CS 542G: The Symmetric Eigenproblem

Robert Bridson

October 15, 2008

1 The Symmetric Eigenproblem

Last time we saw there were strong connections between the SVD and finding the eigenvectors and eigenvalues of symmetric matrices (such as A^TA, AA^T, or a larger matrix with A and A^T as off-diagonal blocks). Finding eigenvalues and eigenvectors of symmetric matrices is important in many problems on its own right; for example, in acoustics, vibration analysis, and related engineering subjects.

For a symmetric real matrix A, we know all eigenvalues are real and that there is a complete orthonormal basis of eigenvectors. However, for general unsymmetric matrices, life is nowhere near as simple: there may be complex eigenvalues, the eigenvectors may not be orthogonal, and there might not even be a full basis of eigenvectors. The algorithms for this case (and how to define a well-posed numerical problem around it) are rather more complicated as a result—thus we will skip over this problem and focus only on the symmetric case.

The classic derivation of the eigenvalues poses them as roots of the characteristic polynomial $\det(A - \lambda I)$ of the matrix; solving the eigenproblem is in some sense equivalent to solving polynomials—at least in exact arithmetic. We haven't discussed how to compute a determinant effectively, but we did see that high degree polynomials can be very badly behaved numerically and are probably best to avoid. Therefore we'll tackle the eigenproblem from a different direction.

However, the link to polynomials does illustrate one important point: finding eigenvalues can't be as straightforward as, say, solving linear systems. For the 2×2 case there is a simple enough exact formula, based on solving a quadratic, which only involves taking a square-root; for the 3×3 case there is still a cubic formula involving cube roots as well, but it's rather more complex, and the 4×4 case is still tractable but pretty nasty. However, for fifth degree and higher polynomials, there is no possible formula for the roots that just involves regular arithmetic and radicals (square roots, cube roots, etc.): Abel proved this

back in the 19th century. This isn't to say exact formulas aren't possible, but they do involve considerably more exotic "special functions" like the family of elliptic functions. In fact, if you start examining this there isn't anything obviously better about functions like cube roots than these lesser known special functions; none of them are typically implemented in hardware, and must be approximated to a given accuracy (round-off precision) with software algorithms.

This is the key point I want to make: for eigenproblems we have to give up on the notion that we can write down a simple terminating algorithm which would give the exact solution if evaluated in exact arithmetic. Instead we will look at **iterative methods**, which will begin with a guess and refine it for a number of steps until the approximation error is deemed small enough (for example, that $\|Ax - \lambda x\|$ is below a tolerance related to round-off error). Smaller errors will require more steps.

However, at least in some cases, it turns out the iterative methods are so wonderfully robust and efficient that for all intents and purposes we can guarantee *a priori* only a fixed number of steps will be required for a particular precision, no matter the matrix A, and thus it's tempting to treat them as **direct methods** (i.e. not iterative).

2 The Power Method

The very simplest eigen-algorithm[1] of all is called the Power Method. Start with a random vector x. If we write it in terms of an orthonormal eigenbasis $\{u_i\}$ of A, $x = \alpha_1 u_1 + \ldots + \alpha_n u_n$ and then multiply by A, we see the result is:

$$\begin{aligned} Ax &= \alpha_1 Au_1 + \ldots + \alpha_n Au_n \\ &= \alpha_1 \lambda_1 u_1 + \ldots + \alpha_n \lambda_n u_n \end{aligned}$$

If we multiply the result by A again, we get:

$$\begin{aligned} A^2 x = A(Ax) &= \alpha_1 \lambda_1 Au_1 + \ldots + \alpha_n \lambda_n Au_n \\ &= \alpha_1 \lambda_1^2 u_1 + \ldots + \alpha_n \lambda_n^2 u_n \end{aligned}$$

Similarly, multiplying k times with A gives:

$$A^k x = \alpha_1 \lambda_1^k u_1 + \ldots + \alpha_n \lambda_n^k u_n$$

Now, as long as the coefficients α_i are nonzero, eventually this sum will be dominated by the component corresponding to the largest eigenvalue in magnitude: if $|\lambda_1| > |\lambda_i|$, say, then the ratio of magnitudes of

[1] One of the fun things about eigenvalues and eigenvectors is that you can use the eigen- prefix on just about any relevant eigenword.

coeffients

$$\frac{|\alpha_1 \lambda_1^k|}{|\alpha_i \lambda_i^k|}$$

will grow arbitrarily large as k goes to infinity.

Assuming the eigenvalues are sorted by decreasing magnitude, so $|\lambda_1|$ is the biggest, this means that after multiplying x by A enough times it will be nearly proportional to an eigenvector associated with λ_1. (This isn't strictly true: we'll see the special problem case in a moment.)

Unfortunately, if $|\lambda_1| \neq 1$, this result $A^k x$ will either blow up or vanish to zero exponentially—eventually being lost to the finite range of floating-point values. Thus to make the Power Method into a usable algorithm, we rescale the vector to be unit-length after every multiply:

- Start with an arbitrary x
- For $k = 1, 2, \ldots$ until convergence:
  - $x \leftarrow Ax$
  - $x \leftarrow \frac{x}{\|x\|}$
- Return x as an eigenvector for the largest eigenvalue of A

This brings up two questions immediately, such as what is the eigenvalue for x (needed, if for nothing else, to evaluate if we've converged adequately), and how efficient can we expect this algorithm to be?

3 The Rayleigh Quotient

The first problem, determining the eigenvalue for a given approximation to an eigenvector, is solved neatly with the **Rayleigh Quotient**, defined for a vector x as:

$$\frac{x^T A x}{x^T x}$$

In the case above, where x is unit-length, obviously this simplifies to $x^T A x$. Clearly if x is an eigenvector, so $Ax = \lambda x$, this quotient does evaluate to λ.

What's even better is that it provides an optimal estimate of the eigenvalue even if x isn't exactly an eigenvector. The Rayleigh quotient minimizes

$$\min_{\lambda} \|Ax - \lambda x\|_2^2$$

which is easy to see by differentiating with respect to λ and setting it to zero:

$$2x^T(Ax - \lambda x) = 0$$

4 The Convergence of the Power Method

The second question, how fast does the Power Method converge, can be answered in asymptotic terms fairly easily. Just as $\alpha_1 \lambda_1^k u_1$ eventually has to be by far the largest magnitude term in the eigen-expansion of $A^k x$, the second largest term must be $\alpha_2 \lambda_2^k u_2$, and this will similarly dominate the rest. Thus eventually the error in our estimate of the eigenvector will be proportional to the ratio

$$\frac{|\lambda_2|^k}{|\lambda_1|^k} = \left(\frac{|\lambda_2|}{|\lambda_1|}\right)^k$$

The smaller the ratio $|\lambda_2|/|\lambda_1|$ is (it must be between 0 and 1), the faster the algorithm converges.

Unfortunately, this ratio depends on A, and it could be arbitrarily close to 1. In fact, it could even be exactly equal to 1. This might be harmless—a repeated largest eigenvalue, in which case the Power Method will converge to a non-unique eigenvector—or it might be serious indeed, if $\lambda_2 = -\lambda_1$, in which case convergence just won't happen at all. For example, you can take a look at the matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

which has eigenvalues 1 and -1. Multiplying by A simply swaps the two entries in x; unless you start with x already equal to one of the eigenvectors $(1, 1)$ or $(1, -1)$, swapping k times doesn't get you any closer to solving the problem.

On the other hand, if there is a good separation between the magnitude of λ_1 and λ_2 then the Power Method can work quite well. In particular, it gives **linear convergence**, which technically means that for k large enough, the error at step $k+1$ is bounded by a constant C times the error at step k:

$$\|e\|_{k+1} \leq C\|e\|_k$$

where $0 \leq C < 1$. Here the constant C is the ratio $|\lambda_2|/|\lambda_1|$. An algorithm with linear convergence will, eventually, improve the error by one decimal digit of accuracy after a constant $\log_{10}(1/C)$ steps.

Incidentally, the "for k large enough" caveat is absolutely critical, just as in $O()$ notation. We generally expect iterative methods to spend a few steps initially behaving poorly (the error might even increase, or fluctuate erratically), depending on how good or bad the initial guess was. We might call this a "transient regime". Eventually however, the error analysis above will take hold (e.g. the u_1 term is significantly larger than the u_2 term, which is significantly larger than all the rest), and the expected convergence behaviour is displayed in what might term the "convergence regime".

5 Shifting

The Power Method can be abysmally slow if the two largest eigenvalues aren't well separated; to make it a more effective algorithm, we need tools that can cause a better separation. In particular, we'll look at modifications we can make to A that don't change the eigenvectors but do change the eigenvalues. The simplest such transformation is called **shifting**: replace A with $A - \mu I$ for some real number μ (the shift).

Observe that if $Ax = \lambda x$, then $(A - \mu I)x = (\lambda - \mu)x$: the eigenvectors are preserved, but the eigenvalues get shifted through subtracting μ.

Ignoring what happens to the other eigenvalues, if μ was much closer to λ_2 then λ_1, this would mean the Power Method applied to the shifted $A - \mu I$ converges according to the ratio

$$\frac{|\lambda_2 - \mu|}{|\lambda_1 - \mu|}$$

which could be significantly closer to zero, and thus give much faster convergence.

The fly in the ointment here (ignoring how we might approximate λ_2 in the first place) is that this shift might cause other eigenvalues to be promoted to the largest. For example, suppose the eigenvalues of A are $\{-1.001, -1, 10, 11\}$. Here 11 is the largest and 10 is the second largest with a convergence rate of $10/11$, but shifting by $\mu = 10$ transforms these to $\{-11.001, -11, 0, 1\}$ where the largest in magnitude is now -11.001 and the convergence rate is $11.0001/11$: much worse.

Next lecture we'll continue this quest by looking at another way to transform A that modifies eigenvalues but leaves eigenvectors unchanged.