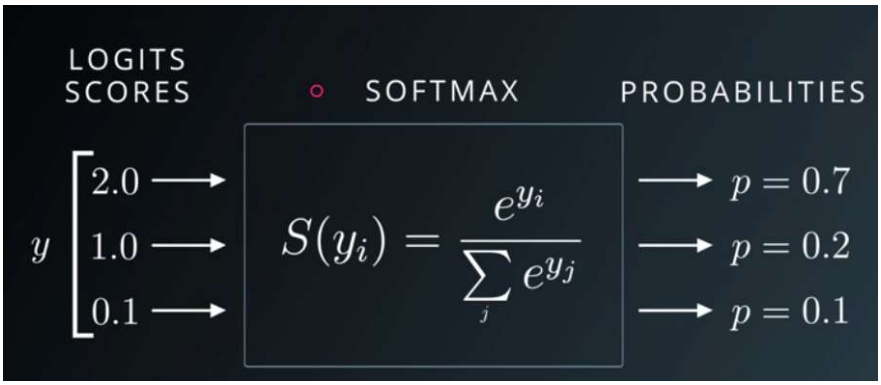
Understand the Softmax Function in Minutes

Uniqtech Co Follow
Jan 31 · 3 min read ★

Learning machine learning? Specifically trying out neural networks and getting on the hype of deep learning? You likely have run into the softmax function, a wonderful *activation function* that turns numbers aka logits into probabilities that sum to one. It's also a core element used in *deep learning classification* tasks (more on this soon in a new section October 5 2018). We will help you understand the Softmax function in a beginner friendly way by showing you exactly how it works—by coding your very own softmax function in python.



Udacity Deep Learning Slide on Softmax

The above Udacity lecture slide shows that softmax function turns [2.0, 1.0, 0.1] into [0.7, 0.2, 0.1] and the probabilities sum to 1.

The softmax is not a black box. It has two components: special number e to some power divide by a sum of some sort.

yi refers to each element in y the logits vector. Let's see it in code

```
logits = [2.0, 1.0, 0.1]
import numpy as np
exps = [np.exp(i) for i in logits]
```

We use numpy.exp(power) to take the special number e to any power we want. We use python list comprehension to iterate through each of the logits, and compute np.exp(logit). Notice the use of plural and singular form. The result is stored in a list called exps.

We just computed the top part of the softmax for each logit. Each of these transformed logit needs to be normalized by another number—the sum of all the transformed logits. This normalized give us nice probabilities outputs that sum to one!

We compute the sum of all the transformed logits and store the sum in sum_of_exps.

```
sum_of_exps = sum(exps)
```

Now we are ready to write the final part of our softmax function: each transformed logits need to be normalized

```
softmax = [j/sum_of_exps for j in exps]
```

Again, we use python list comprehension: we grab each transformed logit using [j for j in exps] divide each j by the sum_of_exps.

List comprehension gives us a list back. When we print the list we get

```
>>> softmax
[0.6590011388859679, 0.2424329707047139,
0.09856589040931818]
>>> sum(softmax)
1.0
```

The output rounds to [0.7, 0.2, 0.1] as seen on the slide at the beginning of this article. They sum nicely to one!

If we hardcore our label data to the vectors below

```
[1,0,0] #cat
[0,1,0] #dog
[0,0,1] #bird
```

The output probabilities are saying 70% sure it is a cat, 20% a dog, 10% a bird. One can see that the initial differences are adjusted to percentages. logits = [2.0, 1.0, 0.1]. It's not 2:1:0.1. Previously, we cannot say that it's 2x more likely to be a cat, because the results were not normalized to sum to one.

Why do we still need fancy machine learning libraries with fancy softmax function? The nature of machine learning training requires ten of thousands of training data. Something as concise as the softmax function needs to be optimize to process each element. Some say that Tensorflow broadcasting is not necessarily faster than numpy's matrix broadcasting though.

Watch this softmax tutorial on Youtube

Visual learner? Prefer watching a youtube video instead? See our tutorial below.

Deeper Dive into Softmax

Softmax is an activation function. Other activation functions include RELU and Sigmoid. It is frequently used in classifications. Softmax output is large if the score (input called logit) is large. Its output is small if the score is small. The proportion is not uniform. Softmax is exponential, can enlarge differences - push one result closer to 1 while another closer to 0. It turns scores aka logits into probabilities. Cross entropy (cost function) is often computed for output of softmax and true labels (encoded in one hot encoding). Here's an example of Tensorflow cross entropy computing function. It computes softmax cross entropy between logits and labels. Softmax outputs sum to 1 makes great probability analysis. Sigmoid outputs don't sum to 1. Remember the takeaway is: the essential goal of softmax is to turn numbers into probabilities.

Coming soon… deep dive of softmax, how it is used in practice, in deep learning models. Is there a more efficient way to calculate Softmax for big datasets? Stay tuned. Get alerts subscribe@uniqtech.co