Lecture 14

Iterative solvers for linear equations

Let $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ be given. We want to solve $Ax = b$. What is the fastest method you know?

We usually learn how to solve systems of linear equations via Gaussian elimination, or perhaps inverting A by first using the SVD, or some other kind of matrix factorization technique such as the LU decomposition. These methods can be very slow; for instance computing the inverse takes $O(n^3)$ time, and even just writing it down will usually require $O(n^2)$ space, even if the original matrix A has a very efficient representation — for instance, it is *sparse*, i.e. it contains only few non-zero entries (think of the adjacency matrix of a graph with low degree).

What we really want is an algorithm whose running time is related to the input size. If A is sparse, it can be specified succinctly by listing only the non-zero entries; the running time of any good algorithm should scale accordingly.

Today we'll see how to achieve this using *iterative methods*. These kinds of algorithms only involve a sequence of matrix-vector multiplications by A (or related matrices). Any such multiplication can be done in time $O(m)$ where m is the number of nonzero entries of A, and hence the sparser A is the faster the algorithm will run. A drawback of these methods is that they never return the exact solution; instead the algorithm will slowly construct a better and better approximation to the solution x, and the running time will depend on the quality of approximation desired.

14.1 First-order Richardson iteration

For any $\alpha \in \mathbb{R}$,

$$\begin{aligned} Ax = b &\iff \alpha Ax = \alpha b \\ &\iff x = (\mathbb{I} - \alpha A)x + \alpha b. \end{aligned}$$

What's the point? We've written x as the fixed point of a simple linear equation. This suggests the following iterative process: set x^0 to anything, say $x^0 = 0$, and then iterate

$$x^t = (\mathbb{I} - \alpha A)x^{t-1} + \alpha b.$$

Note that if this process converges, $x^t \to_{t\to\infty} x^*$ for some $x^* \in \mathbb{R}^n$, then necessarily $Ax^* = b$. We are going to show that there is convergence as long as $\mathbb{I} - \alpha A$ has largest singular value strictly less than 1, and convergence time will depend on how much smaller than 1 it is.

For simplicity assume A is symmetric positive definite (this is not strictly necessary, but convenient), with eigenvalues $0 < \lambda_1 \leq \cdots \leq \lambda_n$. Then the eigenvalues of $\mathbb{I} - \alpha A$ are $1 - \alpha\lambda_i$. Its largest singular value is $\max_i |1 - \alpha\lambda_i|$, and this is minimized by taking $\alpha = 2/(\lambda_1 + \lambda_n)$, in which case we get $1 - 2\lambda_1/(\lambda_1 + \lambda_n)$ for the norm. (Note that we may not know $\lambda_1 + \lambda_n$ a priori. A good guess will do though, as any choice of α such that $\alpha < 2/(\lambda_1 + \lambda_n)$ will work.)

Now let's show convergence. If x^* is such that $Ax^* = b$, we can write for any $t \geq 0$

$$\begin{aligned} x^* - x^t &= ((\mathbb{I} - \alpha A)x^* + \alpha b) - ((\mathbb{I} - \alpha A)x^{t-1} + \alpha b) \\ &= (\mathbb{I} - \alpha A)(x^* - x^{t-1}), \end{aligned}$$

so

$$\begin{aligned} \left\|x^* - x^t\right\| &= \left\|(\mathbb{I} - \alpha A)^t (x^* - x^0)\right\| \\ &\leq \left\|\mathbb{I} - \alpha A\right\|^t \left\|x^*\right\| \\ &\leq 2^{-\frac{2\lambda_1}{\lambda_1+\lambda_n}t} \left\|x^*\right\|. \end{aligned}$$

Therefore the number of iterations required to get relative error $\|x^* - x^t\|/\|x^*\| \leq \varepsilon$ is at most $(\lambda_1 + \lambda_n)/(2\lambda_1)\ln(1/\varepsilon)$. The important term here is λ_n/λ_1, which is called the *condition number* of A — the smaller the better. It is usually denoted κ and is a measure of how "skewed" the linear transformation implemented by A is.

14.2 Speeding up via polynomials

Now here is a really nice trick that gives a quadratic speed-up for the previous algorithm. The vector x^t constructed at the t-th step of the algorithm can be written as

$$x^t = \sum_{i=0}^{t} (\mathbb{I} - \alpha A)^i (\alpha b) = p^t(A)b,$$

where $p_t(A) = \sum_{i=0}^{t} \alpha(\mathbb{I} - \alpha A)^i$. Now if we take $t \to \infty$, what do you recognize?

$$\sum_{i=0}^{\infty} \alpha(\mathbb{I} - \alpha A)^i = \alpha(\mathbb{I} - (\mathbb{I} - \alpha A))^{-1} = \alpha(\alpha A)^{-1} = A^{-1},$$

provided the series converges, which is the case as long as $\|\mathbb{I} - \alpha A\| < 1$. This should be no surprise, since we are solving for $x = A^{-1}b$: what the previous algorithm is doing is simply finding a good polynomial approximation to the inverse!

Now, the obvious question is, can we do better, where here better means obtaining an approximation of similar quality but using a polynomial with a lower degree —- this way we'd have to perform fewer iterations, and thus get a faster algorithm.

What do we need exactly? Given a matrix A, we want p^t such that $\|p^t(A)A - \mathbb{I}\| \leq \varepsilon$, i.e. $|p^t(\lambda_i)\lambda_i - 1| \leq \varepsilon$ for $i = 1, \ldots, n$, where λ_i are the eigenvalues of A (assume again that A is symmetric positive definite, and scale things so that $\|A^{-1}\| \leq 1$). If we have this, then

$$\begin{aligned}\|p_t(A)b - x^*\| &= \|p_t(A)Ax^* - x^*\| \\ &= \|(p_t(A)A - \mathbb{I})x^*\| \\ &\leq \|p_t(A)A - \mathbb{I}\|\|x^*\| \\ &\leq \varepsilon\|x^*\|,\end{aligned}$$

as desired. The following theorem gives us what we need.

Theorem 14.1. *For every $t \geq 1$ and $0 < \lambda_1 \leq \lambda_n$, let $\kappa = \lambda_n/\lambda_1$. There is a polynomial $q^t(x)$ of degree t such that $q^t(0) = 1$ and*

$$|q^t(x)| \leq 2\Big(1 + \frac{2}{\sqrt{\kappa}}\Big)^t \leq 2e^{-\frac{2t}{\sqrt{\kappa}}}, \qquad \forall x \in [\lambda_1, \lambda_n].$$

To see why this is just what we need, note that $q^t(0) = 1$ implies $q^t(x) = 1 - xp^t(x)$ for some p^t of degree $t - 1$. Using this p^t, we'll get an approximation with error ε as long as t is at least $\ln(2/\varepsilon)\sqrt{\kappa}/2$: as promised, a quadratic improvement over the previous algorithm.

The proof of Theorem 14.1 uses a construction based on *Chebyshev polynomials*. If you've never seen those they are worth a look — you'll prove the theorem in your homework.

14.3 A different expansion

Here's another idea. Recall how we're trying to find a fast-converging series expansion of A^{-1}. Suppose $A = \mathbb{I} - B$ for some B with norm strictly less than 1. For instance, B is the normalized adjacency matrix of a graph, and A its normalized Laplacian. Well, not quite — in that case we know that (as long as the graph is connected) B will have one eigenvalue equal to 1. But in fact this is fine, as everything we'll see can just be carried out in the space orthogonal to the $\mathbf{1}$ vector. So we'll ignore the issue for now, and just assume $\|B\| < 1$.

The expansion we used earlier amounts to writing $(\mathbb{I} - B)^{-1} = \mathbb{I} + B + B^2 + \ldots$. This can be written much more succinctly:

$$\mathbb{I} + B + B^2 + B^3 + \ldots = (\mathbb{I} + B)(\mathbb{I} + B^2)(\mathbb{I} + B^4)(\mathbb{I} + B^{2^3}) \cdots . \tag{14.1}$$

To see why this works, expand e.g.

$$(\mathbb{I}+B)(\mathbb{I}+B^2)(\mathbb{I}+B^4) = (\mathbb{I}+B+B^2+B^3)(\mathbb{I}+B^4) = (\mathbb{I}+B+B^2+B^3)+B^4(\mathbb{I}+B+B^2+B^3),$$

etc. So we get 2^t terms in the expansion by taking only t successive products by matrices of the form $(\mathbb{I} + B^{2^i})$. Using the analysis we gave earlier, it is enough to have $2^t \approx \kappa \ln(1/\varepsilon)$, which means that we can take t of order $\ln(\kappa)$: exponential savings!

There is a catch however: now, even if B is sparse to start with, the matrices B^{2^i} will quickly no longer be sparse, and so we lose the main advantage we had from an iterative method in the first place, which is that the complexity of the matrix-vector multiplications to be performed scaled with the number of nonzero entries of A.

In the next lecture we'll see how this can be solved for matrices B that have a special structure — specifically, they are normalized adjacency matrices of graphs. The idea will be to *sparsify* the graph: we'll interpret each B^{2^i} as the adjacency matrix of a certain graph, and figure out a way to approximate it by a *sparse* graph.