MAT 167: Applied Linear Algebra
Lecture 24: Searching by Link Structure I

Naoki Saito

Department of Mathematics
University of California, Davis

November 21 & 24, 2025

Outline

Outline

Introduction

- The most dramatic change in search engine design in the past 15 years or so: incorporation of the Web's *hyperlink structure* (recall *outlinks* and *inlinks* of webpages briefly discussed in Example 4 in Lecture 2).
- Recall LSI (Latent Semantic Indexing), which uses the SVD of a matrix (e.g., a term-document matrix). One cannot use LSI for the entire Web: because it's based on SVD, the computation and storage for the entire Web is simply *not tractable*! (Currently, there are about $m \approx 4.49 \times 10^9$ webpages worldwide).
- The idea of using link structure of the Web is the following:
  1. $\exists$ certain webpages recognized as "go to" places for certain information (called *authorities*);
  2. $\exists$ certain webpages legitimizing those esteemed positions (i.e., authorities) by pointing to them with links (called *hubs*).
  3. It is a mutually reinforcing approach: good hubs $\Longleftrightarrow$ good authorities.

Introduction

- The most dramatic change in search engine design in the past 15 years or so: incorporation of the Web's *hyperlink structure* (recall *outlinks* and *inlinks* of webpages briefly discussed in Example 4 in Lecture 2).
- Recall LSI (Latent Semantic Indexing), which uses the SVD of a matrix (e.g., a term-document matrix). One cannot use LSI for the entire Web: because it's based on SVD, the computation and storage for the entire Web is simply *not tractable*! (Currently, there are about $m \approx 4.49 \times 10^9$ webpages worldwide).
- The idea of using link structure of the Web is the following:
  1. $\exists$ certain webpages recognized as "go to" places for certain information (called *authorities*);
  2. $\exists$ certain webpages legitimizing those esteemed positions (i.e., authorities) by pointing to them with links (called *hubs*).
  3. It is a mutually reinforcing approach: good hubs $\Longleftrightarrow$ good authorities

Introduction

- The most dramatic change in search engine design in the past 15 years or so: incorporation of the Web's *hyperlink structure* (recall *outlinks* and *inlinks* of webpages briefly discussed in Example 4 in Lecture 2).
- Recall LSI (Latent Semantic Indexing), which uses the SVD of a matrix (e.g., a term-document matrix). One cannot use LSI for the entire Web: because it's based on SVD, the computation and storage for the entire Web is simply *not tractable*! (Currently, there are about $m \approx 4.49 \times 10^9$ webpages worldwide).
- The idea of using link structure of the Web is the following:
  1. $\exists$ certain webpages recognized as "go to" places for certain information (called *authorities*);
  2. $\exists$ certain webpages legitimizing those esteemed positions (i.e., authorities) by pointing to them with links (called *hubs*).
  3. It is a mutually reinforcing approach: good hubs $\Longleftrightarrow$ good authorities

Introduction

- The most dramatic change in search engine design in the past 15 years or so: incorporation of the Web's *hyperlink structure* (recall *outlinks* and *inlinks* of webpages briefly discussed in Example 4 in Lecture 2).
- Recall LSI (Latent Semantic Indexing), which uses the SVD of a matrix (e.g., a term-document matrix). One cannot use LSI for the entire Web: because it's based on SVD, the computation and storage for the entire Web is simply *not tractable*! (Currently, there are about $m \approx 4.49 \times 10^9$ webpages worldwide).
- The idea of using link structure of the Web is the following:
  1. $\exists$ certain webpages recognized as "go to" places for certain information (called *authorities*);
  2. $\exists$ certain webpages legitimizing those esteemed positions (i.e., authorities) by pointing to them with links (called *hubs*).
  3. It is a mutually reinforcing approach: good hubs $\Longleftrightarrow$ good authorities

Introduction

- The most dramatic change in search engine design in the past 15 years or so: incorporation of the Web's *hyperlink structure* (recall *outlinks* and *inlinks* of webpages briefly discussed in Example 4 in Lecture 2).
- Recall LSI (Latent Semantic Indexing), which uses the SVD of a matrix (e.g., a term-document matrix). One cannot use LSI for the entire Web: because it's based on SVD, the computation and storage for the entire Web is simply *not tractable*! (Currently, there are about $m \approx 4.49 \times 10^9$ webpages worldwide).
- The idea of using link structure of the Web is the following:
  1. $\exists$ certain webpages recognized as "go to" places for certain information (called *authorities*);
  2. $\exists$ certain webpages legitimizing those esteemed positions (i.e., authorities) by pointing to them with links (called *hubs*).
  3. It is a mutually reinforcing approach: good hubs $\Longleftrightarrow$ good authorities

Introduction

- The most dramatic change in search engine design in the past 15 years or so: incorporation of the Web's *hyperlink structure* (recall *outlinks* and *inlinks* of webpages briefly discussed in Example 4 in Lecture 2).
- Recall LSI (Latent Semantic Indexing), which uses the SVD of a matrix (e.g., a term-document matrix). One cannot use LSI for the entire Web: because it's based on SVD, the computation and storage for the entire Web is simply *not tractable*! (Currently, there are about $m \approx 4.49 \times 10^9$ webpages worldwide).
- The idea of using link structure of the Web is the following:
  1. $\exists$ certain webpages recognized as "go to" places for certain information (called *authorities*);
  2. $\exists$ certain webpages legitimizing those esteemed positions (i.e., authorities) by pointing to them with links (called *hubs*).
  3. It is a mutually reinforcing approach: good hubs $\Longleftrightarrow$ good authorities

- In this lecture and the next, we will discuss two web search algorithms based on link structure (or hyperlinks):
  1. the *HITS* (Hyperlink Induced Topic Search) algorithm due to Jon Kleinberg (1998).
  2. the *PageRank* algorithm of Google (Sergey Brin & Larry Page, 1998)
- Today's lecture, we will focus on the HITS algorithm.

- In this lecture and the next, we will discuss two web search algorithms based on link structure (or hyperlinks):
  1. the *HITS* (Hyperlink Induced Topic Search) algorithm due to Jon Kleinberg (1998).
  2. the *PageRank* algorithm of Google (Sergey Brin & Larry Page, 1998)
- Today's lecture, we will focus on the HITS algorithm.

- In this lecture and the next, we will discuss two web search algorithms based on link structure (or hyperlinks):
  1. the *HITS* (Hyperlink Induced Topic Search) algorithm due to Jon Kleinberg (1998).
  2. the *PageRank* algorithm of Google (Sergey Brin & Larry Page, 1998)
- Today's lecture, we will focus on the HITS algorithm.

- In this lecture and the next, we will discuss two web search algorithms based on link structure (or hyperlinks):
  1. the *HITS* (Hyperlink Induced Topic Search) algorithm due to Jon Kleinberg (1998).
  2. the *PageRank* algorithm of Google (Sergey Brin & Larry Page, 1998)
- Today's lecture, we will focus on the HITS algorithm.

Outline

HITS Method

- Recall: good hubs $\Longleftrightarrow$ good authorities
- Suppose webpage i has an authority score a_i and hub score h_i where $i = 1 : n$.
- Let $\mathscr{E}$ denote the set of all directed edges in a graph of the Web whereby e_{ij} represents the directed edge from node (or webpage) i to node j (meaning that webpage i has a link pointing to webpage j).
- Assume that initial authority and hub scores of webpage i are $a_i^{(0)}$ and $h_i^{(0)}$.
- The HITS method iteratively updates those scores by the following summations:

$$a_i^{(k)} = \sum_j h_j^{(k-1)} \quad \text{where } e_{ji} \in \mathscr{E}; \tag{1}$$

$$h_i^{(k)} = \sum_j a_j^{(k)} \quad \text{where } e_{ij} \in \mathscr{E}, \tag{2}$$

for $k = 1, 2, \ldots$

HITS Method

- Recall: good hubs $\Longleftrightarrow$ good authorities
- Suppose webpage i has an authority score a_i and hub score h_i where $i = 1 : n$.
- Let $\mathscr{E}$ denote the set of all directed edges in a graph of the Web whereby e_{ij} represents the directed edge from node (or webpage) i to node j (meaning that webpage i has a link pointing to webpage j).
- Assume that initial authority and hub scores of webpage i are $a_i^{(0)}$ and $h_i^{(0)}$.
- The HITS method iteratively updates those scores by the following summations:

$$a_i^{(k)} = \sum_j h_j^{(k-1)} \quad \text{where } e_{ji} \in \mathscr{E}; \tag{1}$$

$$h_i^{(k)} = \sum_j a_j^{(k)} \quad \text{where } e_{ij} \in \mathscr{E}, \tag{2}$$

for $k = 1, 2, \ldots$

HITS Method

- Recall: good hubs $\Longleftrightarrow$ good authorities
- Suppose webpage i has an authority score a_i and hub score h_i where $i = 1 : n$.
- Let $\mathscr{E}$ denote the set of all directed edges in a graph of the Web whereby e_{ij} represents the directed edge from node (or webpage) i to node j (meaning that webpage i has a link pointing to webpage j).
- Assume that initial authority and hub scores of webpage i are $a_i^{(0)}$ and $h_i^{(0)}$.
- The HITS method iteratively updates those scores by the following summations:

$$a_i^{(k)} = \sum_j h_j^{(k-1)} \quad \text{where } e_{ji} \in \mathscr{E}; \tag{1}$$

$$h_i^{(k)} = \sum_j a_j^{(k)} \quad \text{where } e_{ij} \in \mathscr{E}, \tag{2}$$

for $k = 1, 2, \ldots$

HITS Method

- Recall: good hubs $\Longleftrightarrow$ good authorities
- Suppose webpage i has an authority score a_i and hub score h_i where $i = 1 : n$.
- Let $\mathscr{E}$ denote the set of all directed edges in a graph of the Web whereby e_{ij} represents the directed edge from node (or webpage) i to node j (meaning that webpage i has a link pointing to webpage j).
- Assume that initial authority and hub scores of webpage i are $a_i^{(0)}$ and $h_i^{(0)}$.
- The HITS method iteratively updates those scores by the following summations:

$$a_i^{(k)} = \sum_j h_j^{(k-1)} \quad \text{where } e_{ji} \in \mathscr{E}; \tag{1}$$

$$h_i^{(k)} = \sum_j a_j^{(k)} \quad \text{where } e_{ij} \in \mathscr{E}, \tag{2}$$

for $k = 1, 2, \ldots$

HITS Method

- Recall: good hubs $\Longleftrightarrow$ good authorities
- Suppose webpage i has an authority score a_i and hub score h_i where $i = 1 : n$.
- Let $\mathscr{E}$ denote the set of all directed edges in a graph of the Web whereby e_{ij} represents the directed edge from node (or webpage) i to node j (meaning that webpage i has a link pointing to webpage j).
- Assume that initial authority and hub scores of webpage i are $a_i^{(0)}$ and $h_i^{(0)}$.
- The HITS method iteratively updates those scores by the following summations:

$$a_i^{(k)} = \sum_j h_j^{(k-1)} \quad \text{where } e_{ji} \in \mathscr{E}; \tag{1}$$

$$h_i^{(k)} = \sum_j a_j^{(k)} \quad \text{where } e_{ij} \in \mathscr{E}, \tag{2}$$

for $k = 1, 2, \ldots$

- The above equations can be recast in matrix notation using the so-called *adjacency matrix* $L = (L_{ij})$ of the directed web graph where

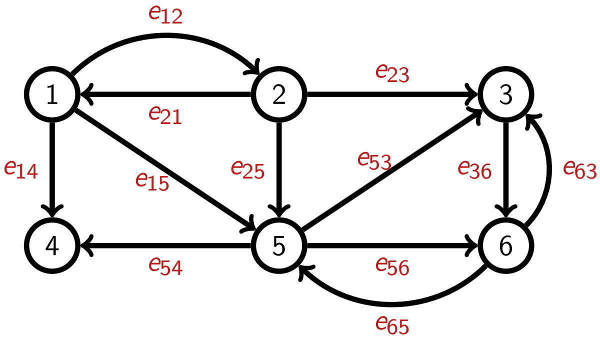
$$L_{ij} = \begin{cases} 1 & \text{if } \exists i, j \text{ s.t. } e_{ij} \in \mathscr{E} \ ; \\ 0 & \text{otherwise.} \end{cases}$$

- For example, consider the following directed web graph of Example 4 in Lecture 2:

- The above equations can be recast in matrix notation using the so-called *adjacency matrix* $L = (L_{ij})$ of the directed web graph where

$$L_{ij} = \begin{cases} 1 & \text{if } \exists i, j \text{ s.t. } e_{ij} \in \mathscr{E} \; ; \\ 0 & \text{otherwise.} \end{cases}$$

- For example, consider the following directed web graph of Example 4 in Lecture 2:



- The adjacency matrix L of this web graph is:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Now, Eqn's (1) and (2) can be rewritten as the matrix-vector multiplications:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} \boldsymbol{h}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L\boldsymbol{a}^{(k)}, \tag{3}$$

where $\boldsymbol{a}^{(k)}, \boldsymbol{h}^{(k)} \in \mathbb{R}^n$ represent the authority and hub scores of n webpages under consideration, respectively.

- Hence, we have:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} L\boldsymbol{a}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = LL^{\mathsf{T}} \boldsymbol{h}^{(k-1)}. \tag{4}$$

- Eqn's (4) are essentially the so-called *power iteration* for computing *the dominant eigenvector* of $L^{\mathsf{T}}L$ and LL^{T}.

- In above the HITS algorithm (3) (as well as (4)), we must *normalize* these vector after each iteration to have $\|\boldsymbol{a}^{(k)}\| = 1$ and $\|\boldsymbol{h}^{(k)}\| = 1$. The most convenient norm is 1-norm in this case.

- The adjacency matrix L of this web graph is:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Now, Eqn's (1) and (2) can be rewritten as the matrix-vector multiplications:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} \boldsymbol{h}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L\boldsymbol{a}^{(k)}, \tag{3}$$

where $\boldsymbol{a}^{(k)}, \boldsymbol{h}^{(k)} \in \mathbb{R}^n$ represent the authority and hub scores of n webpages under consideration, respectively.

- Hence, we have:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} L\boldsymbol{a}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = LL^{\mathsf{T}} \boldsymbol{h}^{(k-1)}. \tag{4}$$

- Eqn's (4) are essentially the so-called *power iteration* for computing *the dominant eigenvector* of $L^{\mathsf{T}}L$ and LL^{T}.

- In above the HITS algorithm (3) (as well as (4)), we must *normalize* these vector after each iteration to have $\|\boldsymbol{a}^{(k)}\| = 1$ and $\|\boldsymbol{h}^{(k)}\| = 1$. The most convenient norm is 1-norm in this case.

- The adjacency matrix L of this web graph is:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Now, Eqn's (1) and (2) can be rewritten as the matrix-vector multiplications:

$$\boldsymbol{a}^{(k)} = L^{\top}\boldsymbol{h}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L\boldsymbol{a}^{(k)}, \tag{3}$$

  where $\boldsymbol{a}^{(k)}, \boldsymbol{h}^{(k)} \in \mathbb{R}^n$ represent the authority and hub scores of n webpages under consideration, respectively.

- Hence, we have:

$$\boldsymbol{a}^{(k)} = L^{\top}L\boldsymbol{a}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = LL^{\top}\boldsymbol{h}^{(k-1)}. \tag{4}$$

- Eqn's (4) are essentially the so-called *power iteration* for computing *the dominant eigenvector* of $L^{\top}L$ and $LL^{\top}$.

- In above the HITS algorithm (3) (as well as (4)), we must *normalize* these vector after each iteration to have $\|\boldsymbol{a}^{(k)}\| = 1$ and $\|\boldsymbol{h}^{(k)}\| = 1$. The most convenient norm is 1-norm in this case.

- The adjacency matrix L of this web graph is:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Now, Eqn's (1) and (2) can be rewritten as the matrix-vector multiplications:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} \boldsymbol{h}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L \boldsymbol{a}^{(k)}, \tag{3}$$

  where $\boldsymbol{a}^{(k)}, \boldsymbol{h}^{(k)} \in \mathbb{R}^n$ represent the authority and hub scores of n webpages under consideration, respectively.

- Hence, we have:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} L \boldsymbol{a}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L L^{\mathsf{T}} \boldsymbol{h}^{(k-1)}. \tag{4}$$

- Eqn's (4) are essentially the so-called *power iteration* for computing *the dominant eigenvector* of $L^{\mathsf{T}} L$ and $L L^{\mathsf{T}}$.

- In above the HITS algorithm (3) (as well as (4)), we must *normalize* these vector after each iteration to have $\|\boldsymbol{a}^{(k)}\| = 1$ and $\|\boldsymbol{h}^{(k)}\| = 1$. The most convenient norm is 1-norm in this case.

- The adjacency matrix L of this web graph is:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Now, Eqn's (1) and (2) can be rewritten as the matrix-vector multiplications:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} \boldsymbol{h}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L\boldsymbol{a}^{(k)}, \tag{3}$$

  where $\boldsymbol{a}^{(k)}, \boldsymbol{h}^{(k)} \in \mathbb{R}^n$ represent the authority and hub scores of n webpages under consideration, respectively.

- Hence, we have:

$$\boldsymbol{a}^{(k)} = L^{\mathsf{T}} L \boldsymbol{a}^{(k-1)}, \quad \boldsymbol{h}^{(k)} = L L^{\mathsf{T}} \boldsymbol{h}^{(k-1)}. \tag{4}$$

- Eqn's (4) are essentially the so-called *power iteration* for computing *the dominant eigenvector* of $L^{\mathsf{T}} L$ and LL^{T}.
- In above the HITS algorithm (3) (as well as (4)), we must *normalize* these vector after each iteration to have $\|\boldsymbol{a}^{(k)}\| = 1$ and $\|\boldsymbol{h}^{(k)}\| = 1$. The most convenient norm is 1-norm in this case.

Power Iteration

- is also known as *power method*
- is an eigenvalue algorithm: given a matrix A, it will produce the largest eigenvalue $\lambda_{\max}$ of A, the corresponding eigenvector $\boldsymbol{v}$
- is a very simple algorithm, but it may converge slowly.
- does not compute a matrix decomposition (e.g., QR, SVD, ...).
- hence can be used when A is a very large *sparse* matrix.

Algorithm: Power Iteration

$\boldsymbol{v}^{(0)}$ = some vector with $\|\boldsymbol{v}^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$\quad \boldsymbol{w} = A\boldsymbol{v}^{(k-1)}$ apply A

$\quad \boldsymbol{v}^{(k)} = \boldsymbol{w}/\|\boldsymbol{w}\|$ normalize

$\quad \lambda^{(k)} = \left(\boldsymbol{v}^{(k)}\right)^{\mathsf{T}} A\boldsymbol{v}^{(k)}$ *Rayleigh* quotient

Power Iteration

- is also known as *power method*
- is an eigenvalue algorithm: given a matrix A, it will produce the largest eigenvalue $\lambda_{\max}$ of A, the corresponding eigenvector $\boldsymbol{v}$
- is a very simple algorithm, but it may converge slowly.
- does not compute a matrix decomposition (e.g., QR, SVD, ...).
- hence can be used when A is a very large *sparse* matrix.

Algorithm: Power Iteration

$\boldsymbol{v}^{(0)} = \text{some vector with } \|\boldsymbol{v}^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$\quad \boldsymbol{w} = A\boldsymbol{v}^{(k-1)}$ — apply A

$\quad \boldsymbol{v}^{(k)} = \boldsymbol{w} / \|\boldsymbol{w}\|$ — normalize

$\quad \lambda^{(k)} = \left(\boldsymbol{v}^{(k)}\right)^{\mathsf{T}} A\boldsymbol{v}^{(k)}$ — *Rayleigh* quotient

Power Iteration

- is also known as *power method*
- is an eigenvalue algorithm: given a matrix A, it will produce the largest eigenvalue $\lambda_{\max}$ of A, the corresponding eigenvector $\boldsymbol{v}$
- is a very simple algorithm, but it may converge slowly.
- does not compute a matrix decomposition (e.g., QR, SVD, ...).
- hence can be used when A is a very large *sparse* matrix.

Algorithm: Power Iteration

$\boldsymbol{v}^{(0)}$ = some vector with $\|\boldsymbol{v}^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$\quad \boldsymbol{w} = A\boldsymbol{v}^{(k-1)}$ apply A

$\quad \boldsymbol{v}^{(k)} = \boldsymbol{w}/\|\boldsymbol{w}\|$ normalize

$\quad \lambda^{(k)} = \left(\boldsymbol{v}^{(k)}\right)^{\mathsf{T}} A\boldsymbol{v}^{(k)}$ *Rayleigh* quotient

Power Iteration

- is also known as *power method*
- is an eigenvalue algorithm: given a matrix A, it will produce the largest eigenvalue $\lambda_{\max}$ of A, the corresponding eigenvector $\boldsymbol{v}$
- is a very simple algorithm, but it may converge slowly.
- does not compute a matrix decomposition (e.g., QR, SVD, ...).
- hence can be used when A is a very large *sparse* matrix.

Algorithm: Power Iteration

$\boldsymbol{v}^{(0)} =$ some vector with $\|\boldsymbol{v}^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$\quad \boldsymbol{w} = A\boldsymbol{v}^{(k-1)}$ apply A

$\quad \boldsymbol{v}^{(k)} = \boldsymbol{w}/\|\boldsymbol{w}\|$ normalize

$\quad \lambda^{(k)} = \left(\boldsymbol{v}^{(k)}\right)^{\mathsf{T}} A\boldsymbol{v}^{(k)}$ *Rayleigh* quotient

Power Iteration

- is also known as *power method*
- is an eigenvalue algorithm: given a matrix A, it will produce the largest eigenvalue $\lambda_{\max}$ of A, the corresponding eigenvector $\boldsymbol{v}$
- is a very simple algorithm, but it may converge slowly.
- does not compute a matrix decomposition (e.g., QR, SVD, ...).
- hence can be used when A is a very large *sparse* matrix.

Algorithm: Power Iteration

$\boldsymbol{v}^{(0)} =$ some vector with $\|\boldsymbol{v}^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$\quad \boldsymbol{w} = A\boldsymbol{v}^{(k-1)}$ apply A

$\quad \boldsymbol{v}^{(k)} = \boldsymbol{w}/\|\boldsymbol{w}\|$ normalize

$\quad \lambda^{(k)} = \left(\boldsymbol{v}^{(k)}\right)^{\mathsf{T}} A\boldsymbol{v}^{(k)}$ *Rayleigh* quotient

Power Iteration

- is also known as *power method*
- is an eigenvalue algorithm: given a matrix A, it will produce the largest eigenvalue $\lambda_{\max}$ of A, the corresponding eigenvector $\boldsymbol{v}$
- is a very simple algorithm, but it may converge slowly.
- does not compute a matrix decomposition (e.g., QR, SVD, ...).
- hence can be used when A is a very large *sparse* matrix.

Algorithm: Power Iteration

$\boldsymbol{v}^{(0)} =$ some vector with $\|\boldsymbol{v}^{(0)}\| = 1$

for $k = 1, 2, \ldots$

$\quad \boldsymbol{w} = A\boldsymbol{v}^{(k-1)}$ apply A

$\quad \boldsymbol{v}^{(k)} = \boldsymbol{w}/\|\boldsymbol{w}\|$ normalize

$\quad \lambda^{(k)} = \left(\boldsymbol{v}^{(k)}\right)^{\mathsf{T}} A\boldsymbol{v}^{(k)}$ *Rayleigh* quotient

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j/\lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j / \lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j / \lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j/\lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j/\lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j/\lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j / \lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Power Iteration: Why?

- Let $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ be the ONB vectors consisting of the eigenvectors of A
- Write $\boldsymbol{v}^{(0)}$ as a linear combination of $\{\boldsymbol{q}_j\}_{j=1:n}$ as

$$\boldsymbol{v}^{(0)} = \alpha_1 \boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_2 + \cdots + \alpha_n \boldsymbol{q}_n$$

- Since $\boldsymbol{v}^{(k)}$ is a constant (say, c_k) multiple of $A^k \boldsymbol{v}^{(0)}$, we have

$$\begin{aligned}
\boldsymbol{v}^{(k)} &= c_k A^k \boldsymbol{v}^{(0)} \\
&= c_k \left(\alpha_1 A^k \boldsymbol{q}_1 + \alpha_2 A^k \boldsymbol{q}_2 + \cdots + \alpha_n A^k \boldsymbol{q}_n \right) \\
&= c_k \left(\alpha_1 \lambda_1^k \boldsymbol{q}_1 + \alpha_2 \lambda_2^k \boldsymbol{q}_2 + \cdots + \alpha_n \lambda_n^k \boldsymbol{q}_n \right) \\
&= c_k \lambda_1^k \left(\alpha_1 \boldsymbol{q}_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \boldsymbol{q}_2 + \cdots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k \boldsymbol{q}_n \right)
\end{aligned}$$

- Note that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$; hence, for $j = 2:n$, $(\lambda_j/\lambda_1)^k \to 0$ as $k \to \infty$.
- Since c_k is chosen such that $\|\boldsymbol{v}^{(k)}\| = 1$, we have $\boldsymbol{v}^{(k)} \to \boldsymbol{q}_1$ and $\lambda^{(k)} \to \boldsymbol{q}_1^{\mathsf{T}} A \boldsymbol{q}_1 = \lambda_1$ as $k \to \infty$.

Outline

Example 4 of Lecture 2

- Recall the adjacency matrix L:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Hence, we have:

$$L^{\mathsf{T}}L = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 3 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 & 3 & 0 \\ 0 & 0 & 1 & 1 & 0 & 2 \end{bmatrix} \quad LL^{\mathsf{T}} = \begin{bmatrix} 3 & 1 & 0 & 0 & 1 & 1 \\ 1 & 3 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 3 & 1 \\ 1 & 2 & 0 & 0 & 1 & 2 \end{bmatrix}$$

Example 4 of Lecture 2

- Recall the adjacency matrix L:

$$L = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

- Hence, we have:

$$L^{\mathsf{T}} L = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 3 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 & 3 & 0 \\ 0 & 0 & 1 & 1 & 0 & 2 \end{bmatrix} \quad L L^{\mathsf{T}} = \begin{bmatrix} 3 & 1 & 0 & 0 & 1 & 1 \\ 1 & 3 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 3 & 1 \\ 1 & 2 & 0 & 0 & 1 & 2 \end{bmatrix}$$

- Now, the eigenvectors corresponding to the largest eigenvalues for these two matrices are as follows (using Julia's `eigen` function):

$$\begin{aligned} \boldsymbol{q}_1(L^\mathsf{T} L) &= (0.226000, 0.182068, 0.606615, 0.372375, 0.598376, 0.226000)^\mathsf{T} \\ \boldsymbol{q}_1(L L^\mathsf{T}) &= (0.458139, 0.568687, 0.0898142, 0.00000, 0.478872, 0.478872)^\mathsf{T} \end{aligned}$$

- Hence, using a simple tie-breaking strategy, we have:

$$\begin{aligned} \text{Authority Ranking} &= (3, 5, 4, 1, 6, 2); \\ \text{Hub Ranking} &= (2, 5, 6, 1, 3, 4). \end{aligned}$$

which are quite reasonable. Recall the web graph:

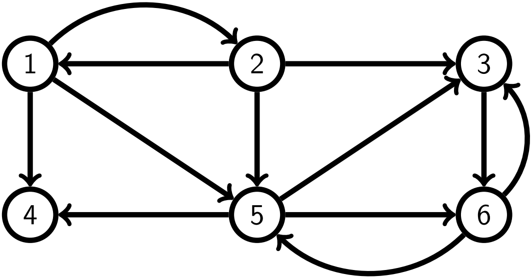
- Now, the eigenvectors corresponding to the largest eigenvalues for these two matrices are as follows (using Julia's `eigen` function):

$$
\begin{aligned}
\boldsymbol{q}_1(L^\mathsf{T} L) &= (0.226000, 0.182068, 0.606615, 0.372375, 0.598376, 0.226000)^\mathsf{T} \\
\boldsymbol{q}_1(L L^\mathsf{T}) &= (0.458139, 0.568687, 0.0898142, 0.00000, 0.478872, 0.478872)^\mathsf{T}
\end{aligned}
$$

- Hence, using a simple tie-breaking strategy, we have:

$$
\begin{aligned}
\text{Authority Ranking} &= (3, 5, 4, 1, 6, 2); \\
\text{Hub Ranking} &= (2, 5, 6, 1, 3, 4).
\end{aligned}
$$

which are quite reasonable. Recall the web graph:



- Now, how about using the Power Iteration in this example?
- With $\boldsymbol{v}^{(0)} = \frac{1}{\sqrt{6}}(1,1,1,1,1,1)^{\mathsf{T}}$ and 10 iteration, we got

$$\begin{aligned}
\boldsymbol{v}^{(10)}(L^{\mathsf{T}}L) &= (0.225992, 0.182069, 0.606614, 0.37239, 0.598363, 0.226021)^{\mathsf{T}} \\
\boldsymbol{v}^{(10)}(LL^{\mathsf{T}}) &= (0.458139, 0.568673, 0.0898284, 0.0000, 0.478895, 0.478864)^{\mathsf{T}}
\end{aligned}$$

- Compare them with

$$\begin{aligned}
\boldsymbol{q}_1(L^{\mathsf{T}}L) &= (0.226000, 0.182068, 0.606615, 0.372375, 0.598376, 0.226000)^{\mathsf{T}} \\
\boldsymbol{q}_1(LL^{\mathsf{T}}) &= (0.458139, 0.568687, 0.0898142, 0.00000, 0.478872, 0.478872)^{\mathsf{T}}
\end{aligned}$$

- The relative ℓ^2 errors are: 2.9665448×10^{-5} and 3.1486126×10^{-5}, respectively.

- Now, how about using the Power Iteration in this example?
- With $\boldsymbol{v}^{(0)} = \frac{1}{\sqrt{6}}(1,1,1,1,1,1)^{\mathsf{T}}$ and 10 iteration, we got

$$\begin{aligned}
\boldsymbol{v}^{(10)}(L^{\mathsf{T}}L) &= (0.225992, 0.182069, 0.606614, 0.37239, 0.598363, 0.226021)^{\mathsf{T}} \\
\boldsymbol{v}^{(10)}(LL^{\mathsf{T}}) &= (0.458139, 0.568673, 0.0898284, 0.0000, 0.478895, 0.478864)^{\mathsf{T}}
\end{aligned}$$

- Compare them with

$$\begin{aligned}
\boldsymbol{q}_1(L^{\mathsf{T}}L) &= (0.226000, 0.182068, 0.606615, 0.372375, 0.598376, 0.226000)^{\mathsf{T}} \\
\boldsymbol{q}_1(LL^{\mathsf{T}}) &= (0.458139, 0.568687, 0.0898142, 0.00000, 0.478872, 0.478872)^{\mathsf{T}}
\end{aligned}$$

- The relative ℓ^2 errors are: 2.9665448×10^{-5} and 3.1486126×10^{-5}, respectively.

- Now, how about using the Power Iteration in this example?
- With $\boldsymbol{v}^{(0)} = \frac{1}{\sqrt{6}}(1,1,1,1,1,1)^{\mathsf{T}}$ and 10 iteration, we got

$$\begin{aligned}
\boldsymbol{v}^{(10)}(L^{\mathsf{T}}L) &= (0.225992, 0.182069, 0.606614, 0.37239, 0.598363, 0.226021)^{\mathsf{T}} \\
\boldsymbol{v}^{(10)}(LL^{\mathsf{T}}) &= (0.458139, 0.568673, 0.0898284, 0.0000, 0.478895, 0.478864)^{\mathsf{T}}
\end{aligned}$$

- Compare them with

$$\begin{aligned}
\boldsymbol{q}_1(L^{\mathsf{T}}L) &= (0.226000, 0.182068, 0.606615, 0.372375, 0.598376, 0.226000)^{\mathsf{T}} \\
\boldsymbol{q}_1(LL^{\mathsf{T}}) &= (0.458139, 0.568687, 0.0898142, 0.00000, 0.478872, 0.478872)^{\mathsf{T}}
\end{aligned}$$

- The relative ℓ^2 errors are: 2.9665448×10^{-5} and 3.1486126×10^{-5}, respectively.

- Now, how about using the Power Iteration in this example?
- With $\boldsymbol{v}^{(0)} = \frac{1}{\sqrt{6}}(1,1,1,1,1,1)^{\mathsf{T}}$ and 10 iteration, we got

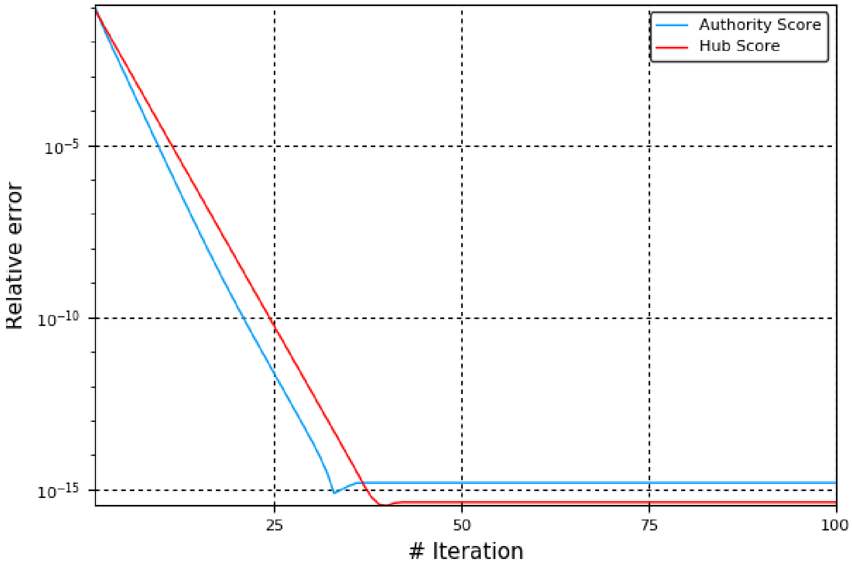
$$
\begin{aligned}
\boldsymbol{v}^{(10)}(L^{\mathsf{T}}L) &= (0.225992, 0.182069, 0.606614, 0.37239, 0.598363, 0.226021)^{\mathsf{T}} \\
\boldsymbol{v}^{(10)}(LL^{\mathsf{T}}) &= (0.458139, 0.568673, 0.0898284, 0.0000, 0.478895, 0.478864)^{\mathsf{T}}
\end{aligned}
$$

- Compare them with

$$
\begin{aligned}
\boldsymbol{q}_1(L^{\mathsf{T}}L) &= (0.226000, 0.182068, 0.606615, 0.372375, 0.598376, 0.226000)^{\mathsf{T}} \\
\boldsymbol{q}_1(LL^{\mathsf{T}}) &= (0.458139, 0.568687, 0.0898142, 0.00000, 0.478872, 0.478872)^{\mathsf{T}}
\end{aligned}
$$

- The relative ℓ^2 errors are: 2.9665448×10^{-5} and 3.1486126×10^{-5}, respectively.

Relative ℓ^2 Errors of Power Iteration Results



How can we construct a web graph in the first place?

- Well, it's good to know the HITS algorithm to get the authority and hub scores of the web sites, but how can we build the underlying web graph related to given query terms?
- That underlying web graph is called a *neighborhood graph* and denoted by $\mathcal{N}$.
- We want all documents (web sites) containing references to the query terms as the nodes in $\mathcal{N}$. There are various ways to do this.
- One simple method consults the inverted term-document file, which lists the column indices (= document id's) of the nonzero entries of the term-document matrix in the rows (terms) corresponding to the query terms.
- Once those documents are included as the nodes of $\mathcal{N}$, construct edges of inlinks and outlinks among them.
- continue to the next page!

How can we construct a web graph in the first place?

- Well, it's good to know the HITS algorithm to get the authority and hub scores of the web sites, but how can we build the underlying web graph related to given query terms?
- That underlying web graph is called a *neighborhood graph* and denoted by $\mathscr{N}$.
- We want all documents (web sites) containing references to the query terms as the nodes in $\mathscr{N}$. There are various ways to do this.
- One simple method consults the inverted term-document file, which lists the column indices (= document id's) of the nonzero entries of the term-document matrix in the rows (terms) corresponding to the query terms.
- Once those documents are included as the nodes of $\mathscr{N}$, construct edges of inlinks and outlinks among them.
- continue to the next page!

How can we construct a web graph in the first place?

- Well, it's good to know the HITS algorithm to get the authority and hub scores of the web sites, but how can we build the underlying web graph related to given query terms?
- That underlying web graph is called a *neighborhood graph* and denoted by $\mathscr{N}$.
- We want all documents (web sites) containing references to the query terms as the nodes in $\mathscr{N}$. There are various ways to do this.
- One simple method consults the inverted term-document file, which lists the column indices (= document id's) of the nonzero entries of the term-document matrix in the rows (terms) corresponding to the query terms.
- Once those documents are included as the nodes of $\mathscr{N}$, construct edges of inlinks and outlinks among them.
- continue to the next page!

How can we construct a web graph in the first place?

- Well, it's good to know the HITS algorithm to get the authority and hub scores of the web sites, but how can we build the underlying web graph related to given query terms?
- That underlying web graph is called a *neighborhood graph* and denoted by $\mathscr{N}$.
- We want all documents (web sites) containing references to the query terms as the nodes in $\mathscr{N}$. There are various ways to do this.
- One simple method consults the inverted term-document file, which lists the column indices (= document id's) of the nonzero entries of the term-document matrix in the rows (terms) corresponding to the query terms.
- Once those documents are included as the nodes of $\mathscr{N}$, construct edges of inlinks and outlinks among them.
- continue to the next page!

How can we construct a web graph in the first place?

- Well, it's good to know the HITS algorithm to get the authority and hub scores of the web sites, but how can we build the underlying web graph related to given query terms?
- That underlying web graph is called a *neighborhood graph* and denoted by $\mathscr{N}$.
- We want all documents (web sites) containing references to the query terms as the nodes in $\mathscr{N}$. There are various ways to do this.
- One simple method consults the inverted term-document file, which lists the column indices (= document id's) of the nonzero entries of the term-document matrix in the rows (terms) corresponding to the query terms.
- Once those documents are included as the nodes of $\mathscr{N}$, construct edges of inlinks and outlinks among them.
- continue to the next page!

How can we construct a web graph in the first place?

- Well, it's good to know the HITS algorithm to get the authority and hub scores of the web sites, but how can we build the underlying web graph related to given query terms?
- That underlying web graph is called a *neighborhood graph* and denoted by $\mathscr{N}$.
- We want all documents (web sites) containing references to the query terms as the nodes in $\mathscr{N}$. There are various ways to do this.
- One simple method consults the inverted term-document file, which lists the column indices (= document id's) of the nonzero entries of the term-document matrix in the rows (terms) corresponding to the query terms.
- Once those documents are included as the nodes of $\mathscr{N}$, construct edges of inlinks and outlinks among them.
- continue to the next page!

- Then, $\mathscr{N}$ is expanded by adding nodes outside of $\mathscr{N}$ that either point to the nodes in $\mathscr{N}$ (inlinks) or are pointed to by the nodes in $\mathscr{N}$ (outlinks).
- This expansion procedure allows some latent semantic associations to be made. For example, for the query term `car`, with the expansion about documents containing `car`, some documents containing `automobile` may now be added to $\mathscr{N}$, hopefully resolving the problem of synonyms.
- However, this expansion procedure may turn $\mathscr{N}$ into a huge graph, e.g., a node to be added into $\mathscr{N}$ may contain a huge number of outlinks.
- Hence, in practice, the maximum number of inlinking nodes and outlinking nodes to add for a particular node in $\mathscr{N}$ is fixed, say, the first 100 nodes or randomly picked 100 nodes among all the inlinking/outlinking nodes.

- Then, $\mathscr{N}$ is expanded by adding nodes outside of $\mathscr{N}$ that either point to the nodes in $\mathscr{N}$ (inlinks) or are pointed to by the nodes in $\mathscr{N}$ (outlinks).
- This expansion procedure allows some latent semantic associations to be made. For example, for the query term `car`, with the expansion about documents containing `car`, some documents containing `automobile` may now be added to $\mathscr{N}$, hopefully resolving the problem of synonyms.
- However, this expansion procedure may turn $\mathscr{N}$ into a huge graph, e.g., a node to be added into $\mathscr{N}$ may contain a huge number of outlinks.
- Hence, in practice, the maximum number of inlinking nodes and outlinking nodes to add for a particular node in $\mathscr{N}$ is fixed, say, the first 100 nodes or randomly picked 100 nodes among all the inlinking/outlinking nodes.

- Then, $\mathscr{N}$ is expanded by adding nodes outside of $\mathscr{N}$ that either point to the nodes in $\mathscr{N}$ (inlinks) or are pointed to by the nodes in $\mathscr{N}$ (outlinks).
- This expansion procedure allows some latent semantic associations to be made. For example, for the query term `car`, with the expansion about documents containing `car`, some documents containing `automobile` may now be added to $\mathscr{N}$, hopefully resolving the problem of synonyms.
- However, this expansion procedure may turn $\mathscr{N}$ into a huge graph, e.g., a node to be added into $\mathscr{N}$ may contain a huge number of outlinks.
- Hence, in practice, the maximum number of inlinking nodes and outlinking nodes to add for a particular node in $\mathscr{N}$ is fixed, say, the first 100 nodes or randomly picked 100 nodes among all the inlinking/outlinking nodes.

- Then, $\mathscr{N}$ is expanded by adding nodes outside of $\mathscr{N}$ that either point to the nodes in $\mathscr{N}$ (inlinks) or are pointed to by the nodes in $\mathscr{N}$ (outlinks).
- This expansion procedure allows some latent semantic associations to be made. For example, for the query term `car`, with the expansion about documents containing `car`, some documents containing `automobile` may now be added to $\mathscr{N}$, hopefully resolving the problem of synonyms.
- However, this expansion procedure may turn $\mathscr{N}$ into a huge graph, e.g., a node to be added into $\mathscr{N}$ may contain a huge number of outlinks.
- Hence, in practice, the maximum number of inlinking nodes and outlinking nodes to add for a particular node in $\mathscr{N}$ is fixed, say, the first 100 nodes or randomly picked 100 nodes among all the inlinking/outlinking nodes.

Strengths of HITS

- HITS presents two ranked lists to the user: one with the most authoritative documents (web sites) to the query; the other with the most "hubby" documents.
- HITS also casts the over all Web Information Retrieval problem as a small problem: finding the dominant eigenvectors of relatively small matrices compared to the entire Web documents.

Strengths of HITS

+ HITS presents two ranked lists to the user: one with the most authoritative documents (web sites) to the query; the other with the most "hubby" documents.
+ HITS also casts the over all Web Information Retrieval problem as a small problem: finding the dominant eigenvectors of relatively small matrices compared to the entire Web documents.

Weaknesses of HITS

- HITS's query dependence: at query time, $\mathscr{N}$ must be build and at least one matrix eigenvector problem solved. This must be done for *each* query.
- HITS's susceptibility to spamming: by adding links to and from a his/her webpage, a user can slightly influence the authority and hub scores of his/her page. A slight change in these scores might be enough to move his/her webpage a few notches up the ranked lists returned to another user. This becomes an especially important issue since a typical user searching webpages generally view only the top 20 pages returned in a ranked list.
    - From a perspective of a webpage owner, adding outlinks from a page is much easier than adding inlinks to that page. So, influencing one's hub score is not difficult.
    - Yet, since hub scores and authority scores share an interdependence and are computed interdependently, an authority score will increase as a hub score increases.
    - Also, since $\mathscr{N}$ is small compared to the entire Web, local changes to the link structure will appear more drastic.

Weaknesses of HITS

- HITS's query dependence: at query time, $\mathscr{N}$ must be build and at least one matrix eigenvector problem solved. This must be done for *each* query.
- HITS's susceptibility to spamming: by adding links to and from a his/her webpage, a user can slightly influence the authority and hub scores of his/her page. A slight change in these scores might be enough to move his/her webpage a few notches up the ranked lists returned to another user. This becomes an especially important issue since a typical user searching webpages generally view only the top 20 pages returned in a ranked list.
  - From a perspective of a webpage owner, adding outlinks from a page is much easier than adding inlinks to that page. So, influencing one's hub score is not difficult.
  - Yet, since hub scores and authority scores share an interdependence and are computed interdependently, an authority score will increase as a hub score increases.
  - Also, since $\mathscr{N}$ is small compared to the entire Web, local changes to the link structure will appear more drastic.

Weaknesses of HITS

- HITS's query dependence: at query time, $\mathscr{N}$ must be build and at least one matrix eigenvector problem solved. This must be done for *each* query.
- HITS's susceptibility to spamming: by adding links to and from a his/her webpage, a user can slightly influence the authority and hub scores of his/her page. A slight change in these scores might be enough to move his/her webpage a few notches up the ranked lists returned to another user. This becomes an especially important issue since a typical user searching webpages generally view only the top 20 pages returned in a ranked list.
  - From a perspective of a webpage owner, adding outlinks from a page is much easier than adding inlinks to that page. So, influencing one's hub score is not difficult.
  - Yet, since hub scores and authority scores share an interdependence and are computed interdependently, an authority score will increase as a hub score increases.
  - Also, since $\mathscr{N}$ is small compared to the entire Web, local changes to the link structure will appear more drastic.

Weaknesses of HITS

- HITS's query dependence: at query time, $\mathscr{N}$ must be build and at least one matrix eigenvector problem solved. This must be done for *each* query.
- HITS's susceptibility to spamming: by adding links to and from a his/her webpage, a user can slightly influence the authority and hub scores of his/her page. A slight change in these scores might be enough to move his/her webpage a few notches up the ranked lists returned to another user. This becomes an especially important issue since a typical user searching webpages generally view only the top 20 pages returned in a ranked list.
  - From a perspective of a webpage owner, adding outlinks from a page is much easier than adding inlinks to that page. So, influencing one's hub score is not difficult.
  - Yet, since hub scores and authority scores share an interdependence and are computed interdependently, an authority score will increase as a hub score increases.
  - Also, since $\mathscr{N}$ is small compared to the entire Web, local changes to the link structure will appear more drastic.

Weaknesses of HITS

- HITS's query dependence: at query time, $\mathscr{N}$ must be build and at least one matrix eigenvector problem solved. This must be done for *each* query.
- HITS's susceptibility to spamming: by adding links to and from a his/her webpage, a user can slightly influence the authority and hub scores of his/her page. A slight change in these scores might be enough to move his/her webpage a few notches up the ranked lists returned to another user. This becomes an especially important issue since a typical user searching webpages generally view only the top 20 pages returned in a ranked list.
  - From a perspective of a webpage owner, adding outlinks from a page is much easier than adding inlinks to that page. So, influencing one's hub score is not difficult.
  - Yet, since hub scores and authority scores share an interdependence and are computed interdependently, an authority score will increase as a hub score increases.
  - Also, since $\mathscr{N}$ is small compared to the entire Web, local changes to the link structure will appear more drastic.

Weaknesses of HITS . . .

- Topic drift: in building $\mathscr{N}$ for a query, it is possible that a very authoritative yet off-topic document be linked to a document containing the query terms. This very authoritative document can carry so much weight that it and its neighboring documents dominate the relevant ranked list returned to the user, skewing the results towards off-topic documents.

References

1. A. N. Langville and C. D. Meyer: "A survey of eigenvector methods for Web information retrieval," *SIAM Review*, vol. 47, no. 1, pp. 135–161, 2005.
2. M. W. Berry and M. Browne: *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, Second Ed., SIAM, 2005.