Chandler Justice - A02313187

November 2, 2023
Suppose we want to define the smallest Eigenvalue:
Input: $A, v_0, \text{tol}, \text{matIter}$

while(error $<$ tol && iter $<$ matIter)
{ v1 = 1 / magnitude(y) * y
w = A * v1
$\lambda_1 = v_1^t * w$

error = abs($\lambda_0 - \lambda_1$)
iter++;
$\lambda_0 = \lambda_1$

$y = v_1$

}
return λ_1, error;
Finding the smallest Eigenvalue: If λ_1 is the largest eigenvector of A, then $\frac{1}{\lambda_1}$ is the smallest eigenvalue of A^{-1}. If λ_n is the smallest eigenvalue of A. the $\frac{1}{n}$ is the largest eigenvalue of A^{-1}

We have the means to compute approximations of λ_1 and λ_n

- λ_1: Use the power method
- λ_n: Use the inverse power method

properties of eigenvalues:

if (λ, v) an eigen pair, $v \neq 0$. this means $Av = \lambda v$. Then for $\mu \in \mathbb{R}$, $(Av - \mu Iv) = (A - \mu I)v = \lambda v - \mu Iv = (\lambda - \mu)v$
$\rightarrow (\lambda - \mu)$ is an eigenvalue of $(A - \mu I)$.
So, $(A - \mu I)v = (\lambda - \mu)v$

The idea is choose μ close our eigenvalue. We can use inverse iteration to derive an approximation of $\lambda - \mu$ and then add μ back to get λ.

Iteratively finding eigenvalues:

$v_0 = a_1 v_1 + a_2 v_2 + ... + a_n v_n$

$Av_0 = a_1(\lambda_1 v_1) + a_2(\lambda_2 v_2) + ... + a_n(\lambda_n v_n)$

where $\lambda_1 / > \lambda_2 \geq ... \geq \lambda_{n-1} / > 0$

Properties

1. $\frac{Av_k}{\|v_k\|} \rightarrow v_1$

2. $\frac{v_k^t A v_k}{v_k^t v_k}$

3. If λ is an eigenvalue of A, the $\frac{1}{\lambda}$ is an eigenvalue of A^{-1}

4. $A\vec{v} = \lambda\vec{v} \rightarrow A\vec{v} - \mu v = (\lambda - \mu)v \rightarrow (A - \mu I)v$.
If λ is an eigenvalue of A, then $\lambda - \mu$ is an eigenvalue of $A \mu I$

Using this approach allows us to utilize parallelism. This means we can use a tool like *openMP* to easily parallelize code written in C, C++, fortran, etc

Lamczos Algorithm

$Ax \rightarrow b$

$$\begin{bmatrix} \sum_{j=i}^{n} a_{i,j} x_j \\ \sum_{j=i}^{n} a_{i,j} x_j \\ ... \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{bmatrix} =$$

$$x_1 \begin{bmatrix} a_{1,1} \\ a_{1,2} \\ ... \\ a_{1,n} \end{bmatrix} + x_2 \begin{bmatrix} a_{2,1} \\ a_{2,2} \\ ... \\ a_{2,n} \end{bmatrix} + ... + x_n \begin{bmatrix} a_{n,1} \\ a_{n,2} \\ ...a_{n,n} \end{bmatrix}$$

```
for(int i = 0; i ¡ n; i++){
double sum = 0.0;
for(int j = 0; j ¡ n; j++){
sum += a[i][j] * x[j];
}
b[i] = sum;
}
```
