Algorithms for symmetric matrices

While the algorithms studied earlier are applicable to general matrices, usually one would like to take advantage of any special properties that a matrix may possess. One of the most commonly occurring property is that of symmetry. The algorithms studied earlier can be modified to take advantage of symmetry, but there are some more algorithms that are specifically designed for symmetric matrices.

We will study four algorithms applicable to symmetric eigenvalue problems:

- Rayleigh quotient iteration
- Jacobi's method
- Bisection method
- Divide and conquer

1 Rayleigh quotient iteration

(Recall) The Rayleigh quotient is defined as the function $R_A : \mathbb{R}^m \setminus \{0\} \to \mathbb{R}$ defined by

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

Also recall that if x is an eigenvector corresponding to eigenvalue λ, then $R_A(x) = \lambda$. So if we know x, the function R_A helps us locate the corresponding eigenvalue.

The problem of estimating λ, given a vector x can be formulated as a least squares problem as follows: find a scalar λ which minimizes $||Ax - \lambda x||_2$. This is a system of linear equations with variable λ, $m \times 1$ matrix x and r.h.s. being the known quantity Ax. Writing the normal equations for this system of equations we get:

$$x^* x \lambda = x^* A x \quad \text{i.e.} \quad \lambda = \frac{x^* A x}{x^* x} = \frac{x^T A x}{x^T x} = R_A(x).$$

Thus, $R_A(x)$ is a reasonable estimate of the eigenvalue to consider if x is an approximate eigenvector.

These ideas can be made more precise by investigating the local behaviour of $R_A(x)$ in a neighbourhood of x using derivatives. It can be shown that the eigenvectors of A are the stationary points of $R_A(x)$ and the eigenvalues of A are the values of $R_A(x)$ at these stationary points. (refer to equation 27.2, chapter 27 of [1]). As a consequence, it can also be shown that if q is an eigenvector of A then

$$R_A(x) - R_A(q) = \mathcal{O}(||x - q||^2) \quad \text{as} \quad x \to q.$$

This implies that the Rayleigh quotient is a *quadratically accurate* estimate of an eigenvalue, i.e. $R_A(x)$ approaches $R_A(q)$ twice as fast as x approaches q. This constitutes the strength of the method.

The above idea is paired with the inverse iteration to give the Rayleigh quotient iteration:

Idea:

- start with a vector $v^{(0)}$
- calculate $R_A(v^{(0)})$, this is the first approximation to $\lambda^{(0)}$,

- apply inverse iteration to $\lambda^{(0)}$ to get an approximation for $v^{(1)}$, then apply R_A to $v^{(1)}$ to get $\lambda^{(1)}$, and so on

$$\begin{pmatrix} \text{approximate} \\ \text{eigenvector } v \end{pmatrix} \underset{\text{inverse iteration}}{\overset{R_A}{\underset{\longleftarrow}{\longrightarrow}}} \begin{pmatrix} \text{approximate} \\ \text{eigenvalue } \lambda \end{pmatrix}$$

Algorithm:

choose $\mathrm{v}^{(0)}$ with $||v^{(0)}|| = 1$
$\lambda^{(0)} = v^{(0)T} A v^{(0)}$ (calculate $R_A(v^{(0)})$)
for *k=1 to ... (convergence)* **do**
  solve (A - $\lambda^{(k-1)} I)w = v^{(k-1)}$ for w (apply inverse iteration)
  $\mathrm{v}^{(k)} = w/||w||_2$
  $\lambda^{(k)} = R_A(v^{(k)}) = v^{(k)T} A v^{(k)}$
end

Theorem 1. *The Rayleigh quotient iteration converges to an eigenvalue/eigenvector pair for all except a set of measure zero (i.e. a negligible set) of starting vectors $v^{(0)}$. When it converges, the convergence is cubic in the following sense: if λ is an eigenvalue of A and v is a vector sufficiently close to the eigenvector x then -*

$$||v^{(k+1)} - x|| = \mathcal{O}(||v^{(k)} - x||^3)$$
$$||\lambda^{(k+1)} - \lambda|| = \mathcal{O}(||\lambda^{(k)} - \lambda||^3).$$

(This means that each iteration triples the number of digits of accuracy, so when the algorithm converges, it converges fast.)

(This is theorem 27.3 of [1] and the proof can be found there. We will proceed without proof.)

2 Jacobi methods

The basic idea of Jacobi methods is to try and reduce the magnitude of off-diagonal elements so that eventually they become small enough and can be declared zero. This is done by a sequence of orthogonal similarity transformations $A \mapsto Q^T A Q$.

The idea is formulated as follows: consider the Frobenius norm of the off-diagonal elements of A -

$$\mathrm{off}(A) = \sqrt{\left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} a_{ij}^2 \right)}.$$

The similarity transformations are applied in such a way that $\mathrm{off}(A)$ is systematically reduced. The tools for doing this are (Givens) rotations of the form:

$$J(p,q,\theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}.$$

In the context of Jacobi's method, these matrices will be called Jacobi rotations. The basic step in a Jacobi eigenvalue procedure involves choosing an index pair (p, q), computing the corresponding cosine-sine pair (c, s) such that

$$\begin{pmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \tag{1}$$

is diagonal and overwriting A with $B = J^T AJ$ where $J = J(p, q, \theta)$.

Observe:

- The matrix B agrees with matrix A in all entries except in rows and columns p and q.
- Since the Frobenius norm is preserved by orthogonal transformations, we have:

$$a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2 = b_{pp}^2 + b_{qq}^2 + 2b_{pq}^2 = b_{pp}^2 + b_{qq}^2.$$

  As a result,

$$\begin{aligned} \text{off}(B)^2 &= ||B||_F^2 - \sum_{i=1}^{n} b_{ii}^2 && (2) \\ &= ||A||_F^2 - \sum_{\substack{i=1 \\ i\neq p,q}}^{n} b_{ii}^2 - b_{pp}^2 - b_{qq}^2 + (a_{pp}^2 + a_{qq}^2) - (a_{pp}^2 + a_{qq}^2) && (3) \\ &= ||A||_F^2 - \sum_{i=1}^{n} a_{ii}^2 + (a_{pp}^2 + a_{qq}^2 - b_{pp}^2 - b_{qq}^2) && (4) \\ &= \text{off}(A)^2 - 2a_{pq}^2 && (5) \\ & && (6) \end{aligned}$$

  which explains how A moves closer to being diagonal with each iteration of the Jacobi algorithm.

- Since A is symmetric, every iteration introduces a symmetric pair of zeroes in A. The zeroes obtained at each step are possibly destroyed at the next step, but the magnitude of these non-zero entries decreases at each step.

We will first consider an algorithm for the Schur decomposition of a 2×2 symmetric matrix $A = \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}$ of the form described in Equation (1). The matrix B of Equation (1) will be diagonal if $b_{pq} = 0$. Note that

$$b_{pq} = a_{pq}(c^2 - s^2) + (a_{pp} - a_{qq})cs. \tag{7}$$

If $a_{pq} = 0$, then we can set $c = 1$ and $s = 0$ and we're done.
Else, define

$$\tau = \frac{a_{qq} - a_{pp}}{2a_{pq}} \quad \text{and} \quad t = s/c,$$

then $t = \tan(\theta)$ satisfies the quadratic $t^2 + 2\tau t - 1 = 0$ (from Equation (7)). To find the desired value of θ, we solve the above quadratic; the two roots are $\{1/(\tau + \sqrt{1+\tau^2}), 1/(\tau - \sqrt{1+\tau^2})\}$. Let t_{min} denote the minimum of these two roots. Since $t_{min} \leq 1$, the rotation angle θ satisfies $|\theta| \leq \pi/4$, which maximizes $c = 1/\sqrt{1+t_{min}^2}$. Further, $s = t_{min}c$. These values minimize the difference between A and the update B:

$$||B - A||_F^2 = 4(1-c) \sum_{i=1, i\neq p,q}^{n} (a_{ip}^2 + a_{iq}^2) + 2a_{pq}^2/c^2.$$

Algorithm for 2×2 symmetric Schur decomposition Given an $n \times n$ symmetric matrix A and integers p and q that satisfy $1 \leq p < q \leq n$, this algorithm computes a pair (c, s) such that if $B = J(p,q,\theta)^T A J(p,q,\theta)$, then $b_{pq} = b_{qp} = 0$.

if $a_{pq} \neq 0$ **then**
  $\tau = (a_{qq} - a_{pp})/2a_{pq}$;
  if $\tau \geq 0$ **then**
    $t = 1/(\tau + \sqrt{1+\tau^2})$;
  else
    $t = 1/(\tau - \sqrt{1+\tau^2})$;
  end
  $c = 1/\sqrt{1+\tau^2}, s = tc$;
else
  $c = 1, s = 0$;
end

We will study the classical Jacobi algorithm. There are other versions and improvements on this algorithm (e.g. the cyclic-by-row Jacobi algorithm and block Jacobi algorithm) which we won't discuss here. You can find more details in [2], Sections 8.5.3 - 8.5.6.

In the classical Jacobi method, the pair (p, q) is chosen so as to maximize the reduction of off(A) in Equation 2. Thus (p, q) is chosen so that a_{pq}^2 is maximal.

Classical Jacobi Algorithm Given a symmetric $A \in \mathbb{R}^{n\times n}$ and a positive tolerance **tol**, this algorithm overwrites A with $V^T AV$ where V is orthogonal and off$(V^T AV) \leq \textbf{tol} \cdot ||A||_F$.

$V = I_n, \delta = \textbf{tol} \cdot ||A||_F$;
while off$(A) > \delta$ **do**
  Choose (p, q) so that $|a_{pq}| = \max_{i\neq j}|a_{ij}|$;
  Obtain (c, s) using Algorithm (2);
  $A = J(p,q,\theta)^T A J(p,q,\theta)$;
  $V = VJ(p,q,\theta)$
end

3 Note on operation counts

Suppose $A \in \mathbb{R}^{m\times m}$.

1. Each step of power iteration involves a matrix-vector multiplication, requiring $\mathcal{O}(m^2)$ flops.

2. Each step of the inverse iteration involves the solution of a linear system, which could require $\mathcal{O}(m^3)$ flops, but this count can be brought down to $\mathcal{O}(m^2)$ if the matrix is processes in advance using some method like LU or QR factorization.

3. For Rayleigh quotient iteration, the matrix to be inverted changes at each step, so its not easy to avoid $\mathcal{O}(m^3)$ flops.

In case of symmetric matrices, all these operation counts can be improved significantly by pre-processing the matrix and making it *tridiagonal*, i.e. a matrix having zeroes everywhere except its diagonal, subdiagonal and superdiagonal. (We will look at this process in the next set of notes.) On a tridiagonal matrix, all three iterations considered above require only $\mathcal{O}(m)$ flops per step.

References

[1] Trefethen and Bau. *Numerical linear algebra.*

[2] Golub and Loan. *Matrix computations*, fourth edition.