Eigenvalues and Eigenvectors of real matrices

Solving $A\vec{x} = \vec{0}$ is one of the most important problems in physics

Applications: geometric transformations
quantum mechanics
Geology
vibration analysis
image processing
⋮

If $\det A \neq 0 \Rightarrow$ only trivial solution $\vec{x} = \vec{0}$ possible

non-trivial solution for $\det A = 0$

Eigenvalues: λ_i for which $A = A(\lambda_i)$ becomes singular
Eigenvectors: $\vec{x}_i$ that corresponds to λ_i

most prominent example: regular Eigenvalue problems:

$$A(\lambda) = A_0 - \lambda \mathbb{1} \quad \text{, i.e.} \quad A_0 \vec{x} = \lambda \vec{x}$$

$$\det(A_0 - \lambda \mathbb{1}) \stackrel{!}{=} 0$$

Eigenvalues can of course be degenerate: $\lambda_i = \lambda_k$

important special matrices: symmetric : $A = A^T$

hermitian : $A = A^\dagger$

Symmetric and hermitian matrices have only real Eigenvalues

orthogonal: $AA^T = \mathbb{1}$

unitary : $AA^\dagger = \mathbb{1}$

Symmetric and hermitian matrices are normal matrices, i.e. $AA^\dagger = A^\dagger A$

Solving the Eigenvalue problem $A\vec{x} = \vec{0}$ is equivalent to diagonalising A

$$U^{-1} A U = D$$

Eigenvalues of D are Eigenvalues of A : $\lambda_i = d_{ii}$

The eigenvectors are the columns of the transformation matrix U

Numerical realizations: Power iteration (v. Mises iteration)

very simple method
does not use matrix decomposition (can be used for large sparse matrices)
will only find the Eigenvalue with the greatest absolute value
may converge slowly

We start with a vector $\vec{v}_0$

Compute $A\vec{v}_0 = \vec{v}_1'$ $\quad \left(= \sum_{i=1}^{N} A \alpha_i \vec{x}_i = \sum_{i=1}^{N} \alpha_i \lambda_i \vec{x}_i \right)$

Normalize $\vec{v}_1 = \frac{\vec{v}_1'}{|\vec{v}_1'|}$

Compute $A\vec{v}_1 = \vec{v}_2'$ $\quad \left(= \sum_{i=1}^{N} \alpha_i \lambda_i^2 \vec{x}_i \right)$

$\vdots$

$$\vec{v}_t = \sum_{i=1}^{N} \alpha_i \lambda_i^t \vec{x}_i$$

For large t, the contribution of the largest eigenvalue will dominate all others:

$$\vec{v}_t \approx \alpha_1 \lambda_1^t \vec{x}_1$$

$$\lim_{t\to\infty} \frac{v_{t+1}^{(l)}}{v_t^{(l)}} \to \lambda_1 \quad \Rightarrow \lim_{t\to\infty} \vec{v}_t \to \vec{x}_1$$

To avoid problems with very small components, one uses

$$\frac{1}{n'} \sum_{\mu} \frac{v_{t+1}^{(\mu)}}{v_t^{(\mu)}} \approx \lambda_1$$

where we sum over all indices μ with obey $|v_t^{(\mu)}| > \varepsilon$

How to compute the eigenvalue with the smallest absolute value?

$$A \vec{x}_i = \lambda_i \vec{x}_i$$
$$\vec{x}_i = \lambda_i A^{-1} \vec{x}_i$$
$$A^{-1} \vec{x}_i = \frac{1}{\lambda_i} \vec{x}_i$$

Eigenvalues of A^{-1} are the inverse eigenvalues of A
The eigenvectors of A and A^{-1} are the same

Use power method on A^{-1} to get the smallest eigenvalue of A

How to improve convergence? Spectral shift
The power method's convergence ~~improves with~~ depends on the ratio $\frac{\lambda_{n-1}}{\lambda_n}$. The bigger this ratio, the faster it converges

With a suitable estimate for λ, i.e. λ_0, we rewrite the eigenvalue problem
$$A' = A - \lambda_0 \mathbb{1}$$

By doing so, we shift the whole spectrum of A by λ_0:
$$\lambda_n \longrightarrow \lambda_n - \lambda_0$$

and achieve
$$\frac{\lambda_{n-1} - \lambda_0}{\lambda_n - \lambda_0} > \frac{\lambda_{n-1}}{\lambda_n}$$ (we are searching for the smallest eigenvalue!)

$$\lim_{t \to \infty} \frac{v_{t,i}^{(t)}}{v_{t+1}^{(t)}} + \lambda_0 \longrightarrow \lambda_n$$

<u>Jacobi method</u>: calculates the full eigenvalue- and eigenvector set
searches for the transformation matrix U

$$U^T A U = D$$

by a successive sequence of similarity operations

$$U_0^T A U_0 = A^{(1)}$$
$$U_1^{T} A^{(1)} U_1 = A^{(2)} = U_1^T U_0^T A U_0 U_1$$
$$\vdots$$
$$U_t^{T} A^{(t)} U_t = A^{(t+1)} = U_t^T U_{t-1}^T \dots U_0^T A U_0 \dots U_{t-1} U_t$$

all these transformations have to result in

$$\lim_{t\to\infty} A^{(t)} \to D$$
$$\lim_{t\to\infty} U_0 U_1 \dots U_t \to U$$

The Jacobi method uses orthogonal rotation matrices of the form

$$U_t(i,j,\varphi) = \begin{array}{cc} \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ \cdots & & \cos\varphi & \cdots & -\sin\varphi & & \cdots \\ & & & 1 & & & \\ \cdots & & \sin\varphi & \cdots & \cos\varphi & & \cdots \\ & & & & & 1 & \\ & & & & & & 1 \end{pmatrix} & \begin{array}{l} \\ \\ \\ i^{th}\ \text{row} \\ \\ j^{th}\ \text{row} \\ \\ \\ \end{array} \\ \quad\quad i^{th}\ \text{col.} \quad j^{th}\ \text{col.} & \end{array}$$

With i,j and the rotation angle φ $U_t(i,j,\varphi)$ is fully determined and we always have

$$U_t^T U_t = \mathbb{1}$$

What happens at iteration t? $U_{t-1}^T A^{(t-1)} U_{t-1} = A^{(t)}$

$$a_{kl}^{(t)} = \sum_{m=1}^{n} \sum_{m'=1}^{n} U_{mk} U_{m'l}\, a_{mm'}^{(t-1)}$$

Symmetry stays intact: $a_{kl}^{(t)} = a_{lk}^{(t)}$

- If i,j not equal to k,l:

$$a_{kl}^{(t)} = \sum_{m=1}^{n} \sum_{m'=1}^{n} \delta_{mk} \delta_{m'l}\, a_{mm'}^{(t-1)} = a_{kl}^{(t-1)}$$

All components of A not having indices i or j stay the same

- If $l=i$, $k=1,\dots,n$ with $k \neq i,j$:

$$a_{ki}^{(t)} = a_{ik}^{(t)} = a_{ki}^{(t-1)} \cos\varphi + a_{kj}^{(t-1)} \sin\varphi$$

- If $l=j$, $k=1,\dots,n$ with $k \neq i,j$

$$a_{kj}^{(t)} = a_{jk}^{(t)} = a_{kj}^{(t-1)} \cos\varphi - a_{ki}^{(t-1)} \sin\varphi$$

- components ii, jj and ij

$$a_{ii}^{(t)} = a_{ii}^{(t-1)} \cos^2\varphi + 2a_{ij}^{(t-1)} \cos\varphi \sin\varphi + a_{jj}^{(t-1)} \sin^2\varphi$$
$$a_{jj}^{(t)} = a_{jj}^{(t-1)} \cos^2\varphi + 2a_{ij}^{(t-1)} \cos\varphi \sin\varphi + a_{ii}^{(t-1)} \sin^2\varphi$$
$$a_{ij}^{(t)} = a_{ij}^{(t-1)} (\cos^2\varphi - \sin^2\varphi) + \left(a_{jj}^{(t-1)} - a_{ii}^{(t-1)}\right) \cos\varphi \sin\varphi$$

How to choose i, j and φ properly?

The goal is to get $A^{(1)}, A^{(2)}, \ldots, A^{(l)}, \ldots$ closer and closer to a diagonal form

A measure of the „diagonality" of $A^{(l)}$ is the sum of the squares of the non-diagonal elements $S^{(l)}$

$$S^{(l)} = 2 \sum_{m=1}^{n-1} \sum_{m'=m+1}^{n} \left(a_{mm'}^{(l)}\right)^2$$

A successful iteration is then identified by $S^{(l-1)} > S^{(l)}$

The difference between the two sums $S^{(l-1)}$ and $S^{(l)}$ can be shown to depend solely on the elements $a_{ij}^{(l-1)}$ and $a_{ij}^{(l)}$.

$$\Delta S^{(l)} = S^{(l-1)} - S^{(l)} = 2\left(\left(a_{ij}^{(l-1)}\right)^2 - \left(a_{ij}^{(l)}\right)^2\right)$$

$\Delta S^{(l)}$ has a maximal value for: maximal $\left|a_{ij}^{(l-1)}\right|$ and
vanishing $a_{ij}^{(l)}$

Optimally, one would therefore choose i and j to be the indices of the biggest non-diagonal element.

As the search for this biggest element can be quite time-consuming, one often uses a simpler (faster) algorithm to choose i and j.

With $a_{ij}^{(l)} \overset{!}{=} 0$ we get an expression for φ

$$\tan 2\varphi = \frac{2a_{ij}^{(l-1)}}{a_{ii}^{(l-1)} - a_{jj}^{(l-1)}}$$

(If $a_{ii}^{(l-1)} = a_{jj}^{(l-1)}$ one has $\varphi = \frac{\pi}{4}$)

By applying these steps we can assure that

$$S^{(0)} > S^{(1)} > S^{(2)} > \ldots > S^{(l)} > \ldots$$

and finally converge to a diagonal form.

As we are also interested in the eigenvectors of the system, we take a look at the transformation matrices

$$B^{(l-1)} = U_0 U_1 \cdot \ldots \cdot U_{l-2} U_{l-1}$$

As before only the elements of the i^{th} and j^{th} column of $B^{(l-1)}$ are changed when multiplied with $U_l(i,j,\varphi)$

$$\left(B^{(l-1)} U_l(i,j,\varphi)\right)_{k,i} = b_{ki} \cos\varphi + b_{kj} \sin\varphi$$

$$k = 1, \ldots, n$$

$$\left(B^{(l-1)} U_l(i,j,\varphi)\right)_{k,j} = b_{kj} \cos\varphi - b_{ki} \sin\varphi$$

Other symmetric eigenvalue problems:

We want to solve the equation

$$\left(A - \lambda S\right)\vec{x} = \vec{0}$$

A is again a symmetric matrix, S is symmetric and positive-definite (A matrix is called positive-definite if $\vec{z}^{\dagger} M \vec{z}$ is positive for every non-zero column vector $\vec{z}$)

Multiplying this equation by S^{-1} to get the previous form $(S^{-1}A - \lambda \mathbb{1})\vec{x} = \vec{0}$ would be a bad choice as the new matrix $S^{-1}A$ would in general no longer be symmetric.

We therefore want to use a different decomposition, which is known as Cholesky decomposition:

$$S = L \cdot L^{T}$$

Here L is a lower triangular matrix $L = \begin{pmatrix} l_{11} & & & & \\ l_{12} & l_{22} & & 0 & \\ l_{13} & l_{23} & l_{33} & & \\ l_{14} & l_{24} & l_{34} & l_{34} & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$

With this decomposition, we can write

$$\left(A - \lambda L L^{T}\right)\vec{x} = \vec{0}$$
$$\left(A\left(L^{T}\right)^{-1} - \lambda L\right)\left(L^{T}\vec{x}\right) = \vec{0}$$
$$\left(L^{-1}A\left(L^{T}\right)^{-1} - \lambda \mathbb{1}\right)\left(L^{T}\vec{x}\right) = \vec{0}$$

$$\left(C - \lambda \mathbb{1}\right)\vec{y} = \vec{0}$$

To calculate the L_{ij} we proceed column-wise (index j). Because of the triangular shape of L we have $i \leq j$

1^{st} col: $s_{11} = L_{11}^2 \implies L_{11} = \sqrt{s_{11}}$

$$s_{i1} = L_{i1} L_{11} \implies L_{i1} = \frac{s_{i1}}{L_{11}} \quad \text{for } i = 2, \dots, n$$

2^{nd} col. $S_{22} = L_{21}^2 + L_{22}^2 \implies L_{22} = \sqrt{S_{22} - L_{21}^2}$

$$S_{i2} = L_{i1} L_{21} + L_{i2} L_{22} \implies L_{i2} = \frac{S_{i2} - L_{i1} L_{21}}{L_{22}} \quad \text{for } i = 3, \dots, n$$

j^{th} col:

$$L_{jj} = S_{jj} - \sum_{t=1}^{j-1} L_{jt}^2$$

$$L_{ij} = \frac{S_{ij} - \sum_{t=1}^{j-1} L_{it} L_{jt}}{L_{jj}} \quad \text{for } i = j+1, \dots, n$$

As one can see, problems will arise for very small or negative arguments of the square root. This ~~must~~ can not happen for positive-definite matrices, so whenever

$$S_{jj} - \sum_{t=1}^{j-1} L_{jt}^2 < 0$$

S is not positive-definite and we may not use the Cholesky decomposition

After reducing the problem to the original form $(A - \lambda \mathbb{1})\vec{x} = \vec{0}$ we can now use the Jacobi method to calculate the eigensystem

For large matrices the Jacobi method is quite slow and other algorithms are preferable, i.e.

- use a non-iterative procedure to rewrite A in a more simple form (Householder algorithm → tridiagonal form)
- Perform a QR or QL decomposition to get eigensystem

General real matrices: What about matrices, that are not symmetric?

In the case of more general matrices without symmetry properties we also want to use a two-part algorithm.

In a first step the matrix is transformed into a simpler form using a non-iterative algorithm

The second step then calculates the eigenvalues of this new matrix

We want to demonstrate this by rewriting a general matrix in Upper Hessenberg Form (UHF) and then use the method by Hyman.