Lecture 12, Sci. Comp. for DPhil Students

Nick Trefethen, Tuesday 27.11.18

Today

III.6 NEOS and COIN-OR
III.7 Constraints and linear programming
III.8 Quadratic programming and Lagrange multipliers
III.9 LP, approximation algorithms, and P=NP?

Handouts

- Maxims about Numerical Mathematics, Computers, Science and Life
- TOP500 November 2018 press release. These days the top machines are around 1e6 cores x 1e11 flops per core = 100 petaflops.
- `m24_lp.m`, `m24ran.m`, and `m25_faraday.m`
- Mathematics of the Faraday cage

Announcements

Assignment 3 returned at end of lecture.

Assignment 4 due 11:00 Tuesday 4 December (Week 9) at Andrew Wiles reception. We will post solutions at that time.

Next term: Tuesdays Weeks 1-7 and Thursdays Weeks 1-3 and 5-6, here in L3. ODEs, PDEs and dynamics. See you Tuesday, January 15.

III.6 NEOS and COIN-OR

NEOS, based at the U. of Wisconsin, used to stand for something like Network-Enabled Optimization Server. Now it doesn't stand for anything. They call it just the

NEOS Server: `www.neos-server.org/neos/`

This is a major resource for information, reports, software, and online solutions. It handles tens of thousands of jobs each month.

See also COIN-OR: `www.coin-or.org` ("Computational Infrastructure for Operations Research"). This site reminds us that so much of optimization sprang from the field of operations research.

III.7 Constraints and linear programming

Many optimization problems have equality and/or inequality constraints.
These are often more of a challenge than the "minimisation" part of the problem.

In linear programming (= LP), the problem is "all" constraints.

In simplest form: seek to find x defined by

$$\min_x f^T x \quad \text{with } Ax \leq b \text{ (componentwise)}$$

x = unknown n-vector
f = given n-vector
A = given $m \times n$ matrix
b = given m-vector

Example

In $X = (x, y)$-plane, seek point X minimizing $f^T X$, $f = (1, 2)^T$ in a triangular region [sketch ->] defined by

$$x \leq y, \quad x \geq -y, \quad y \leq 1$$

that is,

$$x - y \leq 0, \quad -x - y \leq 0, \quad y \leq 1,$$

that is,

```
| 1 -1|  |x|       |0|
|-1 -1|  |y|   <=  |0|
| 0  1|             |1|
```

The solution is obviously $(x, y) = (0, 0)$. But LP problems get far bigger than this.

Discussion of the remarkable importance and history of LP problems.

simplex method - Dantzig late 1940s (he died in 2005)

(How many Economics Nobel Prizes have been based on the simplex method?
Samuelson 1970?
Leontief 1973 for input-output analysis?
Kantorovich & Koopmans 1975 for optimal resource allocation?
Markowitz 1990 for portfolio optimization?)

interior point methods - Karmarkar 1980s

[Google search for "linear programming" - 5,000,000 hits]

[`m24_lp.m` and its variants `m24ran.m`, `m24ranbig.m`]

What Nobel prizes have been closely related to optimization more broadly? I don't have an answer, but one example would be *Density functional theory* (DFT), which won half of the 1998 Chemistry Nobel Prize: Walter Kohn. (The other half, John Pople, was also computational.) Other key names are Hohenberg and Sham.

In quantum mechanics, if you have n electrons, in principle you have to work with a Schrodinger PDE in a space of dimension $3n$. DFT replaces this with an optimization problem in such a space: minimize $E(x)$, where E is an energy functional of the positions of the electrons.

III.8 Quadratic programming and Lagrange multipliers

Let's take a step toward nonlinearity: quadratic programming with linear equality constraints.

Here is a problem with a single constraint. Imagine we seek to find $x \in R^n$ to minimize the energy functional

$$J(x) = \frac{1}{2}x^T Ax - f^T x \qquad \text{with } b^T x = c$$

where

x = unknown n-vector
f = given n-vector
A = given $n \times n$ SPD matrix
b = given n-vector
c = given scalar

More generally, nonlinear optimization problems are often solved by related techniques: locally, we approximate the objective function by a quadratic model and the constraints by a linear model. This is called **sequential quadratic programming (SQP)**.

The standard idea for solving such a problem is to introduce an unknown scalar λ known as a **Lagrange multiplier**. Let L be the **Lagrangian function**

$$L(x, \lambda) = \frac{1}{2}x^T Ax - f^T x + (b^T x - c)\lambda.$$

Now the partial derivative of L with respect to λ is

$$\frac{\partial L}{\partial \lambda} = b^T x - c$$

and the gradient of L with respect to x is

$$\nabla_x L = Ax - f + \lambda b.$$

Suppose we find a stationary point of $L(x, \lambda)$ with respect to both λ and x. Then $\partial L/\partial\lambda = 0$, so the constraint $b^T x = c$ is satisfied. And $\nabla_x L = 0$ tells us $Ax - f = -\lambda b$, or equivalently,

$$\nabla_x J(x) = -\lambda b.$$

Thus the gradient of J points in the direction of the vector b, i.e., orthogonal to the constraint hyperplane $b^T x = c$. So at the point x, $J(x)$ restricted to the constraint hyperplane has its minimum.

Setting these derivatives to zero amounts to a block 2×2 linear algebra problem:

$$\begin{pmatrix} A & b \\ b^T & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ c \end{pmatrix}$$

If we solve this system of equations, we have solved the quadratic programming problem.

One of my research projects of a couple of years ago involved exactly this mathematics: a study of the Faraday cage. Strangely, though this effect dates to 1836, there doesn't seem to be any literature on it until our own paper (Chapman, Hewett, Trefethen) in *SIAM Review* in 2015. To model Faraday shielding, you distribute charge on wires in such a way as to minimize a quadratic energy functional subject to the constraint that the total charge is zero. See handout.

`m25_faraday.m`

III.8 LP and the mysteries of P=NP?

The simple story from the 1970s

P: some problems can be solved in polynomial time. Tractable.

NP: other problems can only be solved in exponential time (assuming P $\neq$ NP). Intractable.

The gulf between P and NP will only get wider as time goes by and machines get faster.

And there's a beautiful theoretical challenge: to prove P $\neq$ NP (Cook & Levin, 1971). This is one of the million-dollar Millennium Prize Problems from the Clay Mathematics Institute (HQ in this building!).

This simple picture lies behind #9 and #10 in my "Maxims".

What actually happened over the decades

Computers got millions of times faster, but somehow, the P vs. NP gulf didn't get wider.

Many problems in NP are often highly tractable in practice. Examples are the Traveling Salesman problem and the Satisfiability problem ("SAT" — this was actually the very first problem to be proved NP-complete). In fact the latter has become one of the workhorses of practical large-scale computing. Knuth gave a SIAM von Neumann Lecture about it 2016 and never even mentioned that the problem was NP-complete! (See v. 4, fascicle 6 of his *Art of Computer Programming* and check out "SAT solvers" on Wikipedia.)

The P vs. NP distinction nowadays seems a bit academic. What's going on?

Explanation 1. Worst-case analysis can be misleading

The P vs. NP theory is based on worst-case analysis. Sometimes this captures the actual behavior of real problems, but sometimes it is wildly wrong. LP is a conspicuous example.

LP, it was eventually proved, is a problem in P, not NP.

The simplex method (dating from the 1940s), nevertheless, is an exponentially slow algorithm. Yet it's used all the time! Reason: "exponentially slow" pertains to the worst case. Those worst cases seem to arise exponentially rarely.

The alternative of interior point methods (dating from the 1980s) is polynomial instead of exponential. These are used too. But they haven't taken over as one might have expected based on P vs. NP theory.

Explanation 2. Optimal vs. near-optimal solutions

The P vs NP theory is based on finding an exact optimum. Often one can come provably close to this by polynomial algorithms, even though computing the exact optimum is a problem in NP.

Very often the polynomial algorithms are based on LP (and also sometimes on other problems of continuous optimization). Indeed, a rule of thumb may be that if a discrete problem is NP-hard, there may be a continuous approximation to it that is polynomial.

α-approximation algorithm: a polynomial-time algorithm that's guaranteed to come within a factor of α of optimal.

Book: Williamson and Shmoys, *The Design of Approximation Algorithms*, Cambridge, 2011.

Perhaps the most important other tool, besides continuous optimization, is randomization. For a range of problems, a combination of LP and randomization and problem-dependent heuristics have proved very powerful.

Examples of surprisingly tractable NP-hard problems

- Edge coloring
- FCC spectrum auction problem
- Integer programming
- Knapsack
- Maximum clique
- Maximum cut (this one is polynomial if you sacrifice 13%)
- Satisfiability
- Set cover
- Traveling salesman
- Vertex coloring

Other NP-hard problems don't yet have effective algorithms. Encryption seems to be one, thank goodness, since the world financial system depends on it.