What is the Log-Sum-Exp Function?

Nicholas J. Higham*

January 5, 2021

The log-sum-exp function takes as input a real n-vector x and returns the scalar

$$\mathrm{lse}(x) = \log \sum_{i=1}^{n} \mathrm{e}^{x_i},$$

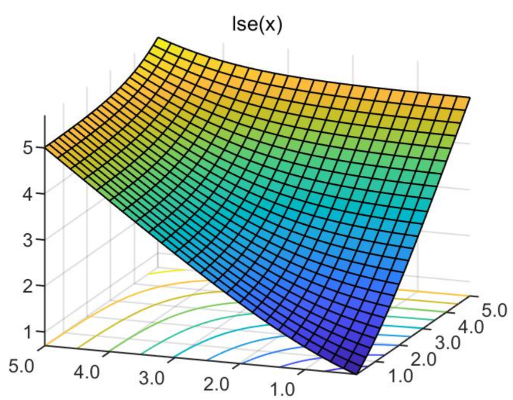
where log is the natural logarithm. It provides an approximation to the largest element of x, which is given by the max function, $\max(x) = \max_i x_i$. Indeed,

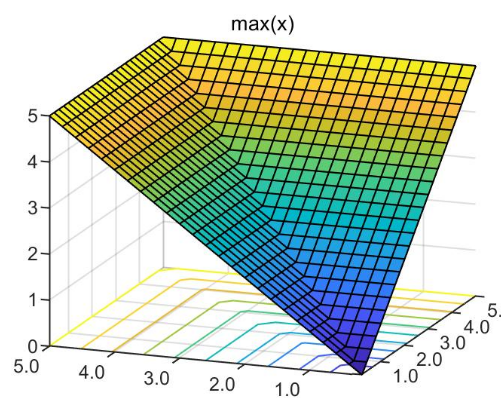
$$\mathrm{e}^{\max(x)} \le \sum_{i=1}^{n} \mathrm{e}^{x_i} \le n\mathrm{e}^{\max(x)},$$

and on taking logs we obtain

$$\max(x) \le \mathrm{lse}(x) \le \max(x) + \log n. \qquad (*)$$

The log-sum-exp function can be thought of as a smoothed version of the max function, because whereas the max function is not differentiable at points where the maximum is achieved in two different components, the log-sum-exp function is infinitely differentiable everywhere. The following plots of $\mathrm{lse}(x)$ and $\max(x)$ for $n = 2$ show this connection.





The log-sum-exp function appears in a variety of settings, including statistics, optimization, and machine learning.

*Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (nick.higham@manchester.ac.uk).

For the special case where $x = [0\ t]^T$, we obtain the function $f(t) = \log(1+\mathrm{e}^t)$, which is known as the softplus function in machine learning. The softplus function approximates the ReLU (rectified linear unit) activation function $\max(t,0)$ and satisfies, by $(*)$,

$$\max(t,0) \le f(t) \le \max(t,0) + \log 2.$$

Two points are worth noting.

- While $\log(x_1+x_2) \ne \log x_1 + \log x_2$, in general, we do (trivially) have $\log(x_1+x_2) = \mathrm{lse}(\log x_1, \log x_2)$, and more generally $\log(x_1+x_2+\cdots+x_n) = \mathrm{lse}(\log x_1, \log x_2, \dots, \log x_n)$.
- The log-sum-exp function is not to be confused with the exp-sum-log function: $\exp \sum_{i=1}^n \log x_i = x_1 x_2 \dots x_n$.

Here are some examples:

```
>> format long e
>> logsumexp([1 2 3])
ans =
     3.407605964444380e+00

>> logsumexp([1 2 30])
ans =
     3.000000000000095e+01

>> logsumexp([1 2 -3])
ans =
     2.318175429247454e+00
```

The MATLAB function `logsumexp` used here is available at `https://github.com/higham/logsumexp-softmax`.

Straightforward evaluation of log-sum-exp from its definition is not recommended, because of the possibility of overflow. Indeed, $\exp(x)$ overflows for $x = 12$, $x = 89$, and $x = 710$ in IEEE half, single, and double precision arithmetic, respectively. Overflow can be avoided by writing

$$\mathrm{lse}(x) = \log \sum_{i=1}^n \mathrm{e}^{x_i} = \log \sum_{i=1}^n \mathrm{e}^a \mathrm{e}^{x_i-a} = \log \left(\mathrm{e}^a \sum_{i=1}^n \mathrm{e}^{x_i-a} \right),$$

which gives

$$\mathrm{lse}(x) = a + \log \sum_{i=1}^n \mathrm{e}^{x_i-a}.$$

We take $a = \max(x)$, so that all exponentiations are of nonpositive numbers and therefore overflow is avoided. Any underflows are harmless. A refinement is to write

$$\mathrm{lse}(x) = \max(x) + \mathrm{log1p}\left(\sum_{\substack{i=1\\ i\ne k}}^n \mathrm{e}^{x_i-\max(x)} \right), \qquad (\#)$$

where $x_k = \max(x)$ (if there is more than one such k, we can take any of them). Here, $\mathrm{log1p}(x) = \log(1+x)$ is a function provided in MATLAB and various other languages that accurately evaluates $\log(1+x)$ even when x is small, in which case $1+x$ would suffer a loss of precision if it was explicitly computed.

Whereas the original formula involves the logarithm of a sum of nonnegative quantities, when $\max(x) < 0$ the shifted formula (#) computes $\mathrm{lse}(x)$ as the sum of two terms of opposite sign, so could potentially suffer from numerical cancellation. It can be shown by rounding error analysis, however, that computing log-sum-exp via (#) is numerically reliable.

References

This is a minimal set of references, which contain further useful references within.

- Pierre Blanchard, Desmond J. Higham, and Nicholas J. Higham, Accurately Computing the Log-Sum-Exp and Softmax Functions, IMA J. Numer. Anal., Advance access, 2020.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, 2016.

Related Blog Posts

- What is Numerical Stability? (2020)

This article is part of the “What Is” series, available from `https://nhigham.com/category/what-is` and in PDF form from the GitHub repository `https://github.com/higham/what-is`.