**What is the Softmax Function?**

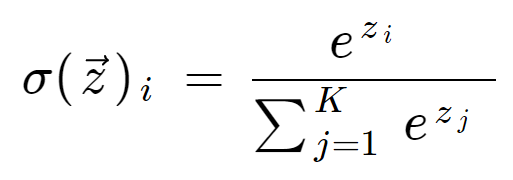
<https://deepai.org/machine-learning-glossary-and-terms/softmax-layer>

The softmax function is a function that turns a [vector](https://deepai.org/machine-learning-glossary-and-terms/vector) of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as [probabilities](https://deepai.org/machine-learning-glossary-and-terms/probability). If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1.

The softmax function is sometimes called the softargmax function, or multi-class [logistic regression](https://deepai.org/machine-learning-glossary-and-terms/logistic-regression). This is because the softmax is a generalization of logistic regression that can be used for multi-class classification, and its formula is very similar to the [sigmoid function](https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function) which is used for logistic regression. The softmax function can be used in a [classifier](https://deepai.org/machine-learning-glossary-and-terms/classifier) only when the classes are mutually exclusive.

Many multi-layer [neural networks](https://deepai.org/machine-learning-glossary-and-terms/neural-network) end in a penultimate layer which outputs real-valued scores that are not conveniently scaled and which may be difficult to work with. Here the softmax is very useful because it converts the scores to a normalized [probability distribution](https://deepai.org/machine-learning-glossary-and-terms/probability-distribution), which can be displayed to a user or used as input to other systems. For this reason it is usual to append a softmax function as the final layer of the neural network.

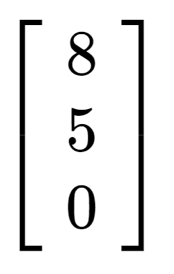
The softmax formula is as follows:



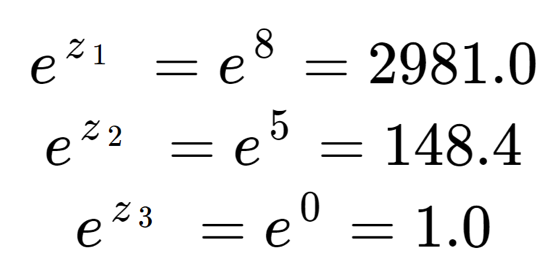
where all the zi values are the elements of the input vector and can take any real value. The term on the bottom of the formula is the normalization term which ensures that all the output values of the function will sum to 1, thus constituting a valid probability distribution.

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/YJ2Ia9on-jmERozwOd9YPRjT_YB8qN30Qqd0xB9sq7WNEdp5GMUsOUqvT3rcRn0bjpehSsXRIbveCGcHITI_gyyBFkbmJtAqUUHDSwNr4rxKk7gGgrMsunKlT7ZSFLnLxlmEC7NZ | The input vector to the softmax function, made up of (z0, ... zK) |
| https://lh3.googleusercontent.com/_FRs4O7QDxh-N-8SUstv6MOMMjCFWe33MqaoQHkGfFJWyR5faP4ZRLIGfOit87vGH-HSlBdlmqApOqjehl97V7GwXuSgfctRYroMJZhmP-RPOvnagRpIrdT_vec_V46rnINyQ_M6 | All the zi values are the elements of the input vector to the softmax function, and they can take any real value, positive, zero or negative. For example a neural network could have output a vector such as (-0.62, 8.12, 2.53), which is not a valid probability distribution, hence why the softmax would be necessary. |
| https://lh6.googleusercontent.com/UgXiQVWpQt3DiYB-MV-sPH77yd0zm-5lv8T_gNeCQnX4mYfCJSn2aJDrhBg-o5gt76y5ba_hmXnTbkEGmvH4HHXAlmrMGAuvvD_p4TZLGKquzO-x1TBwyLHLwxh0xCEZHBbD2oY_ | The standard exponential function is applied to each element of the input vector. This gives a positive value above 0, which will be very small if the input was negative, and very large if the input was large. However, it is still not fixed in the range (0, 1) which is what is required of a probability. |
| https://lh4.googleusercontent.com/7tLf7HZGQf0E7bgIJtueH-mRl3g8Ezw3f1P4tIQPWoWAI8m5QoiMC10J5xep2RFhn6h_94cEOfnA6R71fTAGQBWBLqTOrpbB9NMkJtPc32-WxEeVWdJT_QLSvc43UgrQJPg1-5zm | The term on the bottom of the formula is the normalization term. It ensures that all the output values of the function will sum to 1 and each be in the range (0, 1), thus constituting a valid probability distribution. |
| https://lh4.googleusercontent.com/xJcprBNoJAjAQbF8E2XMf2iKPN5patVakK08hfUVKw0lAkayk7mLn0lWSSpnK0FxfHX--Zm3u694R8RWHKYNqwCNcO0aCbfMAtBecD6rKTJsG9r9l6lFQr2f7MC4SfHQ27IgCY5H | The number of classes in the multi-class classifier. |

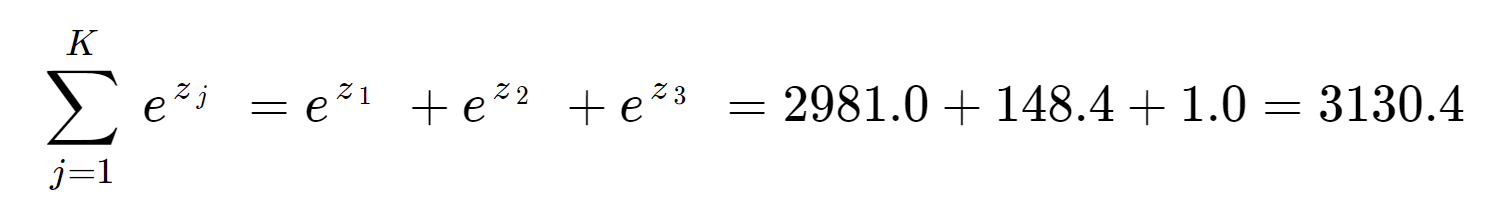
Imagine we have an array of three real values. These values could typically be the output of a [machine learning](https://deepai.org/machine-learning-glossary-and-terms/machine-learning) model such as a neural network. We want to convert the values into a probability distribution.



First we can calculate the exponential of each element of the input array. This is the term in the top half of the softmax equation.

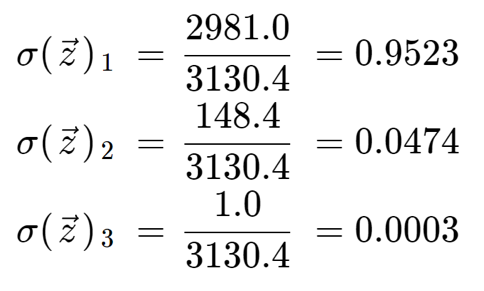


These values do not look like probabilities yet. Note that in the input elements, although 8 is only a little larger than 5, 2981 is much larger than 148 due to the effect of the exponential. We can obtain the normalization term, the bottom half of the softmax equation, by summing all three exponential terms:



We see that the normalization term has been dominated by z1.

Finally, dividing by the normalization term, we obtain the softmax output for each of the three elements. Note that there is not a single output value because the softmax transforms an array to an array of the same length, in this case 3.



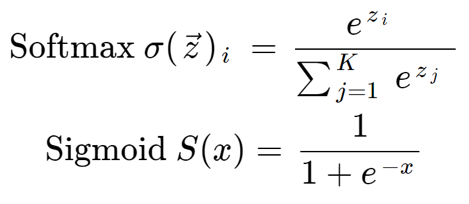
It is informative to check that we have three output values which are all valid probabilities, that is they lie between 0 and 1, and they sum to 1.

Note also that due to the exponential operation, the first element, the 8, has dominated the softmax function and has squeezed out the 5 and 0 into very low probability values.

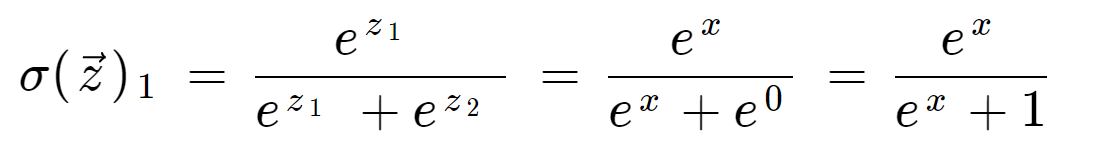
If you use the softmax function in a machine learning model, you should be careful before interpreting it as a true probability, since it has a tendency to produce values very close to 0 or 1. If a neural network had output scores of [8, 5, 0], like in this example, then the softmax function would have assigned 95% probability to the first class, when in reality there could have been more uncertainty in the neural network’s predictions. This could give the impression that the neural network prediction had a high confidence when that was not the case.

## Softmax Function vs Sigmoid Function

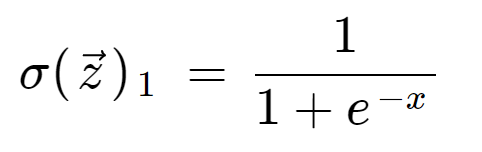
As mentioned above, the softmax function and the sigmoid function are similar. The softmax operates on a vector while the sigmoid takes a scalar.



In fact, the sigmoid function is a special case of the softmax function for a classifier with only two input classes. We can show this if we set the input vector to be [x, 0] and calculate the first output element with the usual softmax formula:



Dividing the top and bottom by ex, we get:

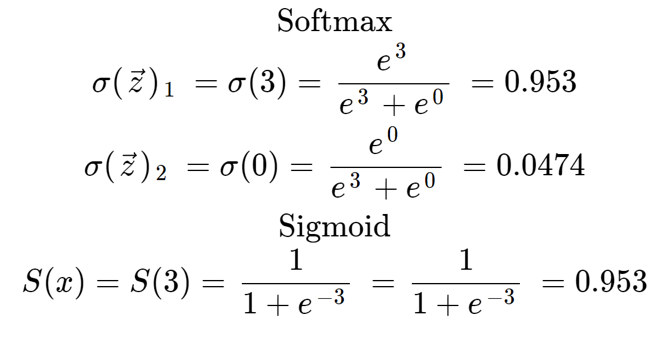


This shows that the sigmoid function becomes equivalent to the softmax function when we have two classes. It is not necessary to calculate the second vector component explicitly because when there are two probabilities, they must sum to 1. So, if we are developing a two-class classifier with logistic regression, we can use the sigmoid function and do not need to work with vectors. But if we have more than two mutually exclusive classes the softmax should be used.

If there are more than two classes and they are not mutually exclusive (a multi-label classifier), then the classifier can be split into multiple binary classifiers, each using its own sigmoid function.

### **Calculating Softmax Function vs Sigmoid Function**

If we take an input vector [3, 0], we can put this into both the softmax and sigmoid functions. Since the sigmoid takes a scalar value we put only the first element into the sigmoid function.



The sigmoid function gives the same value as the softmax for the first element, provided the second input element is set to 0. Since the sigmoid is giving us a probability, and the two probabilities must add to 1, it is not necessary to explicitly calculate a value for the second element.

### **Softmax Function in Neural Networks**

One use of the softmax function would be at the end of a neural network. Let us consider a [convolutional neural network](https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network) which recognizes if an image is a cat or a dog. Note that an image must be either a cat or a dog, and cannot be both, therefore the two classes are mutually exclusive. Typically, the final fully connected layer of this network would produce values like [-7.98, 2.39] which are not normalized and cannot be interpreted as probabilities. If we add a softmax layer to the network, it is possible to translate the numbers into a probability distribution. This means that the output can be displayed to a user, for example the app is 95% sure that this is a cat. It also means that the output can be fed into other machine learning algorithms without needing to be normalized, since it is guaranteed to lie between 0 and 1.

Note that if the network is classifying images into dogs and cats, and is configured to have only two output classes, then it is forced to categorize every image as either dog or cat, even if it is neither. If we need to allow for this possibility, then we must reconfigure the neural network to have a third output for miscellaneous.

## Softmax History

The first known use of the softmax function predates machine learning. The softmax function is in fact borrowed from physics and statistical mechanics, where it is known as the Boltzmann distribution or the Gibbs distribution. It was formulated by the Austrian physicist and philosopher Ludwig Boltzmann in 1868.