A hybrid reordered Arnoldi method to accelerate PageRank computations

Danielle Parker

Final Presentation

Modeling the Web

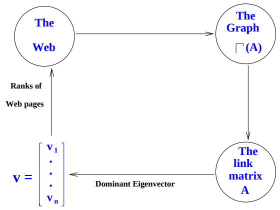


Figure: Model of Page Ranking

Modeling the Web

Definition

Given a matrix A_{nxn}, the graph $\Gamma(A)$ is a directed graph with n nodes P_1, P_2,...,P_n such that

$$P_i \text{ has a link to } P_j \Longleftrightarrow a_{ij} \neq 0$$

Conversely, given a directed graph $\Gamma(A)$, the associated matrix A will be called the link matrix or adjacency matrix.

Modeling the Web

Definition

Given a matrix A_{nxn}, the graph $\Gamma(A)$ is a directed graph with n nodes P_1, P_2,...,P_n such that

$$P_i \text{ has a link to } P_j \iff a_{ij} \neq 0$$

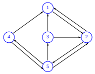
Conversely, given a directed graph $\Gamma(A)$, the associated matrix A will be called the link matrix or adjacency matrix.

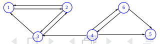
Definition

A directed graph is called **strongly connected** if there exists a path from each node to every other node.

Strongly Connected Vs. Weakly Connected

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 1/3 & 1/3 & 1/3 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \end{bmatrix}$$





Modeling the Web

Theorem

A matrix A_{nxn} is **irreducible** $\iff$ *$\Gamma(A)$ is strongly connected.*

Modeling the Web

Definition

A matrix A_{nxn} is called **(row) stochastic** if it is nonnegative and the sum of the entries in each row is 1.

Modeling the Web

Definition

A matrix A_{nxn} is called **(row) stochastic** if it is nonnegative and the sum of the entries in each row is 1.

Definition

A matrix A_{nxn} is called **reducible** if there is a permutation matrix P_{nxn} and an integer 1 $\leq$ r $\leq$ n-1 such that

$$P^T AP = \begin{bmatrix} C & D \\ 0 & E \end{bmatrix}$$

where 0 is an (n-r)$\times$r block matrix. If a matrix A is not reducible, then it is called **irreducible**.

Modeling the Web

Definition

A nonnegative matrix A_{nxn} is called **primitive** if it is irreducible and has only one eigenvalue of maximum magnitude.

Modeling the Web

Definition

A nonnegative matrix A_{nxn} is called **primitive** if it is irreducible and has only one eigenvalue of maximum magnitude.

- In modeling the web, we use a stochastic, primitive matrix to ensure that there is a unique dominant eigenvalue λ_1 of maximum magnitude 1, which is associated with the dominant eigenvector.

  This dominant eigenvector is called the **PageRank vector**.

Making the web matrix stochastic

- One problem is that our web matrix H is not stochastic because of dangling nodes.

Making the web matrix stochastic

- One problem is that our web matrix H is not stochastic because of dangling nodes.

Definition

Dangling nodes are nodes which have no outlinks, like a PDF or JPEG file.

Making the web matrix stochastic

- One problem is that our web matrix H is not stochastic because of dangling nodes.

Definition

Dangling nodes are nodes which have no outlinks, like a PDF or JPEG file.

- A simple way to address this problem is by replacing the zero row, which occurs because the page has no outlinks, with a probabilistic vector.

Making the web matrix stochastic

From this, we get the definition

$$B = H + au^T.$$

$$\text{where } a_i = \begin{cases} 1 & \text{if page } i \text{ is a dangling node} \\ 0 & \text{otherwise} \end{cases}$$

and u^T is any probabilistic vector.

Making the web matrix irreducible

- A second problem is that the matrix B we just created may be a reducible matrix.

Making the web matrix irreducible

- A second problem is that the matrix B we just created may be a reducible matrix.
- To remedy this, we force every page to be reachable from every other page.

Making the web matrix irreducible

- A second problem is that the matrix B we just created may be a reducible matrix.
- To remedy this, we force every page to be reachable from every other page.
- In reality, this is true, because at any time a user could jump from one page to another by using the URL of the page.

Making the web matrix irreducible

- The chosen method to fix this problem is by adding a perturbation matrix to B, letting

$$G = \alpha B + (1-\alpha)E$$

where $0 < \alpha < 1$ and

$$E = ee^T/n \quad \text{or} \quad E = eu^T$$

where e is a vector of ones.

Making the web matrix irreducible

- The chosen method to fix this problem is by adding a perturbation matrix to B, letting

  $$G = \alpha B + (1 - \alpha)E$$

  where $0 < \alpha < 1$ and

  $$E = ee^T/n \quad \text{or} \quad E = eu^T$$

  where e is a vector of ones.

- We call this stochastic, irreducible matrix G the "Google matrix" and u^T the personalization vector.

Past Methods

- The Power Method

Past Methods

- The Power Method
- The Linear Approach

Past Methods

- The Power Method
- The Linear Approach
- IAD

Past Methods

- The Power Method
- The Linear Approach
- IAD
- Combining reordering with Power, Linear, IAD

Past Methods

- The Power Method
- The Linear Approach
- IAD
- Combining reordering with Power, Linear, IAD

- Refined Arnoldi Algorithm

Past Methods

- The Power Method
- The Linear Approach
- IAD
- Combining reordering with Power, Linear, IAD

- Refined Arnoldi Algorithm
- Arnoldi Extrapolation Algorithm

Past Methods

- The Power Method
- The Linear Approach
- IAD
- Combining reordering with Power, Linear, IAD

- Refined Arnoldi Algorithm
- Arnoldi Extrapolation Algorithm
- Adaptive Arnoldi Algorithm

The Power Method

- The power method was the original approach used by Google's founders Sergey Brin and Larry Page in [1].

The Power Method

- The power method was the original approach used by Google's founders Sergey Brin and Larry Page in [1].

Ranking Pages

Given a number of web pages n the **rank** $r_k(P_j)$ of a given page P_j at step k can be determined iteratively by the formula

$$r_k\left(P_j\right) = \sum_{P_i \in B_{P_j}} \frac{r_{k-1}(P_i)}{|P_i|},$$

where the number of outlinks of a page P is given by $|P|$, B_{P_j} is the set of all pages P with links to P_j and $i, j = 1, 2, \ldots, n, \ i \neq j$.

The Power Method

- If $v_k = \begin{bmatrix} r_k(P_1) & r_k(P_2) & \dots & r_k(P_n) \end{bmatrix}^T$, then the previous equation can be written as

The Power Method

- If $v_k = [r_k(P_1) \quad r_k(P_2) \quad \dots \quad r_k(P_n)]^T$, then the previous equation can be written as

Power Method

$$v_k^T = v_{k-1}^T H \quad \text{where } H_{ij} = \begin{cases} \frac{1}{|P_i|} & \text{if } P_j \text{ has a link from } P_i \\ 0 & \text{otherwise} \end{cases}$$

The Power Method

- The equation $v_k^T = v_{k-1}^T H$, where $k = 0, 1, 2, \dots$ and H is the web matrix, is simply the power method being used to compute the left dominant eigenvector of H.

The Power Method

- The equation $v_k^T = v_{k-1}^T H$, where $k = 0, 1, 2, \ldots$ and H is the web matrix, is simply the power method being used to compute the left dominant eigenvector of H.
- The eigenvector,
$$v = \lim_{k \to \infty} v_k,$$
is called the PageRank vector.

Implementation of the Power Method

- The power method is implemented by

$$v_{k+1}^T = v_k^T G = \alpha v_k^T H + [\alpha v_k^T a + (1-\alpha)] e^T / n$$

where $G = \alpha B + (1-\alpha)E$ and $B = H + au^T$

Implementation of the Power Method

- The power method is implemented by

$$v_{k+1}^T = v_k^T G = \alpha v_k^T H + [\alpha v_k^T a + (1-\alpha)]e^T/n$$

where $G = \alpha B + (1-\alpha)E$ and $B = H + au^T$

- Instead of implementing $v_k^T G$ we choose to use the equivalent equation $\alpha v_k^T H + [\alpha v_k^T a + (1-\alpha)]e^T/n$ which is much cheaper to implement than the original equation since it restores sparsity to the matrix.

Convergence of the Power Method

- To ensure convergence of power method, we need to require that the eigenvalues of H satisfy

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

Convergence of the Power Method

- To ensure convergence of power method, we need to require that the eigenvalues of H satisfy

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

- We call λ_1 the dominant eigenvalue and λ_2 (also denoted by α) the subdominant eigenvalue.

Convergence of the Power Method

- To ensure convergence of power method, we need to require that the eigenvalues of H satisfy

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

- We call λ_1 the dominant eigenvalue and λ_2 (also denoted by α) the subdominant eigenvalue.
- Note the explicit inequality between λ_1 and λ_2. This is sufficient because, while convergence will usually be achieved even if there is more than one dominant eigenvalue, it may be extremely slow, since the rate of convergence is $\frac{\lambda_2}{\lambda_1}$.

Convergence of the Power Method

- The ideal case for this problem is when λ_2 is far from λ_1. However, this would mean that G is not a realistic model of the web. The closer λ_2 is to 1, the more realistic the model is. For some methods, such as the power method, large values of λ_2 slow the convergence time considerably. For this reason, λ_2 is usually chosen to be 0.85.

Convergence of the Power Method

- The ideal case for this problem is when λ_2 is far from λ_1. However, this would mean that G is not a realistic model of the web. The closer λ_2 is to 1, the more realistic the model is. For some methods, such as the power method, large values of λ_2 slow the convergence time considerably. For this reason, λ_2 is usually chosen to be 0.85.

- The main drawback of the power method is the slow convergence it exhibits when λ_2 is close to λ_1 (when $\lambda_2 \longrightarrow 1$). Because of this, other methods have been considered.

Reordering

- Changing the order of the rows and columns of a matrix will not change the structure of the links of the matrix, just the numbering of the nodes.

Reordering

- Changing the order of the rows and columns of a matrix will not change the structure of the links of the matrix, just the numbering of the nodes.
- This led to the idea that reordering the matrix by decreasing row and column degree could drastically simplify computations.

Reordering

- Changing the order of the rows and columns of a matrix will not change the structure of the links of the matrix, just the numbering of the nodes.
- This led to the idea that reordering the matrix by decreasing row and column degree could drastically simplify computations.
- Reordering has been shown to increase the distance between λ_1 and λ_2. It has not yet been proven why.

Reordering

- Changing the order of the rows and columns of a matrix will not change the structure of the links of the matrix, just the numbering of the nodes.
- This led to the idea that reordering the matrix by decreasing row and column degree could drastically simplify computations.
- Reordering has been shown to increase the distance between λ_1 and λ_2. It has not yet been proven why.
- Reordering has been used in conjunction with other methods to improve their convergence time, such as the power method, the linear approach, and IAD. It has been shown to significantly improve the rate of convergence in the power method in [10].

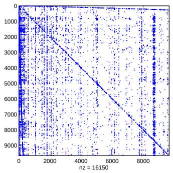
Reordering by Dangling Nodes

After reordering our web matrix H by dangling nodes, it can be rewritten as

$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}$$

where H_{11} is a square matrix that represents the links from nondangling nodes to nondangling nodes and H_{12} is a square matrix that represents the links from nondangling nodes to dangling nodes.

Reordering Example



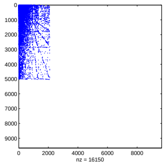


Figure: Matrix without reordering (left) and with reordering (right)

Krylov subspaces

Definition

Let A be a matrix of order n and let $u \neq 0$ be an n vector. Then the sequence

$$u, Au, A^2u, A^3u, \ldots$$

is a **Krylov sequence** based on A and u.

Krylov subspaces

Definition

Let A be a matrix of order n and let $u \neq 0$ be an n vector. Then the sequence

$$u, Au, A^2u, A^3u, \ldots$$

is a **Krylov sequence** based on A and u.

Definition

We call the matrix

$$K_k(A, u) = [\mathrm{u\ Au\ A^2u\ \ldots\ A^{k-1}u}\ ]$$

the kth **Krylov matrix**.

Krylov subspaces

Definition

The space

$$\mathcal{K}_k(A, u) = \mathcal{R}[K_k(A, u)]$$

is called the kth **Krylov subspace**.

Krylov subspaces

Definition

The space

$$\mathcal{K}_k(A, u) = \mathcal{R}[K_k(A, u)]$$

is called the kth **Krylov subspace**.

- More simply, the Krylov subspace is the column space of the Krylov matrix.

Krylov subspaces

- Krylov subspaces offer swift convergence of an approximation to the dominant eigenvector within the subspace.

Krylov subspaces

- Krylov subspaces offer swift convergence of an approximation to the dominant eigenvector within the subspace.
- One advantage of Krylov subspaces is that you keep all the information (the $A^k u$'s) that you have already generated. In the power method, this valuable information from the previously generated vectors is thrown away.

Arnoldi's Process

- Arnoldi's process is used to generate an orthonormal basis of $\mathcal{K}_k(A, u)$.

Arnoldi's Process

- Arnoldi's process is used to generate an orthonormal basis of $\mathcal{K}_k(A, u)$.
- Arnoldi's process can be written in matrix form by

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k$$

or

$$AQ_k = Q_{k+1} \tilde{H}_k$$

where $Q_m = (q_1, q_2, ..., q_m)$ and is orthogonal, $m = k, k+1$, e_k is the kth coordinate vector of dimension k, and $\tilde{H}_k$ is the $(k+1) \times k$ upper Hessenberg matrix identical to H_k except for an additional row of zeroes, where the only nonzero entry is $h_{k+1,k}$.

Arnoldi's Process

- The equation $AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k$ gives us the relationship
$$H_m = Q_m^T A Q_m$$

Arnoldi's Process

- The equation $AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k$ gives us the relationship
$$H_m = Q_m^T A Q_m$$
- This, together with $AQ_k = Q_{k+1} \tilde{H}_k$ gives us an approximate similarity transformation, which is what is used in the iterative Arnoldi algorithm since the approximate eigenvalues are much cheaper to calculate than the exact eigenvalues.

Ritz Values and Ritz Vectors

Definition

Let $x_i^{(k)}$, $y_i^{(k)}$, $i = 1, 2, ..., m$ denote the eigenpairs of H_k and λ_i, ϕ_i denote the eigenpairs of A. Then $x_i^{(k)}$ are the **Ritz values** of A in $K_k(A, u)$. These Ritz values approximate λ_i and the vectors

$$\phi_i^{(k)} = Q_k y_i^{(k)}$$

which approximate ϕ_i, are called the **Ritz vectors** of A in $K_k(A, u)$.

Ritz Values and Ritz Vectors

Definition

Let $x_i^{(k)}$, $y_i^{(k)}$, $i = 1, 2, ..., m$ denote the eigenpairs of H_k and λ_i, ϕ_i denote the eigenpairs of A. Then $x_i^{(k)}$ are the **Ritz values** of A in $K_k(A, u)$. These Ritz values approximate λ_i and the vectors

$$\phi_i^{(k)} = Q_k y_i^{(k)}$$

which approximate ϕ_i, are called the **Ritz vectors** of A in $K_k(A, u)$.

- The Ritz values of H will converge to the eigenvalues of A as $k \to \infty$; however, this does not necessarily imply that the Ritz vectors of H are converging to the eigenvectors of A.

Refined Ritz Values and Refined Ritz Vectors

- A newer idea is to use refined Ritz vectors, which have been shown to converge to the eigenvectors of A if the Ritz values are converging to the eigenvalues of A.

Refined Ritz Values and Refined Ritz Vectors

- A newer idea is to use refined Ritz vectors, which have been shown to converge to the eigenvectors of A if the Ritz values are converging to the eigenvalues of A.
- It has also been shown that algorithms using refined Ritz vectors converge more rapidly than those with normal Ritz vectors in [7].

Refined Ritz Values and Refined Ritz Vectors

- A newer idea is to use refined Ritz vectors, which have been shown to converge to the eigenvectors of A if the Ritz values are converging to the eigenvalues of A.
- It has also been shown that algorithms using refined Ritz vectors converge more rapidly than those with normal Ritz vectors in [7].
- By using refined Ritz vectors, we ensure that, not only are the eigenvalues of H converging to the eigenvalues of A, the eigenvectors of H are also converging to the eigenvectors of A.

Singular Value Decomposition

Definition

For any given matrix A_{mxn} there exists a decomposition

$$A = U\Sigma V^T$$

such that U is an m $\times$ m orthogonal matrix, Σ is an m $\times$ n diagonal matrix, and V is an n $\times$ n orthogonal matrix.

The diagonal values of Σ are called the singular values of A.

The column vectors of U are the left singular vectors of A.

The column vectors of V are the right singular vectors of A.

Solving for Refined Ritz Vectors

- Refined Ritz vectors can be computed by solving the singular value decomposition

$$\tilde{H} - \tilde{I} = U\Sigma V^T$$

where the eigenvalue is not needed in $\tilde{H}$ - $\tilde{I}$ since the eigenvalue for which we are computing the vector in the PageRank problem is equal to 1, which was noted by [6].

Solving for Refined Ritz Vectors

- Refined Ritz vectors can be computed by solving the singular value decomposition

$$\tilde{H} - \tilde{I} = U\Sigma V^T$$

where the eigenvalue is not needed in $\tilde{H}$ - $\tilde{I}$ since the eigenvalue for which we are computing the vector in the PageRank problem is equal to 1, which was noted by [6].

- In general, singular value decompositions are very computationally expensive to implement. However, the matrix $\tilde{H}$ is very small, usually seven or eight rows and columns or less.

Theorem

Theorem

Let the $z_i^{(k)}$ be the right singular vectors of $\tilde{H}_k$ - $x_i^{(k)}\tilde{I}$ associated with $\sigma_{min}(\tilde{H}_k$ - $x_i^{(k)}\tilde{I})$. Then the following relations hold:

$$\begin{aligned} u_i^{(k)} &= Q_k z_i^{(k)} \\ \|(A - x_i^{(k)} I) u_i^{(k)}\| &= \sigma_{min}(\tilde{H}_k - x_i^{(k)}\tilde{I}) \\ &= \sqrt{\|(\tilde{H}_k - x_i^{(k)}\tilde{I}) z_i^{(k)}\|^2 + h_{k+1,k}^2 |e_k z_i^{(k)}|} \end{aligned}$$

Theorem

The vector $u^{(k)}$ is in $\mathcal{K}_k(A, u)$, regardless of the value of $z_i^{(k)}$, since Q_k is an orthonormal basis for $\mathcal{K}_k(A, u)$.

Pseudocode

refinedArnoldi(A, q, k)

```
1: Repeat
2:         $[Q_{k+1}, H_{k+1,k}] = Arnoldi(A, q, k)$
3:         Compute $H_{k,k} v_M = \theta_M v_M$
4:         Compute $H_{k+1,k}$ - $\theta_M \tilde{I} = U \Sigma V^T$
5:         Set $v = V_{*k}$
6:         Set $q = Q_k v$
7: Until $\sigma_{min}(H_{k+1,k} - \theta_M \tilde{I}) < \epsilon$
```

where $\tilde{I}$ is the identity matrix augmented with a row of zeroes, V_{*k} is the kth column of V, $H_{k,k}$ is obtained by excluding the last row from $H_{k+1,k}$, and θ_M is the dominant eigenvalue of $H_{k,k}$ and v_M is the associated dominant eigenvector.

Pseudocode

However, from [6], since we know $\theta_M = 1$, we can simplify our algorithm:

refinedArnoldi(A, q, k)

```
1: Repeat
2:     $[Q_{k+1}, H_{k+1,k}] = Arnoldi(A, q, k)$
3:     Compute $H_{k+1,k}$ - $\tilde{I} = U\Sigma V^T$
4:     Set $v = V_{*k}$
5:     Set $q = Q_k v$
6: Until $\sigma_{min}(H_{k+1,k} - \tilde{I}) < \epsilon$
```

Arnoldi Extrapolation

- In [11], a new algorithm is proposed, using an extrapolation procedure to improve the starting vector q, making the algorithm converge faster.

Arnoldi Extrapolation

- In [11], a new algorithm is proposed, using an extrapolation procedure to improve the starting vector q, making the algorithm converge faster.
- This extrapolation procedure is also used in the iterative Arnoldi algorithm to improve q on each iteration.

Arnoldi Extrapolation

- In [11], a new algorithm is proposed, using an extrapolation procedure to improve the starting vector q, making the algorithm converge faster.
- This extrapolation procedure is also used in the iterative Arnoldi algorithm to improve q on each iteration.
- This method has been shown to converge faster than the power method and the refined Arnoldi method where $\alpha \rightarrow 1$.

Initial Assumptions

By making the assumption that the approximation generated by the Arnoldi-type algorithm, denoted $x^{(k-1)}$, can be expressed as a linear combination of the first three eigenvectors we get the following formulas:

$$x^{(k-1)} = x_1 + \alpha_2\lambda_2 x_2 + \alpha_3\lambda_3 x_3$$

$$x^{(k)} = Ax^{(k-1)} = x_1 + \alpha_2\lambda_2^2 x_2 + \alpha_3\lambda_3^2 x_3$$

$$x_1 = \frac{x^{(k+1)} - (\lambda_2 + \lambda_3)x^{(k)} + \lambda_2\lambda_3 x^{(k-1)}}{(1-\lambda_2)(1-\lambda_3)}$$

$$\hat{x}_1 = \frac{x^{(k+1)} - (\lambda_2 + \lambda_3)x^{(k)} + \lambda_2\lambda_3 x^{(k-1)}}{\left\|x^{(k+1)} - (\lambda_2 + \lambda_3)x^{(k)} + \lambda_2\lambda_3 x^{(k-1)}\right\|_1}$$

Approximations

- The final formula provides a good approximation to the PageRank vector.

Approximations

- The final formula provides a good approximation to the PageRank vector.
- Instead of using the actual values of λ_2 and λ_3, we will use the approximations generated by the Arnoldi-type method.

Approximations

- The final formula provides a good approximation to the PageRank vector.
- Instead of using the actual values of λ_2 and λ_3, we will use the approximations generated by the Arnoldi-type method.
- This also means that the smallest m that we can use in the Arnoldi process is 3, since we need the second and third approximate eigenvalues for the equation.

Cases to Consider

There are two cases that we must consider:

Case 1: $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ are real

In this case, we will use

$$\tilde{x}_1 = \frac{x^{(k+1)} - (\tilde{\lambda}_2 + \tilde{\lambda}_3)x^{(k)} + \tilde{\lambda}_2\tilde{\lambda}_3 x^{(k-1)}}{\left\| x^{(k+1)} - (\tilde{\lambda}_2 + \tilde{\lambda}_3)x^{(k)} + \tilde{\lambda}_2\tilde{\lambda}_3 x^{(k-1)} \right\|_1}$$

as our approximation.

Cases to Consider

Case 2: $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ are conjugate

In this case, we will use

$$\tilde{x}_1 = \frac{x^{(k+1)} - 2Re(\tilde{\lambda}_2)x^{(k)} + |\tilde{\lambda}_2|^2 x^{(k-1)}}{\left\| x^{(k+1)} - 2Re(\tilde{\lambda}_2)x^{(k)} + |\tilde{\lambda}_2|^2 x^{(k-1)} \right\|_1}$$

as our approximation, where $\mathrm{Re}(\tilde{\lambda}_2)$ denotes the real part of $\tilde{\lambda}_2$.

Adaptively Accelerated Arnoldi's

In [12], a new adaptively accelerated Arnoldi method is proposed. This method uses the weighted inner product to speed up convergence. It is proposed to change the weights based on the current residual.

Definition

Let G_{nxn} be a symmetric positive definite matrix and x and y be two vectors. Then a **G-inner product** is defined as

$$(x, y)_G = x^T Gy$$

This inner product is well defined if and only if the matrix G is symmetric positive definite.

Definition

Let G_{nxn} be a symmetric positive definite matrix and x and y be two vectors. Then a **G-inner product** is defined as

$$(x, y)_G = x^T Gy$$

This inner product is well defined if and only if the matrix G is symmetric positive definite.

Definition

Let G and x be defined as before. Then the norm associated with the G-inner product is defined by

$$\|x\|_G = \sqrt{(x, x)_G} = \sqrt{x^T Gx}$$

which is called the **G-norm**.

Adaptively Accelerated Algorithm

- In order to strengthen the weights of the components which converge slowly, we can define ω_i by

$$\omega_i = \frac{|\tilde{r}_i|}{\|\tilde{r}\|_1}$$

where $\tilde{r}$ is the residual computed by the last accelerated Arnoldi process and $\sum_{i=1}^{n} \omega_i = 1$.

Adaptively Accelerated Algorithm

- In order to strengthen the weights of the components which converge slowly, we can define ω_i by

$$\omega_i = \frac{|\tilde{r}_i|}{\|\tilde{r}\|_1}$$

  where $\tilde{r}$ is the residual computed by the last accelerated Arnoldi process and $\sum_{i=1}^{n} \omega_i = 1$.
- The idea is to give a larger weight to the r_i that are converging slowly, in order to accelerate the convergence of the algorithm.

Adaptively Accelerated Algorithm

- In order to strengthen the weights of the components which converge slowly, we can define ω_i by

$$\omega_i = \frac{|\tilde{r}_i|}{\|\tilde{r}\|_1}$$

  where $\tilde{r}$ is the residual computed by the last accelerated Arnoldi process and $\sum_{i=1}^{n} \omega_i = 1$.
- The idea is to give a larger weight to the r_i that are converging slowly, in order to accelerate the convergence of the algorithm.
- Thus, G is given by

$$G = diag\left\{\frac{|\tilde{r}_i|}{\|\tilde{r}\|_1}\right\}$$

  This adaptive changing of the residual is what leads the the faster convergence of the adaptively accelerated Arnoldi algorithm.

Subdominant Eigenvalue

- In [3], high stepping orderings are applied, after the matrix satisfies "Property R", to reduce the magnitude of λ_2. We found that the ordering we are using is more efficient than this reordering.

Subdominant Eigenvalue

- In [3], high stepping orderings are applied, after the matrix satisfies "Property R", to reduce the magnitude of λ_2. We found that the ordering we are using is more efficient than this reordering.
- In [8], some upper bounds on the magnitude of λ_2 are given for irreducible, stochastic matrices; however, this only works after the matrix has been partitioned with IAD, which doesn't apply to our matrices.

Subdominant Eigenvalue

- In [3], high stepping orderings are applied, after the matrix satisfies "Property R", to reduce the magnitude of λ_2. We found that the ordering we are using is more efficient than this reordering.
- In [8], some upper bounds on the magnitude of λ_2 are given for irreducible, stochastic matrices; however, this only works after the matrix has been partitioned with IAD, which doesn't apply to our matrices.
- In [5], they show that if the rows of a matrix are n-dimensional, random variables with $|cov(a_{ij}, b_{ik})| \leq \frac{c}{n^3}$, then $|\lambda_2| \to 0$ as $n \to \inf$. When we tried this, we found that the covariance was always 0, so this method would not work for us.

Subdominant Eigenvalue

- In [4], they reorder the matrix into NCD form with partitioned diagonal blocks formed by using the strongly connected components. Then, using ILU as a preconditioner, different Krylov subspace methods are used. We chose to use our reordering instead of this, since we believe it is more efficient.

Subdominant Eigenvalue

- In [4], they reorder the matrix into NCD form with partitioned diagonal blocks formed by using the strongly connected components. Then, using ILU as a preconditioner, different Krylov subspace methods are used. We chose to use our reordering instead of this, since we believe it is more efficient.
- In [2], bounds are given on the subdominant eigenvalue of stochastic matrices with nonnegative eigenvalues. This does not apply to us, since our eigenvalues can be negative or even complex.

Reordered Refined Arnoldi

- Since reordering significantly sped up the convergence time of the power method in [10], we decided to apply it to the Refined Arnoldi algorithm to see if the convergence time of it could be sped up as well.

Reordered Refined Arnoldi

- Since reordering significantly sped up the convergence time of the power method in [10], we decided to apply it to the Refined Arnoldi algorithm to see if the convergence time of it could be sped up as well.
- The reordering we used was by decreasing row and column degree. In our tests, this was shown to move λ_2 farthest away from λ_1, thus maximizing the convergence speed.

Reordered Refined Arnoldi

- Since reordering significantly sped up the convergence time of the power method in [10], we decided to apply it to the Refined Arnoldi algorithm to see if the convergence time of it could be sped up as well.
- The reordering we used was by decreasing row and column degree. In our tests, this was shown to move λ_2 farthest away from λ_1, thus maximizing the convergence speed.
- The results of this reordering vs. not reordering are shown in the following tables.

Reordering Hybrid

	California	Stanford	CNR	Stanford-Berkley
Without Reordering	.04092	5.26132	6.97566	9.40766
With Reordering	.02372	1.4956	2.01628	3.95372
Speedup	42.0%	71.6%	71.1%	56.0%

Table: Comparison of CPU times with $\alpha = .85$ and $m = 5$

	California	Stanford	CNR	Stanford-Berkley
Without Reordering	.04486	8.07578	10.43478	13.9722
With Reordering	.02678	1.48892	1.98436	3.93864
Speedup	40.3%	81.6%	81.0%	71.8%

Table: Comparison of CPU times with $\alpha = .90$ and $m = 5$

	California	Stanford	CNR	Stanford-Berkley
Without Reordering	.07072	15.87858	21.43492	26.11446
With Reordering	.03278	1.53726	1.99698	4.45082
Speedup	53.6%	90.3%	90.7%	83.0%

Table: Comparison of CPU times with $\alpha = .95$ and $m = 5$

	California	Stanford	CNR	Stanford-Berkley
Without Reordering	.17402	62.59826	90.77086	109.47492
With Reordering	.03468	1.74194	2.02384	4.56164
Speedup	80.1%	97.2%	97.8%	95.8%

Table: Comparison of CPU times with $\alpha = .99$ and $m = 5$

New Initial Guess

- We tried to improve the initial guess of the Arnoldi-based algorithms by using the column sum of the web matrix H.

New Initial Guess

- We tried to improve the initial guess of the Arnoldi-based algorithms by using the column sum of the web matrix H.
- The only matrix that this helped the majority of the time was the Stanford web matrix.

New Initial Guess

- We tried to improve the initial guess of the Arnoldi-based algorithms by using the column sum of the web matrix H.
- The only matrix that this helped the majority of the time was the Stanford web matrix.
- The results of this are shown in the following tables.

	$\alpha = 0.85$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.99$
Without New Initial Guess	5.26132	8.07578	15.87858	62.59826
With New Initial Guess	5.4839	7.70132	11.85016	54.8347
Speedup	-4.2%	4.6%	25.4%	12.4%

Table: Comparison of CPU times of Refined Arnoldi algorithm on Stanford matrix with $m = 5$

	$\alpha = 0.85$	$\alpha = 0.90$	$\alpha = 0.95$	$\alpha = 0.99$
Without New Initial Guess	6.69214	6.19414	8.8023	29.38416
With New Initial Guess	2.97144	4.39606	6.5456	103.97394
Speedup	55.6%	29.0%	25.6%	-253.8%

Table: Comparison of CPU times of Extrapolated Arnoldi algorithm on Stanford matrix with $m = 3$

Conclusions

- We observed that the reordering algorithm we use separates λ_2 from λ_1 the most, thus giving us the fastest convergence time.

Conclusions

- We observed that the reordering algorithm we use separates λ_2 from λ_1 the most, thus giving us the fastest convergence time.
- We also observed that combining reordering with the refined Arnoldi algorithm significantly improves the convergence time of the algorithm.

Conclusions

- We observed that the reordering algorithm we use separates λ_2 from λ_1 the most, thus giving us the fastest convergence time.
- We also observed that combining reordering with the refined Arnoldi algorithm significantly improves the convergence time of the algorithm.
- After studying the generalized Arnoldi, we observed that there is no significant improvement over the refined Arnoldi algorithm presented in [6].

Conclusions

- We observed that the reordering algorithm we use separates λ_2 from λ_1 the most, thus giving us the fastest convergence time.
- We also observed that combining reordering with the refined Arnoldi algorithm significantly improves the convergence time of the algorithm.
- After studying the generalized Arnoldi, we observed that there is no significant improvement over the refined Arnoldi algorithm presented in [6].
- Changing the initial guess vector to the sum of the columns of the web matrix did not significantly improve the convergence time, except on the Stanford matrix. In most other cases, it actually converged more slowly than when the initial guess was a random vector.

Open Questions

- Why does λ_2 separate from λ_1 when a matrix is reordered?

Open Questions

- Why does λ_2 separate from λ_1 when a matrix is reordered?
- Is it possible to associate our reordering with some explicit permutation matrices and/or similarity transformations?

Open Questions

- Why does λ_2 separate from λ_1 when a matrix is reordered?
- Is it possible to associate our reordering with some explicit permutation matrices and/or similarity transformations?
- Is it possible to construct an ordered Schur factorization $Q^T AQ = T$, where the entries in the block upper triangular matrix T are all nonnegative? If so, we can apply some results on the spectral radius of the nonnegative matrix B (whose dominant eigenvalue is λ_2).

Open Questions

- Why does λ_2 separate from λ_1 when a matrix is reordered?
- Is it possible to associate our reordering with some explicit permutation matrices and/or similarity transformations?
- Is it possible to construct an ordered Schur factorization $Q^T AQ = T$, where the entries in the block upper triangular matrix T are all nonnegative? If so, we can apply some results on the spectral radius of the nonnegative matrix B (whose dominant eigenvalue is λ_2).
- Is there any relationship between our reordering and preconditioning via ILU?

References I

- S. Brin, L.Page, R. Motwami, T. Winograd "The PageRank citation ranking: bringing order to the web". Technical Report 1999-0120, Standford University (1999).
- LJ. CvetKovic, V. Kostic. J.M. Pena "Eigenvalue Localization and Refinements for Matrices Related to Positivity". SIAM. (2011).
- T. Dayar "State Space Orderings for Gauss-Seidel in Markov Chains Revisited". SIAM. (1998).
- T. Dayar, W. Stewart "Comparison of Partitioning Techniques for Two-level Iterative Solvers on Large, Sparse Markov Chains". SIAM. (2000).

References II

- G. Goldberg and M. Neumann "Distribution of Subdominant Eigenvalues of Matrices with Random Rows". SIAM. (2003).
- G. H. Golub, C. Greif "An Arnoldi-Type Algorithm for Computing Page Rank". BIT Numerical Mathematics. (2006).
- Z. Jia "Refined Iterative Algorithms Based on Arnoldi's Process for Large Unsymmetric Eigenproblems". Linear Algebra and its Applications 259:1-23. Elsevier Science Inc. (1997).
- S. Kirkland "A cycle-based bound for subdominant eigenvalues of stochastic matrices". Linear and Multilinear Algebra. (2009).

References III

- S. Kirkland "Girth and Subdominant Eigenvalues for Stochastic Matrices". Electronic Journal of Linear Algebra. (2005).
- S. Osborne, J. Rebaza, E. Wiggins "On Accelerating the PageRank Computation". Internet Mathematics Vo. 6, No. 2: 157-171.(2009).
- G. Wu, Y. Wei. "An Arnoldi-Extrapolation algorithm for computing PageRank". Journal of Computational and Applied Mathematics 234:3196-3212. (2010).

References IV

J. Yin, G. Yin, M. Ng "On adaptively accelerated Arnoldi method for computing PageRank". Numerical Linear Algebra with Applications 19:73-85. Wiley Online Library (2012).

Arnoldi's Method

Arnoldi(A, q, k)

```
1:  $q_1 = q / \|q\|_2$
2:  for $j = 1$ to $k$
3:      $z = Aq_j$
4:      for $i = 1$ to $j$
5:          $h_{i,j} = q_i^T z$
6:          $z = z - h_{i,j} q_i$
7:      end for
8:      $h_{i,j} = \|z\|_2$
9:      if $h_{j+1,j} = 0$, quit
10:     $q_{j+1} = z / h_{j+1,j}$
11: end for
```

Restarted Arnoldi's Method

restartedArnoldi(A, q, k)

```
1: Repeat
2:     $[Q_{k+1}, H_{k+1,k}] = Arnoldi(A, q, k)$
3:     Compute $H_{k,k} v_M = \theta_M v_M$
4:     Set $q = Q_k v_M$
5: Until $\|Aq - q\|_2 < \epsilon$
```

where $H_{k,k}$ is obtained by excluding the last row from $H_{k+1,k}$, θ_M is the dominant eigenvalue, v_M is the associated dominant eigenvector, and ϵ is a predetermined tolerance.

Extrapolation Method

```
extrap(A, $x^{(k-1)}$, $\tilde{\lambda}_2$, $\tilde{\lambda}_3$, tol)
1: $x^{(k)} = Ax^{(k-1)}$
2: $r = \frac{\left\|x^{(k)} - x^{(k-1)}\right\|_1}{\left\|x^{(k-1)}\right\|_1}$
3: if tol $< r <$ 1$e$ - 2
4:         if $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$ are real
5:                 $x^{(k+1)} = Ax^{(k)}$
6:                 $x^{(k)} = x^{(k+1)} - (\tilde{\lambda}_2 + \tilde{\lambda}_3)x^{(k)} + \tilde{\lambda}_2\tilde{\lambda}_3 x^{(k-1)}$
7:         end if
```

Extrapolation Method, cont.

extrap(A, $x^{(k-1)}$, $\tilde{\lambda}_2$, $\tilde{\lambda}_3$, k, tol)

```
8:      if $\tilde{\lambda}_2$ and $\tilde{\lambda}_2$ are conjugate
9:          $x^{(k+1)} = Ax^{(k)}$
10:         $x^{(k)} = x^{(k+1)} - 2Re(\tilde{\lambda}_2)x^{(k)} + |\tilde{\lambda}_2|^2 x^{(k-1)}$
11:     end if
12:     $n = \left\|x^{(k)}\right\|_1$
13:     $x^{(k)} = \frac{x^{(k)}}{n}$
14:     $k = k + 1$
15: end if
```

Initial Conditions for Algorithm

Before we run the main code of the algorithm, we must set some initial conditions:

m = 3
k = 0
r = 1
r_0 = r
restart = 0
α = 0.85
β = α - 0.2
maxit = 6
tolnorm = 0.1
n = 1

Extrapolation Algorithm

extrapArn(A, v, m, β, maxit, restart, tolnorm)

```
1: while $restart < maxit$ and $r > tol$
2:     $x = Av$
3:     $r = \frac{\|x-v\|_1}{\|v\|_1}$
4:     $x = \frac{x}{\|x\|_1}$
5:     if $\frac{r}{r_0} > \beta$
6:         $restart = restart + 1$
7:     end if
8:     $v = x$
9:     $k = k + 1$
10:    $r_0 = r$
11: end while
```

Extrapolation Algorithm, cont.

extrapArn(A, v, m, β, maxit, restart, tolnorm)

```
12: while $r > tol$
13:     $[Q_{m+1}, \tilde{H}_m]$ = Arnoldi(A, x, m)
14:     Compute eigenpairs of $H$ and select $\tilde{\lambda}_2$ and $\tilde{\lambda}_3$
15:     Compute SVD: $\tilde{H}_m - \tilde{I} = U\Sigma V^T$
16:     $v = Q_m V(:, m)$
17:     $x = Q_{m+1}(\tilde{H}V(:, m))$
18:     $r = \frac{\|x-v\|_1}{\|v\|_1}$
19:     $x = \frac{x}{\|x\|_1}$
20:     $restart = 0$
21:     $r_0 = r$
22:     $n = 1;$
```

Extrapolation Algorithm, cont.

extrapArn(A, v, m, β, maxit, restart, tolnorm)

```
23:        while $r > tol$ and $n \geq tolnorm$
24:            [x, r, k, n] = extrap(A, x, $\lambda_2$, $\lambda_3$, k, tol)
25:            if $\frac{r}{r_0} > \beta$
26:                $restart = restart + 1$
27:            end if
28:            $r_0 = r$
29:        end while
30:    end while
```

The Generalized Arnoldi Method

GArnoldi(A, G, q_0, m)

```
1:  $\tilde{q}_1 = \frac{q_0}{\|q_0\|_G}$
2:  for $j$ = 1, 2,..., $m$
3:      $w = A\tilde{q}_j$
4:      for $k$ = 1, 2,..., $j$
5:          $\tilde{h}_{k,j} = (w, \tilde{q}_k)_G$
6:          $w = w - \tilde{h}_{k,j}\tilde{q}_k$
7:      end for
8:      $\tilde{h}_{j+1,j} = \|w\|_G$
9:      if $h_{j+1,j} = 0$
10:         stop and exit
10:     else
11:         $q_{j+1} = \frac{w}{\tilde{h}_{j+1,j}}$
12:     end if
13: end for
```

Adaptively Accelerated Algorithm

adaptiveArnoldi(A, G, q, m, tol

```
1: while true
2:     $[Q_{m+1}, H_{m+1,m}] =$ GArnoldi(A, G, q, m)
3:     $\tilde{H}_{m+1,m} - \tilde{I} = U\Sigma V^T$
4:     $q = Q_m v_m$
5:     $r = \sigma_m Q_{m+1} u_m$
6:     if $\|r\|_1 <$ tol
7:         stop and exit
8:     end if
9:     $G = \text{diag}\left\{\frac{|r_i|}{\|r\|_1}\right\}$
10: end while
```