The Lanczos Algorithm

Shane Harding

February 17, 2015

Outline of talk

Introduction

Brief History

The Lanczos algorithm was developed by Cornelius Lanczos. He developed the method while working at Boeing in the 40s. It was then forgotten for a number of years due to the lack of computers meaning that it couldn't be properly utilised. When it was then "rediscovered" it was modified multiple times to fix some instability issues.

The Lanczos Algorithm is used to solve large scale eigenvalue problems. So give a large $n \times n$ matrix A we have:

Eigenvalues and eigenvectors

$$Av_i = \lambda_i v_i$$

Where the vectors v_i are the eigenvectors and the scalars λ_i are the eigenvalues.

Power Iteration

- The Lanczos algorithm is an adaptation of the power method.
- The power methods (or power iteration or Von Mises iteration) is an eigenvalue algorithm.
- This algorithm will only produce one eigenvalue and may converge slowly.
- It finds the eigenvalue with the greatest absolute value.

Power Iteration

The power method can be summarised as follows: If x_0 is some random vector and $x_{n+1} = Ax_n$ then if we consider the limit of n being large we find that $\frac{x_n}{||x_n||}$ approaches the normed eigenvector with the greatest eigen value.

- Then if $A = Udiag(\sigma_i)U'$ is the eigendeecomposition of A then $A^n = Udiag(\sigma_i^n)U'$.
- As n gets big this will be dominated by the largest eigenvalue

The Lanczos Algorithm

We wish to solve the system:

$$Ax = b$$

where $A \in \mathbb{R}^{N \times N}$ and $b \in \mathbb{R}^N$. Let $x_0 \in \mathbb{R}^N$ be our initial guess at the solution, and $r_0 = b - Ax_0$ be the residual of our guess.
We will calculate the tridiagonal, symmetric matrix $T_{mm} = V_m^* A V_m$
Let $v_1 = \frac{r_0}{\beta_1}$ where $\beta_1 = ||r_0||$

```
1. set r_0 = b - Ax_0; beta_1 = abs(r_0); v_1 = r_0/beta_1;
2. for i = 1,...,m do
     r_i = A v_i - beta_i v_{i-1}
     alpha_i = (r_i,v_i)
     r_i = r_i - alpha_i v_i
     beta_{i+1} = abs(r_i)
     if(beta_{i+1} < tau)
        m = i
        break
     else
        v_{i+1} = r_i/beta_{i+1}
```

Thank you!