The Breaking of a Mathematical Curse

A serendipitous discovery by a Columbia computer scientist may make it much easier to integrate multivariate functions

THEORETICAL COMPUTER SCIENTISTS CALL IT the "curse of dimensionality." It's the ugly tendency of numerical algorithms to become impossibly slow when they're applied to problems involving more than a few variables. Take integrals. Everybody remembers them from calculus, where, in a simple form, they are used to find the area under a curve. With many more variables, integrals can be used for far more complex purposes: solving differential equations that describe the weather, say, or doing the computations in quantum field theory implied by Feynman diagrams. These integrals usually can't be solved exactly—they're too complex—but the solutions can be approximated through the use of computer algorithms. And that's where the curse of dimensionality comes in: For straightforward algorithms, the amount of computation required for a good approximation grows exponentially with the number of variables.

Now Henryk Wozniakowski, a computer scientist at Columbia University, has found a highly efficient "average case" algorithm that breaks the curse. By the standards of computational complexity, Wozniakowski's algorithm requires little more effort to compute integrals in a thousand variables than it does in one or two—and it may ultimately offer large savings of time in many areas of scientific computation.

Wozniakowski's achievement is especially interesting because it combines the serendipitous exploitation of a result in a related field with a bit of that good old motivator, cash on the barrel. But understanding the story requires taking in a little background. The precise problem he took on was to approximate the integral of a continuous function by sampling its values at a finite set of points. (For theoretical simplicity, the integration is restricted to the d-dimensional "unit cube," meaning that there are d variables, each ranging in value between 0 and 1.) However, it's impossible for any algorithm to guarantee a given level of accuracy uniformly for all continuous functions. Therefore, when they operate in this realm, computer scientists look instead at what happens for an "average" function. To do this they use a technical definition of "average" that is analogous, for the set of continuous functions, to the familiar, bell-shaped normal-curve distribution for the set of real numbers.

In this average-case setting, the complexity of an approximate integration algorithm is measured (roughly speaking) as the number of sample points required to achieve a given level of accuracy—approximating the numerical answer to 8 decimal places, say—for an "average" function. Alternatively, the complexity can be measured as the average error incurred by using a given set of sample points. The question then becomes: Given that you want a certain average accuracy, how many sample points do you need to use and where should they be located?

A natural, intuitive answer might be to place points at equally spaced intervals on a d-dimensional grid. And this is indeed the best arrangement—but only in the most limited case: that of one dimension, where the "cube" is a line segment. For functions of more than one variable, this approach succumbs to the curse of dimensionality. The number of points needed to keep the average error less than e is roughly proportional to $1/e^d$. This means, for instance, that to maintain 8-place accuracy, the amount of computation increases by a factor of 10^8 with each additional variable, which makes integration in more than a few variables all but impossible.

> **Usually we have to sweat more, but that wasn't the case for this problem.**
>
> Henryk Woźniakowski

Wozniakowski's result not only breaks the curse of dimensionality, it actually finds the best location for the sample points—in any dimension. Tell his algorithm what dimension you want to work in and how much average error you're willing to tolerate, and it will reply by telling you how many points you need and where you should put them. Alternatively, you can tell it how many points you're willing to use, and it will tell you where to put them to minimize the average error. What breaks the curse is that the number of points is almost independent of the dimension, and is roughly proportional just to $1/e$. More precisely, it's proportional to $(1/e)(\log(1/e))^{(d-1)/2}$.

The new algorithm offers an alternative to the currently favored technique for numerical integration. Known as the Monte Carlo method, it is, as the name suggests, a gambler's approach: Base your approximation on values of the function at points that are chosen completely at random. Wozniakowski's algorithm may beat out the Monte Carlo method because it typically calls for fewer points to stay within a given average error. However, Wozniakowski notes, the comparison is somewhat misleading because the meaning of "average" is different for the two approaches.

Numerical integration is a staple of scientific and statistical computing, with applications ranging from computation chemistry—where theorists want to understand the interactions among large collections of atoms—to geophysical exploration, where researchers try to use ground-level measurements to figure out things like where to drill for oil. The savings resulting from efficient algorithms can be enormous, especially as greater accuracy is called for—and ultimately Wozniakowski's discovery may make those savings possible.

Wozniakowski attributes the discovery to a combination of luck and diligence. He was working with Joe Traub, also of Columbia, on a completely different problem concerning the Monte Carlo method when he realized that a certain mathematical identity in effect translated the problem of multivariate integration into yet another problem known as L_2 discrepancy. To Wozniakowski's delight, the L_2 discrepancy problem had been solved, by Klaus Roth of Imperial College in London. Wozniakowski had little more to do than translate Roth's solution back into terms of multivariate integration. "Usually we have to sweat more, but that wasn't the case for this problem," he says.

Traub jokingly adds that there might have been a financial incentive. The two theorists had often thought about the problem, but had never seriously worked on it—mostly because they thought it was too hard. Then, on a whim, during a seminar presentation, Traub offered a cash reward. A week later, his colleague Wozniakowski had the answer. "Just coincidence," deadpans Wozniakowski. In this case he may be right: The cash prize was $64. ■ BARRY A. CIPRA