Notes for 2016-03-09

Orthogonal iteration to QR

The *QR iteration* is the workhorse for solving the nonsymmetric eigenvalue problem. Unfortunately, while the iteration itself is simple to write, the derivation sometimes appears to be a work of black magic. In fact, the QR iteration is essentially the subspace iteration we have already seen, re-cast in a different form.

1. The orthogonal iteration $\underline{Q}^{(k+1)} R^{(k)} = A\underline{Q}^{(k)}$ is a generalization of the power method. In fact, the first column of this iteration is *exactly* the power iteration. In general, the first p columns of $\underline{Q}^{(k)}$ are converging to an orthonormal basis for a p-dimensional invariant subspace associated with the p eigenvalues of A with largest modulus (assuming that there aren't several eigenvalues with the same modulus to make this ambiguous).

2. If all the eigenvalues have different modulus, orthogonal iteration ultimately converges to the orthogonal factor in a Schur form

$$AU = UT$$

   What about the T factor? Note that $T = U^* AU$, so a natural approximation to T at step k would be

$$A^{(k)} = (\underline{Q}^{(k)})^* A \underline{Q}^{(k)},$$

   and from the definition of the subspace iteration, we have

$$A^{(k)} = (\underline{Q}^{(k)})^* \underline{Q}^{(k+1)} R^{(k)} = Q^{(k)} R^{(k)},$$

   where $Q^{(k)} \equiv (\underline{Q}^{(k)})^* \underline{Q}^{(k+1)}$ is unitary.

3. Note that

$$A^{(k+1)} = (\underline{Q}^{(k+1)})^* A^{(k)} \underline{Q}^{(k+1)} = (Q^{(k)})^* A^{(k)} Q^{(k)} = R^{(k)} Q^{(k)}.$$

Thus, we can go from $A^{(k)}$ to $A^{(k+1)}$ directly without the orthogonal factors from subspace iteration, simply by computing

$$A^{(k)} = Q^{(k)} R^{(k)}$$
$$A^{(k+1)} = R^{(k)} Q^{(k)}.$$

This is the QR iteration.

Practical problems

There are two major problems with the basic QR iteration:

1. Each step of the QR iteration requires a QR factorization, which is an $O(n^3)$ operation. This is rather expensive, and even in the happy case where we might be able to get each eigenvalue with a constant number of steps, $O(n)$ total steps at a cost of $O(n^3)$ each gives us an $O(n^4)$ algorithm. Given that everything else we have done so far costs only $O(n^3)$, an $O(n^4)$ cost for eigenvalue computation seems excessive.

2. Like the power iteration upon which it is based, the basic iteration converges linearly, and the rate of convergence is related to the ratios of the moduli of eigenvalues. Convergence is slow when there are eigenvalues of nearly the same modulus, and nonexistent when there are eigenvalues with the same modulus.

We now turn to each of these in turn.

Hessenberg matrices and $O(n^2)$ QR steps

A matrix H is *upper Hessenberg* if it has nonzeros only in the upper triangle and the first subdiagonal. For example, the nonzero structure of a 5-by-5 Hessenberg matrix is

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

For any square matrix A, we can find a unitarily similar Hessenberg matrix $H = Q^*AQ$ in $O(n^3)$ time (a topic for next time). Because H is similar to A, they have the same eigenvalues; but as it turns out, the special structure of the Hessenberg matrix makes it possible to run QR in $O(n^2)$ per iteration.

The special structure of the Hessenberg matrix makes the Householder QR routine very economical. The Householder reflection computed in order to introduce a zero in the $(j+1, j)$ entry needs only to operate on rows j and $j+1$. Therefore, we have

$$Q^*H = W_{n-1}W_{n-2}\ldots W_1H = R,$$

where W_j is a Householder reflection that operates only on rows j and $j+1$. Computing R costs $O(n^2)$ time, since each W_j only affects two rows ($O(n)$ data). Now, note that

$$RQ = R(W_1W_2\ldots W_{n-1});$$

that is, RQ is computed by an operation that first mixes the first two columns, then the second two columns, and so on. The only subdiagonal entries that can be introduced in this process lie on the first subdiagonal, and so RQ is again a Hessenberg matrix. Therefore, one step of QR iteration on a Hessenberg matrix results in another Hessenberg matrix, and a Hessenberg QR iteration step can be performed in $O(n^2)$ time.

As it happens, the Hessenberg QR step can be written in terms of *bulge chasing*. The picture (in the 5-by-5 case) is as follows. We start with the original Hessemberg matrix, and first apply an orthogonal transformation (the first in a QR factorization) to the first two rows and then symmetrically to the first two columns. This introduces a nonzero (a "bulge") in the $(3,1)$ position. Marking the elements modified in the first step with a star, we have:

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

From here, our goal is to "chase" the bulge out of the Hessenberg structure. We start by applying an orthogonal transformation to rows 2 and 3 to remove the $(3,1)$ element; applying the same transformation to columns 2 and 3

introduces a bulge element in the $(4,2)$ position; marking the newly modified elements with stars, we have the new structure

$$\begin{bmatrix} \times & * & * & \times & \times \\ * & * & * & * & * \\ 0 & * & * & * & * \\ & * & * & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

Continuing for two more step in a similar fashion, we have a nonzero in the $(5,3)$ position, and then can restore the structure the rest of the way to a Hessenberg form.

Inverse iteration and the QR method

When we discussed the power method, we found that we could improve convergence by a spectral transformation that mapped the eigenvalue we wanted to something with large magnitude (preferably much larger than the other eigenvalues). This was the *shift-invert* strategy. We already know there is a connection leading from the power method to orthogonal iteration to the QR method, which we can summarize with a small number of formulas. Let us see if we can follow the same path to uncover a connection from inverse iteration (the power method with A^{-1}, a special case of shift-invert in which the shift is zero) to QR. If we call the orthogonal factors in orthogonal iteration $\underline{Q}^{(k)}$ ($\underline{Q}^{(0)} = I$) and the iterates in QR iteration $A^{(k)}$, we have

$$A^k = \underline{Q}^{(k)} \underline{R}^{(k)} \tag{1}$$

$$A^{(k)} = (\underline{Q}^{(k)})^* A (\underline{Q}^{(k)}). \tag{2}$$

In particular, note that because $R^{(k)}$ are upper triangular,

$$A^k e_1 = (\underline{Q}^{(k)} e_1) r_{11}^{(k)};$$

that is, the first column of $\underline{Q}^{(k)}$ corresponds to the kth step of power iteration starting at e_1. What happens when we consider negative powers of A? Inverting (1), we find

$$A^{-k} = (\underline{R}^{(k)})^{-1} (\underline{Q}^{(k)})^*$$

The matrix $\tilde{R}^{(k)} = (\underline{R}^{(k)})^{-1}$ is again upper triangular; and if we look carefully, we can see in this fact another power iteration:

$$e_n^* A^{-k} = e_n^* \tilde{R}^{(k)} (\underline{Q}^{(k)})^* = \tilde{r}_{nn}^{(k)} (\underline{Q}^{(k)} e_n)^*.$$

That is, the last column of $\underline{Q}^{(k)}$ corresponds to a power iteration converging to a *row* eigenvector of A^{-1}.

Shifting gears

The connection from inverse iteration to orthogonal iteration (and thus to QR iteration) gives us a way to incorporate the shift-invert strategy into QR iteration: simply run QR on the matrix $A - \sigma I$, and the (n, n) entry of the iterates (which corresponds to a Rayleigh quotient with an increasingly-good approximate row eigenvector) should start to converge to $\lambda - \sigma$, where λ is the eigenvalue nearest σ. Put differently, we can run the iteration:

$$\begin{aligned} Q^{(k)} R^{(k)} &= A^{(k-1)} - \sigma I \\ A^{(k)} &= R^{(k)} Q^{(k)} + \sigma I. \end{aligned}$$

If we choose a good shift, then the lower right corner entry of $A^{(k)}$ should converge to the eigenvalue closest to σ in fairly short order, and the rest of the elements in the last row should converge to zero.

The shift-invert power iteration converges fastest when we choose a shift that is close to the eigenvalue that we want. We can do even better if we choose a shift *adaptively*, which was the basis for running Rayleigh quotient iteration. The same idea is the basis for the *shifted QR iteration*:

$$Q^{(k)} R^{(k)} = A^{(k-1)} - \sigma_k I \tag{3}$$

$$A^{(k)} = R^{(k)} Q^{(k)} + \sigma_k I. \tag{4}$$

This iteration is equivalent to computing

$$\begin{aligned} \underline{Q}^{(k)} \underline{R}^{(k)} &= \prod_{j=1}^{n} (A - \sigma_j I) \\ A^{(k)} &= (\underline{Q}^{(k)})^* A (\underline{Q}^{(k)}) \\ \underline{Q}^{(k)} &= Q^{(k)} Q^{(k-1)} \dots Q^{(1)}. \end{aligned}$$

When we have transformed in advance to Hessenberg form, shifting can be naturally incorporated into the "bulge-chasing" picture. The only thing that changes is the choice of the first orthogonal transformation, which is chosen based on looking at the first step of QR factorization on the shifted matrix rather than the original matrix.

What should we use for the shift parameters σ_k? A natural choice is to use $\sigma_k = e_n^* A^{(k-1)} e_n$, which is the same as $\sigma_k = (\underline{Q}^{(k)} e_n)^* A (\underline{Q}^{(k)} e_n)$, the Rayleigh quotient based on the last column of $\underline{Q}^{(k)}$. This simple shifted QR iteration is equivalent to running Rayleigh iteration starting from an initial vector of e_n, which we noted before is locally quadratically convergent. This strategy (Wilkinson shifts) is close to what is done in practice, though we usually do something special when the matrix is real in order to apply pairs of complex shifts together (the "double shift" strategy), and an occasional "exceptional" shift is needed to keep the iteration from getting stuck at times.

Modern QR and beyond

Modern QR iteration involves several tricks to make things run fast on large matrices. *Multi-shift* techniques get several shifts at a time to run overlapping QR iterations; *aggressive early deflation* techniques similarly take advantage of the fact that there may be several eigenpairs that converge nearly simultaneously. A practical implementation that takes into account all these tricks is a bit more complex than the basic QR iteration, enough that it usually makes sense to rely on the codes in LAPACK (or in MATLAB or SciPy, which call LAPACK).

So far, we have considered the case of *general nonsymmetric* matrices. In the *symmetric* case, the Hessenberg reduction takes one all the way to a tridiagonal, and it is possible to do one QR step in $O(n)$ time. There are more efficient methods than QR for computing all the eigenvalues of a tridiagonal; but even for QR, the initial tridiagonal reduction (which costs $O(n^3)$) dominates the cost of the later work.