How does the Softmax activation function work?

<https://www.machinecurve.com/index.php/2020/01/08/how-does-the-softmax-activation-function-work/>

When you’re creating a neural network for classification, you’re likely trying to solve either a binary or a multiclass classification problem. In the latter case, it’s very likely that the [activation function](https://www.machinecurve.com/index.php/2019/09/04/relu-sigmoid-and-tanh-todays-most-used-activation-functions/) for your final layer is the so-called **Softmax activation function**, which results in a multiclass probability distribution over your target classes.

Okay: Softmax. It always “returns a probability distribution over the target classes in a multiclass classification problem”

The final layer of the neural network, *without the activation function*, is what we call the **“logits layer”** .

 It simply provides the final outputs for the neural network. In the case of a four-class multiclass classification problem, that will be four neurons – and hence, four outputs.

*the odds of something to occur* must be a positive real number, e.g. 0.238. Since the sum of probabilities must be equal to 1, no probability can be >1. Hence, any probability therefore lies somewhere in the range [0,1].

*A****discrete probability distribution****is a probability distribution that can take on a countable number of values.*

*A****continuous probability distribution****is a probability distribution with a cumulative distribution function that is [absolutely continuous](https://en.wikipedia.org/wiki/Absolute_continuity).*

So, while a discrete distribution can take a certain amount of values – four, perhaps  – and is therefore rather ‘blocky’ with one probability per value, a continuous distribution can take *any* value, and probabilities are expressed as being in a range.

For each outcome (each neuron represents the outcome for a target class), we’d love to know the individual probabilities, but of course they must be relative to the other target classes in the machine learning problem. Hence, probability distributions, and specifically discrete probability distributions.

### **The Softmax function**

The Softmax function allows us to express our inputs as a discrete probability distribution. Mathematically, this is defined as follows:

𝑆𝑜𝑓𝑡𝑚𝑎𝑥(𝑥𝑖)=𝑒𝑥𝑝(𝑥𝑖)/∑𝑗𝑒𝑥𝑝(𝑥𝑗))

Intuitively, this can be defined as follows: for each value (i.e. input) in our input vector, the Softmax value is the *exponent of the individual input* divided by a sum of *the exponents of all the inputs*.

This ensures that multiple things happen:

* Negative inputs will be converted into nonnegative values, thanks to the exponential function.
* Each input will be in the interval (0,1).
* As the *denominator* in each Softmax computation is the same, the values become proportional to each other, which makes sure that together they sum to 1.

This, in return, allows us to “interpret them as probabilities” (Wikipedia, 2006). Larger input values correspond to larger probabilities, at exponential scale, once more due to the exponential function.

Softmax can: besides having nice properties with regards to normalization (as we saw before), it can be differentiated. Hence, it’s very useful for optimizing your neural network.