

The Arnoldi and Lanczos algorithms

Tyler Chen

The Arnoldi and Lanczos algorithms for computing an orthonormal basis for Krylov subspaces are at the core of most Krylov subspace methods. Essentially, these algorithms are the Gram-Schmidt procedure applied to the vectors v, Av, A^2v, A^3v, \dots in a clever way.

The Arnoldi algorithm

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

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

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

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

Since we obtain q_{k+1} by orthogonalizing Aq_k against $\{q_0, q_1, \dots, q_k\}$ then q_{k+1} is in the span of these vectors, there exist some c_i so that,

$$q_{k+1} = c_0 q_0 + c_1 q_1 + \dots + c_k q_k + c_{k+1} Aq_k$$

We can rearrange this (using new scalars d_i) to,

$$Aq_k = d_0 q_0 + d_1 q_1 + \dots + d_{k+1} q_{k+1}$$

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). While I’m not going to derive them here, since 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) their explicit expressions can be easily written down.

Since Q is orthogonal then, $Q^*AQ = H$, so H and A are similar. This means that finding the eigenvalues and vectors of H will give us the eigenvalues and vectors of A . However, since H is upper Hessenburg, then solving the eigenproblem is easier than for a general matrix.

The Lanczos algorithm

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,

$$Aq_k = \beta_{k-1}q_{k-1} + \alpha_kq_k + \beta_kq_{k+1}$$

where $\alpha_1, \dots, \alpha_n$ are the diagonal entries of T and $\beta_1, \dots, \beta_{n-1}$ are the off diagonal entries of T . The Lanczos algorithm is an efficient way of computing this decomposition.

I will present a brief derivation for the method motivated by the three term recurrence above. Since we know that the q_k satisfy the three term recurrence, we would like the method to store as few of the q_k as possible (i.e. take advantage of the three term recurrence as opposed to the Arnoldi algorithm).

Suppose that we have q_k , q_{k-1} , and the coefficient β_{k-1} . We need to expand the Krylov subspace to find q_{k+1} in a way that takes advantage of the three term recurrence. To do this we can expand the subspace by computing Aq_k and then orthogonalizing Aq_k against q_k and q_{k-1} . By the three term recurrence, Aq_k will be orthogonal to q_j for all $j \leq k-2$ so we do not need to explicitly orthogonalize against those vectors.

We orthogonalize,

$$\tilde{q}_{k+1} = Aq_k - \alpha_kq_k - \langle Aq_k, q_{k-1} \rangle q_{k-1}, \quad \alpha_k = \langle Aq_k, q_k \rangle$$

and finally normalize,

$$q_{k+1} = \tilde{q}_{k+1} / \beta_j, \quad \beta_j = \|\tilde{q}_{k+1}\|$$

Note that this is not the most “numerically stable” form of the algorithm, and care must be taken when implementing the Lanczos method in practice. We can improve stability slightly by using $Aq_k - \beta_{k-1}q_{k-1}$ instead of Aq_k when finding a vector in the next Krylov subspace. This allows us to ensure that we have orthogonalized q_{k+1} against q_k and q_{k-1} rather than just q_k . It also

ensures that the tridiagonal matrix produces is symmetric in finite precision (since $\langle Aq_k, q_{k-1} \rangle$ may not be equal to β_j in finite precision).

We can implement Lanczos iteration in numpy. Here we assume that we only want to output the diagonals of the tridiagonal matrix T , and don't need any of the vectors.

```
def lanczos(A,q0,max_iter):
    alpha = np.zeros(max_iter)
    beta = np.zeros(max_iter)
    q_ = np.zeros(len(q0))
    q = q0/np.sqrt(q0@q0)

    for k in range(max_iter):
        qq = A@q-(beta[k-1]*q_ if k>0 else 0)
        alpha[k] = qq@q
        qq -= alpha[k]*q
        beta[k] = np.sqrt(qq@qq)
        q_ = np.copy(q)
        q = qq/beta[k]

    return alpha,beta
```