18.409 An Algorithmist's Toolkit November 21, 2009

Lecture 21

Lecturer: Jonathan Kelner *Scribe: Yan Zhang*

1 Solving Linear Systems: A Brief Overview

Given an invertible $n \times n$ matrix A and an n-vector b, we would like to solve the matrix equation $Ax = b$. One way to do so is to invert A and multiply both sides by A^{-1}. While this approach is theoretically valid, there are several problems with it in practice. Computing the inverse of a large matrix is expensive and susceptible to numerical error due to the finite precision of floating-point numbers. Moreover, matrices which occur in real problems tend to be sparse and one would hope to take advantage of such structure to reduce work, but matrix inversion destroys sparsity.

So what would a numerical analyst do? A better method is Gaussian elimination, or equivalently, LU factorization followed by back-substitution. This technique is competitive when the matrix A is dense and unstructured, and it also has the advantage of allowing solution of $Ax = b$ for multiple values of b with little additional work. However, it still fails to make use of the fact that systems encountered in practice are rarely dense and unstructured. As we will see in the next few lectures, iterative methods are the technique of choice for solving such systems.

2 A First Iterative Method

2.1 An Example

Consider the system

$$\begin{pmatrix} 100 & 3 & -2 \\ 1 & 200 & 5 \\ -4 & 3 & 100 \end{pmatrix} x = \begin{pmatrix} 800 \\ 1000 \\ 500 \end{pmatrix}.$$

While computing the exact solution by hand would be a tedious task, it is a simple matter to find an approximate solution. Roughly speaking, we expect the behavior of our system to be governed by the "large" diagonal entries of the matrix A, so if we just pretend that all off-diagonal entries are zero, the solution we obtain should still be reasonably close to the correct answer. Of course, once we ignore the off-diagonal entries, solving the system is easy, and we get as a first approximation $x_1 = (8, 5, 5)^T$.

How close does our approximation come to solving the system? Multiply A by x_1 to get $(805, 1033, 483)^T$. Subtracting from the desired result b, we find that we are off by $e_1 = (-5, -33, 17)^T$. Now this suggests a way to improve our estimate: since our system is linear, we can adjust our approximation x_1 by applying the same technique as before with e_1 on the right rather than b. Adding this adjustment gives an improved approximation $x_2 = (7.95, 4.835, 5.17)^T$, and clearly we can iterate the procedure as many times as we wish in hopes of obtaining better and better estimates converging to the true solution. It turns out that in this example one more iteration already achieves accuracy of about four significant figures: our next approximation is $(7.9584, 4.8310, 5.1730)^T$, while the actual answer is $(7.9585, 4.8309, 5.1734)^T$ to four decimals. In fact, convergence is exponential: the number of correct digits increases linearly with the number of iterations.

2.2 A Bit Harder

One might argue that the above example was contrived, since our approximation scheme depended on the fact that the diagonal entries of A were much larger than the off-diagonal entries. However, consider the

system

$$\begin{pmatrix} 100 & -1 & -4 \\ 100 & 100 & 3 \\ 100 & 100 & 100 \end{pmatrix} x = \begin{pmatrix} 100 \\ 200 \\ 300 \end{pmatrix}.$$

Again, while computing the exact answer would take some work, we can tell at a glance that the solution should be close to $(1, 1, 1)^T$. In this case, the above-diagonal entries are all small, and once we ignore these, we can easily solve the remaining lower-triangular system. As before, we may now iteratively improve our solution by finding the error and repeating the procedure, converging geometrically to the correct answer.

2.3 General Idea

Why do both of these matrices work? One was "almost diagonal" while the other was "almost lower-triangular." This suggests that the important common attribute of the matrices A is the existence of a decomposition

$$A = L + S,$$

where L is "large"—accounting for "most of A"—and easy to invert, while S is "small." We reason that L^{-1} would thereby be a good approximation of A^{-1}. Therefore, we define

$$\begin{aligned} x_1 &= L^{-1}b, \\ r_1 &= b - Ax_1. \end{aligned}$$

We can perform iterative updates according to

$$\begin{aligned} x_{k+1} &= x_k + L^{-1}r_k, & (1) \\ r_{k+1} &= b - Ax_{k+1}. & (2) \end{aligned}$$

In the k-th stage, x_k is our current approximate solution to $Ax = b$ and r_k is called the *residual*.

Note that this iterative approach never requires us to invert A: instead, we need only know how to multiply vectors by L^{-1}. Aside from this, only the (inexpensive) operations of matrix-vector multiplication and vector arithmetic are required. Thus, if we know an efficient way of computing $L^{-1}y$ given a vector y—or alternatively, are given a "black box" that performs this operation—then we may infer a method for approximately solving $Ax = b$ which may be much faster than the standard techniques for computing the exact solution.

2.4 Analysis

Of course, for this method to be useful, we need to know that our iterations do actually improve our estimate. We would also like a bound on the improvement at each stage so that we know when to stop. To obtain these results, we need to make precise the notions of L and S being "large" and "small."

Consider the product

$$L^{-1}A = L^{-1}(L + S) = I + L^{-1}S.$$

This gives us some intuition that L^{-1} should be a good approximation of A^{-1} when $L^{-1}S$ is "small" compared to the identity matrix I. Proceeding with the analysis, let x denote the actual solution to $Ax = b$. Substituting $A = L + S$, we get $Lx = -Sx + b$, or equivalently,

$$x = -L^{-1}Sx + L^{-1}b.$$

Define $M = -L^{-1}S$ and $z = L^{-1}b$ and observe that we can rewrite our iterative step as the recurrence

$$\begin{aligned} x_{k+1} &= x_k + L^{-1}r_k \\ &= x_k + L^{-1}(b - Ax_k) \\ &= x_k + L^{-1}b - L^{-1}Lx_k - L^{-1}Sx_k \\ &= Mx_k + z. \end{aligned}$$

Note that x is a fixed point of this recurrence because it leaves zero residual: $r = b - Ax = 0$ by definition of x. In other words, $x = Mx + z$.

Now define the *error* at step k to be $e_k = x_k - x$ and observe

$$\begin{aligned} e_{k+1} &= x_{k+1} - x \\ &= Mx_k + z - x \\ &= M(x + e_k) + z - x \\ &= (Mx + z - x) + Me_k \\ &= Me_k. \end{aligned}$$

It follows immediately that $e_k = M^{k-1}e_1$, and in fact

$$e_k = -M^k x,$$

since we could have started our iteration at $x_0 = 0$ in which case $e_0 = -x$. Thus, we can think of the error growing roughly as a matrix power[1]. We pause here to make a definition.

Definition 1 *The* spectral radius ρ *of a symmetric matrix M is the absolute value of its largest eigenvalue:*

$$\rho = |\lambda_{\max}|.$$

Observe that it follows from the definition that (in the symmetric case)

$$||M^n x|| \leq \rho^n ||x||,$$

so if $\rho < 1$, then powers of M converge exponentially to zero at a rate given by ρ. The same holds for general M if we replace "eigenvalue" by "singular value." Summarizing, we have the following result.

Theorem 2 *Suppose A is a square matrix admitting a decomposition $A = L + S$ where L is invertible and the largest singular value of $L^{-1}S$ has absolute value $\rho < 1$. Then the iteration given by (1), (2) for solving $Ax = b$ converges to the correct answer as ρ^k.*

2.5 Further Remarks

As a side note, the two specific examples we began with are cases of *Jacobi iteration*, in which the matrix A is decomposed as $D + S$ with D diagonal and S small; and *Gauss-Seidel iteration*, where $A = L + S$ with L lower triangular and S small.

Also, one may wonder why we want to work specifically with matrices that look like these. One good explanation is that in physics, many "natural" matrices tend to have larger diagonal values, since we are considering the transition matrix of a physical state near equilibrium.

3 Setup for More Iterative Methods

3.1 Assumptions

For the remainder of this lecture, we will restrict our attention to solving $Ax = b$ for $n \times n$ square matrices A that are symmetric and positive definite. Note that positive definiteness implies nonsingularity. These conditions may at first glance appear to be very restrictive, but in fact we claim we can reduce any nondegenerate square linear system to such a problem. Indeed, we need only observe that for an invertible matrix A,

$$Ax = b \quad \text{iff} \quad A^T Ax = A^T b,$$

[1] This is similar to our analysis of stablization in random walks!

and the matrix A^TA is positive definite.

It is worth noting that while it is clear that the above reduction is theoretically valid, it is less clear whether or not such a reduction is practical. While the matrix product A^TA has the advantage of positive definiteness, it raises several other concerns. For one, matrix multiplication could be as expensive as solving the system in the first place and could destroy sparsity properties. Additionally, one might worry about the effects of replacing A with A^TA on convergence speed and condition number. As we shall see, however, the trick to getting around these issues is to never actually compute A^TA. Instead, since our algorithms will only use this matrix in the context of multiplying by a vector, we can perform such multiplications from right to left via two matrix-vector multiplications, thus avoiding the much more expensive matrix-matrix multiplication.

3.2 Converting a Linear Problem to a Quadratic One

Having assumed now that we are dealing with a symmetric positive definite matrix A, we can recast our linear system $Ax = b$ as the condition that the vector x minimizes the quadratic form

$$f(x) = \frac{1}{2}x^TAx - bx + c.$$

Indeed, the gradient of f is given by

$$\nabla f(x) = \frac{1}{2}(A + A^T)x - b = Ax - b$$

because A is symmetric, and since A is positive definite, the quadratic form f is strictly convex, hence has a unique minimizer x given by $\nabla f(x) = 0$. In this case, level (contour) sets of $f(x)$ are ellipsoids with axes along the eigenvectors of A and axis lengths inversely proportional to the eigenvalues of A.

What happens if our assumptions on A are violated? If A is nonsymmetric, vanishing of the gradient is no longer equivalent to the condition $Ax = b$: instead, we get $\frac{1}{2}(A + A^T)x = b$. If A is negative definite, $\nabla f(x) = 0$ gives a maximum rather than a minimum, and if A is symmetric but neither positive nor negative definite, then vanishing of the gradient generally gives a saddle point. For more geometric intuition and figures (some of which are reproduced in the lecture slides), we refer to [1].

4 Steepest Descent

4.1 Motivation

We now discuss the technique of steepest descent, also known as *gradient descent*, which is a general iterative method for finding local minima of a function f. The idea is that given a current estimate x_i, the gradient $\nabla f(x_i)$—or more precisely, its negative—gives the direction in which f is decreasing most rapidly. Hence, one would expect that taking a step in this direction should bring us closer to the minimum we seek. Keeping with our previous notation, we will let x denote the actual minimizer, x_i denote our i-th estimate, and

$$e_i = x_i - x, \tag{3}$$
$$r_i = b - Ax_i = -Ae_i \tag{4}$$

denote the i-th error term and residual, respectively.

The question now is how to decide what step size to use at each iteration. A logical approach is to choose the step α_i such that the updated estimate $x_{i+1} = x_i - \alpha_i \nabla f(x_i)$ minimizes $f(x_{i+1})$ among all such x_{i+1}. In general, the solution to this *line search* may or may not have a closed form, but in our case of f a quadratic form, we can determine the minimizing α_i explicitly. Indeed, we need only notice that at the minimum along a line, the gradient is orthogonal to the line. Now the negative gradient at the $i + 1$-st step

$$-\nabla f(x_{i+1}) = b - Ax_{i+1} = r_{i+1}$$

turns out just to equal the $i+1$-st residual, so our orthogonality relation reduces to the condition that successive residuals be orthogonal:

$$r_{i+1}^T r_i = 0.$$

Expanding out

$$\begin{aligned} r_{i+1} &= b - Ax_{i+1} \\ &= b - A(x_i + \alpha_i r_i) \\ &= r_i - \alpha_i A r_i \end{aligned}$$

and substituting into the previous equation gives (using $A = A^T$)

$$\alpha r_i^T A r_i = \alpha (A r_i)^T r_i = r_i^T r_i,$$

and thus we have a formula for computing the step size along r_i in terms of just r_i itself.

Remark It is important to remember that the *residuals* $r_i = b - Ax_i$ measure the difference between our objective b and the result Ax_i of our approximation in "range space," whereas the *errors* $e_i = x_i - x$ measure the difference between our approximation and the true solution in "domain space." Thus, the previous orthogonality relation that holds for residual vectors does *not* mean that successive error vectors in the domain are orthogonal. It *does*, however, imply that successive *differences* between consecutive approximations are orthogonal because these differences $x_{i+1} - x_i = \alpha_i r_i$ are proportional to the residuals.

4.2 Algorithm

To summarize the development thus far, we have obtained an iterative algorithm for steepest descent with the following update step:

$$r_i = b - Ax_i \tag{5}$$

$$\alpha_i = \frac{r_i^T r_i}{r_i^T A r_i} \tag{6}$$

$$x_{i+1} = x_i + \alpha_i r_i. \tag{7}$$

As an implementation note, we point out that the runtime of this algorithm is dominated by the two matrix-vector multiplications: Ax_i (used to compute r_i) and Ar_i (used in finding the step size α_i). In fact, it is enough to do just the latter multiplication because as we saw before, we can alternatively write

$$r_{i+1} = r_i - \alpha_i A r_i,$$

so that after the first step we can find residuals by reusing the computation Ar_i, which was already done in the previous step. In practice, one needs to be careful about accumulation of roundoff errors, but this problem may be resolved by using (5) every once in a while to recalibrate.

4.3 Analysis

Before dealing with general bounds on the rate of convergence of steepest descent, we make the preliminary observation that in certain special cases, steepest descent converges to the exact solution in just one step. More precisely, we make the following claim.

Claim 3 *If the current error vector e_i is an eigenvector of A, then the subsequent descent step moves directly to the correct answer. That is, $e_{i+1} = 0$.*

Proof Apply (5)–(7) and the definition of the error (3) to find

$$e_{i+1} = e_i + \frac{r_i^T r_i}{r_i^T A r_i} r_i, \tag{8}$$

giving the change in the error from step i to step $i+1$. In the case that e_i is an eigenvector of A, say with eigenvalue λ, we have from (4) that $r_i = -Ae_i = -\lambda e_i$, and hence (8) reduces to

$$e_{i+1} = e_i + \frac{1}{\lambda}(-\lambda e_i) = 0.$$

■

Remark The above result tells us that steepest descent works instantly for error vectors in the eigenspaces of A. These spaces have dimensions equal to the multiplicities of the corresponding eigenvalues, and in particular, if A is a multiple of the identity, then steepest descent converges immediately from any starting point. In general, we are not nearly so lucky and the eigenspaces each have dimension 1, but it is worth noting that even in this case convergence is qualitatively different from that of our first iterative approach: there are particular directions along which steepest descent works perfectly, whereas our first approach only gave the correct answer in the trivial case in which the error was already zero.

In light of the preceding remark, we can expect that convergence should be faster along some directions than others, and we will see that this is indeed the case. Before jumping headlong into the convergence analysis, however, it is worthwhile to define a more convenient measure of error.

Definition 4 *The* energy norm *of a vector e is given by*

$$||e||_A = e^T A e. \tag{9}$$

Motivation for this definition will be provided in the next lecture; for now, we simply take for granted that it obeys the usual properties of a norm—and hence produces the same qualitative notion of convergence—but lends itself to a cleaner convergence bounds. We will satisfy ourselves with simply stating the result and focus on discussing its consequences, since the proof is just a computation using (8) and (9). A more intuitive line of reasoning will also come in the next lecture.

Theorem 5 *Let e_i denote the error vector at step i of steepest descent. Let $\{v_j\}_{j=1}^n$ be a normalized eigenbasis of A with corresponding eigenvalues λ_j, and let $e_i = \sum_j \xi_j v_j$ denote the expansion of e_i with respect to this eigenbasis. Then*

$$||e_{i+1}||_A^2 = ||e_i||_A^2 \left(1 - \frac{(\sum_j \xi_j^2 \lambda_j^2)^2}{(\sum_j \xi_j^2 \lambda_j^3)(\sum_j \xi_j^2 \lambda_j)}\right). \tag{10}$$

The general result (10) is quite a mouthful, but fortunately we can understand its flavor just by looking at the two-dimensional case. In this case we have only two eigenvectors v_1 and v_2. Assume $\lambda_1 > \lambda_2$, so the *condition number* of A is $\kappa = \lambda_1/\lambda_2$. Define $\mu = \xi_1/\xi_2$ to be the ratio of the components of e_i along the basis vectors. Then (10) simplifies to

$$\frac{||e_{i+1}||_A^2}{||e_i||_A^2} = 1 - \frac{(\kappa^2 + \mu^2)^2}{(\kappa + \mu^2)(\kappa^3 + \mu^2)}.$$

Note that the form of the expression on the right corroborates our preliminary observations. If the condition number $\kappa = 1$, convergence occurs instantly, and if κ is close to 1, convergence occurs quickly for all values of μ. If κ is large, convergence still occurs instantly if $\mu = 0$ or ∞, but now the rate of convergence varies substantially with μ, with the worst case being when e_i is closer to the smaller eigenvector than the larger one by a factor of κ, i.e., $\mu = \pm\kappa$ (see the lecture slides or [1] for helpful pictures).

4.4 Some Motivation

To summarize, we have seen that the performance of steepest descent varies depending on the error direction and can sometimes be excellent; however, in the worst case (obtained by maximizing the factor on the right side of (10) over all ξ_j) convergence is still only geometric.

The problem, as can be seen in the lecture figures, is that steepest descent has the potential to "zig-zag too much." We will see in the next lecture how the method of *conjugate gradients* overcomes this issue. The big idea here is that the so-called "zig-zagging" comes from situations when the ellipsoidal curves are very skew; the disparity between the magnitudes of the axes of the ellipses causes us to take very tiny steps. Note we can then think of the energy norm is really a normalization of the ellipses into spheres, which removes this issue.

References

[1] Shewchuk, Jonathan. "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain." August 1994. `http://www.cs.cmu.edu/~jrs/jrspapers.html`.

18.409 Topics in Theoretical Computer Science: An Algorithmist's Toolkit
Fall 2009