9.5 FROM RAW NUMBERS TO PROBABILITIES: SOFTMAX

In many cases, the outcome of an AI system is a choice between a set of given possibilities. That situation is in fact the standard one with the most visible of today's AI tools: Large Language Models. The way an LLM functions is that at each step it selects the next "token to be output. It does so by using a probability distribution for the various possibilities and selecting the one with the highest probability. (It may apply a "temperature" parameter to select, occasionally, an element of slightly lower probability, with the goal of introducing some flexibility into a text that might otherwise sound too uniform.)

In such situations, we need a neural-network outcome that is one of the elements of a probability distribution. This is not what we would get in the output layer based on the discussions so far: they yield a vector of numbers, the result of applying an affine transformation $W * X + B$. Those numbers, often called **logits**, can have values in an arbitrary range; what matters is their order (if logit a is 45 and b is 48, the choice associated with b should have a higher probability). Softmax will turn the vector of logits into a probability distribution.

Here is an example of the application of softmax. Assume that an LLM or other natural-language processing tool is looking for filling in the dots after the sentence start

After many attempts, the match finally…

I am being a bit devious here, or at least mindful of chapter 11 (where this example will help us understand *transformers*): the word "match" has several independent meanings (sports game, lighting device, potential spouse…). Let us assume that the neural network has given us 6 possible completions, each with a certain logit representing its value:

In the last example, "match" is used as in "matchmaker".

Item	flared	began	succeeded	failed	ended	ignited
Logit	**2.0**	**1.5**	**2.1**	**0.5**	**0.2**	**1.8**

"Flared" and "ignited" work for matches as lighting tools ("flared" might also make sense for a sports game), "began" works for matches as well as sports games and spousal partnerships, and so on. The result of applying softmax, shown as a bar chart, is the following (next page):

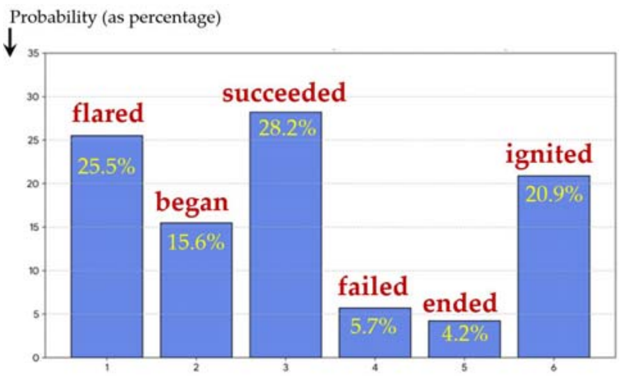


Figure 9.8: Softmax example results

Let us now see what the softmax transformation is, ensuring this result. It has four essential properties:

- The s**um of all probabilities** (expressed above as percentages) **is 1** (100 as percentages). This property is a consequence of the result being a "probability distribution". The distribution is discrete, meaning that the only choices with non-zero probability are the given logits. No other value is possible.

- Softmax **preserves order**. If $a < b$, then the probability of b will be greater than the probability of a. (Mathematically, softmax is a *monotonic* function.) *Or "monotone".*

- Softmax **magnifies the differences**. Larger values get a significant boost. The phrase "exponentially greater", so much abused by journalists and others, is literally applicable here, as the function's formula below will show.

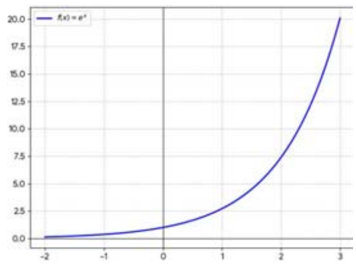
- Softmax is **translation-invariant**, meaning that we can add a given value to all the logits without changing the result of their softmax. This property is important in practice since the various affine translations applied to tokens (in particular the biases) can produce results in highly diverse ranges.

- Softmax is **differentiable**: the formula below uses an exponential function with a well-defined derivative, simple to compute.

Consider a vector of logits z with n elements $z\ [1], z\ [2], \ldots\ z\ [n]$. To compute the i-th element of the softmax, we start with

$$term\ [i] = e^{z\ [i]}$$

In other words, we replace the i-th logit by its exponentiated value $e^{z[i]}$, also written $exp\ (z\ [i])$: e to the power $z\ [i]$. Here e is the constant known as Euler's number, approximately equal to 2.71828… The function e^x grows extremely fast (*exponentially* fast, in the proper sense of the term), much faster than any "polynomial" function of the form x^n even for a very high constant n. As a reminder from your math courses, the beginning of the graph of the exponential function appears on the right.



Graph of e^x

The use of the exponential addresses one of the goals mentioned above: magnifying the differences between the logits. The largest logits will get a boost moving them even further away from their rivals. (Kind of like in real life: the rich get richer. Too bad for the others.)

The *term* [i] values are not the actual softmax values because they are not probabilities yet: they have no reason to sum up to 1. We turn them into probabilities summing up to 1 by dividing them by the sum of all of them:

$$softmax\ (z)\ [i] = term\ [i]\ /\ TOTAL$$

$$\text{-- where } TOTAL \text{ is } \sum_{j=1}^{n} term\ [j]$$

Here is the detail of the computation for the preceding example (with the result again multiplied by 100 to express the probabilities as percentages):

Item	Logit $z\ [i]$	$e^{z[i]}$	Probability (percentage) $e^{z[i]}\ /\ (TOTAL) * 100$
flared	2.0	7.39	**25.5%**
began	1.5	4.48	**15.5%**
succeeded	2.1	8.17	**28.2%**
failed	0.5	1.65	**5.7%**
ended	0.2	1.22	**4.2%**
ignited	1.8	6.05	**20.9%**
TOTAL		28.96	**100%**

As noted, being able to compute the derivatives is an important condition for nonlinearities. The partial derivative of $softmax\ (z)\ [i]$ with respect to $z\ [i]$, which we may call $\sigma\ (i)$, is simple (thanks to the property of the exponential function that it is its own derivative): it is $\sigma(i) * (\delta(i, j) - \sigma(j))$, defining $\delta(i, j)$ (known as the “Kronecker delta”) to be 1 if $i = j$ and 0 otherwise.

Thanks to these qualities, softmax has become an essential part of today’s neural network architectures, particular at the last step — the output layer — to choose between the identified completion possibilities.

9.6 SIGMOID

The next nonlinearity example is less commonly used but still deserves a mention since you might encounter it in the literature. Like softmax, the sigmoid activation function will give us a probability, but instead of handling an arbi-