5.Eigen

April 22, 2025

0.1 STAT 207: Eigenvalues and Eigenvectors

Zhe Fei (zhe.fei@ucr.edu)

- NAS Chapter 8

Finding the eigenvalues and eigenvectors of a symmetric matrix is one of the basic tasks of computational statistics.

- Application 1: **PCA** a random m-vector X with covariance matrix Ω,

$$\Omega = UDU^T,$$

where $D = \text{diag}(\lambda_1, \lambda_2, .., \lambda_m)$ and U is the orthogonal matrix of eigenvectors.

- Application 2: If Ω is the covariance matrix of a normally distributed random vector X with mean $E(X) = \mu$, then the quadratic form and the determinant

$$(x-\mu)\Omega^{-1}(x-\mu) = [U^t(x-\mu)]^t D^{-1} U^t (x-\mu)$$
$$\det(\Omega) = \prod_i \lambda_i$$

appearing in the density of X.

0.1.1 Jacobi's Method

- ideas for proving convergence of iterative methods in general
- easy to implement with parallel computing

Basic idea:

- repeatedly applying a sequence of similarity transformations to the matrix until its off-diagonal elements are sufficiently small to be considered zero.
- the diagonal elements of the matrix represent its eigenvalues,
- the rows (or columns) of the transformed matrix correspond to its eigenvectors.

Apply the rotation

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

to any row k and column ℓ of the $m \times m$ symmetrix matrix $A = (a_{ij})$. WLOG, we take $k = 1, \ell = 2$ and

$$U = \begin{pmatrix} R & 0 \\ 0^T & I_{m-2} \end{pmatrix},$$

then the upper-left block of $B = U^T A U$ becomes:

$$\begin{aligned} b_{11} &= a_{11}\cos^2\theta - 2a_{12}\cos\theta\sin\theta + a_{22}\sin^2\theta \\ b_{12} &= (a_{11} - a_{22})\cos\theta\sin\theta + a_{12}(\cos^2\theta - \sin^2\theta) \\ b_{22} &= a_{11}\sin^2\theta + 2a_{12}\cos\theta\sin\theta + a_{22}\cos^2\theta. \end{aligned}$$

Further,

$$b_{12} = \frac{a_{11} - a_{22}}{2}\sin(2\theta) + a_{12}\cos(2\theta).$$

To force $b_{12} = 0$,

$$\begin{cases} \tan(2\theta) = \frac{2a_{12}}{a_{22}-a_{11}} & \text{if } a_{22} - a_{11} \neq 0 \\ \theta = \pi/4 & \text{if } a_{22} - a_{11} = 0 \end{cases}$$

And

$$b_{11} = a_{11} - a_{12}\tan(\theta), b_{22} = a_{22} + a_{12}\tan(\theta).$$

Because $\|B\|_F^2 = \|A\|_F^2$,

$$b_{11}^2 + b_{22}^2 = a_{11}^2 + a_{22}^2 + 2a_{12}^2,$$

which implies the off-diagonal part

$$\text{off}(B) = \text{off}(A) - 2a_{12}^2.$$

Finally,

$$\Omega_n = U_n^T ... U_1^T \Omega U_1 ... U_n.$$

Classical Jacobi: search for the largest $|a_{ij}|$ at each iteration. **What is the rate of convergence?**

Parallel Jacobi:

- Distribute rows of A to multiple processors.
- Perform computation based on the owner-computes rule.
- Perform all-all broadcasting after each iteration.

reference

[1]:
```python
import numpy as np

def jacobi_eigenvalue(A, tol=1e-8):
    # initialize matrix V as identity matrix
    V = np.eye(A.shape[0])
    while True:
        # get the index of the largest off-diagonal element
        max_idx = np.argmax(np.abs(np.triu(A, 1))) # flattened upper tri mat
        i, j = divmod(max_idx, A.shape[1])

        # calculate the rotation angle
        if A[i,i] == A[j,j]:
            theta = np.pi / 4
        else:
```

```python
            theta = 0.5 * np.arctan(2 * A[i,j] / (A[i,i] - A[j,j]))

        # create the rotation matrix
        S = np.eye(A.shape[0])
        S[i,i] = np.cos(theta)
        S[j,j] = np.cos(theta)
        S[i,j] = -np.sin(theta)
        S[j,i] = np.sin(theta)

        # update A and V with the rotation
        A = S.T @ A @ S
        V = V @ S

        # check if the off-diagonal elements are below tolerance
        if np.max(np.abs(np.triu(A, 1))) < tol:
            break

    # return the diagonal elements of A as eigenvalues
    eigenvalues = np.diag(A)

    # sort eigenvalues and eigenvectors
    idx = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[idx]
    eigenvectors = V[:,idx]

    return eigenvalues, eigenvectors
```

[2]:
```python
# generate a random matrix
A = np.random.randint(1, 10, size=(4, 4))

# Make the matrix symmetric
A = np.tril(A) + np.tril(A, -1).T

print(A)
```

```
[[6 2 7 4]
 [2 8 5 7]
 [7 5 9 8]
 [4 7 8 3]]
```

[4]:
```python
w, v = jacobi_eigenvalue(A)

print(w)
print(v)
```

```
[23.57512911  5.93479954  0.09322972 -3.60315837]
[[ 0.40923118 -0.58156807  0.70278377  0.02008474]
```

```
 [ 0.46710256  0.73985457  0.34981776 -0.33475059]
 [ 0.61999028 -0.29769955 -0.59550827 -0.41515892]
 [ 0.47953842  0.16052656 -0.17056497  0.84568417]]
```

0.1.2 The Rayleigh Quotient

Definition:

$$R(x) = \frac{x^T A x}{x^T x}$$

for $x \neq 0$.

Let A have eigenvalues $\lambda_1, \dots, \lambda_m$ and corresponding orthonormal eigenvectors $u_1, \dots, u_m$.

With the unique presentation $x = \sum_{i=1}^{m} c_i u_i$,

$$R(x) = \frac{\sum_{i=1}^{m} \lambda_i c_i^2}{\sum_{i=1}^{m} c_i^2}.$$

Therefore, $\lambda_1 \leq R(x) \leq \lambda_m$ and the equality $R(u_m) = \lambda_m$.

Hence, $R(x)$ is maximized by $x = u_m$ and correspondingly minimized by $x = u_1$. The following generalizes this result.

NAS Proposition 8.3.1 (Courant-Fischer) Let V_k be a k-dimensional subspace of $\mathbb{R}^m$. Then

$$\begin{aligned} \lambda_k &= \min_{V_k} \max_{x \in V_k,\ x \neq 0} R(x) \\ &= \max_{V_{m-k+1}} \min_{y \in V_{m-k+1},\ y \neq 0} R(y). \end{aligned}$$

The next proposition shows how much the eigenvalues of a symmetric matrix change under a symmetric perturbation of the matrix.

NAS Proposition 8.3.2 Let the $m \times m$ symmetric matrices A and $B = A + \Delta A$ have ordered eigenvalues $\lambda_1, \dots, \lambda_m$ and $\mu_1, \dots, \mu_m$, respectively. Then the inequality

$$|\lambda_k - \mu_k| \leq \|\Delta A\|_2$$

holds for all $k \in \{1, \dots, m\}$.

0.1.3 Finding a Single Eigenvalue

The **power method** iterates:

$$u_n = \frac{1}{\|A u_{n-1}\|_2} A u_{n-1}$$

to find the dominant eigenvector whenever A is diagonalizable.

- To find the eigenvalue with smallest absolute value, use the inverse power method with A^{-1} instead of A.
- To find any eigenvalue λ_i, with μ close to λ_i, update using $(A - \mu I)^{-1}$.

The **Rayleigh quotient iteration algorithm** to find the dominant eigenvalue λ_m, with $\mu_n = u_n^T A u_n$,

$$u_n = \frac{1}{\|(A-\mu_{n-1}I)^{-1}u_{n-1}\|_2}(A-\mu_{n-1}I)^{-1}u_{n-1}.$$

- It converges at a cubic rate;
- Also works for the smallest eigenvalue

[ ]:

```

```