SOFTWARE ASSIGNMENT

EE24BTECH11061 - Rohith Sai

Eigenvectors and Eigenvalues

Eigenvectors

An **eigenvector** of a square matrix A is a vector that, when transformed by A, does not change its direction. Mathematically, if A is an $n \times n$ matrix, then an eigenvector $\mathbf{x}$ associated with A is a non-zero vector that satisfies:

$$A\mathbf{x} = \lambda\mathbf{x}$$

where:

- $\mathbf{x}$ is a non-zero vector.
- λ is the eigenvalue associated with $\mathbf{x}$.

This equation means that, the action of A on $\mathbf{x}$ (multiplying A by $\mathbf{x}$) is the same as simply scaling $\mathbf{x}$ by the factor λ. So, instead of changing the direction of $\mathbf{x}$, the matrix A only changes its length.

Eigenvalues

An **eigenvalue** is a scalar associated with a particular eigenvector. It tells us how much the eigenvector is scaled (stretched, shrunk, or flipped) when multiplied by the matrix A. So, in the equation:

$$A\mathbf{x} = \lambda\mathbf{x}$$

λ is the eigenvalue, which can have several effects on the eigenvector $\mathbf{x}$:

- If $\lambda > 1$, the eigenvector is stretched.
- If $0 < \lambda < 1$, the eigenvector is compressed.
- If $\lambda = 1$, the eigenvector stays the same length.
- If $\lambda = 0$, the eigenvector is mapped to the zero vector (a ”degenerate” case, often meaning that A has some singular or non-invertible behavior).
- If $\lambda < 0$, the eigenvector is flipped (pointing in the opposite direction).

Computing Eigenvalues

There are several methods to calculate eigenvalues of a matrix.

1) **Power Iteration**
2) **Inverse Iteration**
3) **Rayleigh Quotient Iteration**
4) **QR Algorithm**
5) **Householder Reduction**

Power Iteration

- **Description:** Power iteration is an iterative method that finds the dominant eigenvalue (the eigenvalue with the largest magnitude) and its corresponding eigenvector.
- **Process:** Repeatedly multiply an arbitrary vector $\mathbf{x}$ by a matrix A and normalize the result. The vector will eventually converge to the dominant eigenvector, and the eigenvalue can be estimated from the ratio of successive vector norms.

Inverse Iteration

- **Description:** Inverse iteration is a modification of power iteration that can be used to find an eigenvalue close to a given initial estimate. It is useful for finding non-dominant eigenvalues.
- **Process:** For an approximate eigenvalue λ_0, the matrix $(A - \lambda_0 I)^{-1}$ is used in power iteration. This method can yield an eigenvector associated with an eigenvalue close to λ_0.

Rayleigh Quotient Iteration

- **Description:** Rayleigh quotient iteration is an iterative method that converges cubically to an eigenvalue near an initial guess. It is similar to inverse iteration but uses the Rayleigh quotient to update the approximation of the eigenvalue.
- **Process:** At each step, the Rayleigh quotient is computed to update the eigenvalue estimate.

QR Algorithm

- **Description:** The QR algorithm is one of the most widely used methods for finding all eigenvalues of a matrix. It iteratively factors the matrix A into QR (where Q is an orthogonal matrix and R is upper triangle) and then form $A_{k+1} = RQ$. This process eventually leads to a matrix that is nearly diagonal, with eigenvalues on the diagonal.
- **Suitability:** Effective for both symmetric and non-symmetric matrices and can be adapted to find both real and complex eigenvalues.

Householder Reduction

- **Description:** Householder transformations are used to reduce a matrix to a simpler form, such as upper Hessenberg form (for general matrices) or tridiagonal form (for symmetric matrices), before applying other algorithms like the QR algorithm.
- **Suitability:** Mainly used as a preprocessing step to improve the efficiency of other eigenvalue algorithms

The following table summarizes about all the methods to calculate eigenvalues:

Method	Matrix Type	Finds	Pros	Cons
Power Iteration	Any	Largest eigenvalue	Simple and efficient for largest eigenvalue	Only finds dominant eigenvalue
Inverse Iteration	Any	One eigenvalue	Finds non-dominant eigenvalues	Needs good initial guess, inversion
Rayleigh Quotient Iteration	Any	One eigenvalue	Very fast convergence near eigenvalue	Expensive and needs a good initial guess
QR Algorithm	Any	All eigenvalues	Effective for dense matrices	Computationally intensive
Householder Reduction	Any	Used in QR, others	Simplifies matrix for other methods	Primarily a preprocessing step

Before choosing any algorithm there are a few things to keep in mind:

- Time Complexity
- Memory Usage
- Convergence Rate

Time Complexity

Time complexity describes how the computation time of an algorithm grows with the size of the input, typically measured in terms of big-O notation.

Memory Usage

Memory usage describes how much memory an algorithm needs relative to the input size.

Convergence Rate

Convergence rate describes how quickly an iterative algorithm approaches the solution. Faster convergence means fewer iterations are needed for an acceptable accuracy.

The table for the time complexity, memory usage, convergence rate for all the above algorithms is given below:

Algorithm	**Time Complexity**	**Memory Usage**	**Convergence Rate Order**
Power Iteration	$O(n^2)$ per iteration	$O(n)$	Linear ($\lvert\lambda_2/\lambda_1\rvert$)
Inverse Iteration	$O(n^3)$	$O(n^2)$	Linear (if close to eigenvalue (λ))
Rayleigh Quotient Iteration	$O(n^3)$ per iteration	$O(n^2)$	Cubic (superlinear near eigenvalue)
QR Algorithm	$O(n^3)$ for dense, $O(kn)$ for sparse	$O(n^2)$	Quadratic to cubic (depending on shifts)
Householder Reduction	$O(n^3)$	$O(n^2)$	Not iterative (used for tridiagonalization)

Conclusion

From the above two tables, I came to the conclusion to use the Householder Reduction followed by QR decomposition to compute eigenvalues. This is for the following reasons:

- **Faster Convergence of QR Algorithm:**
  Householder reduction transforms the matrix into a bidiagonal form, allowing the QR algorithm to converge more quickly. This reduces the number of iterations needed to reach convergence, making the method more efficient than direct diagonalization.
- **Reduction in Computational Complexity:**
  The Householder transformation simplifies the matrix to a bidiagonal form, which is easier for QR decomposition to handle. This results in lower computational cost, especially for large matrices, compared to methods like direct diagonalization.
- **Preservation of Orthogonality:**
  Householder reflections are orthogonal, preserving the matrix's eigenvalues and ensuring numerical stability. This prevents error accumulation, making the method reliable for ill-conditioned matrices.

- **Good Convergence Behavior:**
  The QR algorithm with Householder reduction converges quickly, even for large or ill-conditioned matrices. This is faster than methods like Power Iteration, making it more suitable for large-scale eigenvalue problems.
- **Time Complexity and Memory Usage:**
  The time complexity of the Householder reduction and QR algorithm is $O(n^3)$, with reduced iterations improving efficiency. Memory usage is $O(n^2)$, which is manageable for large matrices and more efficient than methods like Power Iteration.

References

- Gilbert Strang *MIT OpenCourseWare*. Available at:
  Computing Eigenvalues and Singular Values and Eigenvalues and Eigenvectors
- Gilbert Strang, *Linear Algebra and Its Applications*, Wellesley-Cambridge Press, 5th edition, 2016. Available at:
  https://math.mit.edu/ gs/linearalgebra/
- Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 4th edition, 2013. More details:
  https://jhupbooks.press.jhu.edu/title/matrix-computations
- MIT OpenCourseWare, *Linear Algebra Course*, 2010. Available at:
  https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/