Mathematics of the Attention Layer

Sirvan Almasi

April 2024

Many explanations of the attention layer in the transformer-based networks jump into the variables, such as Q, K, and V but do not develop a strong intuition of why do we need them from first principle. This is my intuition for it. Some details are skipped such as positional encoding and multi-head attention to develop the intuition for the necessary parts.

We start with a sequence of tokens, represented by X. You might imagine token to be a word. Each token x_i with a dimension of d_{model}. This x_i is known as the embedding vector and the dimension is also known as the hidden size or the embedding size.

$$X = [x_1, x_2, x_3, \cdots, x_n] \tag{1}$$

The first thing we want to do with this sequence is to capture some *relationship* between the tokens. Or to learn some contextual information across the tokens. Imagine you have a matrix as shown below, where the matrix value is some score indicating high probability of relationship between them. We will call this matrix C for now:

$$C = \begin{array}{c} \\ x_1^T \\ x_2^T \\ \vdots \\ x_i^T \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & \cdots & x_i \end{array} \\ \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1i} \\ s_{21} & s_{22} & \cdots & s_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ s_{i1} & s_{i2} & \cdots & s_{ii} \end{bmatrix} \end{array} \tag{2}$$

So, let's capture this contextual information with two variables that can be *learnt* over the course of the training. So, how do we end up with an equation as shown in Eq. 2, C? C has a shape of $\mathbb{R}^{d_{model} \times d_{model}}$. Firstly, we need two matrices and each of the matrices need some weight matrix multiplied by the input X (that's where C gets its dimensions from).

$$C = XW^Q(XW^K)^T \tag{3}$$

We have labelled the two weight matrices to differentiate them. We can simplify this by letting $Q = XW^Q$ and $K = XW^K$.

$$C = QK^T \tag{4}$$

Now, we have landed in the first part of the self-attention equation and you might recognise this.

C captures the relationship between the sequence of tokens nicely and we have a way of learning this. What about the *meaning* of the words? The context reveals alot of information about the words. Think about the word 'bank', it could have multiple meanings. How do we capture such meaning? We can create another matrix to store such information. This new matrix will obviously need to have a relationship between the token position as shown in C and the information represented in the new matrix. Let's call this new matrix V, and it will have the form of XW^V. Each row of the V matrix corresponds to a position in the input sequence and contains a dense representation of the information at that position. Now we can combine it with C:

$$QK^TV \tag{5}$$

The beauty of the attention equation is the V and how, using the context, we can start to learn the meaning of the words. And after some time (after millions of dollars) we can start learning the representation of the world just through tokens and words (though a poor representation). The context part of the equation is normalised and ran through a softmax function to help us train the model better.

$$\text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$