



Softmax vs LogSoftmax



Abhirami V S · Follow

4 min read · Oct 10, 2021



Listen



Share

softmax is a mathematical function which takes a vector of K real numbers as input and converts it into a probability distribution (generalized form of logistic function, refer figure 1) of K probabilities proportional to the exponential of input numbers, that is, before applying softmax, the vector components can be negative or greater than 0, but after applying softmax, each component will be in the interval $[0,1]$, and the components will add up to 1, so that we can interpret these values as probabilities.

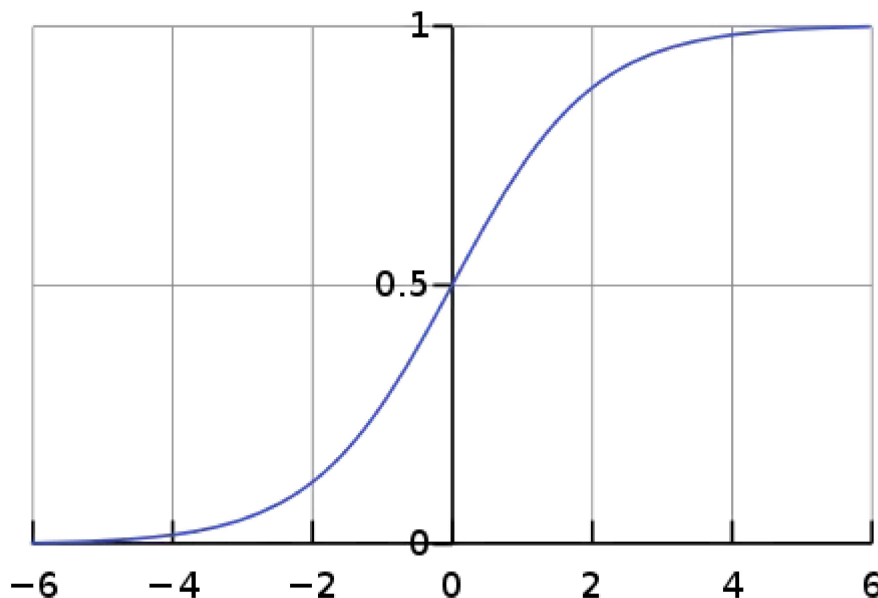


Figure 1

Have you wondered the why the name 'softmax'?

Well, according to 'Neural Networks for Pattern Recognition' book,

“The term softmax is used because this activation function represents a smooth version of the winner-takes-all activation model in which the unit with the largest input has output +1 while all other units have output 0.”

-Page 238.

softmax function is defined by the formula,

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \rightarrow \text{Eq 1}$$

for $i = 1, 2, \dots, K$ and $z = z_1, z_2, \dots, z_K$

Equation 1

Exponential function is applied to each element Z_i of the input vector Z and normalize these values by dividing by the sum of all these exponentials. This normalization ensures that sum of components in output vector is 1.

The most common use of softmax in Machine Learning/ Deep Learning is it's use as an activation function in neural networks. Activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The activation function introduces the non-linearity into the output of a neuron.

In a multi-class classification problem, the network is configured to output N values, one for each class in the classification task. Softmax function is used to normalize the outputs, converting them from weighted some values to probabilities, summing up to 1. Each value in the output of softmax function is the probability representing each class.

The softmax function is a good way of normalizing any value from $[-\infty, +\infty]$ by applying an exponential function. However, it may create issues sometimes as we get large output values for small values of input. For example,

for $x = 19$,

$$e^{19} = 178482300.96$$

For $x = 20$,

$$e^{20} = 485165195.41.$$

As the numbers are too big the exponents will probably blow up (computer cannot handle such big numbers) giving Nan as output. Also, dividing large numbers from Equation 1, can be numerically unstable.

Log softmax is advantageous over softmax for improved numerical performance and gradient optimization.

Log softmax is the log of softmax function, mathematically,

$$\log \left(\frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \right) \rightarrow \text{Eq 2}$$

$$\log \left(\frac{a}{b} \right) = \log a - \log b, \text{ using this on equation 2,}$$

$$\rightarrow z_i - \log \left(\sum_{j=1}^K e^{z_j} \right)$$

$$\approx z_i - \max(z) \text{ since sum is dominated by the largest number.}$$

Equation 2

At the heart of using log-softmax over softmax is the use of log probabilities over probabilities, a log probability is simply a logarithm of a probability. The use of log probabilities means representing probabilities on a logarithmic scale, instead of the standard

unit interval. Since the probabilities of independent events multiply, and logarithms convert multiplication to addition, log probabilities of independent events add. Log probabilities are thus practical for computations.

We can implement log softmax using PyTorch,

```
def log_softmax(x):  
    return x - x.exp().sum(-1).log().unsqueeze(-1)
```

We can directly use log softmax, using nn.LogSoftmax too. Implementation will be shown below.

We are creating a tensor filled with random numbers from a normal distribution with mean 0 and variance 1 as input.

```
In [1]: 1 import torch  
        2 from torch import nn  
        3  
        4 input = torch.randn(2,3)  
        5 input  
  
Out[1]: tensor([[ 0.3452, -0.0267,  0.4066],  
                [ 0.4798, -0.6728,  1.1408]])
```

Softmax and logsoftmax function is applied on this input,

```
In [2]: 1 softmax_fn = nn.Softmax(dim=1)  
        2 output = softmax_fn(input)  
        3 output  
  
Out[2]: tensor([[0.3633, 0.2505, 0.3863],  
                [0.3074, 0.0971, 0.5955]])  
  
In [7]: 1 log_softmax = nn.LogSoftmax(dim=1)  
        2 log_out = log_softmax(input)  
        3 log_out  
  
Out[7]: tensor([[ -1.0126, -1.3845, -0.9512],  
                [ -1.1794, -2.3320, -0.5184]])
```

Both the output values are stored in variables, output and log_out

To make sure that, logsoftmax is the logarithm of softmax function, we can take the exponential of log_out and see whether it is same as output.

```
In [8]: 1 outp = torch.exp(log_out)
        2 outp
Out[8]: tensor([[0.3633, 0.2505, 0.3863],
                [0.3074, 0.0971, 0.5955]])
```

As you can see, output and outp (exponential of logsoftmax output) are same.

Happy Learning!!!

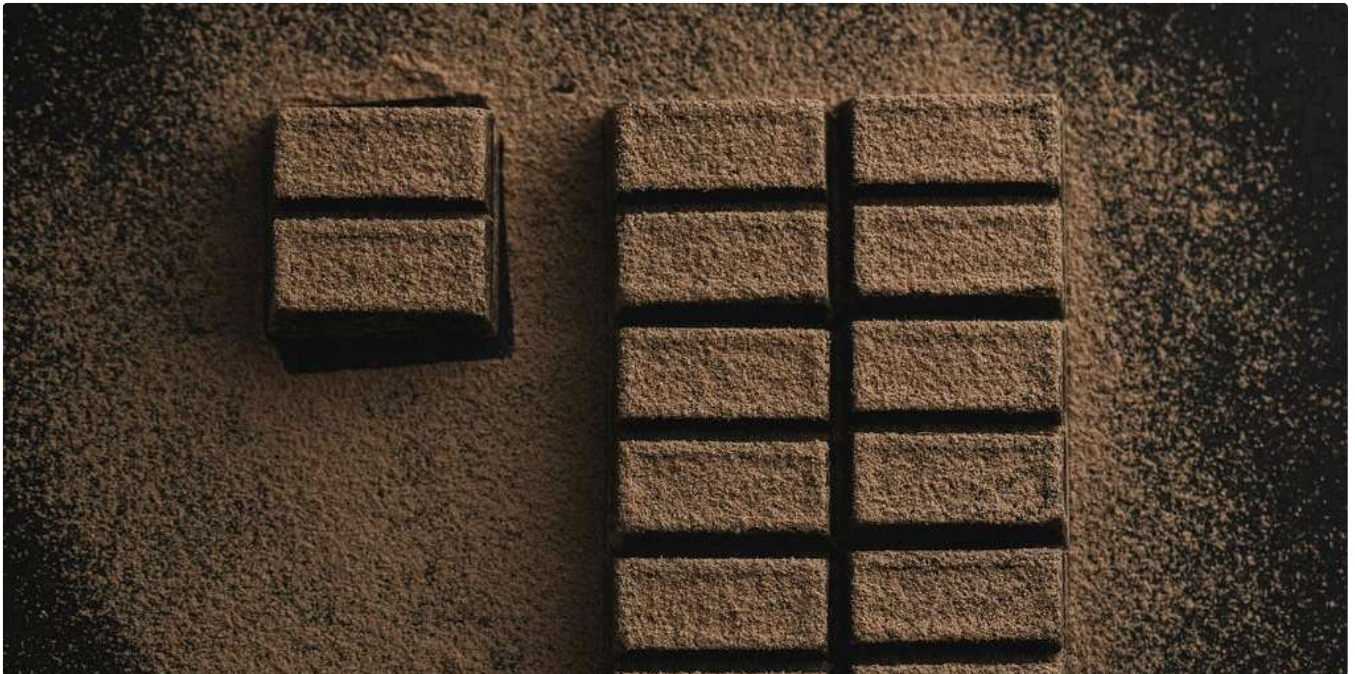
[Softmax](#)[Logsoftmax](#)[Pytorch](#)[Activation Functions](#)[Loss Function](#)[Follow](#)


Written by Abhirami V S

42 Followers

Senior Machine Learning Engineer at Conga with more than 5 years of experience. Connect with me on LinkedIn: <https://www.linkedin.com/in/abhiramivs/>

More from Abhirami V S



 Abhirami V S

Chunking Methods: All to know about it

In this blog, we will discuss about chunking methods to be used in LLM Applications and points to consider while choosing a chunking...

5 min read · Mar 7, 2024




12



1




 Abhirami V S

NumPy Array vs Pandas Series

Have you ever thought Pandas Series is similar to NumPy array?

3 min read · Jan 26, 2023



 Abhirami V S


Bayes theorem with Python!

Probability is a fundamental concept in the realm of statistics and decision-making. It allows us to quantify uncertainty and make...

4 min read · Jul 22, 2023





 Abhirami V S

Build Your Own Question Answering System Using RAG

Using Fully Open Source Models (Mistral7B)

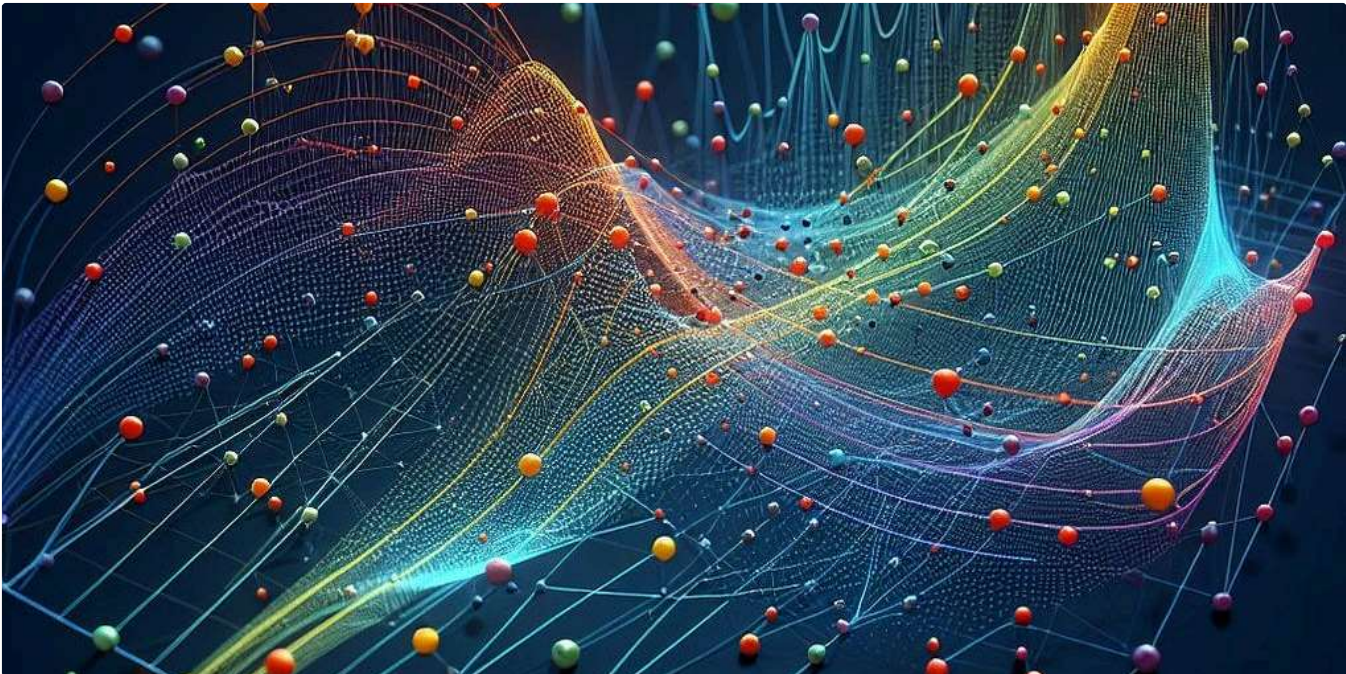
5 min read · Jan 20, 2024

 20 



See all from Abhirami V S

Recommended from Medium



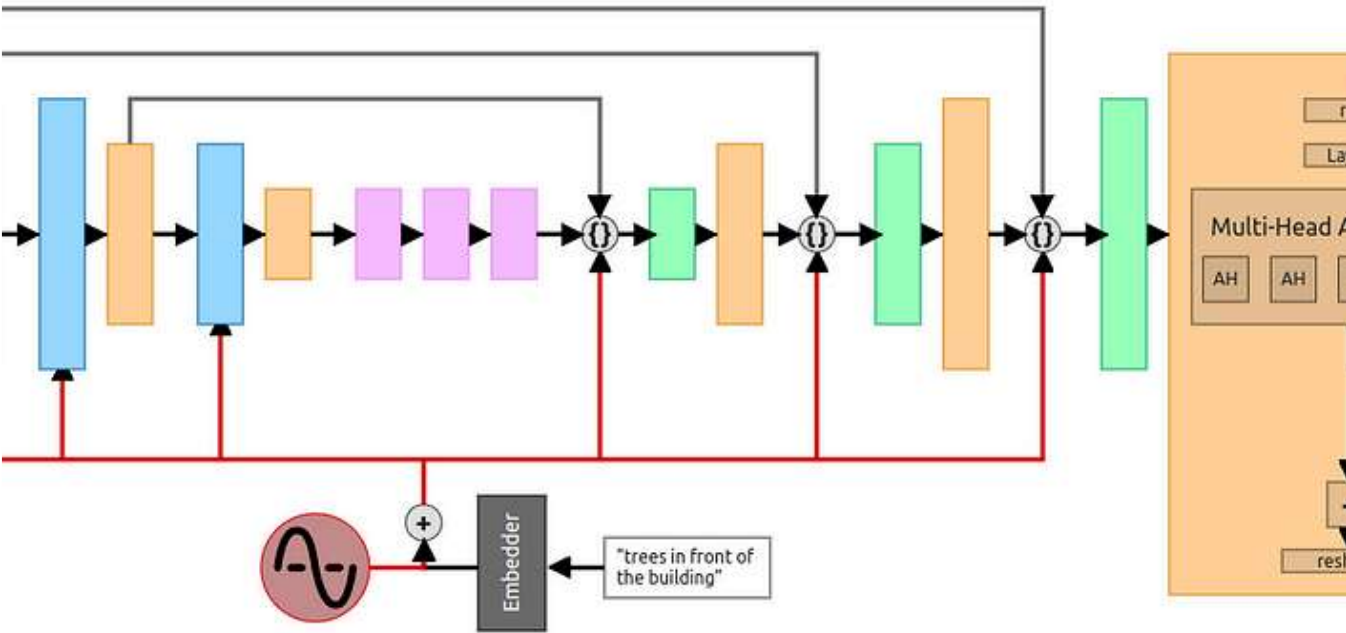
 Tim Sumner in Towards Data Science

A New Coefficient of Correlation

What if you were told there exists a new way to measure the relationship between two variables just like correlation except possibly...

10 min read · Mar 31, 2024

 3.4K  43



 Kemal Erdem (burnpiro)

Step by Step visual introduction to Diffusion Models.

How the diffusion models works under the hood? Visual guide to diffusion process and model architecture.

15 min read · Nov 9, 2023

 333  2



Lists



Natural Language Processing

1431 stories · 926 saves



 Ciaran Bench

Monte Carlo Dropout: a practical guide

12 min read · Nov 12, 2023

 12  1



<u>Index</u>	<u>Interpretation</u>
0	dist between word at position i and $i-4$
1	dist between word at position i and $i-3$
2	dist between word at position i and $i-2$
3	dist between word at position i and $i-1$
4	dist between word at position i and i
5	dist between word at position i and $i+1$
6	dist between word at position i and $i+2$
7	dist between word at position i and $i+3$
8	dist between word at position i and $i+4$



Ngieng Kianyew

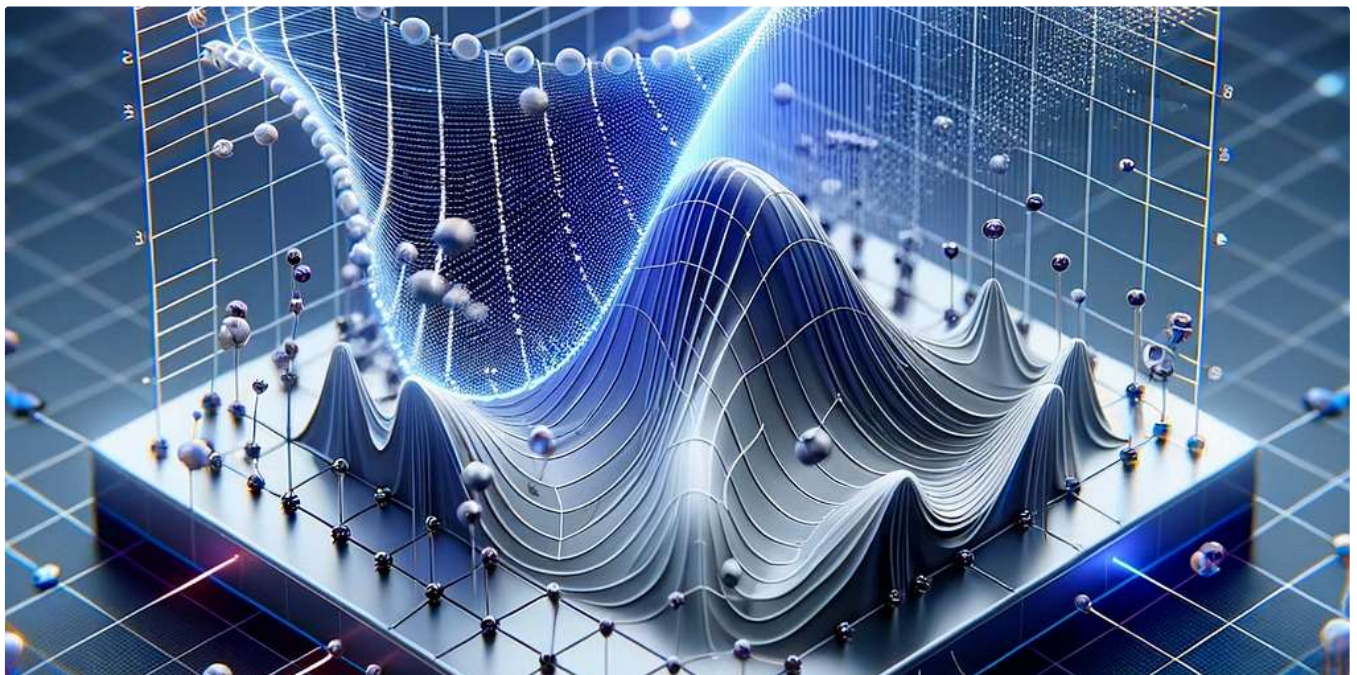
What is Relative Positional Encoding

How does it work and differ from absolute positional encoding?

11 min read • Jan 7, 2024



22



Juan C Olamendy

Understanding ReLU, LeakyReLU, and PReLU: A Comprehensive Guide

Why should you care about ReLU and its variants in neural networks?

6 min read · Dec 4, 2023



105



1



Fahadp

Cracking the Code: Machine Learning for Next Word Predictions

Let Me Predict What Comes Next in Your Text!

10 min read · Dec 15, 2023

[See more recommendations](#)