The Jacobi Eigenvalue Algorithm

Titus Parker, Math118

May 2024

1 Introduction

Finding eigenvalues to a linear system is supremely useful in many real-life situations. The reproduction of an infectious disease is in many ways, determined by the largest eigenvalue. Quantum physics relies heavily on linear algebra, and energy states in Hamiltonian systems correspond to eigenvalues of the linear system. In artificial intelligence, finding eigenvalues is important for tools like dimension reduction and clustering.

How, then, can we find eigenvalues of a linear system? In elementary linear algebra classes, we are taught to solve the 'characteristic polynomial', typically represented as

$$\det(A - \lambda I) = \lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0 = 0 \tag{1.1}$$

for some $c_n \in \mathbf{F}$.
While this does find the eigenvalues of a known linear system, it also is extremely computationally costly for a sufficiently 'large' matrix, as polynomial rootfinding is an ill-conditioned problem even if the underlying eigenvalue problem is well-conditioned. In practice, therefore, we need *shortcuts* - approximation methods to find eigenvalues of a given matrix. These can, in fact, reduce the time taken to calculate eigenvalues of a large linear system from the order of months or even years, to minutes or even seconds. The most famous of these algorithms is the QR algorithm, which writes a given matrix as the product of an upper triangular matrix and an orthogonal matrix.

Carl Gustav Jacob Jacobi was a German mathematician working in the early 19th century; he had prolific output in elliptic functions, differential equations, and number theory to name a few fields. This paper focuses on his Eigenvalue algorithm, an alternative to the QR algorithm, of finding eigenvalues of a matrix.

2 Jacobi's Eigenvalue Algorithm

According to Trefethen and Bau (1997), algorithms of numerical linear algebra are rely on putting zeros into matrices. Thus, the Jacobi Eigenvalue Algorithm is a sequence of computations to produce an eigenvalue-revealing factorization of A, converging to a diagonal matrix of eigenvalues in the diagonal entries. First then, we introduce the *Givens Matrix* to show the underlying motivation of rotations for the algorithm.

2.1 Motivation

Consider a 2x1 position vector $A = \begin{bmatrix} x \\ y \end{bmatrix}$. It turns out we can rotate this matrix to produce a vector $\vec{v} = \lambda \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ for λ some constant as follows:

$$J(x, y, \theta) = GA = \begin{bmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \lambda \\ 0 \end{bmatrix} \tag{2.1.1}$$

This is called a Givens Rotation, where γ and σ are $cos\theta = x/r$ and $sin\theta = -y/r$ respectively. To now produce a 2x2 matrix, instead of a 2x1 matrix, consisting of eigenvalues of A using (2.1.1) let

$$G = \begin{bmatrix} cos(\theta) = \gamma & -sin(\theta) = -\sigma \\ \sigma = sin(\theta) & cos(\theta) = \gamma \end{bmatrix} \tag{2.1.2}$$

such that

$$A' = G^\top AG = \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \tag{2.1.3}$$

where $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

Thus,

$$A' = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}^\top \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{2.1.4}$$

produces two eigenvalues λ_0 and λ_1, both on the matrix's diagonal, by solving for $tan(2\theta) = \frac{2b}{a-d}$. In a 2 dimensional matrix, this only takes one computation!

2.2 Description

Generalising this to a multidimensional context, we select the $i^{th}$and j^{th} columns and rows of A, whose 2x2 submatrix taken from then is rotated as in section 2.1. to search for eigenvalues we create a matrix $A' = G^\top AG$ where G is some multidimensional n by n matrix $G(i, j, \theta)$ of the form

$$G = \begin{bmatrix} 1 & ... & 0 & ... & 0 & ... & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \gamma & \vdots & -\sigma & \vdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \vdots & \sigma & \vdots & \gamma & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & ... & 0 & ... & 0 & ... & 1 \end{bmatrix} \tag{2.2.1}$$

where $a_{i,i} = a_{j,j} = \gamma$, $a_{i,j} = -\sigma$ and $a_{j,i} = \sigma$ respectively. We have 1 on the other diagonals, and 0 everywhere else.

The matrix multiplication $A' = G^\top AG$ sets each $a'_{i,j} = a'_{j,i}$ (for $i \neq j$) to 0 through our selection of θ as above. The algorithm sequentially performs these rotations, setting A' to A at each step, until the matrix A converges to a diagonal matrix.

The term-by-term breakdown of A' is as follows:

$$\begin{aligned} &a'_{i,i} = c^2 a_{i,i} - 2sca_{i,j} + s^2 a_{j,j} \\ &a'_{j,j} = s^2 a_{i,i} + 2sca'_{i,j} + sca'_{i,i} \\ &a'_{i,j} = a'_{j,i} = (c^2 - s^2)a_{i,j} + sc(a_{ii} - a_{j,j}) \\ &a'_{i,k} = a'_{k,i} = ca_{i,k} - sa_{j,k} \text{ for } k \neq i, j \\ &a'_{j,k} = a'_{k,j} = ca_{i,k} + sa_{j,k} \text{ for } k \neq i, j \\ &a'_{k,l} = a_{k,l} \text{ for } k, l \neq i, j \end{aligned} \tag{2.2.2}$$

This process selects the $i^{th}$and j^{th} columns and rows of A, to rotate them by a given angle, setting the selected non-diagonal elements to 0; to optimise this algorithmically, we select the largest off-diagonal entry $d_{i,j}$, defining i,j in the process. The algorithm also has O(n) cost when the pivot is known. When it is not, we have to search through non-diagonal entries, which has O(n^2) cost. Ultimately, then, this algorithm produces an almost-diagonal matrix whose diagonal elements converge to eigenvalues of the original matrix A.

2.3 Convergence

In this section, we prove that the off-diagonal entries in a sequence of matrices, obtained by the Jacobi Eigenvalue Algorithm, converge to 0. This proof is thanks to Suli and Mayer's 'Introduction to Numerical Analysis'.

To prove this convergence, we introduce some new terminology:

Definition 2.1. *Let $A \in \mathbb{R}^{n\times n}_{\mathbb{SYM}}$. The quantity*

$$\|A\|_F = (\sum_{i=1}^{n}\sum_{i=1}^{n} a_{ij}^2)^{1/2}$$

*is called the **Frobenius norm** of $A \in \mathbb{R}^{n\times n}$.*

Interpreting A as an element of a linear space of dimension n^2, the Frobenius norm is the 2-norm of A; however, it is not subordinate to the 2-norm in $\mathbb{R}^n$.

Now, we introduce a necessary lemma involving the Frobenius norm.

Lemma 1. *: The sum of squares of the elements of a symmetric matrix is invariant under an orthogonal transformation; IE, if $A \in \mathbb{R}^{n\times n}_{\mathbb{SYM}}$ and $B = R^\top AR$, where $R \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, then*

$$\sum_{i=1}^{n}\sum_{i=1}^{n} b_{ij}^2 = \sum_{i=1}^{n}\sum_{i=1}^{n} a_{ij}^2 \tag{2.3.1}$$

Expressing (2.3.1) in the language of the Frobeius norm, we say the Frobenius norm of a symmetric matrix A is invariant under an orthogonal transformation: $\|R^\top AR\|_F = \|A\|_F$.

Proof. The sum of squares of the elements of A is the same as the trace of A^2 for

$$Trace(A^2) = \sum_{i=1}^{n}(A^2)_i i = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}a_{ji} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2 \tag{2.3.2}$$

since A is symmetric. Along similar lines, as B is symmetric,

$$Trace(B^2) = \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}^2 \tag{2.3.3}$$

. Thus, we show $Trace(B^2) = Trace(A^2)$:

$$B^2 = (R^\top AR)(R^\top AR) = R^\top A^2 R \tag{2.3.4}$$

by orthogonality of R. Thus, B^2 is an orthogonal transformation of A^2, which implies that B^2 and A^2 have the same eigenvalues, and thus the same trace. □

Using this lemma, we are now ready to prove the convergence of the Jacobi eigenvalue algorithm.

Theorem 1. *Let $A^{(0)}, A^{(1)}, ..., A^{(k)}$ be a sequence of matrices generated from the Jacobi algorithm, where $A^{(0)} = A$, our starting matrix. Furthermore, suppose this $A \in \mathbb{R}^{n\times n}_{\mathbb{SYM}}$. Then,*

$$\lim_{k\to\infty} \sum_{i,j=1,i\neq j}^{n} [(A^k)_{ij}]^2 = 0 \tag{2.4.1}$$

.
Moreover,

$$\lim_{k\to\infty} \sum_{i=1}^{n} [(A^k)_{ii}]^2 = Trace(A^2) \tag{2.4.2}$$

This essentially means that, as we iterate through Givens rotations, the off diagonal elements converge to 0, and the diagonals converge to eigenvalues of A.

Proof. Let a_{pq} be the off-diagonal element of A with largest absolute value, and let A' = $G^\top AG$ for G defined as in (2.2.1).
Thus, (2.1.4) implies holds for $A = \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix}$, and our choices of θ and Lemma 1 implies

$$a\prime^2_{pp} + 2a\prime^2_{pq} + a\prime^2_{qq} = a^2_{pp} + 2a^2_{pq} + a^2_{qq}$$

Writing

$$S(A) = \sum_{i,j=1}^{n} a^2_{ij}, D(A) = \sum_{i=1}^{n} a^2_{ii} \text{ and } L(A) = \sum_{i,j=1,i,j\neq 1}^{n} a^2_{ij}$$

it follows that $S(a) = D(A) + L(A)$. Now, $S(A') = S(A)$ by Lemma 1. Thus, $D(A') + L(A') = D(A) + L(A)$. Therefore, the diagonal entries of A' are the same as A, except those 'selected' in rows p and q. Because $(a\prime_p q)^2 = 0$, $a\prime^2_{pp} + a\prime^2_{qq} = a^2_{pp} + 2a^2_{pq} + a^2_{qq}$.
This implies

$$D(A') = D(A) + 2a^2_{pq}, \text{ and } L(A') = L(A) - 2a^2_{pq} \tag{2.4.3}$$

We know a_{pq} is the largest off-diagonal element of A; thus L(A) $\leq Na^2_{pq}$ where $N = n(n-1)$ the number of off-diagonal elements. Therefore, we bound L(A') as

$$L(A') \leq (1 - 2/N)L(A) \tag{2.4.4}$$

Proceeding via induction on k for $A^(k)$ generated via the algorithm in (2.2.2), we see from (2.4.4) that

$$0 \leq L(A^{(k)} \leq (1 - 2/N)^k L(A) \text{ for k=1,2,3..., N} \geq 2 \tag{2.4.5}$$

Thus, $\lim_{k\to\infty} L(A^{(k)}) = 0$ From (2.4.2) then, $\forall k \geq 0$,

$$Trace(A^2) = S(A) = S(A^{(k)}) = D(A^{(k)} + L(A^{(k)} \tag{2.4.6}$$

. Passing $k \to \infty$, we see that $Trace(A^2) = \lim_{k\to\infty} D(A^{(k)})$. (2.4.5) and (2.4.6) both then imply the theorem, that non-diagonal elements converge to 0, and the sum of diagonal elements converges to the trace of A as required. $\square$

3 References

Endre Suli, and David Mayers. An Introduction to Numerical Analysis.Cambridge University Press, 2003.

N. Lloyd Trefethen, and David Bau III. Numerical Linear Algebra.SIAM, 1997.

Timo Heister, and Leo G. Rebholz. Scientific Computing: For Scientists and Engineers.De Gruyter, 2015.