Computational Methods
CMSC/AMSC/MAPL 460

Eigenvalues and Eigenvectors

Ramani Duraiswami,
Dept. of Computer Science
Eigen Values of a Matrix

• Definition:

• A $N \times N$ matrix $A$ has an eigenvector $x$ (non-zero) with corresponding eigenvalue $\lambda$ if

$$Ax = \lambda x$$

• This means

$$Ax - \lambda x = 0 \quad \Rightarrow \quad (A - \lambda I)x = 0$$

• If a matrix vector product gives a zero vector, then either the vector is zero, or the matrix has zero determinant (is singular).

• Here this means $\det(A - \lambda I) = 0$
Left and Right Eigenvectors

- Right eigenvector of a matrix $A$ is
  \[ Ax = \lambda x \]
- For a $N \times N$ matrix we can also define a left matrix product
  \[ y^t A = v^t \]
- So if we have
  \[ y^t A = \lambda y^t \]
  then $y$ is a left eigenvector of $A$
- If $A$ is symmetric $A = A^t$
- $(Ax)^t = x^t A^t = x^t A = (\lambda x)^t = \lambda x^t$
- So left and right eigenvectors of a symmetric matrix are the same
Symmetric Matrices

• A matrix is symmetric if its transpose is equal to itself
• \( A \) is symmetric if \( A^t = A \) For a complex matrix \( A^H = A \)
• Eigenvalues and Eigenvectors of a real symmetric (complex hermitian) matrix are real and eigenvectors are orthogonal.
Characteristic Equation

- $Ax = \lambda x$ can be written as

  $$(A - \lambda I)x = 0$$

  which holds for $x \neq 0$, so $(A - \lambda I)$ is singular and

  $$\det(A - \lambda I) = 0$$

- This is called the characteristic polynomial. If $A$ is $n \times n$ the polynomial is of degree $n$ and so $A$ has $n$ eigenvalues, counting multiplicities.
Example

\[ A = \begin{pmatrix} 4 & 3 \\ 1 & 2 \end{pmatrix} \quad A - \lambda I = \begin{pmatrix} 4 - \lambda & 3 \\ 1 & 2 - \lambda \end{pmatrix} \]

\[ \det(A - \lambda I) = 0 \quad \Rightarrow \quad (4 - \lambda)(2 - \lambda) - (1)(3) = 0 \]

\[ \lambda^2 - 6\lambda + 5 = 0 \quad \Rightarrow \quad (\lambda - 5)(\lambda - 1) = 0 \]

• Hence the two eigenvalues are 1 and 5.
Example (continued)

- Once we have the eigenvalues, the eigenvectors can be obtained by substituting back into $(A - \lambda I)x = 0$.
- This gives eigenvectors $(1 -1)^T$ and $(1 1/3)^T$.
- Note that we can scale the eigenvectors any way we want.
- Determinant are not used for finding the eigenvalues of large matrices.
Positive Definite Matrices

- A complex matrix $A$ is *positive definite* if for every nonzero complex vector $x$ the quadratic form $x^H A x$ is real and:

$$x^H A x > 0$$

where $x^H$ denotes the conjugate transpose of $x$ (i.e., change the sign of the imaginary part of each component of $x$ and then transpose).
Eigenvalues of Positive Definite Matrices

- If $A$ is positive definite and $\lambda$ and $x$ are an eigenvalue/eigenvector pair, then:

$$Ax = \lambda x \quad x^HAx = \lambda x^Hx$$

- Since $x^HAx$ and $x^Hx$ are both real and positive it follows that $\lambda$ is real and positive.
Properties of Positive Definite Matrices

- If $A$ is a positive definite matrix then:
  - $A$ is nonsingular.
  - The inverse of $A$ is positive definite.
  - Gaussian elimination can be performed on $A$ without pivoting.
  - The eigenvalues of $A$ are positive.
Hermitian Matrices

- A square matrix for which $A = A^H$ is said to be an *Hermitian* matrix.
- If $A$ is real and Hermitian it is said to be *symmetric*, and $A = A^T$.
- Every Hermitian matrix is positive definite.
- Every eigenvalue of an Hermitian matrix is real.
- Different eigenvectors of an Hermitian matrix are orthogonal to each other, i.e., their scalar product is zero.
Eigen Decomposition

- Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of the $n \times n$ matrix $A$ and $x_1, x_2, \ldots, x_n$ the corresponding eigenvectors.
- Let $\Lambda$ be the diagonal matrix with $\lambda_1, \lambda_2, \ldots, \lambda_n$ on the main diagonal.
- Let $X$ be the $n \times n$ matrix whose $j$th column is $x_j$.
- Then $AX = X \Lambda$, and so we have the eigen decomposition of $A$:
  \[ A = X^t \Lambda X^{-1} \]
- This requires $X$ to be invertible, thus the eigenvectors of $A$ must be linearly independent.
Powers of Matrices

• If \( A = X^t \Lambda X^{-1} \) then:
  \[
  A^2 = (X^t \Lambda X^{-1})(X^t \Lambda X^{-1}) = X^t \Lambda (X^{-1}X) \Lambda X^{-1} = X^t \Lambda X^{-1}^2
  \]
  Hence we have:
  \[
  A^p = X^t \Lambda X^{-1}
  \]
  • Thus, \( A^p \) has the same eigenvectors as \( A \), and its eigenvalues are \( \lambda_1^p, \lambda_2^p, \ldots, \lambda_n^p \).
  • We can use these results as the basis of an iterative algorithm for finding the eigenvalues of a matrix.
The Power Method

• Label the eigenvalues in order of decreasing absolute value so $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|.$

• Consider the iteration formula:

$$y_{k+1} = Ay_k$$

where we start with some initial $y_0$, so that:

$$y_k = A^ky_0$$

• Then $y_k$ converges to the eigenvector $x_1$ corresponding the eigenvalue $\lambda_1$. 
Proof

- We know that $A^k = X \Lambda^k X^{-1}$, so:
  
  $$y_k = A^k y_0 = X \Lambda^k X^{-1} y_0$$

- Now we have:

  $$\Lambda^k = \begin{pmatrix} 
  \lambda_1^k \\
  \lambda_2^k \\
  \vdots \\
  \lambda_n^k 
  \end{pmatrix} = \begin{pmatrix} 
  \lambda_1 \\
  \lambda_2 \\
  \vdots \\
  \lambda_n 
  \end{pmatrix}^k$$

- The terms on the diagonal get smaller in absolute value as $k$ increases, since $\lambda_1$ is the dominant eigenvalue.
Proof (continued)

• So we have

\[
y_k = \lambda_1^k \begin{pmatrix} \vdots & \cdots & \vdots \\ x_1 & \cdots & x_n \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \lambda_1^k c_1 x_1
\]

• Since \( \lambda_1^k c_1 x_1 \) is just a constant times \( x_1 \) then we have the required result.
Example

- Let \( A = \begin{bmatrix} 2 & -12; 1 & -5 \end{bmatrix} \) and \( y_0 = \begin{bmatrix} 1 & 1 \end{bmatrix} \)’
- \( y_1 = -4\begin{bmatrix} 2.50 & 1.00 \end{bmatrix} \)’
- \( y_2 = 10\begin{bmatrix} 2.80 & 1.00 \end{bmatrix} \)’
- \( y_3 = -22\begin{bmatrix} 2.91 & 1.00 \end{bmatrix} \)’
- \( y_4 = 46\begin{bmatrix} 2.96 & 1.00 \end{bmatrix} \)’
- \( y_5 = -94\begin{bmatrix} 2.98 & 1.00 \end{bmatrix} \)’
- \( y_6 = -190\begin{bmatrix} 2.99 & 1.00 \end{bmatrix} \)’
- The iteration is converging on a scalar multiple of \( \begin{bmatrix} 3 & 1 \end{bmatrix} \)’, which is the correct dominant eigenvector.
Rayleigh Quotient

- Note that once we have the eigenvector, the corresponding eigenvalue can be obtained from the Rayleigh quotient:

\[
\frac{\text{dot}(Ax, x)}{\text{dot}(x, x)}
\]

where \( \text{dot}(a, b) \) is the scalar product of vectors \( a \) and \( b \) defined by:

\[
\text{dot}(a, b) = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n
\]

- So for our example, \( \lambda_1 = -2 \).
Scaling

• The $\lambda_1^k$ can cause problems as it may become very large as the iteration progresses.
• To avoid this problem we scale the iteration formula:
  \[ y_{k+1} = A(y_k/r_{k+1}) \]
  where $r_{k+1}$ is the component of $Ay_k$ with largest absolute value.
Example with Scaling

• Let $A = \begin{bmatrix} 2 & -12; 1 & -5 \end{bmatrix}$ and $y_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}'$
• $Ay_0 = \begin{bmatrix} -10 & -4 \end{bmatrix}'$ so $r_1 = -10$ and $y_1 = \begin{bmatrix} 1.00 & 0.40 \end{bmatrix}'$
• $Ay_1 = \begin{bmatrix} -2.8 & -1.0 \end{bmatrix}'$ so $r_2 = -2.8$ and $y_2 = \begin{bmatrix} 1.0 & 0.3571 \end{bmatrix}'$
• $Ay_2 = \begin{bmatrix} -2.2857 & -0.7857 \end{bmatrix}'$ so $r_3 = -2.2857$ and $y_3 = \begin{bmatrix} 1.0 & 0.3437 \end{bmatrix}'$
• $Ay_3 = \begin{bmatrix} -2.1250 & -0.7187 \end{bmatrix}'$ so $r_4 = -2.1250$ and $y_4 = \begin{bmatrix} 1.0 & 0.3382 \end{bmatrix}'$
• $Ay_4 = \begin{bmatrix} -2.0588 & -0.6912 \end{bmatrix}'$ so $r_5 = -2.0588$ and $y_5 = \begin{bmatrix} 1.0 & 0.3357 \end{bmatrix}'$
• $Ay_5 = \begin{bmatrix} -2.0286 & -0.6786 \end{bmatrix}'$ so $r_6 = -2.0286$ and $y_6 = \begin{bmatrix} 1.0 & 0.3345 \end{bmatrix}'$
• $r$ is converging to the correct eigenvector -2.
Scaling Factor

- At step $k+1$, the scaling factor $r_{k+1}$ is the component with largest absolute value is $A y_k$.
- When $k$ is sufficiently large $A y_k \simeq \lambda_1 y_k$.
- The component with largest absolute value in $\lambda_1 y_k$ is $\lambda_1$ (since $y_k$ was scaled in the previous step to have largest component 1).
- Hence, $r_{k+1} \rightarrow \lambda_1$ as $k \rightarrow \infty$. 
function [lambda, y] = powerMethod(A, y, n)
for (i=1:n)
    y = A * y;
    [c j] = max(abs(y));
    lambda = y(j);
    y = y / lambda;
end
Convergence

- The Power Method relies on us being able to ignore terms of the form $(\lambda_j / \lambda_1)^k$ when $k$ is large enough.
- Thus, the convergence of the Power Method depends on $|\lambda_2|/|\lambda_1|$.
- If $|\lambda_2|/|\lambda_1|=1$ the method will not converge.
- If $|\lambda_2|/|\lambda_1|$ is close to 1 the method will converge slowly.
Orthonormal Vectors

- A set $S$ of nonzero vectors are *orthonormal* if, for every $x$ and $y$ in $S$, we have $\text{dot}(x,y)=0$ (orthogonality) and for every $x$ in $S$ we have $\|x\|_2=1$ (length is 1).
The QR Algorithm

• The QR algorithm for finding eigenvalues is based on the QR factorisation that represents a matrix $A$ as:

$$A = QR$$

where $Q$ is a matrix whose columns are orthonormal, and $R$ is an upper triangular matrix.

• Note that $Q^H Q = I$ and $Q^{-1} = Q^H$.

• $Q$ is termed a *unitary* matrix.
QR Algorithm without Shifts

\[ A_0 = A \]

for \( k = 1, 2, \ldots \)

\[ Q_k R_k = A_k \]

\[ A_{k+1} = R_k Q_k \]

end

Since:

\[ A_{k+1} = R_k Q_k = Q_k^{-1} A_k Q_k \]

then \( A_k \) and \( A_{k+1} \) are similar and so have the same eigenvalues.

\( A_{k+1} \) tends to an upper triangular matrix with the same eigenvalues as \( A \). These eigenvalues lie along the main diagonal of \( A_{k+1} \).
QR Algorithm with Shift

\[ A_0 = A \]
for k=1,2,…
\[ s = A_k(n,n) \]
\[ Q_kR_k = A_k - sI \]
\[ A_{k+1} = R_kQ_k + sI \]
end

Since:
\[ A_{k+1} = R_kQ_k + sI \]
\[ = Q_k^{-1}(A_k - sI)Q_k + sI \]
\[ = Q_k^{-1}A_kQ_k \]
so once again \( A_k \) and \( A_{k+1} \) are similar and so have the same eigenvalues.

The shift operation subtracts \( s \) from each eigenvalue of \( A \), and speeds up convergence.
MATLAB Code for QR Algorithm

• Let $A$ be an $n \times n$ matrix

$$n = \text{size}(A, 1);$$
$$I = \text{eye}(n, n);$$
$$s = A(n,n); \ [Q,R] = \text{qr}(A-s*I); \ A = R*Q+s*I$$

• Use the up arrow key in MATLAB to iterate or put a loop round the last line.
Deflation

- The eigenvalue at $A(n,n)$ will converge first.
- Then we set $s=A(n-1,n-1)$ and continue the iteration until the eigenvalue at $A(n-1,n-1)$ converges.
- Then set $s=A(n-2,n-2)$ and continue the iteration until the eigenvalue at $A(n-2,n-2)$ converges, and so on.
- This process is called *deflation*.