CS475 / CM 375
Lecture 18: Nov 10, 2011

QR Method with Shifts

Google Page Rank

Reading: [TB] Chapter 29

Reduction to Hessenberg Algorithm

For $k = 1,2,\dots,n-2$

$x = A(k+1:n,k)$

$v_k = sign(x_1)||x||e_1 + x$

$v_k = v_k/||v_k||$

$Q_k^T \times$ {

for $j = k, k+1, \dots, n$

$A(k+1:n,j) = A(k+1:n,j) - 2v_k\left(v_k^T A(k+1:n,j)\right)$

end }

$\times Q_k$ {

for $i = 1,2,\dots,n$

$A(i,k+1:n) = A(i,k+1:n) - 2(A(i,k+1:n)v_k)v_k^T$

end }

end

Symmetric Case

- If $A = A^T$, then $(Q^T AQ)^T = Q^T AQ$ is also symmetric
- A symmetric Hessenberg matrix $\longrightarrow$ tridiagonal matrix
- Two-phase process:

Shift QR Algorithm

- QR algorithm is both simultaneous iteration and simultaneous inverse iteration
  - Can apply shift technique
- Algorithm (Shifted QR)

$A^{(0)} = A$

For $k = 1,2, \dots$

  Pick a shift $\mu^{(k)}$

  $Q^{(k)} R^{(k)} \leftarrow A^{(k-1)} - \mu^{(k)} I$ (QR factorization)

  $A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I$

End

Shift QR Algorithm

- Similar to regular QR, we can show that
  $A^{(k)} = \left(\underline{Q}^{(k)}\right)^T A(\underline{Q}^{(k)})$ where $\underline{Q}^{(k)} = Q^{(1)} \dots Q^{(k)}$
- Derivation:

Shift QR Algorithm

- We can also show that
  $\left(A - \mu^{(k)} I\right)\left(A - \mu^{(k-1)} I\right) \dots \left(A - \mu^{(1)} I\right) = \underline{Q}^{(k)} \underline{R}^{(k)}$
- Derivation:

Shift QR Algorithm

- Continued derivation:

- If the shifts are good eigenvalue estimates, the last column of $\underline{Q}^{(k)}$ converges quickly to an eigenvector.

Rayleigh quotient shift

- To estimate the eigenvalue corresponding to the eigenvector approximated by the last column of $\underline{Q}^{(k)}$:

$$\mu^{(k)} = \left(\underline{q}_n^{(k)}\right)^T A \left(\underline{q}_n^{(k)}\right)$$

- Equivalent to applying RQI on e_n
  - i.e., QR algo has cubic convergence to that eigenvector
- Note: $A^{(k)} = \left(\underline{Q}^{(k)}\right)^T A \underline{Q}^{(k)}$

$$A_{nn}^{(k)} = \left(\underline{q}_n^{(k)}\right)^T A \left(\underline{q}_n^{(k)}\right) = \mu^{(k)}$$

$\therefore \mu^{(k)}$ comes for free!

Google PageRank

- Problem: give a ranking, PageRank, to all webpages.

- Idea: surfing the web is like a random walk
  → a Markov chain or Markov process.
  - PageRank = the limiting probability that an infinitely dedicated random surfer visits any particular page.
  - A page has high rank if other pages with high rank link to it.

Google PageRank

- Example:

Google PageRank

- Define connectivity matrix G by

$$g_{ij} = \begin{cases} 1 & \text{if } \exists \text{ a link from page } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

$G =$

- The j^{th} column of G shows the links on the j^{th} page.

Google PageRank

- Let $p =$ prob. that the random walk follows a link and $1 - p =$ prob. that an arbitrary page is chosen
  - Typically $p = 0.85$

- Define $a_{ij} = p \frac{g_{ij}}{\sum_i g_{ij}} + (1 - p)\frac{1}{n}$
  to be the prob. of jumping from page j to page i

Google PageRank

- Properties of A:
  - Entries between 0 and 1: $0 < a_{ij} < 1$
  - Columns sum to 1:
    $\sum_i a_{ij} = p \frac{\sum_i g_{ij}}{\sum_i g_{ij}} + (1-p)\frac{1}{n}\sum_i 1 = p + (1-p) = 1$
- By Ferron-Frobenius theorem, a matrix A with the above properties admits a vector x such that $Ax = x$
  i.e., x is the eigenvector corresponding to eigenvalue 1

Google PageRank

- Normalize x such that $\sum_i x_i = 1$. Then x is the state vector of the Markov chain & is Google's PageRank!
- The elements of x are all positive and less than 1.
- In our example, $x =$

Google PageRank

- To compute PageRank:
  - Setup A
  - Compute largest eigenvector by: