Basic properties of the soft maximum

John Cook
Department of Biostatistics, Unit 1409
The University of Texas, M. D. Anderson Cancer Center
Houston, Texas 77030, USA
cook@mdanderson.org

September 7, 2011

Abstract

This note presents the basic properties of the soft maximum, a smooth approximation to the maximum of two real variables. It concludes by looking at potential numerical difficulties with the soft maximum and how to avoid these difficulties.

Keywords: soft maximum, optimization, overflow, underflow

1 Introduction

It is often necessary in applications to take the maximum of two numbers. But the simple function

$$f(x, y) = \max(x, y)$$

can be difficult to work with because it has a sharp corner along the line $x = y$. Sometimes you want an alternative that sands down the sharp edges of the maximum function. One such alternative is the soft maximum. The soft maximum $g(x, y)$ has three desirable properties.

1. $g(x, y)$ infinitely differentiable everywhere.
2. $g(x, y)$ is convex.
3. $\lim_{|x-y|\to\infty} = \max(x, y)$.

This report call the maximum function the "hard" maximum to make it easier to compare to the soft maximum.

2 Analytical properties

The soft maximum of two variables is the function

$$g(x, y) = \log(\exp(x) + \exp(y)).$$

See *Convex Optimization* by Stephen Boyd and Lievan Vandenberghe, Cambridge University Press, 2004.

The definition of soft maximum can be extended to more than two variables by taking

$$g(x_1, x_2, \ldots, x_n) = \log(\exp(x_1) + \exp(x_2) + \cdots + \exp(x_n)).$$

To see that the soft maximum approximates the hard maximum, note that if x is a little bigger than y, $\exp(x)$ will be a lot bigger than $\exp(y)$. That is, exponentiation exaggerates the differences between x and y. If x is significantly bigger than y, $\exp(x)$ will be so much bigger than $\exp(y)$ that $\exp(x) + \exp(y)$ will essentially equal $\exp(x)$ and the soft maximum will be approximately $\log(\exp(x)) = x$, the hard maximum.

The soft maximum approximates the hard maximum and is a convex function just like the hard maximum. But the soft maximum is smooth. It has no sudden changes in direction and is infinitely differentiable. These properties make it useful in convex optimization algorithms.

Notice that the accuracy of the soft maximum approximation depends on scale. If you multiply x and y by a large constant, the soft maximum will be closer to the hard maximum. For example, g(1, 2) = 2.31, but g(10, 20) = 20.00004. This suggests you could control the "hardness" of the soft maximum by generalizing the soft maximum to depend on a parameter k.

$$g(x, y; k) = \log(\exp(kx) + \exp(ky))/k$$

You can make the soft maximum as close to the hard maximum as you like by making k large enough. For every value of k the soft maximum is

differentiable, but the partial derivatives near $x = y$ become larger as k increases. In the limit the partial derivatives become infinite as the soft maximum converges to the hard maximum.

3 Numerical properties

The most obvious way to compute the soft maximum function in C would be

```
double SoftMaximum(double x, double y)
{
    return log( exp(x) + exp(y) );
}
```

This works for some values of x and y, but fails if x or y is large. For example, if we use this to compute the soft maximum of 1000 and 200, the result is numerical infinity when using standard (IEEE 754) double precision arithmetic. The value of exp(1000) is too big to represent in a floating point number, so it is computed as infinity. The value of exp(200) is finite, but the sum of an infinity and a finite number is infinity, and the log function applied to infinity returns infinity. See *What every computer scientist should know about floating-point arithmetic* by David Goldberg, Computing Surveys, March, 1991. Available at `http://bit.ly/gwt4Z4`.

We have the opposite problem if we try to compute the soft maximum of -1000 and -1200. In this computation exp(−1000) and exp(−1200) both underflow to zero, and the log function returns negative infinity for the logarithm of zero.

Fortunately it is not hard to fix the function `SoftMaximum` to avoid overflow and underflow. If we shift both arguments by a constant c,

$$\log(\exp(x - c) + \exp(y - c)) = \log(\exp(x) + \exp(y)) - c.$$

and so

$$\log(\exp(x) + \exp(y)) = \log(\exp(x - c) + \exp(y - c)) + c.$$

If we pick c to be the (hard) maximum of x and y, then one of the calls to `exp` has argument 0 (and so returns 1) and the other has a negative argument. This means the following code cannot overflow.

```
double SoftMaximum(double x, double y)
{
    double maximum = max(x, y);
    double minimum = min(x, y);
    return maximum + log( 1.0 + exp(minimum - maximum) );
}
```

The call to `exp(minimum - maximum)` could possibly underflow to zero, but in that case the code returns `maximum`. And in that case the return value is very accurate: if `maximum` is much larger than `minimum`, then the soft maximum is essentially equal to `maximum`.

The equation for the soft maximum implemented above has a few advantages in addition to avoiding overflow. It makes it clear that the soft maximum is always greater than the maximum. Also, it shows that the difference between the hard maximum and the soft maximum is controlled by the spread of the arguments. The soft maximum is nearest the hard maximum when the two arguments are very different and furthest from the hard maximum when the two arguments are equal.