The Lanczos Algorithm for Large-Scale Eigenvalue Problems

Sam Handler

08/01/07
The Eigenvalue Problem

- Given a square matrix $A$, find values $\lambda_i$ and vectors $v_i$ such that
  \[ Av_i = \lambda_i v_i \]
  - Values $\lambda_i$ are called *eigenvalues*
  - Vectors $v_i$ are called *eigenvectors*
Quantum Mechanics

- The Time-Independent Schrödinger equation

\[ H\psi = E\psi \]

where \( H \) is the Hamiltonian operator, \( \psi \) is a wave function, and \( E \) is the energy in the system.

- This can be simplified so that \( H \) can be represented as a matrix and \( \psi \) as a vector.

- Eigenvalues correspond to energy levels of the system; the eigenvectors represent the corresponding wave-functions.
Graph Theory

- Given a graph $G$, the *centrality* of a vertex is a measure of how “important” a vertex is to a graph.

- The importance of a node is proportional to the sum of the importance of the nodes adjacent to it.

- Centrality is found by determining the eigenvector associated with the largest eigenvalue.

- This measure forms the basis for Google’s PageRank™ algorithm.
The Lanczos Algorithm

- Reduces a large, complicated eigenvalue problem into a smaller, simpler one
- Approximates the eigenvalues of a matrix
- Finds the smallest and largest eigenvalues fastest
A is the matrix, $q_1$ is a random vector with $|q_1| = 1$

$q_0 = 0, \beta_1 = 0$

for $i = 1$ to $m$:

$u = Aq_i - \beta_i q_{i-1}$

$\alpha_i = u \cdot q_i$

$u = u - \alpha_i q_i$

$\beta_{i+1} = |u|$

$q_{i+1} = u / \beta_{i+1}$

Then find the eigenvalues of

$$T = \begin{pmatrix}
\alpha_1 & \beta_2 & & \\
\beta_2 & \alpha_2 & \beta_3 & \\
& \ddots & \ddots & \\
& & \ddots & \alpha_{m-1} & \beta_m \\
& & & \beta_m & \alpha_m
\end{pmatrix}$$
The Orthogonality Problem

- Each $q_i$ should be orthogonal to all other $q$ vectors.

- Due to limited precision when storing vectors, new $q$ vectors slowly become less orthogonal.
Reorthogonalization

- Periodically reorthogonalize the current $q$ vector against all previous $q$ vectors.
- Takes a lot of time - is only done when necessary
- Use simple recurrence relations to estimate level of nonorthogonality - reorthogonalize when this level gets too large.
- In practice, reorthogonalize about every 10-15 iterations.
Implementation Notes

- Test the eigenvalues every 10 loops (adjustable).
- Test for convergence by testing the sum of the smallest (or largest) eigenvalues.
- The eigenvectors of $A$ can be calculated as

$$v_i = \begin{pmatrix} q_0 & \ldots & q_n \end{pmatrix} w_i$$

where $w_i$ is an eigenvector of $T$. 
Performance

- Performance is largely determined by disk speed.

  Example: On a matrix of size $n = 108,384$, performing 1000 iterations took 4437 seconds, but only 269 seconds (6%) were spent performing computations; the rest were spent waiting for the disk.

- The time spent waiting for disk should decrease with larger vectors.
  - For a given operation, computing time increases linearly with vector size, while load time is nearly constant.
Future Directions

Goal: Further increase speed and scale of calculations

- Better handling of vector storage
  - Keep vectors in memory longer

- Store fewer vectors
  - Faster to regenerate vectors than load them

- Parallelize for multiple-processor machines
Questions?