Practical No. 4

Adriana Bukała
ab394064@students.mimuw.pl

April 30, 2022

1 Part 1

2 Part 2

a)

Output c would be approximately equal to v_j, if $a_j \approx 1$ and $a_i \approx 0$ for all $i \neq j$, i.e. if $k_j^T q >> k_i^T q$ for $i \neq j$.

b)

Output c would be an average on value vectors v_a, v_b, if following requirements were satisfied:

1. $a_a \approx a_b \approx \frac{1}{2}$
2. $a_i \approx 0$ for $i \notin \{a, b\}$

Similarly to the previous example, that requires $k_b^T q \approx k_a^T q >> k_i^T q$. Setting query to

$$q = m(k_a + k_b), m >> 0$$

would result in $k_a^T q = k_b^T q = m$ and $k_i^T q = 0$, what would lead to c being an average of v_a and v_b.

c)

Question 1

Based on the assumption that covariance matrices are $\sum_i = \alpha I$ for a vanishingly small α, i.e. tokens are barely correlated, it is expected that $k_i \approx \mu_i$. Another assumption is that all μ_i are orthogonal and of unit norm, which makes this problem similar to the one in subsection b). Basing on previous solution, query should be set to

$$q = m(\mu_a + \mu_b), m >> 0$$

Question 2

For all $k_i, i \neq a$ it is still expected to $k_i \approx \mu_i$. However $k_a \sim \mathcal{N}(\mu_a, \alpha I + \frac{1}{2}(\mu_a \mu_a^T))$. Since α is still negligibly small, k_a could be approximated by $k_a \approx \epsilon \mu_a$ for $\epsilon \sim \mathcal{N}(1, \frac{1}{2})$. Sampling $\{k_1, ..., k_n\}$ multiple times would result in $k_i^T q \approx 0$ for all $i \neq \{a, b\}$, as discussed in previous question. That leads to

$$c \approx v_a a_a + v_b a_b$$

$$a_a \approx \frac{exp(\epsilon m)}{exp(\epsilon m) + exp(m)} = \frac{exp(\epsilon m)}{exp(\epsilon m)(1 + exp((1-\epsilon)m))} = \frac{1}{1 + exp((1-\epsilon)m)}$$

$$a_b \approx \frac{exp(m)}{exp(\epsilon m) + exp(m)} = \frac{exp(m)}{exp(m)(exp((\epsilon - 1)m) + 1)} = \frac{1}{1 + exp((\epsilon - 1)m)}$$

Since $\epsilon \sim \mathcal{N}(1, \frac{1}{2})$, its values would mostly oscillate between 0 and 2, what gives two cases:

1. when $\epsilon \to 0$ then $a_a \approx 0, c \to v_b$
2. when $\epsilon \to 2$ then $a_b \approx 0, c \to v_a$

d)

Question 1

Based on previous observations and solutions:

$$q_1 = q_2 = m(\mu_a + \mu_b), m >> 0$$

Question 2

$$c = \frac{1}{2}(c_1 + c_2)$$

$$c_1 \approx v_a a_a + v_b a_b = \frac{1}{1 + exp((1-\epsilon)m)} v_a + \frac{1}{1 + exp((\epsilon - 1)m)} v_b$$

$$c_2 \approx v_a a_a + v_b a_b = \frac{1}{1 + exp((1-\epsilon)m)} v_a + \frac{1}{1 + exp((\epsilon - 1)m)} v_b$$

Similarly to the previous subsection, values of c would oscillate between v_a and v_b for different samples. However, since the number of attention heads was increased to 2, variation of these oscillations would be reduced, which is a benefit of multi-head attention. With $n > 2$ attention heads, ϵ values would approach their expected value (1), what would result in $c \approx \frac{1}{2}(v_a + v_b)$ as desired in this example.

e)

Question 1

$$c_2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 =$$

$$\frac{exp(x_1^T x_2)x_1 + exp(x_2^T x_2)x_2 + exp(x_3^T x_2)x_3}{exp(x_1^T x_2) + exp(x_2^T x_2) + exp(x_3^T x_2)} =$$

$$\frac{exp((u_d + u_b)^T u_a)(u_d + u_b) + exp(u_a^T u_a)u_a + exp((u_c + u_b)^T u_a)(u_c + u_b)}{exp((u_d + u_b)^T u_a) + exp(u_a^T u_a) + exp((u_c + u_b)^T u_a)} =$$

$$\frac{exp(0)(u_d + u_b) + exp(u_a^T u_a)u_a + exp(0)(u_c + u_b)}{exp(0) + exp(u_a^T u_a) + exp(0)} =$$

$$\frac{u_d + u_b + exp(\beta^2)u_a + u_c + u_b}{1 + exp(\beta^2) + 1} \approx \frac{exp(\beta^2)u_a}{exp(\beta^2)}$$

$$c_2 \approx u_a$$

It is impossible for c_2 to approximate u_b by adding u_b or u_d to x_2. For example if u_b would be added to x_2:

$$c_2 = \frac{exp(0)(u_d + u_b) + exp((u_a + u_b)^T(u_a + u_b))(u_a + u_b) + exp(0)(u_c + u_b)}{exp(0) + exp((u_a + u_b)^T(u_a + u_b)) + exp(0)} \approx$$

$$\frac{exp((u_a + u_b)^T(u_a + u_b))(u_a + u_b)}{exp((u_a + u_b)^T(u_a + u_b))} =$$

$$\frac{exp(\beta^2 + 0 + 0 + \beta^2)(u_a + u_b)}{exp(\beta^2 + 0 + 0 + \beta^2)} =$$

$$\frac{exp(2\beta^2)(u_a + u_b)}{exp(2\beta^2)} =$$

$$u_a + u_b$$

Similarly, if u_d would be added to x_2, then c_2 would approximate $u_a + u_d$.

Question 2

$$V := \frac{u_b u_b^T - u_c u_c^T}{\beta^2}$$

$$v_1 = V(u_d + u_b) = \frac{0 - 0 + u_b u_b^T u_b - 0}{\beta^2} = \frac{u_b \beta^2}{\beta^2} = u_b$$

$$v_2 = V u_a = 0$$

$$v_3 = V(u_c + u_b) = \frac{0 - u_c u_c^T u_c + u_b u_b^T u_b - 0}{\beta^2} = \frac{u_b \beta^T - u_c \beta^T}{\beta^2} = u_b - u_c$$

Now output vectors of interest would be:

$$c_1 = a_{11} v_1 + a_{12} v_2 + a_{13} v_3 = a_{11} u_b + a_{13}(u_b - u_c)$$

$$c_2 = a_{21} v_1 + a_{22} v_2 + a_{23} v_3 = a_{21} u_b + a_{23}(u_b - u_c)$$

To have $c_1 \approx u_b - u_c$ it is needed that $a_{13} \approx 1$ and $a_{11} \approx 0$, i.e. $k_3^T q_1 >> k_1^T q_1$ (a_{12} is not significant, since $v_2 = 0$). While for $c_2 \approx u_b$ it is needed that $a_{21} \approx 1$ and $a_{23} \approx 0$, i.e. $k_1^T q_2 >> k_3^T q_2$.

$$K := I$$

$$Q := u_d u_a^T + u_c u_d^T$$

$$k_3^T q_1 = (u_c + u_b)^T (u_d u_a^T + u_c u_d^T)(u_d + u_b) = u_c^T u_c u_d^T (u_d + u_b) = u_c^T u_c u_d^T u_d = \beta^4$$

$$k_1^T q_1 = (u_d + u_b)^T (u_d u_a^T + u_c u_d^T)(u_d + u_b) = u_d^T u_d u_a^T (u_d + u_b) = 0$$

$$k_1^T q_2 = (u_d + u_b)^T (u_d u_a^T + u_c u_d^T) u_a = (u_d^T u_d u_a^T) u_a = u_d^T u_d u_a^T u_a = \beta^4$$

$$k_3^T q_2 = (u_c + u_b)^T (u_d u_a^T + u_c u_d^T) u_a = (u_c^T u_c u_d^T) u_a = 0$$

This setup meets all aforementioned requirements.

3 Part 3

d)

DEV SET: correct: 6.0 out of 500.0: 1.2%
LONDON BASELINE: correct: 25.0 out of 500.0: 5.0%

f)

DEV SET: correct: 94.0 out of 500.0: 18.8%

g)

Question 1

Linformer is based on the observation that self-attention is approximately a low-rank matrix. Authors utilized k-dimensional projections of key and value matrices obtained with singular value decomposition algorithm, where $k << n$. This operation decreases both time and computation complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(nk)$, although it was shown in Theorem 2. that when $k = \mathcal{O}(\frac{d}{\epsilon^2})$, where d is a hidden dimension of key and value projection, it is possible to compute self-attention in linear time with ϵ error.

Question 2

BigBird is based on the graph sparsification problem, i.e. authors achieved linear complexity of self-attention by reducing number of attending tokens. BigBird has three building blocks of attention mechanism: 1) global attention, i.e. a set of g global tokens, which attend on all parts of sequence; 2) random attention, i.e. a set of r random tokens, to which all tokens attend; 3) window attention, i.e. a set of w local neighboring tokens, to which all tokens attend. This approach created an attention mechanism, which scales to much longer sequence lengths (even 8x longer) than standard transformers such as BERT.

Question 3

Firstly, authors prove Turing completeness of sparse encoder-decoder transformers. Secondly, authors prove that sparse attention mechanism used in a standalone encoders, such as BERT, are universal approximators of sequence to sequence functions. Finally, authors empirically showed that BigBird is as powerful as standard transformers by benchmarking BigBird against other models and achieving comparable results.

*h)

Question 1

Performers utilize mechanism called *Fast Attention Via positive Orthogonal Random features* (FAVOR+) and new methods for softmax and Gaussian kernels approximation proposed by the authors. Performers achieve nearly linear time and sub-quadratic memory consumption.

Question 2

Trigometric softmax approximation could lead to highly unstable results (i.e. unstable learning curve) preventing training or leading to sub-optimal models, or even NaN values during training. That is a result of applying random feature maps with potentially negative dimension values

Question 3

Authors utilize positive orthogonal random features instead of trigometric functions. Positive random features are more stable for low kernel values, which caused stability problems for trigonometric approximations.

Question 4

Sampling orthogonal random features (ORFs) reduces variance of both softmax and Gaussian kernel estimators, meaning that fewer random features could be used to achieve comparable results, while decreasing computational complexity.

4 Part 4

a)

Pretrained model had access to more data, which is a key in deep learning applications, especially in natural language processing. Even though model was not asked to predict birth places at the time of pretraining, it is highly likely that it learnt some patterns. Non-pretrained model almost always predicted Berlin or Brooklyn.

b)

Such indeterminacy of the model could:

1. cause interpretability problems. One of major concerns of deep learning applications for healthcare are problems with explainability of predictions or model itself. However, it becomes even more challenging with model's indeterminacy and lack of training data (which is often a case for sensitive data such as medical records),

2. cause bias and/or stereotype. Getting a well-structured, realistic answer, aligning with one's prior beliefs about the question, could only strengthen their biased or stereotypical views.

c)

Even though model did not see specific name, it could have seen a similar one. More specifically, it could have seen similar name in its latent space, so model's strategy would be to predict birth place of the most similar name seen at pretraining or fine-tuning time.