

EVENT

Intro to Cascading Retrieval: Boost RAG and search precision by up to 48%

[Learn more >](#)[← Learn](#)

Softmax Activation Function: Everything You Need to Know



Bala Priya C

Jun 30, 2023

Share:

ML Foundations



Jump to section

[Multiclass Classification Revisited](#)[Can You Use Sigmoid or Argmax Activations Instead?](#)[The Softmax Activation Function, Explained](#)[Equivalence of the Sigmoid, Softmax Activations for N = 2](#)[Summing Up](#)[Further Reading](#)**Don't miss the next one...**

Get an email the next time we publish an article about machine learning and similarity search.

[Get Updates](#)



Have you ever trained a neural network to solve the problem of multiclass classification? If yes, you know that the raw outputs of the neural network are often very difficult to interpret. The **softmax activation function** simplifies this for you by making the neural network's outputs easier to interpret!

The softmax activation function transforms the raw outputs of the neural network into a vector of *probabilities*, essentially a probability distribution over the input classes. Consider a multiclass classification problem with N classes. The softmax activation returns an output vector that is N entries long, with the entry at index i corresponding to the probability of a particular input belonging to the class i .

In this tutorial, you'll learn all about the softmax activation function. You'll start by reviewing the basics of multiclass classification, then proceed to understand why you cannot use the sigmoid or argmax activations in the output layer for multiclass classification problems.

Finally, you'll learn the mathematical formulation of the softmax function and implement it in Python.

Let's get started.

Multiclass Classific

Recall that in *binary* classification, a ConvNet trained to classify whet




Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

classifier, whereas, in *multiclass* classification, there are *more than* two possible classes.

Let's consider the following example: You're given a dataset containing images of pandas, seals, and ducks. You'd like to train a neural network to predict whether a previously unseen image is that of a seal, a panda, or a duck.

Notice how the input class labels below are one-hot encoded, and the classes are *mutually exclusive*. In this context, mutual exclusivity means that a given image can only be *one* of {seal, panda, duck} at a time.

Class	Value	One-Hot Encoding
0		$[1, 0, 0]$
1		$[0, 1, 0]$
2		$[0, 0, 1]$

Multiclass Classification Example (Image by the author)

Can You Use Sigmoid or Argmax Activations Instead?

In this section, you'll learn why the sigmoid and argmax functions are not the optimal choices for the output layer in a multiclass classification problem.

Limitations of the Sigmoid

Mathematically, the sigmoid activation function is not suitable for multiclass classification because it squashes all inputs onto the

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{1 + e^z}$$

where, $\sigma(z) \rightarrow 0$ as $z \rightarrow -\infty$
and $\sigma(z) \rightarrow 1$ as $z \rightarrow \infty$

Sigmoid Function Equation (Image by the author)

The sigmoid function takes in any *real* number as the input and maps it to a number between 0 and 1. This is exactly why it's well-suited for binary classification.

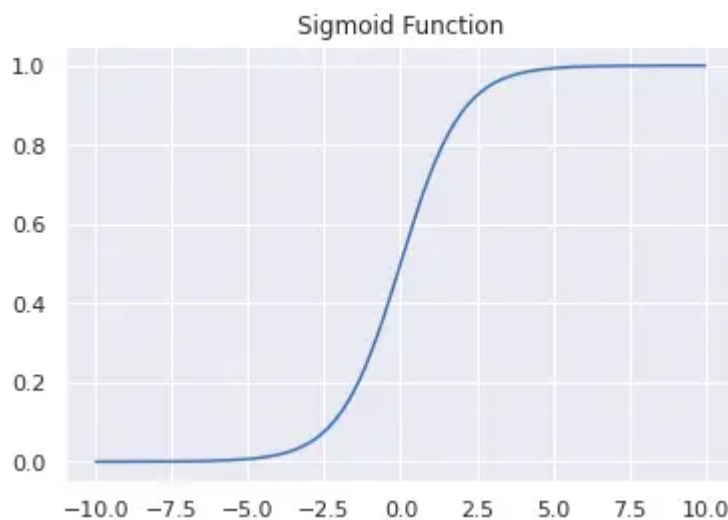
▶ You may run the following code cell to plot the values of the sigmoid function over a range of numbers.

```
1 import numpy as np
2 import seaborn as sns
3
4 def sigmoid(x):
5     exp_x = np.exp(x)
6     return np.divide(exp_x, (1 + exp_x))
7
8 x = np.linspace(-10, 10, num=200)
9 exp_x = np.exp(x)
10 sigmoid_arr = sigmoid(x)
11
12 sns.set_theme()
13 sns.lineplot(x = x, y = sigmoid_arr).set(title='Sigmoid Function')
```



Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.



Plot of the Sigmoid Function

Let's go back to our example of classifying whether an input image is that of a panda or not. In this case, let z be the raw output of the neural network. If $\sigma(z)$ is the probability that the given image belongs to class 1 (is a panda), then $1 - \sigma(z)$ is the probability that the given image does not belong to class 1 and is not a panda. You can think of $\sigma(z)$ as a *probability score*.

You can now fix a threshold, say T , and predict that class whose probability score is *greater* than the chosen threshold.

However, this won't quite work when you have more than two classes. Softmax to the rescue!

In fact, you can think of the softmax function as a **vector generalization** of the sigmoid activation. We'll revisit this later to confirm that for *binary* classification—when $N = 2$ —the softmax and sigmoid activations are *equivalent*.

Limitations of the Argmax Function

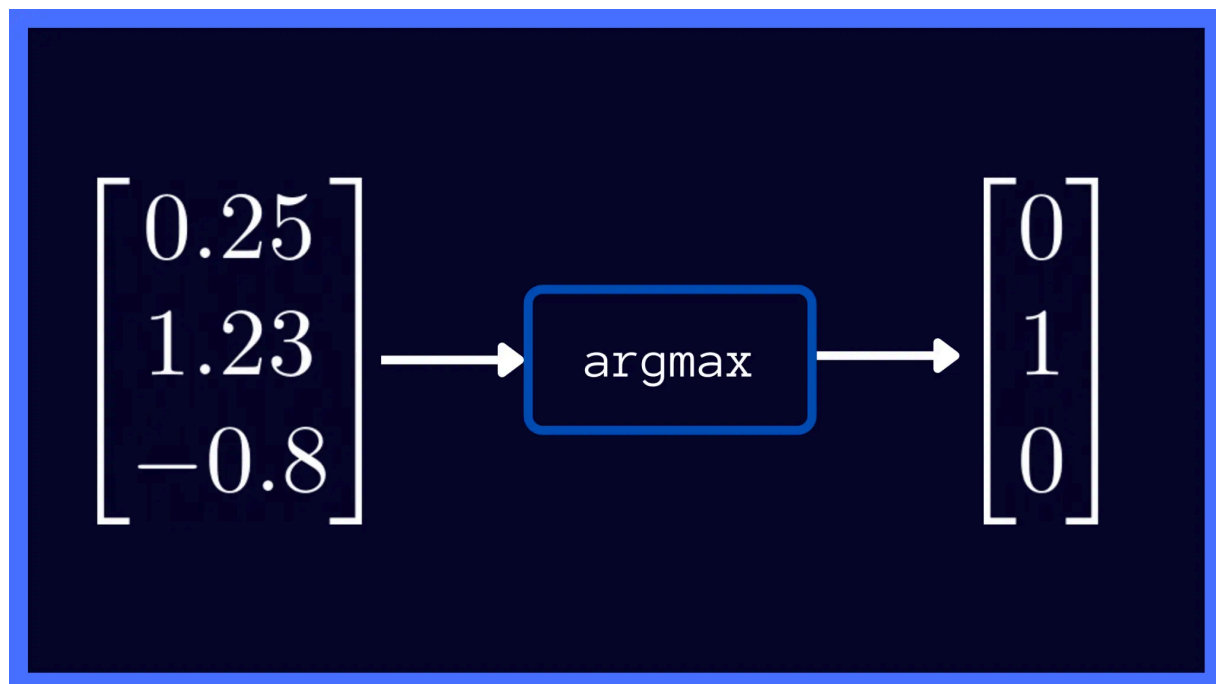
The argmax function returns the **index** of the maximum value in the input array.

Let's suppose the neural network's output is $[-0.8, 0.5, 0.2]$. In this case, the maximum value is at index 1. In our classification example, index 1 corresponds to the class "panda", so the network is predicted to be that of a panda.

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

In vector notation, you'll have 1 at the index where the maximum occurs (at index 1 for the vector z). And you'll have 0 at all other indices.



Argmax Output (Image by the author)

One limitation with using the argmax function is that its *gradients* with respect to the raw outputs of the neural networks are always *zero*. As you know, it's the backpropagation of gradients that facilitates the learning process in neural networks.

As you'll have to plug in the value 0 for all gradients of the argmax output during backpropagation, you cannot use the argmax function in training. Unless there's



From a probabilistic viewpoint, notice how the argmax function puts all the mass on index 1: the predicted class and 0 elsewhere. So it's straightforward to infer the predicted class label from the argmax output. However, we would like to know how likely the image is to be that of a panda, a seal, or a duck, and the softmax scores help us with just that!

The Softmax Activation

It's finally time to learn about softmax. It takes in a vector of **raw outputs** of **probability scores**.

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

The equation of the softmax function is given as follows:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}}$$

Softmax Function Equation (Image by the author)

Here,

- \mathbf{z} is the vector of raw outputs from the neural network
- The value of $e \approx 2.718$
- The i -th entry in the softmax output vector $\text{softmax}(\mathbf{z})$ can be thought of as the predicted probability of the test input belonging to class i .

From the plot of e^x , you can see that, regardless of whether the input x is positive, negative, or zero, e^x is always a positive number.



Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

Recall that in our example, $N = 3$ as we have 3 classes: {seal, panda, duck}, and the valid indices are 0, 1, and 2. Suppose you're given the vector $\mathbf{z} = [0.25, 1.23, -0.8]$ of raw outputs from the neural network.

Let's apply the softmax formula on the vector \mathbf{z} , using the steps below:

1. Calculate the exponent of each entry.
2. Divide the result of step 1 by the sum of the exponents of all entries.

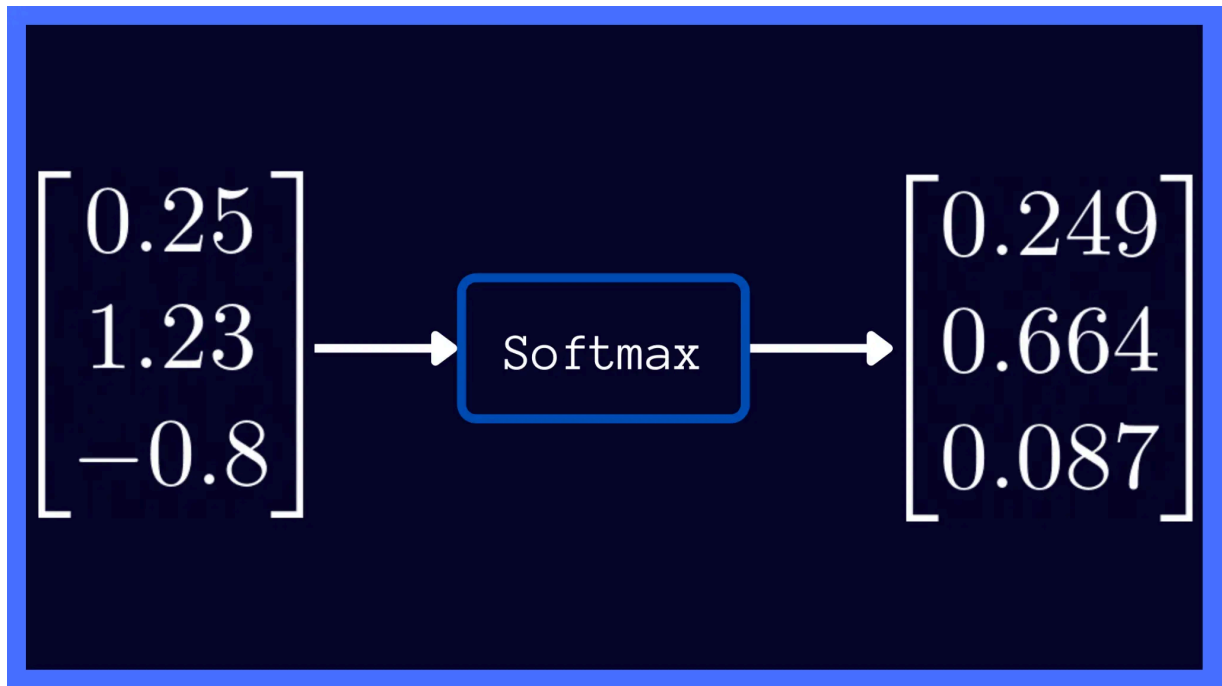
$$\begin{aligned} Pr[y_i = seal] &= softmax(\mathbf{z})_0 \\ &= \frac{e^{0.25}}{e^{0.25} + e^{1.23} + e^{-0.8}} = 0.249 \end{aligned} \quad \img alt="Illustration of a seal" data-bbox="623 298 723 358"/>
$$\begin{aligned} Pr[y_i = panda] &= softmax(\mathbf{z})_1 \\ &= \frac{e^{1.23}}{e^{0.25} + e^{1.23} + e^{-0.8}} = 0.664 \end{aligned} \quad \img alt="Illustration of a panda" data-bbox="358 378 438 458"/>
$$\begin{aligned} Pr[y_i = duck] &= softmax(\mathbf{z})_2 \\ &= \frac{e^{-0.8}}{e^{0.25} + e^{1.23} + e^{-0.8}} = 0.087 \end{aligned} \quad \img alt="Illustration of a duck" data-bbox="633 473 693 538"/>$$$$$$

Computing softmax scores for the 3 classes (Image by the author)

▶ Now that we've computed the softmax scores, let's collect them into a vector for succinct representation, as shown below:

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.



Softmax Output (Image by the author)

From the softmax output above, we can make the following observations:

- In the vector \mathbf{z} of raw outputs, the maximum value is 1.23, which on applying softmax activation maps to 0.664: the largest entry in the softmax output vector. Likewise, 0.25 and -0.8 map to 0.249 and 0.087: the second and the third largest entries in the softmax output respectively. Thus, applying softmax preserves the *relative ordering* of scores.
- All entries in the softmax output vector are between 0 and 1.
- In a multiclass classification problem, where the classes are mutually exclusive, notice how the entries of the softmax output sum up to 1: $0.664 + 0.249 + 0.087 = 1$.

This is exactly why you can think of softmax output as a probability distribution over the input classes, that makes it *readily interpretable*.

As a next step, let's examine the softmax output for our example.

In the vector $\text{softmax}(\mathbf{z}) = [0.664, \dots]$ value. This means there's a 66.4% which from our one-hot encoding

And the input image has a 29.4% of being a duck.

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

Therefore, applying softmax gives *instant* interpretability, as you know how *likely* the test image is to belong to each of the 3 classes. In this particular example, it's *highly likely* to be a panda and *least likely* to be a duck.

It now makes sense to call the argmax function on the softmax output to get the predicted class label. As the predicted class label is the one with the highest probability score, you can use `argmax(softmax(z))` to obtain the predicted class label. In our example, the highest probability score of 0.664 occurs at index 1, corresponding to class 1 (panda).

How to Implement the Softmax Activation in Python

In the previous section, we did some simple math to compute the softmax scores for the output vector **z**.

Now let's translate the math operations into equivalent operations on NumPy arrays. You may use the following code snippet to get the softmax activation for any vector **z**.

```
1 import numpy as np
2
3 def softmax(z):
4     '''Return the softmax output of a vector.'''
5     exp_z = np.exp(z)
6     sum = exp_z.sum()
7     softmax_z = np.round(exp_z/sum,3)
8     return softmax_z
```



We can parse the definition of the softmax function:

- The function takes in one required parameter **z**, a vector, and returns the softmax output vector `softmax_z`.
- We use `np.exp(z)` to compute `exp_z`.
- Next, we call `sum` on the array `exp_z` to get the total sum of the exponentiated values.

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

- We then divide each entry in `exp_z` by the sum and round off the result to 3 decimal places, storing the result in a variable, say, `softmax_z`.
- Finally, the function returns the array `softmax_z`.

You may now call the function with the output array `z` as the argument and verify that the scores are identical to what we had computed manually.

```
1 z = [0.25, 1.23, -0.8]
2 softmax(z)
3
4 # Output
5 array([ 0.249, 0.664, 0.087])
```



Are you wondering if normalizing each value by the sum of entries will suffice, to get relative scores? Let's see why it's not an efficient solution.

Why Won't Normalization by the Sum Suffice

Why use something math-heavy as the softmax activation? Can we not just divide each of the output values by the sum of all outputs?

Well, let's try to answer this by taking a few examples.

Use the following function to return the array normalized by the sum.

```
1 def div_by_sum(z):
2     sum_z = np.sum(z)
3     out_z = np.round(z/sum_z, 3)
4     return out_z
```



1 Consider `z1 = [0.25, 1.23, -0.8]` though the entries in the returned negative values. We still aren't able

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

```
1 z1 = [0.25, 1.23, -0.8]
2 div_by_sum(z1)
3
4 # Output
5 array([ 0.368,  1.809, -1.176])
```



2 Let $\mathbf{z2} = [-0.25, 1, -0.75]$. In this case, all elements in the vector sum up to zero, so the denominator will always be 0. When you divide by the sum to normalize, you'll face runtime warnings, as division by zero is not defined.

```
1 z2 = [-0.25, 1, -0.75]
2 div_by_sum(z2)
3
4 # Output
5 RuntimeWarning: divide by zero encountered in true_divide
6 array([-inf,  inf, -inf])
```



3 In this example, $\mathbf{z3} = [0.1, 0.9, 0.2]$. Let's check both the softmax and normalized scores.

```
1 z3 = [0.1, 0.9, 0.2] # ratio: 1:9:2
2 print(div_by_sum(z3))
3 print(softmax(z3))
4
5 # Output
6 [0.083 0.75  0.167] # ratio: 1:9:2
7 [0.231 0.514 0.255]
```



As shown in the code cell above, we can see the normalized scores as probabilities in the array $\mathbf{z3}$. In this example, the

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

However, you can't guarantee that the neural network's raw output won't sum up to 0 or have negative entries.

4 In this example, $\mathbf{z4} = [0, 0.9, 0.1]$. Let's check both the softmax and normalized scores.

```
1 z4 = [0,0.9,0.1]
2 print(div_by_sum(z4))
3 print(softmax(z4))
4
5 # Output
6 [0.  0.9 0.1]
7 [0.219 0.539 0.242]
```



As you can see, when one of the entries is 0, upon calling the `div_by_sum` function, the entry is still 0 in the normalized array. However, in the softmax output, you can see that 0 has been mapped to a score of 0.219.

In some sense you can think of the softmax activation function as a softer version of the argmax function: It *maximizes* the probability score corresponding to the predicted output label. At the same time, it's *soft* because it does assign some probability mass to the less likely classes as well, unlike the argmax function that puts the entire probability mass of 1 on the maximum, and 0 everywhere else.

In essence, the softmax activation can be perceived as a smooth approximation to the argmax function.

Equivalence of the Sigmoid, Softmax Activations for $N = 2$

Now let's revisit our earlier claim that the softmax function is equivalent for binary classification.

Recall that in binary classification, we use the neural network's output to get a value in the range [0, 1].

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

When you're using the softmax function for multiclass classification, **the number of nodes in the output layer = the number of classes N**.

You can think of binary classification as a special case of multiclass classification. Assume that the output layer has two nodes: one outputting the score z and the other 0.

Effectively, there's *only one* node as the other is not given any weight at all. The raw output vector now becomes $\mathbf{z} = [z, 0]$. Next, we may go ahead and apply softmax activation on this vector \mathbf{z} and check how it's equivalent to the sigmoid function we looked at earlier.

$$\begin{aligned} \text{softmax}(\mathbf{z})_0 &= \frac{e^z}{e^0 + e^z} = \frac{e^z}{1 + e^z} = \sigma(z) \\ \text{softmax}(\mathbf{z})_1 &= \frac{e^0}{e^0 + e^z} = \frac{1}{1 + e^z} = 1 - \sigma(z) \end{aligned}$$

Equivalence of Sigmoid & Softmax Activations (Image by the author)

Observe how the softmax activation scores in this case are the same as the sigmoid activation scores: $\sigma(z)$ and $1 - \sigma(z)$.

And with this, we wrap up our discussion on the softmax activation function. Let's quickly summarize all that we've learned.

Summing Up

In this tutorial, you've learned the

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

- How to use the softmax function as output layer activation in a multiclass classification problem.
- The working of the softmax function—how it transforms a vector of raw outputs into a vector of probabilities. And how you can interpret each entry in the softmax output as the probability of the corresponding class.
- How to interpret the softmax activation as an extension of the sigmoid function to multiclass classification, and their equivalence for binary classification where the number of classes $N = 2$.

In the next tutorial, we'll delve deep into cross-entropy loss—a widely-used metric to assess how well your multiclass classification model performs.

Until then, check out other interesting NLP tutorials on vector search, algorithms, and more. Happy learning!

Further Reading

[1] [Lesson on Backpropagation, Chain Rule, and Vectorization from CS231n](#)

[2] [Layer Activation Functions: Keras API Reference](#)

[3] [Implementation of Softmax Regression from Scratch](#)

Share:



Further Reading

Building a Reliable, Curated, and Accurate RAG System with Cleanlab and Pinecone

Four features of the Assistant

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

Vectors and Graphs: Better Together



Product

[Overview](#)[Documentation](#)[Integrations](#)[Trust and Security](#)

Solutions

[Customers](#)[RAG](#)[Semantic Search](#)[Multi-Modal Search](#)[Candidate Generation](#)[Classification](#)

Resources

[Learning Center](#)[Community](#)[Pinecone Blog](#)[Support Center](#)[System Status](#)[What is a Vector Database?](#)[What is Retrieval Augmented Generation \(RAG\)?](#)[Multimodal Search
Whitepaper](#)[Classification Whitepaper](#)

Company

[About](#)[Partners](#)[Careers](#)[Newsroom](#)[Contact](#)

Legal

[Customer Terms](#)[Website Terms](#)[Privacy](#)[Cookbook](#)[Cookbook](#)

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.

© Pinecone Systems, Inc. | San Francisco, CA

Pinecone is a registered trademark of Pinecone Systems, Inc.

Don't miss the next one...

Get an email the next time we publish an article about machine learning and similarity search.