TECHNICAL PERSPECTIVE:

FINDING A GOOD NEIGHBOR, NEAR AND FAST

by Bernard Chazelle

You haven't read it yet, but you can already tell this article is going to be one long jumble of words, numbers, and punctuation marks. Indeed, but look at it differently, as a text classifier would, and you will see a single point in high dimension, with word frequencies acting as coordinates. Or take the background on your flat panel display: a million colorful pixels teaming up to make quite a striking picture. Yes, but also one single point in 10^6-dimensional space—that is, if you think of each pixel's RGB intensity as a separate coordinate. In fact, you don't need to look hard to find complex, heterogeneous data encoded as clouds of points in high dimension. They routinely surface in applications as diverse as medical imaging, bioinformatics, astrophysics, and finance.

Why? One word: geometry. Ever since Euclid pondered what he could do with his compass, geometry has proven a treasure trove for countless computational problems. Unfortunately, high dimension comes at a price: the end of space partitioning as we know it. Chop up a square with two bisecting slices and you get four congruent squares. Now chop up a 100-dimensional cube in the same manner and you get 2^{100} little cubes—some Lego set! High dimension provides too many places to hide for searching to have any hope.

Just as dimensionality can be a curse (in Richard Bellman's words), so it can be a blessing for all to enjoy. For one thing, a multitude of random variables cavorting together tend to produce sharply concentrated measures: for example, most of the action on a high-dimensional sphere occurs near the equator, and any function defined over it that does not vary too abruptly is in fact nearly constant. For another blessing of dimensionality, consider Wigner's celebrated *semicircle law*: the spectral distribution of a large random matrix (an otherwise perplexing object) is described by a single, lowly circle. Sharp measure concentrations and easy spectral predictions are the foodstuffs on which science feasts.

But what about the curse? It can be vanquished. Sometimes. Consider the problem of storing a set S of n points in $\mathbf{R}^d$ (for very large d) in a data structure, so that, given any point q, the nearest $p \in S$ (in the Euclidean sense) can be found in a snap. Trying out all the points of S is a solution—a slow one. Another is to build the Voronoi diagram of S. This partitions $\mathbf{R}^d$ into regions with the same answers, so that handling a query q means identifying its relevant region. Unfortunately, any solution with the word "partition" in it is likely to raise the specter of the dreaded curse, and indeed this one lives up to that expectation. Unless your hard drive exceeds in bytes the number of particles in the universe, this "precompute and look up" method is doomed.

What if we instead lower our sights a little and settle for an approximate solution, say a point $p \in S$ whose distance to q is at most $c = 1 + \varepsilon$ times the smallest one? Luckily, in many applications (for example, data analysis, lossy compression, information retrieval, machine learning), the data is imprecise to begin with, so erring by a small factor of $c > 1$ does not cause much harm. And if it does, there is always the option (often useful in practice) to find the exact nearest neighbor by enumerating all points in the vicinity of the query: something the methods discussed below will allow us to do.

The pleasant surprise is that one can tolerate an arbitrarily small error and still break the curse. Indeed, a zippy query time of $O(d \log n)$ can be achieved with an amount of storage roughly $n^{O(\varepsilon^{-2})}$. No curse there. Only one catch: a relative error of, say, 10% requires a prohibitive amount of storage. So, while theoretically attractive, this solution and its variants have left practitioners unimpressed.

Enter Alexandr Andoni and Piotr Indyk [1], with a new solution that should appeal to theoretical and applied types alike. It is fast and economical, with software publicly available for slightly earlier incarnations of the method. The starting point is the classical idea of *locality-sensitive hashing* (LSH). The bane of classical hashing is collision: too many keys hashing to the same spot can ruin a programmer's day. LSH turns this weakness into a strength by hashing high-dimensional points into bins on a line in such a way that only nearby points collide. What better way to meet your neighbors than to bump into them? Andoni and Indyk modify LSH in critical ways to make neighbor searching more effective. For one thing, they hash down to spaces of logarithmic dimension, as opposed to single lines. They introduce a clever way of cutting up the hashing image space, all at a safe distance from the curse's reach. They also add bells and whistles from coding theory to make the algorithm more practical.

Idealized data structures often undergo cosmetic surgery on their way to industrial-strength implementations; such an evolution is likely in this latest form of LSH. But there is no need to wait for this. Should you need to find neighbors in very high dimension, one of the current LSH algorithms might be just the solution for you.

Biography

Bernard Chazelle (chazelle@cs.princeton.edu) is a professor of computer science at Princeton University, Princeton, NJ.

Reference

1. Andoni, A. and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'06)*.