eigenvalues_eigenvectors

August 21, 2025

1 Eigenvalues & Eigenvectors — Theory and Algorithms

This notebook covers definitions, properties, and implements numerical algorithms from scratch: - Power iteration (dominant eigenpair) - Simple QR iteration (for all eigenvalues) and compares with NumPy's built-in solvers.

1.1 1. Definitions

An eigenvector (v) and scalar ( ) satisfy (A v = v).

[1]:

```python
import numpy as np

def power_iteration(A, num_iters=1000, tol=1e-10):
    n = A.shape[0]
    b = np.random.randn(n)
    b = b / np.linalg.norm(b)
    lambda_old = 0.0
    for i in range(num_iters):
        b1 = A @ b
        b1_norm = np.linalg.norm(b1)
        b = b1 / b1_norm
        lambda_new = b.T @ (A @ b)
        if abs(lambda_new - lambda_old) < tol:
            break
        lambda_old = lambda_new
    return lambda_new, b

def qr_eigenvalues(A, num_iters=1000, tol=1e-10):
    A_k = A.copy().astype(float)
    n = A.shape[0]
    for i in range(num_iters):
        Q, R = np.linalg.qr(A_k)
        A_k1 = R @ Q
        if np.linalg.norm(np.triu(A_k1, k=1)) < tol:
            break
        A_k = A_k1
    return np.diag(A_k1)
```

```
# Example matrix
A = np.array([[4, 1, -2],
              [1, 3, 0],
              [-2, 0, 1]], dtype=float)

lam, vec = power_iteration(A)
print('Power iteration dominant eigenvalue:', lam)
print('Power iteration eigenvector (normalized):', vec)

print('\nQR iteration eigenvalues:', qr_eigenvalues(A))
print('\nNumPy eigenvalues:', np.linalg.eigvals(A))
```

```
Power iteration dominant eigenvalue: 5.346462189540798
Power iteration eigenvector (normalized): [-0.84716746 -0.36103704  0.38981989]

QR iteration eigenvalues: [ 5.34646219  2.72224563 -0.06870782]

NumPy eigenvalues: [-0.06870782  5.34646219  2.72224563]
```

1.2 2. Verifications and properties

We verify eigen-decomposition numerically and discuss orthogonality for symmetric matrices.

[2]:

```
eigvals, eigvecs = np.linalg.eig(A)
print('Eigenvalues (numpy):', eigvals)
print('Eigenvectors (numpy columns):\n', eigvecs)
# Verify Av = lambda v for first eigenpair
lam0 = eigvals[0]; v0 = eigvecs[:,0]
print('\nCheck Av - lam v (should be near 0):', np.linalg.norm(A @ v0 - lam0 *␣
 ↪v0))
```

```
Eigenvalues (numpy): [-0.06870782  5.34646219  2.72224563]
Eigenvectors (numpy columns):
 [[ 0.46582728 -0.84716665 -0.25556529]
 [-0.15179916 -0.36103997  0.92011258]
 [ 0.87175797  0.38981894  0.29678146]]

Check Av - lam v (should be near 0): 1.3386053023199622e-15
```