Krylov Subspace Methods

Tyler Chen

Introduction

Solving a linear system of equations $Ax = b$ is one of the most important tasks in modern science. Applications such as weather forecasting, medical imaging, and training neural nets all require repeatedly solving linear systems.

Loosely speaking, methods for linear systems can be separated into two categories: direct methods and iterative methods. Direct methods such as Gaussian elimination manipulate the entries of the matrix A in order to compute the solution $x = A^{-1}b$. On the other hand, iterative methods generate a sequence $x_0, x_1, x_2, \ldots$ of approximations to the true solution $x = A^{-1}b$, where hopefully each iterate is a better approximation to the true solution.

At first glance, it may seem that direct methods are better. After all, after a known number of steps you get the exact solution. In many cases this is true, especially when the matrix A is dense and relatively small. The main drawback to direct methods is that they are not able to easily take advantage of sparsity. That means that even if A has some nice structure and a lot of entries are zero, direct methods will take the same amount of time and storage to compute the solution as if A were dense. This is where iterative methods come in. Often times iterative methods require only that the product $x \mapsto Ax$ be able to be computed. If A is sparse the product can be done cheaply, and if A has some known structure, a you might not even need to construct A. Such methods are aptly called "matrix free".

The rest of this piece gives an introduction to Krylov subspace methods, a common type of iterative method. My intention is not to provide a rigorous explanation of the topic, but rather to provide some (hopefully useful) intuition about where these methods come from and why they are useful in practice. I assume some linear algebra background (roughly at the level of a first undergrad course in linear algebra).

If you are a bit rusty on your linear algebra I suggest taking a look at [some resource]. If you want a more rigorous introduction to iterative methods I suggest Anne Greenbaum's book. For a overview of modern analysi of the Lanczos and Conjugate Gradient methods I suggest Gerard Meurant and Zdenek Strakos's report.

Krylov subspaces

sequence of subspaces contained in the previous

1. easy to construct
2. eventually should contain solution
3. easy to optimize (some quantity) over

The k-th Krylov subspace generated by a square matrix A and a vector v is defined to be,

$$\mathcal{K}_k(A, v) = \text{span}\{v, Av, \ldots, A^{k-1}v\}$$

These subspaces are relatively easy to construct because we can get them by repeatedly applying A to v. On the other hand, it's not so clear that they will ever contain our solution $x = A^{-1}b$ or that it is easy to optimize over them.

We first show that $\mathcal{K}_k(A, b)$ will eventually contain our solution by the time $k = n$. While this result comes about naturally later from our description of some algorithms I think it is useful to immediately relate polynomials with Krylov subspace methods, as the two are intimately related.

Suppose A has characteristic polynomial,

$$p_A(t) = \det(tI - A) = c_0 + c_1 t + \cdots + c_{n-1}t^{n-1} + t^n$$

It turns out that $c_0 = (-1)^n \det(A)$ so that c_0 is nonzero if A is invertible.

The Cayley-Hamilton Theorem states that a matrix satisfies its own characteristic polynomial. This means,

$$0 = p_A(A) = c_0 I + c_1 A + \cdots c_{n+1}A^{n-1} + A^n$$

Moving the identity term to the left and dividing by $-c_0$ we can write,

$$A^{-1} = -(c_1/c_0)I - (c_2/c_0)A + \cdots - (1/c_0)A^{n-1}$$

This says that A^{-1} can be written as a polynomial in A! In particular,

$$x = A^{-1}b = -(c_1/c_0)b - (c_2/c_0)Ab + \cdots - (1/c_0)A^{n-1}b$$

That means that the solution to $Ax = b$ is a linear combination of $b, Ab, A^2b, \ldots, A^{n-1}b$. This observation is the motivation behind Krylov subspace methods. In particular, Krylov subspace methods work by building low degree polynomial approximations to A^{-1} using powers of A.

We then see that $x = A^{-1}b \in \mathcal{K}_n(A, b)$.

Arnoldi Algorithm

The Arnoldi algorithm for computing an orthonormal basis for Krylov subspaces is at the core of most Krylov subspace methods. Essentially, the Arnoldi algorithm is the Gram-Schmidt procedure applied to the vectors $v, Av, A^2v, A^3v, \ldots$ in a clever way.

Recall that given a set of vectors $v_0, v_1, \ldots, v_k$ the Gram-Schmidt procedure computes an orthonormal basis $q_0, q_1, \ldots, q_k$ so that for all $j \leq k$,

$$\text{span}\{v_0, \ldots, v_j\} = \text{span}\{q_0, \ldots, q_j\}$$

The trick behind the Arnoldi algorithm is the fact that you do not need to construct the whole set $v, Av, A^2v, \ldots$ ahead of time. Instead, you can compute Aq_j in place of $A^{j+1}v$ once you have found an orthonormal basis $q_0, q_1, \ldots, q_j$ spanning $v, Av, \ldots, A^jv$.

If we assume that $\text{span}\{v, Av, \ldots A^jv\} = \text{span}\{q_0, \ldots, q_j\}$ then q_j can be written as a linear combination of $v, Av, \ldots, A^jv$. Therefore, Aq_j will be a linear combination of $Av, A^2v, \ldots, A^{j+1}v$. In particular, this means that $\text{span}\{q_0, \ldots, q_j, Aq_j\} = \text{span}\{v, Av, \ldots, A^{j+1}v\}$. Therefore, we will get exactly the same set of vectors by applying Gram-Schmidt to $\{v, Av, \ldots, A^kv\}$ as if we compute Aq_j once we have computing q_j.

Since we obtain q_{j+1} by orthogonalizing Aq_j against $\{q_0, q_1, \ldots, q_j\}$ then q_{j+1} is in the span of these vectors. That means we can can some c_i so that,

$$q_{j+1} = c_0q_0 + c_1q_1 + \cdots + c_jq_j + c_{j+1}Aq_j$$

If we rewrite using new scalars d_i we have,

$$Aq_j = d_0q_0 + d_1q_1 + \cdots + d_{j+1}q_{j+1}$$

This can be written in matrix form as,

$$AQ = QH$$

where H is "upper Hessenburg" (like upper triangular but the first subdiagonal also has nonzero entries). In fact, while we have not showed it, the entries of H come directly from the Arnoldi algorithm (just like how the entries of R in a QR factorization can be obtained from Gram Schmidt).

Conjugate Gradient

When A is Hermetian, then $Q^*AQ = H$ is also Hermetian. Since H is upper Hessenburg and Hermitian, it must be tridiagonal! This means that the q_j satisfy a three term recurrence,

$$Aq_j = \beta_{j-1}q_{j-1} + \alpha_jq_j + \beta_jq_{j+1}$$

where $\alpha_1, \ldots, \alpha_n$ are the diagonal entries of T and $\beta_1, \ldots, \beta_{n-1}$ are the off diagonal entries of T.

When A is symmetric positive definite (all eigenvalues are positive) then a common method for solving $Ax = b$ is Conjugate Gradient.