高级人工智能课程汇报

信息科学与工程学院
Gu Rui
220220942871

June 25 2023

Contents

1 Attention exploration (22 points)

Multi-headed self-attention is the core modeling component of Transformers. In this question, we’ ll get some practice working with the self-attention equations, and motivate why multi-headed self-attention can be preferable to single-headed self-attention. Recall that attention can be viewed as an operation on a *query* $q \in \mathbb{R}^d$, a set of *value* vectors $\{v_1, ..., v_n\}$, $v_i \in \mathbb{R}^d$, and a set of *key* vectors $\{k_1, ..., k_n\}$, $k_i \in \mathbb{R}^d$, specified as follows:

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

$$\alpha_i = \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp(k_j^T q)} \tag{2}$$

with $_i$ termed the “attention weights”. Observe that the output $c \in \mathbb{R}^d$ is an average over the value vectors weighted with respect to $_i$.

(a) (4 points) **Copying in attention.** One advantage of attention is that it’ s particularly easy to “copy” a value vector to the output c. In this problem, we’ ll motivate why this is the case.

i. (1 point) **Explain** why α can be interpreted as a categorical probability distribution.

Answer: Eq.1 has shown that this is a fuzzy query, and we cannot directly match a key vector k with the query vector q, and can only give each key a certain probability distribution weight (i.e. α_{ij}) to get the final output result.

ii. (2 points) The distribution α 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 α puts almost all of its weight on some α_j , where $j \in \{1, \ldots, n\}$(*i.e.* $\alpha_j \gg \sum_{i \neq j} \alpha_i$). What must be true about the query q and/or the keys $\{k_1, \ldots, k_n\}$?

Answer: According to the calculation method of Eq.2, if the query vector q has a very high similarity to a key k_i (the dot product is large), and q is basically vertical to other bonds (the point product is zero), then α_i will be maximized.

iii. (1 point) Under the conditions you gave in (ii), **describe** what properties the output c might have.

Answer: Under the conditions described in (ii), the output vector c will be heavily influenced by the value vector v_j associated with the key vector k_j that received the majority of the attention weight. At this point, the c is approximately equal to v_i

iv. (1 point) **Explain** (in two sentences or fewer) what your answer to (ii) and (iii) means intuitively.

Answer: When the dot product (similarity) between a specific word key and a query significantly outweighs the dot products of other word keys with the same query, the attention output corresponding to that specific word will closely resemble its associated value. This behavior can be likened to "copying" the value into the output.

(b) (7 points) **An average of two**. Instead of focusing on just one vector v_j , a Transformer model might want to incorporate information from *multiple* source vectors. Consider the case where we instead want to incorporate information from **two** vectors v_a and v_b, with corresponding key vectors k_a and k_b.

i. (3 points) How should we combine two d-dimensional vectors v_a, v_b into one output vector c in a way that preserves information from both vectors? In machine learning, one common way to do so is to take the average: $c = \frac{1}{2}(v_a + v_b)$. It might seem hard to extract information about the original vectors va and v_b from the resulting c, but under certain conditions one can do so. In this problem, we' ll see why this is the case.

Suppose that although we don' t know v_a or v_b, we do know that v_a lies in a subspace A formed by the m basis vectors $\{a_1, a_2, \ldots, a_m\}$, while v_b lies in a subspace B formed by the p basis vectors $\{b_1, b_2, \ldots, b_p\}$. (This means that any va can be expressed as a linear combination of its basis vectors, as can v_b. All basis vectors have norm 1 and orthogonal to each other.)

Additionally, suppose that the two subspaces are orthogonal; i.e. $a_j^\top b_k = 0$ for all j, k. Using the basis vectors $\{a_1, a_2, \ldots, a_m\}$, construct a matrix M such that for arbitrary vectors $v_a \in A$ and $v_b \in B$, we can use M to extract v_a from the sum vector $s = v_a + v_b$. In other words, we want to construct M such that for any v_a, v_b, $M_s = v_a$.

Note: both M and v_a, v_b should be expressed as a vector in $\mathbb{R}^d$, not in terms of vectors from A and B.

Hint: Given that the vectors $\{a_1, a_2, \ldots, a_m\}$ are both *orthogonal* and *form a basis* for v_a, we know that there exist some $c_1, c_2, \ldots, c_m$ such that $v_a = c_1a_1 + c_2a_2 + \ldots + c_ma_m$. Can you create a vector of these weights c?

Answer: Assume that A is a matrix of concatenated basis vectors $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$ and B is a matrix of concatenated basis vector $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_p\}$. Linear combinations of vectors v_a and v_b can then be expressed as:

$$\mathbf{v}_a = c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \ldots + c_m\mathbf{a}_m = \sum_{i=1}^{m} c_i\mathbf{a}_i = A\mathbf{c}$$

$$\mathbf{v}_b = d_1\mathbf{b}_1 + d_2\mathbf{b}_2 + \ldots + d_p\mathbf{b}_m = \sum_{j=1}^{p} d_j\mathbf{b}_j = B\mathbf{d}$$

We need to construct such M which, when multiplied with $\mathbf{v}_b$, produces $\mathbf{0}$ and, when multiplied with $\mathbf{v}_a$, produces the same vector (in terms of its own space). Let M have the following form:

$$M = \sum_{i=1}^{m} \lambda_i\mathbf{a}_i\mathbf{a}_i^\top$$

Where $\lambda_i, i = 1, \dots, m$ is the undetermined coefficient, it is derived as follows

$$\begin{aligned} M\mathbf{s} = \mathbf{v}_a \Leftrightarrow & \, M\mathbf{v}_a + M\mathbf{v}_b = \mathbf{v}_a \\ & \Leftrightarrow \left(\sum_{i=1}^{m} \lambda_i \mathbf{a}_i \mathbf{a}_i^\top\right) \left(\sum_{i=1}^{m} c_i \mathbf{a}_i + \sum_{j=1}^{p} d_i \mathbf{b}_i\right) = \sum_{i=1}^{m} c_i \mathbf{a}_i \\ & \Leftrightarrow \sum_{i=1}^{m} \lambda_i c_i \mathbf{a}_i \mathbf{a}_i^\top \mathbf{a}_i = \sum_{i=1}^{m} c_i \mathbf{a}_i \quad \text{(orthogonal property)} \\ & \Leftrightarrow \sum_{i=1}^{m} (\lambda_i c_i \mathbf{a}_i^\top \mathbf{a}_i) \mathbf{a}_i = \sum_{i=1}^{m} c_i \mathbf{a}_i \\ & \Rightarrow \lambda_i c_i \mathbf{a}_i^\top \mathbf{a}_i = c_i \\ & \Rightarrow \lambda_i = \frac{1}{\mathbf{a}_i^\top \mathbf{a}_i}, i = 1, \dots, m. \end{aligned}$$

It is easy to see that, since $\mathbf{a}_j^\top b_k = 0$ for all j, k, $A^\top B = 0$. And we know that in terms of $\mathbb{R}^d$ (not in terms of A and B), $\mathbf{v}_a$ is just a collection of constants c. Thus the results of M are as follows

$$M = \sum_{i=1}^{m} \frac{\mathbf{a}_i \mathbf{a}_i^\top}{\mathbf{a}_i^\top \mathbf{a}_i} = A^\top$$

ii. (4 points) As before, let v_a and v_b be two value vectors corresponding to key vectors k_a and k_b, respectively. Assume that (1) all key vectors are orthogonal, so $k_i^\top k_j$ for all $i \neq j$; and (2) all key vectors have norm 1.[1] **Find an expression** for a query vector q such that $c \approx \frac{1}{2}(v_a + v_b)$.[2]

Answer: Assume that c is approximated as follows:

$$c \approx \frac{1}{2}\mathbf{v}_a + \frac{1}{2}\mathbf{v}_b$$

This means we want $\alpha_a \approx 0.5$ and $\alpha_b \approx 0.5$, which can be achieved when (whenever $i \neq a$ and $i \neq b$):

$$\mathbf{k}_a^\top q \approx \mathbf{k}_b^\top q \gg \mathbf{k}_i^\top \mathbf{q}$$

Like explained in the previous question, if the dot product is big, the probability mass will also be big and we want a balanced mass between α_a and α_b. q will be largest for

[1] Recall that a vector x has norm 1 if $x^\top x = 1$.

[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.

k_a and k_b when it is a large multiplicative of a vector that contains a component in k_a direction and in k_b direction:

$$\mathbf{q} = \beta(\mathbf{k}_a + \mathbf{k}_b), \quad \text{where} \beta \gg 0$$

Now, since the keys are orthogonal to each other, it is easy to see that:

$$\mathbf{k}_a^\top q = \beta; \mathbf{k}_b^\top q = \beta; \mathbf{k}_i^\top q = 0, \quad \text{whever } i \neq a \text{ and } i \neq b$$

Thus when we exponentiate, only $\exp(\beta)$ will matter, because $\exp(0)$ will be insignificant to the probability mass. We get that:

$$\alpha_a = \alpha_b = \frac{\exp(\beta)}{n - 2 + 2\exp(\beta)} \approx \frac{\exp(\beta)}{2\exp(\beta)} \approx \frac{1}{2}, \text{ for } \beta \gg 0$$

(c) (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. Consider a set of key vectors $\{k_1, \ldots, k_n\}$ that are now randomly sampled, $k_i \sim \mathcal{N}(_i, \Sigma_i)$, where the means $_i \in \mathbb{R}^d$ are known to you, but the covariances Σ_i are unknown. Further, assume that the means $_i$ are all perpendicular; $\mu_i^\top \mu_j = 0$ if $i \neq j$, and unit norm, $\|\mu_i\| = 1$.

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

ii. (3 points) Though single-headed attention is resistant to small perturbations in the keys, some types of larger perturbations may pose a bigger issue. Specifically, 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 $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$ for vanishingly small α (as shown in Fig. 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 $\{k_1, \ldots, k_n\}$ multiple times, and use the q vector that you defined in part i., what qualitatively do you expect the vector c will look like for different samples?

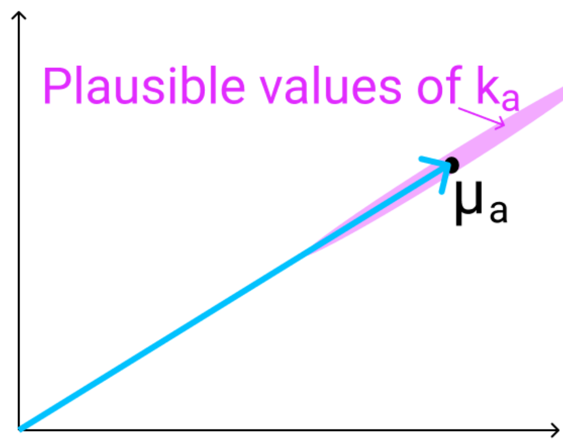

Figure 1: The vector μ_a (shown here in 2D as an example), with the range of possible values of k_a shown in red. As mentioned previously, k_a points in roughly the same direction as μ_a, but may have larger or smaller magnitude.

(d) (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 we' ve presented it in this homework, except two query vectors (q_1 and q_2) are defined, which leads to a pair of vectors (c_1 and c_2), each the output of single-headed attention given its respective query vector. The final output of the multi-headed attention is their average,$\frac{1}{2}(c_1 + c_2)$. As in question 1(c), consider a set of key vectors $\{k_1, \ldots, k_n\}$ that are randomly sampled, $k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, where the means μ_i are known to you, but the covariances Σ_i are unknown. Also as before, assume that the means μi are mutually orthogonal; $\mu_i^\top \mu_j = 0$ if $i \neq j$, and unit norm,$\|\mu_i\| = 1$.

i. (1 point) Assume that the covariance matrices are $\Sigma_i = \alpha I$, for vanishingly small α. Design q_1 and q_2 such that c is approximately equal to $\frac{1}{2}(v_a + v_b)$.

ii. (2 points) Assume that the covariance matrices are $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$ for vanishingly small α, and $\Sigma_i = \alpha I$ for all $i \neq a$. Take the query vectors q_1 and q_2 that you designed in part i.

    What, qualitatively, do you expect the output c to look like across different samples of the key vectors? Please briefly explain why. You can ignore cases in which $k_a^\top q_i < 0$.

2 Pretrained Transformer models and knowledge access (35 points)

You' ll train a Transformer to perform a task that involves accessing knowledge about the world – knowledge which isn' t provided via the task' s training data (at least if you want to generalize outside the training set). You' ll find that it more or less fails entirely at the task. You' ll then learn how to pretrain that Transformer on Wikipedia text that contains world knowledge, and find that finetuning that Transformer on the same knowledge-intensive task enables the model to access some of the knowledge learned at pretraining time. You' ll find that this enables models to perform considerably above chance on a held out development set.

The code you' re provided with is a fork of Andrej Karpathy' s minGPT. It' s nicer than most research code in that it' s relatively simple and transparent. The "GPT" in minGPT refers to the Transformer language model of OpenAI, originally described in this paper [1]. As in previous assignments, you will want to develop on your machine locally, then run training on HuaWei Could. You' ll need around 5 hours for training, so budget your time accordingly!

References

[1] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding with unsupervised learning. Technical report, OpenAI (2018).

[2] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research 21, 140 (2020), 1–67.

[3] Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models. arXiv preprint arXiv:2005.00743 (2020).