$. Because $\mathbf{q}_b$ is c times the magnitude of $\mathbf{q}_a$, however, $\mathbf{q}_b$'s dot products are normally distributed defined by $\mathcal{N}(c\mu, (c\sigma)^2)$. Therefore, using the approximate entropy of the softmax function, the entropy of the attention weights for w_b will be

$$H[\alpha_b] = ln(T-1) - \frac{(c\sigma)^2}{2} \tag{20}$$

$$\tag{21}$$

The derivative with respect to c is:

$$\frac{dH}{dc} = -c\sigma^2 \tag{22}$$

$$\tag{23}$$

So, the entropy generally decreases faster with increasing c based on our approximate entropy formula.

c) Finally, consider a third token, $\mathbf{q}_c$ which is a scaled version of $\mathbf{q}_a$ with added noise. Specifically, $\mathbf{q_c} \approx c * \mathbf{q}_a$ such that the dot products have an added noise ϵ where $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ and σ_ϵ is sufficiently small. Compute the entropy of the attention weights associated with $\mathbf{q}_c$. As c grows large, how sensitive is the entropy of attention to noise (e.g., σ_ϵ)? [3]

[1] For those interested, this is based on the second order Taylor expansion.

[2] It may be helpful for part b and c to research operations on normal distributions and compositions of normal distributions.

[3] This is roughly a lower bound on gauging the sensitivity of attention in general to noise.

Solution:

Because the noise is independent of the distribution of dot products, the dot products associated with $\mathbf{q}_c$ are now normally distributed under $\mathcal{N}(\mu_a, c^2\sigma_a^2+\sigma_\epsilon^2)$. So, the entropy of the attention weights are now:

$$H[\alpha_b] = ln(T-1) - \frac{(c^2\sigma^2 + \sigma_\epsilon^2)}{2} \tag{24}$$

$$\tag{25}$$

The sensitivity (derivative) of the entropy with respect to σ_ϵ is then

$$\frac{dH}{d\sigma_e} = -\sigma_\epsilon \tag{26}$$

$$\tag{27}$$

Therefore, the sensitivity of the entropy to noise is not dependent on the magnitude of the query vector.

Week 7

Problem 5 (0 points) *Topics: Truncation sampling and temperature scaling*

You have a language model with a vocabulary of 100 tokens, and you look at the outputted probability distribution with varying hyperparameters temperature, $\mathcal{T}$, and top-p.

a) In particular, assume that the temperature is $\mathcal{T} \in \{0.5, 0.75\}$ and the top-p value is also $p \in \{0.5, 0.75\}$. Figure 2 shows the resulting distributions with each combination of these hyperparameter values relative to the default distribution ("dist-base" in black). First, what combinations of $\mathcal{T}$ and p are likely represented by each of dist-0 to dist-3? What features of the distributions lead you to these answers?

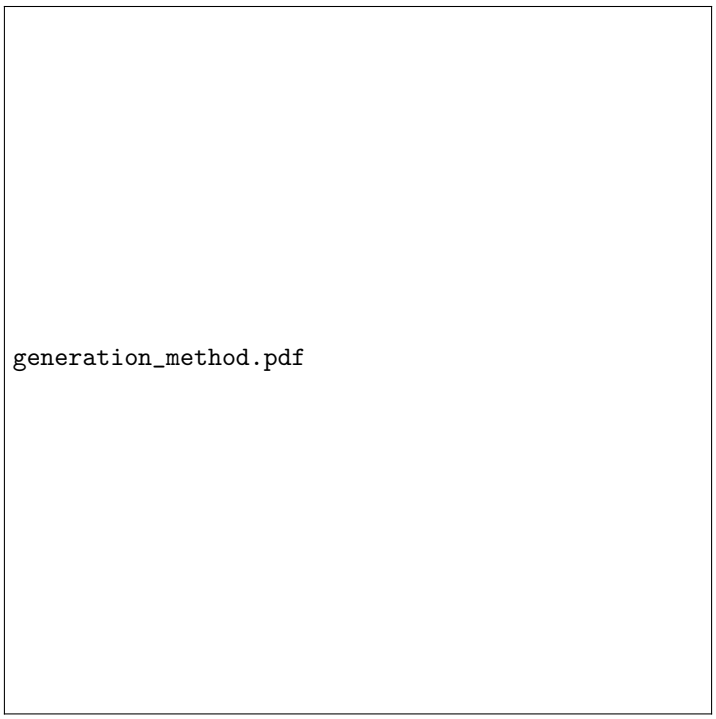


Figure 2: Probability distribution variants with different temperature and top-p values.

Solution:

The settings for each distribution are as follows:

- **dist-0:** $\mathcal{T} = 0.75, p = 0.5$
- **dist-1:** $\mathcal{T} = 0.5, p = 0.5$
- **dist-2:** $\mathcal{T} = 0.75, p = 0.75$
- **dist-3:** $\mathcal{T} = 0.5, p = 0.75$

By observation, dist-1 should correspond to the lowest temperature and top-p because it has the highest peak and considers the fewest tokens. Similarly, dist-2 should correspond to the highest for both values as it is the flattest and considers the most tokens. dist-0 and dist-3 are harder to distinguish. However, you could consider that dist-3 appears to have more probability mass over tokens than dist-0 and additionally, dist-3 appears to be more peaked than dist-0. Therefore, dist-3 should have the higher top-p and lower temperature compared to dist-0. For top-p, you should primarily focus on the probability mass of tokens considered or how many tokens are considered overall. For temperature, you should instead focus on the shape of the distribution, and whether it is flat or peaked.

b) More formally, consider the following formulation of temperature scaling:

$$p_{\mathcal{T},\theta}(w \mid x, y_{<t}) = \text{softmax(z)} \tag{28}$$

$$p_{\mathcal{T},\theta}(w \mid x, y_{<t}) = \frac{p_\theta(w \mid x, y_{<t})^{1/\mathcal{T}}}{\sum_{w \in \mathcal{V}} p_\theta(w \mid x, y_{<t})^{1/\mathcal{T}}} \tag{29}$$

Where z represents the unnormalized output of the model corresponding to token w. You then apply top-p truncation to the resulting distribution, renormalizing based on the accepted tokens. Given two tokens in the vocabulary, w_a and w_b, compute the ratio $\frac{p_{\mathcal{T},\theta}(w_a|x,y_{<t})}{p_{\mathcal{T},\theta}(w_b|x,y_{<t})}$ in terms of z_a and z_b. How does this ratio change with respect to temperature? How does it change with respect to p?

Solution:

The ratio can be derived as:

$$\frac{p_{\mathcal{T},\theta}(w_a \mid x, y_{<t})}{p_{\mathcal{T},\theta}(w_b \mid x, y_{<t})} = \frac{p_\theta(w_a \mid x, y_{<t})^{1/\mathcal{T}}}{\sum_{w \in \mathcal{V}} p_\theta(w \mid x, y_{<t})^{1/\mathcal{T}}} \Big/ \frac{p_\theta(w_b \mid x, y_{<t})^{1/\mathcal{T}}}{\sum_{w \in \mathcal{V}} p_\theta(w \mid x, y_{<t})^{1/\mathcal{T}}} \tag{30}$$

$$= \frac{p_\theta(w_a \mid x, y_{<t})^{1/\mathcal{T}}}{p_\theta(w_b \mid x, y_{<t})^{1/\mathcal{T}}} \tag{31}$$

$$= \frac{(e^{z_a})^{1/\mathcal{T}}}{(e^{z_b})^{1/\mathcal{T}}} \tag{32}$$

$$= e^{\frac{z_a - z_b}{\mathcal{T}}} \tag{33}$$

Then, either by observation or taking the derivative of this ratio with respect to $\mathcal{T}$, we can see that temperature scales the difference in model outputs between tokens. Specifically, extremely large or increasing temperature brings the ratio closer to 1, flattening the distribution or reducing the difference in probability between tokens.

When normalizing for top-p truncation, the denominator for both tokens will be the same. Therefore, the ratio will only be changed if either token is not included in the resulting set of accepted tokens. A smaller p will generally result in one or both tokens being more likely to be excluded, but assuming that both are in the accepted set, top-p truncation does not impact the ratio of probabilities between tokens..

Problem 6 (0 points) *Topics: Pretraining/Finetuning*

You have a very large language model, $p_{\theta_L}(w \mid w_<t)$, but you realize that this is very expensive to run inference with. So, you have the great idea of training a smaller model on the outputs of a larger model (this is known as *model distillation*) on a smaller dataset.

a) With your smaller model initialized randomly, $p_{\theta_S}(w \mid w_{<t})$, you define the following loss objective for a given sequence of length T:

$$L = \sum_{i=1}^{T} \mathrm{KL}\left[p_{\theta_L}(w_t \mid w_{<t}) \| p_{\theta_S}(w_t \mid w_{<t})\right] \tag{34}$$

$$\tag{35}$$

Interpret what this loss objective is doing (i.e., explain how the KL divergence encourages the behavior we are looking for). What is the primary weakness of this loss objective? (*Hint: think about the assumptions being made in order for this to work well and whether they are realistic.*)

Solution:

The current loss objective updates the weights of the smaller model to match the outputted next-token *distributions* of the larger model. Importantly, it isn't just training the smaller model to make the same predictions, but attempts to match the overall distribution. In particular, this objective assumes that the teacher model has very few flaws and models language well enough to serve as ground truth. This weakness could be stated in many ways, such as the potential of overfitting on the teacher model, not letting the small model learn any information about real language data, a lack of diversity in training.

b) To address the problem above, you update the loss objective by adding a term that considers only the weights of the small model and the ground truth pretraining data:

$$L = (1-\alpha)\sum_{t=1}^{T} -\log(p_{\theta_S}(y_t \mid w_{<t})) + \alpha \sum_{t=1}^{T} \mathrm{KL}\left[p_{\theta_L}(w_t \mid w_{<t}) \| p_{\theta_S}(w_t \mid w_{<t})\right] \tag{36}$$

$$\tag{37}$$

where α is a scalar controlling how much weight is given to matching the larger model vs. the ground truth data. After doing so, you notice that the outputs of your large model are very confident (i.e., most probability mass is placed on a single token) because the larger model has already been trained on the data used for distillation. After training, what are the student models predictions likely to look like on unseen data (this doesn't need to be a mathematical argument)? Why might this still be a problem or a weakness?

Solution:

There are a couple of things that could be said about this. One possible answer is that the highly confident predictions of the larger model may reduce *diversity* during training. This is because KL Divergence is scaled by the probability of the reference distribution (the distribution output by the large model) and so tokens with low probabilities will contribute little to the overall loss. Therefore, the loss will encourage the student model to place high probability on the top-prediction of the larger model, particularly in cases where the larger model matches the ground truth data, but place little weight on the other tokens in the distribution. These other tokens carry richer information about what the large model has learned, but it has little impact if the large models distribution is highly peaked.

c) You remember from lecture ways to control the probability distributions of language models and think one of those methods may work to help solve this problem and encourage more diversity during training. Explain how the issue you identify above may be resolved. *Challenge:* Modify the loss objective to incorporate this change.

Solution:

From the lecture on generation, recall that *temperature scaling* is one way to modify the overall shape of a distribution and can be used to encourage more diverse sampled generations. In this case, you can do the same to the large models outputs to encourage the smaller model to learn more information about the tail of the distribution.
Specifically, the loss objective would instead look like the following:

$$L = (1-\alpha)\sum_{t=1}^{T} -\log(p_{\theta_S}(y_t \mid w_{<t})) \tag{38}$$

$$+ \alpha\mathcal{T}^2 \sum_{t=1}^{T} \mathrm{KL}\left[p_{\mathcal{T},\theta_L}(w_t \mid w_{<t}) \| p_{\mathcal{T},\theta_S}(w_t \mid w_{<t})\right] \tag{39}$$

where (40)

$$p_{\mathcal{T},\theta}(w_t \mid w_{<t}) = \frac{e^{z_{w_t}/\mathcal{T}}}{\sum_{w_j \in \mathcal{V}} e^{z_{w_j}/\mathcal{T}}} \tag{41}$$

After applying temperature scaling to the model outputs, a constant factor (specifically $\mathcal{T}^2$) is needed to rescale the KL divergence component of the loss. Importantly, both the small and large model probability distributions are temperature scaled, which ensures that the small model is able to learn from the flatter/softened distribution of the teacher, but does not actually learn to predict the flatter distribution. Scaling the larger model distribution alone would lead to a less certain smaller model.