Math 6610  

10/21/13

- Now building up to building up to the most widely used general purpose algorithm for computing all eigenvalues and eigenvectors of a general dense matrix.

- The QR algorithm.

- This is what MATLAB does!

- Good example for a typical process in mathematics. Start with something simple and keep refining it until it grows into something quite sophisticated.

- Recall the power method.

- Suppose $A$ an $n \times n$ real and

  $|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$

  $Ax_i = \lambda_i x_i, \quad x_i \neq 0$

- $\lambda_1$ is the dominant eigenvalue.
- Given $q(0)$
  
  For $k = 0, 1, 2, \ldots$ until satisfied
  
  $$z^{(k+1)} = Aq^{(k)}$$
  
  $$q^{(k+1)} = \frac{z^{(k+1)}}{||z^{(k+1)}||}$$

- First question: given an approximate eigenvector $q = q^{(k+1)}$, say, how do we estimate the corresponding eigenvalue?

  $$F(x) = ||Aq - 2q||^2 = \min_{||x||=1} ||x||^2$$
  
  $$F(x) = (Aq - 2q)^T(Aq - 2q)$$
  
  $$= q^TAAq - 2q^TAq + 2q^Tq + 2^2q^Tq$$

  $$F(x) = 2\lambda q^Tq - q^T(A + A^T)q = 0$$

  $$\lambda = \frac{q^T(A + A^T)q}{2q^Tq}$$

In the special case that $A = A^T$ this turns into

$$\lambda = \frac{q^TAq}{q^Tq} \quad \text{The Rayleigh Quotient}$$
- What can go wrong?
- No dominant eigenvalue
- \( \lambda \) real, multiple. The iteration converges to a vector in the corresponding invariant subspace
- \( \lambda \), complex. Get an oscillation

Simplest example

\[
A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad \lambda = \pm i
\]

\[
q^{(0)} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad q^{(1)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \ldots \text{ etc}
\]

- The main problem, however, is that the method converges only slowly if

\[
\left| \frac{x_1}{x_2} \right| \text{ close to } 1.
\]
There are several modifications of the power method.

- **Shift of origin**

  Apply the power method to
  \[ B = A - \mu I \]

  The eigenvalues of B are \( 2, -\mu \), and the PM will converge to the dominant eigenvalue of B.

- **Inverse Iteration**

  Apply the PM to \( A^{-1} \).

  of course we never invert a matrix!

  \[ q(0) \text{ given} \]

  For \( k = 0, 1, 2, \ldots \)

  \[ A z^{(k+1)} = q^{(k)} \]

  \[ q^{(k+1)} = \frac{z^{(k+1)}}{\|z^{(k+1)}\|} \]

- This is a good example when we solve many linear systems with the same matrix and different right hand sides, and where we know the new RHS only after solving the previous system.
Shift of Origin and Inverse Iteration can be combined.

Apply PM to \((A - \mu I)^{-1}\):

\[ A - \mu I = PLU \]

\[ q(0) \text{ given} \]

For \( k = 0, 1, 2, \ldots \):

Solve \[ Ly(k+1) = q(k) \]

Solve \[ Uz(k+1) = y(k+1) \]

\[ q(k+1) = \frac{z(k+1)}{\|z(k+1)\|} \]

- This converges to the dominant eigenvalue \( \eta \) of \((A - \lambda I)^{-1}\).

- The corresponding eigenvalue of \( A \) is \( \lambda \):

\[ \eta = \frac{1}{\lambda - \mu} \]

\[ \lambda = \frac{1}{\eta} + \mu \]
So in principle we could find all eigenvalues (even complex ones)

- Interesting complication

- we want $\mu$ close to $2$, since the closer it is the larger is the dominant eigenvalue $\lambda$

- But if $\mu$ is an eigenvalue then $A - \mu I$ is singular

- so the closer $\mu$ is to 2 the larger is the condition number of $A - \mu I$

- we have the rare case that we really do want to solve an ill-conditioned system

- It turns out that is OK.

The power method is temperamental.

Interestingly, LAPACK (which underlies MATLAB) does not provide an option of computing a single eigenvalue/vector pair.

Nonetheless, PM is the starting point for the QR algorithm.

How can we modify the PM to find several eigenvalue/vector pairs?

**Idea 1:**

\[ y^{(0)} \text{ given, } \quad y^{(0)} \text{ n x r matrix} \]

\[ z^{(k+1)} = A \cdot y^{(k)} \]

\[ y^{(k+1)} = \frac{z^{(k+1)}}{\|z^{(k+1)}\|} \]

This is no good! The columns of \( y^{(k)} \) do not interact and we are running the PM in each column separately.

We have to make sure that the columns remain linearly independent.
- how independent?
- how about making them orthonormal?

- **Orthogonal Iteration.**

  \[ Q_0 = Q \quad n \times r \quad Q^TQ = I \]

  - For \( k = 1, 2, \ldots \) e.g. \( Q_0 = I \)

  \[ Z_k = A Q_{k-1} \]

  \[ Q_k R_k = Z_k \quad Q_k^TQ_k = I \quad R_k \text{ upper } \Delta \]

  - if \( r = 1 \) this is just the PM

  - moreover, as far as the first column of \( Q_k \) is concerned this is just orthogonal iteration with \( r = 2 \)

  - similarly for the first \( s = 3, 4, \ldots, r \) columns

  - we are running \( r \) orthogonal iterations simultaneously.
OK, how about \( r = n \)? (find all eigenpairs)

Define

\[ T_k = Q_k^T A Q_k \]

\( T_k \) is similar to \( A \), since \( Q_k^{-1} = Q_k^T \)

It turns out that the \( T_k \) satisfy the recursion (with \( Q_0 = I \))

\[ T_0 = A \]

For \( k = 1, 2, 3, \ldots \)

Factor \( T_{k-1} = \hat{Q}_{k-1} R_{k-1} \)

Set

\[ T_k = R_{k-1} \hat{Q}_{k-1} \]

more succinctly (replacing "T" with "A" and overwriting \( A \))

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

\[ A = QR \quad \text{(compute)} \]

\[ A = RQ \quad \text{(overwrite)} \]

That's why it's the CQR algorithm.
Let's see why this works

\[ T_k = Q_k^T A Q_k \quad A Q_{k-1} = Z_k = Q_k R_k \]

\[ T_{k-1} = Q_{k-1}^T A Q_{k-1} = Q_{k-1}^T Z_k = Q_{k-1} Q_k R_k \]

\[ T_k = Q_k^T A Q_k = Q_k^T A Q_{k-1} R_{k-1} Q_k^T Q_k \]

= \[ Q_k^T Z_k Q_{k-1} Q_k^T Q_k \]

= \[ Q_k^T \overset{\sim}{Q}_k Q_{k-1} Q_k^T Q_k \]

= \[ I R_k Q_{k-1} Q_k \]

= \[ R_k Q_{k-1} Q_k \]

= \[ Q_k \]

---

The process converges if the eigenvalues are real and distinct.

- converges slowly
- we need to refine it.
- make it faster
- handle complex eigenvalues