The scaling factor in self-attention

William Chuang

October 4, 2023

1 Introduction

In the context of deep learning neural networks, attention algorithms are designed for determining connections by weights between two elements in a sequential inputs. Self-attention was proposed by Vaswani et. al.[2] in 2017. Suppose a sequential input $X \in \mathbb{R}^N \times \mathbb{R}^{d_x}$ is given. Denote $Q^t := (XW_q)^t \in \mathbb{R}^{d_q} \times \mathbb{R}^N, K^t := (XW_k)^t \in \mathbb{R}^{d_k} \times \mathbb{R}^N$, and $V^t := (XW_v)^t \in \mathbb{R}^{d_v} \times \mathbb{R}^N$. The notations for denoting column vectors of each these three matrices are: $q_i^t \in \mathbb{R}^{d_q}$, $k_i^t \in \mathbb{R}^{d_k}$, and $q_i^v \in \mathbb{R}^{d_v}$. The j-th component of each column vector is denoted as $q_{ij}^t \in \mathbb{R}$, $k_{ij}^t \in \mathbb{R}$, and $q_{ij}^v \in \mathbb{R}$.

Now, in a standard implementation of a transformer-based model or any models that have self-attention mechanism implemented in it, W_k, W_v, and W_q are initiated in a way to satisfy the following four conditions:

(i) $W_q \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_q}$, $W_k \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_k}$, and $W_v \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_v}$,

(ii) the variance of the inner product of each column vector in X^t with each column vector in either W_q, W_k, or W_v to be unit (all these inner products results in entries matrices: Q^t, K^t, and V^t),

(iii) each $q_i^t = (q_{i1}, ..., q_{id_q})^t$ and each $k_i^t = (k_{i1}, ..., k_{id_k})^t$ to be random variables with zero means, i.e. $\mathbb{E}\left(q_i^t\right) = \mathbb{E}\left(\{q_{ij}\}_{j=1}^{d_q}\right) = 0 = \mathbb{E}\left(\{k_{ij}\}_{j=1}^{d_k}\right) = \mathbb{E}\left(k_i^t\right), \forall i \in \{1, ..., N\}$, and unit variances, i.e $\mathrm{Var}\left(q_i^t\right) = \mathrm{Var}\left(\{q_{ij}\}_{j=1}^{d_q}\right) = 1 = \mathrm{Var}\left(\{k_{ij}\}_{j=1}^{d_k}\right) = \mathrm{Var}\left(k_i^t\right), \forall i \in \{1, ..., N\}$, and

(iv) $q_{i_1}^t$ and $k_{i_2}^t$ are all independent in the following ways: $\mathrm{Cov}\left(q_{i_1}^t, k_{i_2}^t\right) = 0, \forall i_1, i_2$, $\mathrm{Cov}\left(q_{i_1}^t, q_{i_2}^t\right) = 0, \forall i_1 \neq i_2,$, $\mathrm{Cov}\left(k_{i_1}^t, k_{i_2}^t\right) = 0, \forall i_1 \neq i_2$.

Furthermore, by convention, let $d_k = d_q$. Then, with the four above conditions satisfied, the any scalar products of column vectors in Q^t and K have variance d_k (see Appendix).

Definition 1. *The Softmax function is defined using the Boltzmann distribution function: Given a sequential inputs (i.e. consider the finite sequence as a vector):* $\{z_i\}, z_j \in \mathbb{R}, \forall j \in \{1, ..., n\}$*, then*

$$Softmax(z_i) := \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

Furthermore, this distribution can be scaled by using a parameter β, and β is the inverse of the temperature of the physical system described by the distribution:

$$\text{Softmax}(\beta z_i) := \frac{\exp(\beta z_i)}{\sum_j \exp(\beta z_j)}.$$

In order to take inputs that are tensors, for each component of a tensor $a_{i_1 i_2 \ldots i_n}$, the softmax is operated on its last dimension i_n, and also output a tensor which has indices $i_1 \ldots i_n$. For example, for a $N \times d$ tensor (i.e. a matrix), the softmax is taking on its last dimension, i.e. which equals d. The output is also an $N \times d$ matrix, say it is denoted by $\text{softmax}(a_{i_1 i_2 \ldots i_n})_{N \times d}$. Then, each entry $\text{softmax}(a_{i_1 i_2})_{ij}$ of the matrix $\text{softmax}(a_{i_1 i_2})_{N \times d}$ is defined as

$$\text{softmax}(a_{i_1 i_2})_{ij} := \frac{\exp(\beta a_{ij})}{\sum_{i_1=1}^{N} \exp(\beta a_{i_1 j})}.$$

Operating on the last dimension means to fix the last index in the normalization. In this example that means to consider each column vector in $a_{i_1 i_2}$ as a set of data or a physical system that Boltzmann distribution was applied to, and the result $\text{softmax}(a_{i_1 i_2})_{ij}$ is the probability of the configuration a_{ij}. The denominator is also called the normalization factor or the partition function that it sums all possible configurations using to given data $a_{i_1 i_2}$ where $i_2 = j$ to be the input of the exponential function $\exp(\beta s), s \in \mathbb{R}$.

Furthermore, for a fixed j, the probability $\text{softmax}(a_{i_1 i_2})_{ij}$ can be consider as the interaction strength between the i-th and j-th particles.

In the following example, the self-attention mechanism, the i-th particle would be the column vector q_i^t in the matrix Q^t, and the j-th particle would be the column vector k_j^t in the matrix K^t.

Definition 2 (Self-attention). *With the notations Q, K and V, defined as above, the self-attention mechanism is the following function*

$$\mathit{Attention}(Q, K, V) := \mathit{softmax}\left(\beta\left(QK^T\right)\right)V,$$

where $\beta := \frac{1}{\sqrt{d_k}} \in \mathbb{R}$.

In self-attention, this β is $\frac{1}{\sqrt{d_k}}$, and T is the standard deviation of the scalar product $\sqrt{d_k}$.

By definition of softmax, apply softmax to the matrix $\frac{QK^T}{\sqrt{d_k}}$ means to apply Softmax function on each column of the matrix $\left(\frac{QK^T}{\sqrt{d_k}}\right)_{N\times N}$, and for defining the self-attention function, take a matrix multiplication between softmax $\left(\frac{QK^T}{\sqrt{d_k}}\right)_{N\times N}$ and the value matrix $V_{N\times d_v}$.

The question that asked and answered by this paper is the following.

Because of the random initiation of each weight matrices, if this self-attention, i.e. weights assigning process were run in parallel, then the model can learn from more than one different perspectives. In [2], this notion is defined as multi-head attention. Each self-attention function $\mathrm{Attention}\,(Q, K, V)_i$ is called head_i, where i is from 1 to H. The output of all H heads are concatenated to a tensor denoted by $\mathrm{Concat}\,(\mathrm{head}_1, ..., \mathrm{head}_H)$, then defined Multihead(Q,K,V) := $\mathrm{Concat}\,(\mathrm{head}_1, ..., \mathrm{head}_H)\, W^O$, where $W^O \in \mathbb{R}^{H\cdot d_v} \times \mathbb{R}^{d_{\mathrm{out}}}$.

Let $\epsilon > 0$ be given. Because of the mathematical form of the Boltzmann distribution, if the input is $z_i + \epsilon$, then $\mathrm{Softmax}(\beta z_i + \epsilon) = \mathrm{Softmax}(\beta z_i)$. However, a naive but natural question to ask is what if the scaling factor β in Boltzmann distribution, or the self-attention function, is modified into the form β^ϵ or $\beta \cdot \epsilon$? It is necessary to ask this question, since we want to be able to justify why and when it makes sense to use the scaling factor $\beta = \frac{1}{\sqrt{d_k}}$ in the self-attention function. Is it sufficient to ask this question? That is the question this paper is going to answer. The next section provides a solution to this question.

2 Algorithm

To be able to make sense to use the scaling factor $\beta = \frac{1}{\sqrt{d_k}}$ in the self-attention function, the four assumptions are necessary.

Within the four assumptions, the first assumption is usually fixed when self-attention is considered to be applied.

Since the way query and key matrices are defined, i.e. $Q = XW_q$ and $K = XW_k$, assumption (iii) and (iv) are based on assumption (ii).

Suppose the assumptions (ii) is relaxed, the variance of the scalar product $q_i k_i$ is approximated to $\left(d_x \sigma_X^2 \sigma_{W_q}^2\right) \cdot \left(d_x \sigma_X^2 \sigma_{W_k}^2\right) \cdot d_k$. Since the training set is given, i.e. it is fixed, and so are d_x and d_k, σ_{W_q} and σ_{W_k} are the actual moving factors.

What can cause σ_{W_q} and σ_{W_k} to be changed? In the training phase, the gradient descent algorithm or the backpropagation is used and the algorithm is going to shift the initial probability distribution of each entry in the random matrices W_q and W_k to the final probability distribution at a local minimum in the moduli space (or on the loss surface).

Hence, the scaling factor $\beta = \frac{1}{\sqrt{d_k}}$ is only valid at the initial point given the four assumptions at the initial point in the moduli space, and it can hold true if the thermodynamic system described by the Boltzmann distribution is in statics, i.e. its temperature is fixed. However, since the initial point is random and unlikely to be the final point in the moduli space throughout the training mode, the assumption (ii) is not valid right after the training of the model that used self-attention function is initiated. Without assumption (ii), assumption (iii) and (iv) are invalid throughout the training and predicting modes.

To solve this problem, the solution could be written as an algorithm that it measures σ_{W_q} and σ_{W_k} and updates the scaling factor β in after each iteration or batch during the training and the best scaling factor β that is going to be used in the predicting mode could be determined by the best model chosen from the ensemble of all trained models.

This also answers the question asked in the previous section that it shows the sufficiency to ask the question.

In practice, 709 is the upper bound of the exponent of the exponential function based e in lots of machines before the machines start to overflow. Likewise, on each machine, there exists a non-negative integer N such that the machine starts to overflow the result. Hence, an alternative algorithm is: firstly try $\beta \in \left\{10^N, \frac{1}{10^N}\right\}$ for $N = 0, 1, 2, ...,$ then run a binary search to narrow down the range of the best scaling factor β during the training after either each iteration or batch.

A further meaning of this algorithm is shown in the next section by using an empirical example.

3 Empirical example

The task of the empirical example is to flip a given string of non-negative integers. In other words, the goal is to train a multihead in a transformer to learn a function that maps each index i of an integer in a string to the index $(n-i)+1$ in the new string that will be returned by the model. Since the model needs to learn to swap the first and last elements in the string, it could be a challenging task for recurrent neural nets if the length of the input string is large.

In this empirical example, when the string length is 20, the range of the non-negative integers is between 0 and 100, and the number of samples used in training is 15000, 1000 samples for validation, and 100000 samples for testing.

Then, without running the above alternative algorithm, i.e. when the scaling factor β is $\frac{1}{\sqrt{d_k}}$ the validation accuracy is 1.51%, and the test accuracy is 1.50%. To have a stable test accuracy above 95%, the training sample size needs to be increased from 15000 to 25000.

However, by applying the above alternative algorithm, the scaling factor is in $O(1)$, a constant ~ 5, with training sample size at 15000, the validation accuracy and test accuracy can already reach above 95% stably. Thus, the above algorithms can not only compress the training set when the training samples are limited or rare, but also reduce the training time to reach the best accuracy.

Furthermore, the model that achieved the above accuracy did not achieve it by memorizing the training examples. Since a 100% accuracy is also achievable within the ensemble, and once achieved that 100% accuracy the model can flip any input string with the same length and the range of the values of its each word. However, if it was by memorizing, then the model had to memorize 100^{20} strings with length 20 which greatly exceed the number of parameters of the model (less than 8000).

4 Discussion

5 Conclusion

6 Appendix: Deriving the scaling factor $\frac{1}{\sqrt{d_k}}$ used in [2]

Let X and Y be random variables.

Firstly, recall the definition of variance[1]:

Definition 3.

$$\sigma^2(X) := \inf_{a \in \mathbb{R}} \mathbb{E}\left[(X-a)^2\right].$$

Hence if $X \notin L^2$, then $\mathrm{Var}[X] = \infty$. If $X \in L^2$, then $\mathbb{E}\left[(X-a)^2\right] = \mathbb{E}(X^2) - 2a\mathbb{E}(X) + a^2$ has a minimum at $a = \mathbb{E}(X)$, and the expression of the variance of X can be rewritten using the minimum:

$$\mathrm{Var}[X] = \mathbb{E}\left[(X-a)^2\right] = \mathbb{E}\left[(X-\mathbb{E}[X])^2\right].$$

Proposition 1.

$$Var[X] = \mathbb{E}(X^2) + (\mathbb{E}[X])^2 - 2(\mathbb{E}[x])^2 = \mathbb{E}(X^2) - (\mathbb{E}X)^2.$$

Proof.

$$\mathrm{Var}[X] = \mathbb{E}\left[X^2 + (\mathbb{E}X)^2 - 2X\mathbb{E}X\right]$$

$$= \mathbb{E}(X^2) + (\mathbb{E}[X])^2 - 2(\mathbb{E}[x])^2 = \mathbb{E}(X^2) - (\mathbb{E}X)^2.$$

□

Secondly,

Definition 4. *the covariance of X and Y is:*

$$Cov[X,Y] := \mathbb{E}\left[(X-\mathbb{E}X)(Y-\mathbb{E}Y)\right].$$

Proposition 2.

$$Cov[X,Y] = \mathbb{E}XY - \mathbb{E}X\mathbb{E}Y.$$

Proof.

$$\mathrm{Cov}[X,Y] = \mathbb{E}\left[XY - X\mathbb{E}Y - Y\mathbb{E}X + \mathbb{E}X\mathbb{E}Y\right]$$

$$= \mathbb{E}XY - \mathbb{E}X\mathbb{E}Y - \mathbb{E}X\mathbb{E}Y + \mathbb{E}X\mathbb{E}Y$$

$$= \mathbb{E}XY - \mathbb{E}X\mathbb{E}Y.$$

□

Then, the expectation value of X^2Y^2 can be written in the following expression:

$$\mathbb{E}X^2Y^2 = \mathrm{Cov}[X^2,Y^2] + \mathbb{E}X^2\mathbb{E}Y^2$$

$$= \mathrm{Cov}[X^2,Y^2] + (\mathrm{Var}X + (\mathbb{E}X)^2)(\mathrm{Var}Y + (\mathbb{E}Y)^2).$$

Proposition 3. *If X and Y are independent and both have zero means, then*

$$Var[XY] = \ VarX\, VarY.$$

Proof. By applying the above results, the variance of the product of X and Y can be derived:

$$\mathrm{Var}[XY] = \mathbb{E}X^2Y^2 - (\mathbb{E}XY)^2$$

$$= \mathbb{E}X^2Y^2 - (\mathbb{E}Y\mathbb{E}X + \mathrm{Cov}[X,Y])^2$$

$$= \mathrm{Cov}[X^2,Y^2] + (\mathrm{Var}X + (\mathbb{E}X)^2)(\mathrm{Var}Y + (\mathbb{E}Y)^2) - (\mathbb{E}Y\mathbb{E}X + \mathrm{Cov}[X,Y])^2.$$

Since X and Y are independent, then $\mathrm{Cov}[X^2,Y^2] = 0$, and

$$\mathrm{Var}[XY] = (\mathrm{Var}X + (\mathbb{E}X)^2)(\mathrm{Var}Y + (\mathbb{E}Y)^2) - (\mathbb{E}Y\mathbb{E}X + \mathrm{Cov}[X,Y])^2.$$

Since X and Y are independent and both have zero means, i.e. $\mathbb{E}X = 0 = \mathbb{E}Y$, then

$$\mathrm{Var}[XY] = \mathrm{Var}X\mathrm{Var}Y.$$

$\square$

Proposition 4. *Let $X_i, i \in \{1, ..., n\}$ be n independent variables.*

$$Var\left(\sum_i X_i\right) = \sum_i VarX_i.$$

Proof. Then

$$\begin{aligned}
\mathrm{Var}\left(\sum_i X_i\right) &= \mathbb{E}\left(\left(\sum_i X_i\right)^2\right) - \left(\mathbb{E}\sum X_i\right)^2 \\
&= \mathbb{E}\left(\sum_i \sum_j X_i X_j\right) - \left(\sum_i \mathbb{E}X_i\right)^2 \\
&= \sum_i \sum_j \left(\mathbb{E}X_i X_j - \mathbb{E}X_i \mathbb{E}X_j\right) \\
&= \sum_i \sum_j \mathrm{Cov}\left(X_i, X_j\right).
\end{aligned}$$

Since X_i are independent, then $\mathrm{Cov}\left(X_i, X_j\right) = 0, \forall i \neq j$, and

$$\begin{aligned}
\mathrm{Var}\left(\sum_i X_i\right) &= \sum_i \mathrm{Cov}\left(X_i, X_i\right) \\
&= \sum_i \mathrm{Var}X_i.
\end{aligned}$$

$\square$

Then we can derive the assumption in [2] for the variance of QK^T written in the hidden dimension d_k:

Proposition 5. *Let q_i and k_i be random variables with zero means, i.e. $\mathbb{E}q_i = 0 = \mathbb{E}k_i, \forall i$, and unit variances, i.e $Varq_i = 1 = Vark_i$, where $i \in \{1, ..., d_k\}$, and they are all independent in the* *following ways:*

$$Cov\,(q_i, k_i) = 0, \forall i,$$

$$Cov\,(q_i, q_j) = 0, \forall i \neq j, \;\; and$$

$$Cov\,(k_i, k_j) = 0, \forall i \neq j.$$

Then

$$Var\left(QK^T\right) = d_k.$$

Proof.

$$\begin{aligned}
&\mathrm{Var}\left(QK^T\right) \\
&= \mathrm{Var}\left(\sum_{i=1}^{d_k} q_i k_i\right) \\
&= \sum_i \mathrm{Var}\left(q_i k_i\right) \\
&= \sum_i \mathrm{Var}\left(q_i\right) \mathrm{Var}\left(k_i\right) \\
&= \sum_i 1 \cdot 1 \\
&= d_k.
\end{aligned}$$

□

The first equality is because in [2], Q and K are row vectors, and QK^T is taking their scalar product. The second equality is because $\mathrm{Cov}\left(q_i, q_j\right) = 0, \forall i \neq j$, and $\mathrm{Cov}\left(k_i, k_j\right) = 0, \forall i \neq j$. Then, the variance of the sum is the sum of variances. The third equality is because of the conditions $\mathbb{E}q_i = 0 = \mathbb{E}k_i, \forall i$ and $\mathrm{Cov}\left(q_i, k_i\right) = 0, \forall i$, hence we have the variance of the product is the product of variances.

The authors in [2] want to scale the result of QK^T using the standard deviation of the scalar product, so QK^T is multiplied by a factor

$$\frac{1}{\sqrt{\mathrm{Var}\left(QK^T\right)}} = \frac{1}{\sqrt{d_k}}.$$

References

[1] Gerald B Folland, *Real analysis: modern techniques and their applications*, vol. 40, John Wiley & Sons, 1999.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems **30** (2017).