Why softmax?

Zhihan Hu

March 2025

Softmax is one of the most widely used scheme to model probability. For example, in LLM, you're predicting the next words. And softmax will be used to model the probability of each vocab in the vocabulary to be the next word. In this document, I will talk about why softmax is so popular compared to other probability modeling scheme.

Suppose we have n classes

$$x_1, ...x_n$$

the probability to predict x_i is shown in the following equation

$$p(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{1}$$

For example, in a multi-class neural network with 5 classes, the output vector is [0.3, 0.7, 1.2, -0.2, 0.7]. After applying softmax to each element, we have [0.14184833, 0.21161285, 0.3488906, 0.08603536, 0.21161285], which is the modeled probability for each label.

When I first learned about softmax, I was wondering, for a long time, why softmax is the most appropriate scheme to model probability. Why not directly use simple normalization, i.e

$$p(x_i) = \frac{x_i}{\sum_{j=1}^{n} x_j} \tag{2}$$

to model the probability. The reasons can be concluded to the following points
1. Avoid negative logits issue:
Suppose we have an output vector [-2, 3, 1, -1, 7]. Then the probability of predicting the first label, if using simple normalization, will be -0.25, which is obviously impossible since probability should always be greater than 0.
2. Lack of exponential scaling:
Suppose we have an output vector [2, 5, 1]. Then the probability of predicting each label, if using simple normalization, will be [0.25, 0.625, 0.125]. But when using softmax, the result will be [0.047, 0.91, 0.043]. We can see a scaling up on the dominant element, and a scaling down on the other elements. With such scaling, the training process will be faster as the true label value does not need

to be dominantly larger than other elements.
3. low complexity in gradient computation:
When using softmax, the gradient computation requires far less complexity to compute gradient. Let's suppose the logits are z = $[z_1, ..., z_n]$, and the output vector y after applying softmax is y = $[y_1, ..., y_n]$. Then, the derivative of

$$\frac{\delta y_i}{\delta z_i} = y_i(1 - y_i) \tag{3}$$

$$\frac{\delta y_i}{\delta z_j} = -y_i y_j \tag{4}$$

Whereas if using simple normalization, we have

$$\frac{\delta y_i}{\delta z_i} = \frac{1}{\sum z_j} - p_i \tag{5}$$

$$\frac{\delta y_i}{\delta z_j} = -\frac{p_i}{\sum z_j} \tag{6}$$

As you can see, a sum over operation is required. And the numerical stability is not good, since $\sum z_j$ might be large.