Computing Approximate Derivatives

Sometimes gradient and/or hessian information is unavailable even for smooth functions. We can approximate derivatives using finite differences with care!

Suppose we wish to determine $\nabla f(x)$ for the particular point x. We can approximate each component by considering the limit definition of a derivative.

$$\frac{\partial f}{\partial x_i}(x) = \lim_{h \to 0} \frac{f(x+h\hat{x}_i) - f(x)}{h}$$

We can, for some small enough h, say that

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x+h\hat{x}_i) - f(x)}{h}.$$

(one extra function evaluation at $x+h\hat{x}_i$. This is about the same cost as a direct calculation.)

In an optimization setting we do not want to spend a lot of time deciding a good value for h — this would involve many extra function evaluations. Instead, can we determine a likely good h just by considering the properties of the numerical computation.

Consider the numerical task of estimating a derivative of a given function using limited precision arithmetic.

Let $f(x) = \sin x$, $g(x) = \dfrac{\sin(x+\delta) - \sin x}{\delta}$.

So $g(x)$ is an estimate for $f'(x)$, and we will consider the question of how to choose δ. If we have infinite precision arithmetic then we can choose as small a δ as we desire because $\lim_{\delta \to 0} g(x) = f'(x)$.

However, suppose we have 9-digit precision. Let's compute $g(1)$ and see how it compares with $f'(1) = \cos 1 = 0.540302305$.

We have $f(1) = 0.841470984$ (9 digits!)

δ	$f(x+\delta)$	$g(x)$
10^{-9}	0.841470985	1.
10^{-8}	0.841470990	0.6211
10^{-7}	0.841471038	0.54839
10^{-6}	0.841471525	0.541108
10^{-5}	0.841476387	0.5403786
$10^{-4.5}$		0.540314413
10^{-4}	0.841525010	0.54026832
10^{-3}	0.842010866	0.539882289
10^{-2}	0.846831844	0.5360860617
10^{-1}	0.891207360	0.49736376
1	0.909297426	0.067826442

When δ is too small we do not have sufficient precision for a meaningful computation.

We choose to keep half of the precision as a compromise.
$\delta = \sqrt{10^{-9}} = 3\times10^{-5}$.

When δ is too large we are finding (accurately) the slope of a secant line. This may not be close to the slope of the function.

(Double precision computing uses $eps = 2^{-52} \approx 2.2\times10^{-16}$, so $\delta = 2^{-26} \approx 1.6\times10^{-8}$)

The above analysis is appropriate for problems where typical values of x are 1. So, to generalize this concept we can either

(a) Let $\delta = (\sqrt{eps}\,) \max\{|x_i|, |\text{typ } x_i|\}$

(b) Rescale problem variables: $\bar{x}_i = \dfrac{x_i}{|\text{typ } x_i|}$

We might also employ central differencing methods in an attempt to improve accuracy. Using this strategy, we have

$$\frac{\partial f}{\partial x_i}(x) = \frac{1}{2}\left(\lim_{h\to 0} \frac{f(x+h\hat{x}_i) - f(x)}{h} + \lim_{h\to 0} \frac{f(x) - f(x+h\hat{x}_i)}{h} \right)$$

$$= \frac{1}{2} \lim_{h\to 0} \frac{f(x+h\hat{x}_i) - f(x-h\hat{x}_i)}{h}$$

$$\approx \frac{f(x+h\hat{x}_i) - f(x-h\hat{x}_i)}{2h}$$

This method requires $2n$ extra function evaluations, but has the benefit of increased accuracy.

This approach is usually avoided in an iterative optimization context in which iterates are approximate solutions to a line search.

Suppose we wanted to estimate hessian information.

$$[\nabla^2 f]_{ij} = \frac{\partial}{\partial x_j}\frac{\partial f}{\partial x_i}(x) = \frac{\partial}{\partial x_j}[\nabla f(x)]_i \approx \frac{[\nabla f(x+k\hat{x}_j)]_i - [\nabla f(x)]_i}{k}$$

Symmetry of the hessian reduces the total number of extra gradient evaluations to $\frac{1}{2}n(n+1) < n^2$.

$k = \sqrt{eps}$ for the same reasons as before.

We could estimate hessian entries using only function evaluations

$$[\nabla^2 f]_{ij} = \frac{\partial}{\partial x_j}\frac{\partial f}{\partial x_i}(x) \approx \frac{\partial}{\partial x_j}\frac{f(x+h\hat{x}_i)-f(x)}{h}$$

$$\approx \frac{1}{k}\left(\frac{f(x+h\hat{x}_i+k\hat{x}_j)-f(x+k\hat{x}_j)}{h} - \frac{f(x+h\hat{x}_i)-f(x)}{h}\right)$$

$$= \frac{f(x+h\hat{x}_i+k\hat{x}_j) - f(x+k\hat{x}_j) - f(x+h\hat{x}_i) + f(x)}{hk}$$

This estimate requires $\frac{1}{2}n(n+1)+n = \frac{1}{2}n(n+3)$ extra function evaluations.

How should we choose h and k? In this case, we can keep 1/3 of the precision overall by choosing $h = k = (eps)^{1/3} \approx 10^{-5}$.

Of course, estimating the hessian is usually too costly to be part of a regular and efficient strategy. However, it can be a useful part of some derivative free methods (MATH 565 or 567).