Linear programming in low dimensions is easy

Aaron Defazio

September 13, 2016

I recently came across this remarkable randomized algorithm due to Seidel (1991) for linear programming in 2 dimensions that takes only time $O(n)$ for n constraints (in expectation). It's definitely worth a close inspection, so I thought I would write a short article about it.

Note that my presentation of this algorithm assumes that there are no redundant or colinear constraints, that the solution is bounded, and that the solution for each subproblem encountered is unique. These assumptions are easy to remove without slowing the algorithm down, but they complicate the presentation a little.

Algorithm 1 Seidel's randomized LP algorithm in two dimensions

```
FUNCTION lpsolve
INPUT: Set $H$ of constraints & vector $c$.
OUTPUT: A point $v$ in the polytope defined by $H$ minimizing $c^T v$.
BASE CASE: If $|H| = 2$ then return the unique intersecting point.
NON BASE CASE:
    1. Pick a constraint $B$ uniformly at random from $H$.
    2. $v' = \text{lpsolve}(H - \{B\}, c)$.
    3. If $v'$ is in the half-space defined by $B$ RETURN $v = v'$, otherwise:
    4. RETURN $v$, the solution of the 1D LP consisting of the projection of $H$ & $c$ onto $B$.
```

This algorithm is quite remarkable in its simplicity. It takes the form of an recursive algorithm, where we solve a base case, then add additional constraints 1 at a time. Every time we add a constraint, we consider if it changes the solution. It's only when the old solution v' is outside the added constraint that we have to do any work, namely solving a simple 1D LP, which is trivially $O(m)$ time for m constraints.

If we had to solve the 1D LP subproblem at every step the algorithm would have quadratic running time ($\sum_{i=1}^{n} i = O(n^2)$). We avoid this because we pick the constraint B to recurse on at each step randomly. Clearly we only have to solve that 1D LP if the randomly chosen constraint B forms part of the new solution to the problem (ignoring degenerate solutions), and since a solution is the intersection of 2 constraints, there is only a 2 in $|H|$ chance that we randomly pick such a constraint.

It follows that the cost of each recursion in expectation is $|H|$ times $2/|H|$ which is just 2, and we recurse n times, giving us a $O(n)$ running time in expectation.

In the original presentation by Seidel, the algorithm is given in a general form, usable in higher dimensions as well. The difficulty is that the running time quickly gets out of hand as the dimension d increases. The main obstacle is step 4 above, you essentially need to solve a $d-1$ dimensional LP, which is only slightly easier than your original LP in higher dimensions.

On a side note, there is an interesting discussion of this algorithm and related ones in the Wikipedia page for LP-type problems.

References

Seidel, Raimund (1991), "Small-dimensional linear programming and convex hulls made easy", Discrete and Computational Geometry, 6 (5): 423–434