CS 542G: More on the Symmetric Eigenproblem

Robert Bridson

October 20, 2008

1 Shift-and-Invert

Last time we saw how shifting (replacing A with $A - \mu I$ in the power method) could possibly help, in adjusting the eigenvalue distribution which controls convergence while leaving eigenvectors unchanged. We'll now add another transformation, matrix inversion, which makes this far more powerful.

Observe that for an eigenpair where $Ax = \lambda x$, then multiplying both sides by A^{-1} if it exists gives:

$$\begin{aligned} x &= \lambda A^{-1} x \\ \Leftrightarrow \quad A^{-1} x &= \frac{1}{\lambda} x \end{aligned}$$

Thus x is also an eigenvector of A^{-1}, with eigenvalue $1/\lambda$.

Combining this with shifting gives the **shift-and-invert** algorithm: apply the power method to $(A - \mu I)^{-1}$. This transforms the eigenvalues to $1/(\lambda - \mu)$. Suppose μ is closest to λ_i, and second closest to λ_j of the original matrix A; for the transformed matrix the largest eigenvalue in magnitude must be $1/(\lambda_i - \mu)$ and the second largest is $1/(\lambda_j - \mu)$. Therefore the iteration will converge to λ_i's eigenvector at a rate

$$\frac{\lambda_i - \mu}{\lambda_j - \mu}$$

Clearly, the closer μ is to λ_i (relative to its distance from λ_j) the faster the convergence.

You might expect something to go wrong if $\mu = \lambda_i$: indeed, the matrix $A - \mu I$ is then singular and can't be inverted. However, as long as it's not exactly equal—even if $A - \mu I$ is exceedingly ill-conditioned—the numerics aren't a problem, as this will serve just to make the desired eigenvector component of x blow up that much faster.

Notice that shift-and-invert is also a very powerful tool, in comparison to the plain power method, since we can use it to pick out any eigenpair we care about—not just the largest eigenvalue.

2 Rayleight Quotient Iteration

Combining shift-and-invert with the Rayleigh quotient gives a particularly powerful algorithm, the Rayleigh Quotient Iteration. Here we take a step of shift-and-invert for some μ to get an updated x; then we use the Rayleigh quotient to improve the estimate μ of the associated eigenvalue, and repeat:

- Begin with an initial unit-length vector x and guess at the eigenvalue μ
- Until converged ($\|(A - \mu I)x\|$ is small enough):
  - Update $x \leftarrow (A - \mu I)^{-1}x$ (i.e. solve the linear system $(A - \mu I)x_{\text{new}} = x_{\text{old}}$)
  - Normalize $x \leftarrow \frac{x}{\|x\|}$
  - Update $\mu = x^T Ax$

As x converges, μ also converges to the eigenvalue, accelerating the convergence of both. In fact, it can be shown this gives **cubic convergence**: for k large enough, the error satisfies

$$\|e_{k+1}\| \leq C\|e_k\|^3$$

Here C is some constant, but the important part (compared to the linear convergence of the power method we saw last time) is the exponent of three, hence the name "cubic". Roughly speaking this means that once convergence starts, the number of *digits* of accuracy will triple each iteration. We don't have that many digits to worry about: even in double precision, this means we will essentially only need two or three more iterations once we enter the convergence regime.

One final word is in order about the initial choice of μ. Sometimes the user will have a good idea of what μ should be, but quite often will just be interested in either the smallest magnitude or largest magnitude eigenpair. Luckily it's fairly easy to come up with a sensible guess for μ in both these cases. For the smallest $\mu = 0$ is obviously appropriate: zero is closest to the smallest magnitude eigenvalue. For the largest, we have two cases, depending on whether the λ is positive or negative. Note that $\|Ax\| \leq \|A\|\|x\|$ for any induced matrix norm, which implies that $\|A\|$ is always an upper bound on the magnitude of all the eigenvalues; we can choose $\mu = \pm\|A\|$ for some easily computed matrix norm (like the 1-norm or infinity-norm) to get an initial value that's closest to either the largest positive eigenvalue or largest negative eigenvalue. The caveat in all these cases, however, is that depending on the initial guess for x, we might not converge to exactly the eigenpair we wanted: for example, if x is already an eigenvector, no matter what the initial μ is x won't converge to a different eigenvector.

3 Orthogonal Iteration

So far we've just looked at methods which find a single eigenpair. In many cases we want more, perhaps all of them. The convergence of the power method to an eigenvector of the largest magnitude eigenvalue depended on the initial guess x containing a nonzero component in that eigenspace. In practice, apart from specially-structured matrices, rounding errors will almost always guarantee that this component is nonzero in fact, so it's not something to worry about too much—but it does give us a clue on how to get other eigenvectors.

Suppose we have already found the largest magnitude eigenpair, with eigenvector u_1. If we want to now find u_2, an eigenvector for the second largest eigenvalue[1], which we know can be chosen to be orthogonal to u_1. The power method will work as long as our guess for u_2 has a zero component for u_1, i.e. is orthogonal to u_1. We thus add a step to the algorithm where we orthogonalize x with respect to u_1: doing this every iteration will further guarantee that we're not hurt by rounding error reintroducing a nonzero component in the direction of u_1.

We can continue this for the third largest and so on down the line: once we've found the first k largest eigenpairs, we can run power method with a step to orthogonalize our guess x against each of these k eigenvectors, and it should converge to the $k+1$'st eigenpair.

However, we can do better: solve for the set of eigenvectors we want simultaneously. This is **Orthogonal Iteration**. If we are looking for the k largest magnitude eigenvalues and eigenvectors, we begin with k linearly independent vectors $x_1, \ldots x_k$, where x_1 will hopefully converge to the biggest, x_2 to the second biggest, and so on. Each iteration we multiply all of them by A, then normalize x_1, make x_2 orthogonal to x_1 and then normalize it, make x_3 orthogonal to x_1 and x_2 and then normalize it, etc. We've seen exactly this orthogonalization before: it's the Gram-Schmidt process!

We know from before that Gram-Schmidt (or preferably, Modified Gram-Schmidt) is just one algorithm for computing the QR factorization. We can write orthogonal iteration as follows then:

- Begin with an $n \times k$ full rank matrix X with the initial guesses at eigenvectors as the columns.
- Until converged:
  - Multiply $X \leftarrow AX$
  - Factor $QR = X$ with MGS
  - Replace $X \leftarrow Q$

[1] This also will cover the case where the largest magnitude eigenvalue was repeated, and we want to find a second eigenvector for this eigenspace.

When we have converged, the R factor, of course, should be diagonal (containing the eigenvalues).

4 The QR Method

Applying orthogonal iteration to find all n eigenvectors leads to something called the QR method for eigenproblems (not to be confused with the plain QR factorization). In particular, if we're working on all n eigenvectors, we can use the superior Householder method for the QR factorization.

One of the great advantages Householder QR has over MGS, discussed earlier, is that it is guaranteed to produce an orthogonal Q, even if the input matrix is singular. For example, if A had a zero eigenvalue (was singular), then orthogonal iteration with MGS won't ever be able to determine the corresponding eigenvector: multiplying by A will zero out any component in that eigenspace, and MGS can't put it back into Q. On the other hand, Householder QR will automatically produce the eigenvector as the last column of Q—converging instantly. Taking this idea further leads to one of the many enhancements that have been added to the QR method: if we have a good guess at an eigenvalue μ, then the shifted $A - \mu I$ will have an eigenvalue very close to zero, for which Householder QR will produce an extremely good estimate of the eigenvector.

There are several other important enhancements to this strategy that go into the final QR method: refer to Golub and Van Loan if you're curious about other additions (you may be surprised that the ultimate version bears only a passing resemblence to orthogonal iteration at first glance, despite being derived from it). Ultimately the algorithm can be made fast enough to return all eigenvalues and eigenvectors in a small factor more than the time it takes to solve a linear system. LAPACK of course provides this and related algorithms for you; however, more targeted algorithms such as Rayleigh Quotient Iteration are not provided.

5 Rayleigh-Ritz

One of the more important additions to QR, without which it wouldn't work, is a way to handle eigenvalues of equal magnitude but opposite sign. If you recall, this is a case where the power method can fail to converge at all. Luckily for the symmetric case this can only occur with a pair of eigenvalues, negatives of each other. When we detect we have two vectors which are resisting convergence—probably indicating their eigenvalues are equal or nearly equal in magnitude—we can turn to another method which can

tease them apart.[2]

The **Rayleigh-Ritz** algorithm is a generalization of the Rayleigh quotient to multiple vectors: given a set of vectors which may not be eigenvectors but do approximately some union of eigenspaces, Rayleigh-Ritz works out an optimal estimate of the associated eigenvalues. As a bonus, it also returns an optimal estimate of linear combinations of the input vectors that approximate the eigenvectors.

Let U be an $n \times k$ matrix with orthonormal[3] columns, which span a suspected collection of eigenspaces. We are looking for a $k \times k$ diagonal matrix D that will contain the estimated eigenvalues, and a $k \times k$ matrix V which will give the linear combinations of the columns of U that approximate the eigenvectors. (Note that the eigenvectors should be orthogonal, and since U has orthonormal columns, V should be orthogonal.) In other words, we want V and D so that $A(UV) \approx (UV)D$. Setting this up in a least-squares sense gives a problem

$$\min_{D,V} \|AUV - UVD\|_F^2$$

which is a natural generalization of the Rayleigh quotient problem (where $V = 1$ and $D = \lambda$).

Let $\bar{U}$ be any $n \times (n-k)$ matrix where $(U|\bar{U})$ is orthogonal (so $\bar{U}$ also has orthonormal columns which are orthogonal to those of U). Multiplying by an orthogonal matrix such as $(U|\bar{U})^T$ doesn't change the Frobenius norm; it leaves us the problem

$$\min_{D,V} \left\| \begin{pmatrix} U^T AUV - U^T UVD \\ \bar{U}^T AUV - \bar{U}^T UVD \end{pmatrix} \right\|_F^2$$

Note that $U^T U = I$, but $\bar{U}^T U = 0$, and that the Frobenius norm squared can be separated, so this simplifies to:

$$\min_{D,V} \|(U^T AU)V - VD\|_F^2 + \|\bar{U}^T AUV\|_F^2$$

The second term's Frobenius norm isn't changed through multiplying by V^T on the right (since V is orthogonal), which makes it apparent it only depends on the choice of U, not on V or D: it's irrelevant to the minimization. So our problem reduces to:

$$\min_{D,V} \|(U^T AU)V - VD\|_F^2$$

With $\bar{A} = U^T AU$, a $k \times k$ symmetric matrix, it's clear this can be made exactly zero simply by choosing V to be a set of eigenvectors of $\bar{A}$ and D the diagonal matrix of its eigenvalues.

[2] Again, as a cautionary note, if you look at the ultimate expression of the QR algorithm the link to this section is a bit obscure, but what we are about to discuss does underly the approach.

[3] Orthogonality isn't strictly necessary, but simplifies Rayleigh-Ritz.

This is the Raleigh-Ritz procedure: solve the $k \times k$ eigenproblem for $\bar{A} = U^T AU$ which gives optimal estimates of the eigenvalues of A "projected" onto the subspace spanned by U, and eigenvectors estimated by UV. The critical point is that if k is much smaller than n, finding the eigenpairs of $\bar{A}$ will be much easier than working on A; in particular, if $k = 2$ than $\bar{A}$ is a 2×2 symmetric matrix whose eigenpairs can be found directly by solving a quadratic equation.

Rayleigh-Ritz is, seen from another perspective, another example of taking a high dimensional problem, choosing a smaller dimension sub-space we guess can approximate the solution, and finding an optimal guess from within that subspace. We will come back to this concept again for other problems.