Georgia State University
Digital Archive @ GSU

Mathematics Theses | Department of Mathematics and Statistics

5-28-2009

Applications of Linear Algebra to Information Retrieval

Jhansi Lakshmi Vasireddy
jhansiv@yahoo.com

Follow this and additional works at: http://digitalarchive.gsu.edu/math_theses

Recommended Citation
Vasireddy, Jhansi Lakshmi, "Applications of Linear Algebra to Information Retrieval" (2009). *Mathematics Theses.* Paper 71.

This Thesis is brought to you for free and open access by the Department of Mathematics and Statistics at Digital Archive @ GSU. It has been accepted for inclusion in Mathematics Theses by an authorized administrator of Digital Archive @ GSU. For more information, please contact digitalarchive@gsu.edu.

APPLICATIONS OF LINEAR ALGEBRA TO INFORMATION RETRIEVAL

by

JHANSI LAKSHMI VASIREDDY

Under the Direction of Dr. Frank J. Hall

ABSTRACT

Some of the theory of nonnegative matrices is first presented. The Perron-Frobenius theorem is highlighted. Some of the important linear algebraic methods of information retrieval are surveyed. Latent Semantic Indexing (LSI), which uses the singular value decomposition is discussed. The Hyper-Text Induced Topic Search (HITS) algorithm is next considered; here the power method for finding dominant eigenvectors is employed. Through the use of a theorem by Sinkohrn and Knopp, a modified HITS method is developed. Lastly, the PageRank algorithm is discussed. Numerical examples and MATLAB programs are also provided.

INDEX WORDS: Eigenvalue, Eigenvector, Nonnegative matrix, Perron-Frobenius theorem, Latent Semantic Indexing, Hyper-Text Induced Topic Search, Power method, PageRank

APPLICATIONS OF LINEAR ALGEBRA TO INFORMATION RETRIEVAL

by

JHANSI LAKSHMI VASIREDDY

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2009

Copyright by
Jhansi Lakshmi Vasireddy
2009

APPLICATIONS OF LINEAR ALGEBRA TO INFORMATION RETRIEVAL

by

JHANSI LAKSHMI VASIREDDY

Committee Chair: Frank J. Hall

Committee: Marina Arav
Zhongshan Li
Michael Stewart

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August 2009

ACKNOWLEDGEMENTS

The author wishes to gratefully acknowledge the assistance of Dr. Frank J. Hall without whose guidance this thesis would not have been possible. The author is also very grateful to Dr. Marina Arav, Dr. Zhongshan Li and Dr. Michael Stewart for their valuable feedback and suggestions.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

1. Introduction and Preliminaries 1

2. Nonnegative Matrices 3

3. Latent Semantic Indexing 6

4. Hyper-text Induced Topic Search 18

5. PageRank 29

References 39

1. Introduction and Preliminaries

We first look at the definitions of some basic terminology from matrix theory. The symbols $M_{m,n}(R)$ and $M_n(R)$ represent the set of all $m \times n$ and $n \times n$ real matrices, respectively. A square matrix $D = [d_{ij}]$ is called a diagonal matrix if $d_{ij} = 0$ whenever $i \neq j$. We can denote a diagonal matrix $D \in M_n(R)$ by $D = \text{diag}(d_{11}, \cdots, d_{nn}) = \text{diag}(\mathbf{x})$, where $\mathbf{x} = (d_{11}, \cdots, d_{nn})^T$ is an $n \times 1$ column vector. The matrix $I = \text{diag}(1, \cdots, 1)$, where $I \in M_n(R)$, is called an $n \times n$ identity matrix. A matrix is called a row stochastic matrix if the sum of the elements of each of the rows is equal to 1. A matrix is doubly stochastic if all row and column sums are equal to 1. Let A be an $n \times n$ matrix and $\mathbf{x}$ be an $n \times 1$ column vector (also denoted as $\mathbf{x} \in R^n$). Then, the scalar λ is called an eigenvalue of the matrix A if it satisfies the equation $A\mathbf{x} = \lambda\mathbf{x}$, for some $\mathbf{x} \neq 0$ (i.e, the scalar λ is a solution to the equation $\det(\lambda I - A) = 0$), and the nonzero vector $\mathbf{x}$ is called an eigenvecor associated with the eigenvalue λ. The set of all $\lambda \in C$ that are eigenvalues of the matrix $A \in M_n(R)$ is called the spectrum of A and is denoted by $\sigma(A)$. The spectral radius of A is $\max\{|\lambda| : \lambda \in \sigma(A)\}$, and is denoted by $\rho(A)$. An eigenvalue having the maximum magnitude is called a dominant eigenvalue. An eigenvector corresponding to a dominant eigenvalue is called a dominant eigenvector. The algebraic multiplicity of an eigenvalue is defined as the multiplicity of the corresponding root of the characteristic polynomial. The geometric multiplicity of an eigenvalue is the number of linearly independent eigenvectors corresponding to that eigenvalue. A matrix $A \in M_n(R)$ is real symmetric if $A^T = A$. An $n \times n$ real symmetric matrix A is said to be positive semi-definite if $\mathbf{x}^T A\mathbf{x} \geq 0$, for all $\mathbf{x} \in R^n$. For a positive semi-definite matrix A, each eigenvalue $\lambda \geq 0$.

A matrix $A \in M_n(R)$ is said to be reducible if $n \geq 2$, and there is a permutation matrix $P \in M_n(R)$, and some integer r with $1 \leq r \leq n-1$, such that

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

where $B \in M_r(R), D \in M_{n-r}(R), C \in M_{r,n-r}(R)$, and $0 \in M_{n-r,r}(R)$ is a zero matrix.
A matrix $A \in M_n(R)$ is said to be irreducible if it is not reducible.
The directed graph of $A \in M_n(R)$, denoted by $\Gamma(A)$, is the directed graph on n nodes $P_1, P_2, \cdots, P_n$ such that there is a directed arc in $\Gamma(A)$ from P_i to P_j if $a_{ij} \neq 0$.

A matrix $A \in M_n(R)$ is said to be partly decomposable if $n = 1$ and $A = 0$, or, $n \geq 2$ and there are permutation matrices $P, Q \in M_n(R)$ and some integer r with $1 \leq r \leq n-1$, such that

$$PAQ = \begin{bmatrix} B & C \\ 0 & D \end{bmatrix}$$

where $B \in M_r(R), D \in M_{n-r}(R), C \in M_{r,n-r}(R)$, and $0 \in M_{n-r,r}(R)$ is a zero matrix.
A matrix $A \in M_n(R)$ is said to be fully indecomposable if it is not partly decomposable.

In Chapter 2 of this thesis we will first present some of the theory of nonnegative matrices. The Perron-Frobenius theorem for nonnegative, irreducible matrices is highlighted. We then survey some of the important linear algebraic methods of information retrieval. In Chapter 3 we discuss Latent Semantic Indexing (LSI), which uses the singular value decomposition. The Hyper-Text Induced Topic Search (HITS) algorithm is considered in Chapter 4; here the power method for finding dominant eigenvectors is employed. Through the use of a theorem by Sinkhorn and Knopp, a modified HITS method is developed. The PageRank algorithm is explained in Chapter 5. Numerical examples and MATLAB programs are also provided.

2. Nonnegative Matrices

In this chapter we survey some of the theory of nonnegative matrices. First, we give a few results on matrices in $M_n(R)$. Through the use of the Jordan canonical form, the following foundational result about convergent matrices can be established, see, [HJ06], page 138.

Theorem 2.1 Let $A \in M_n(R)$. Then $\lim_{m\to\infty} A^m = 0$ if and only if $\rho(A) < 1$.

As a consequence, we have the following.

Theorem 2.2 Let $\|\cdot\|$ be a matrix norm on $M_n(R)$. Then $\lim_{m\to\infty} \|A^m\|^{\frac{1}{m}} = \rho(A)$.

Proof: If $\sigma(A) = \{\lambda_1, \cdots, \lambda_n\}$, then $\sigma(A^m) = \{\lambda_1^m, \cdots, \lambda_n^m\}$.

So, $[\rho(A)]^m = (\max_{1\le i\le n} |\lambda_i|)^m = \rho(A^m) \le \|A^m\|$

and so $\rho(A) \le \|A^m\|^{\frac{1}{m}}$ for all m. (1)

Take $\epsilon > 0$ and let $\widetilde{A} = \frac{1}{\rho(A)+\epsilon}A$. Then, $\sigma(\widetilde{A}) = \{\frac{1}{\rho(A)+\epsilon}\lambda_1, \cdots, \frac{1}{\rho(A)+\epsilon}\lambda_n\}$.

Hence, $\rho(\widetilde{A}) = \frac{\rho(A)}{\rho(A)+\epsilon} < 1$.

So, by Theorem 2.1, $\lim_{m\to\infty} (\widetilde{A})^m = 0$.

Then $\|(\widetilde{A})^m\| \to 0$ for any matrix norm, since all vector norms on the n^2- dimensional vector space $M_n(R)$ are equivalent.

So, there exists N such that if $m \ge N$, then $\|(\widetilde{A})^m\| \le 1$

or $\frac{1}{[\rho(A)+\epsilon]^m}\|A^m\| \le 1$

or $\|A^m\| \le [\rho(A)+\epsilon]^m$

or $\|A^m\|^{\frac{1}{m}} \le \rho(A)+\epsilon$

or $\|A^m\|^{\frac{1}{m}} - \rho(A) \le \epsilon$.

With (1) we then have

$|\|A^m\|^{\frac{1}{m}} - \rho(A)| \le \epsilon$ for $m \ge N$.

Thus the result holds. ■

Theorem 2.3 Let $A, B \in M_n(R)$. If $|A| \leq B$, then $\rho(A) \leq \rho(|A|) \leq \rho(B)$.

Proof: First, $|A^m| \leq |A|^m$ for all m.

Also, $|A| \leq B \Rightarrow |A|^m \leq B^m$.

So, $|A^m| \leq |A|^m \leq B^m$.

Hence, $\| \, |A^m| \, \|_F^{\frac{1}{m}} \leq \| \, |A|^m \, \|_F^{\frac{1}{m}} \leq \|B^m\|_F^{\frac{1}{m}}$, (where $\|A\|_F = (\sum_{i,j=1}^{n} a_{ij}^2)^{\frac{1}{2}}$).

But, $\| \, |A^m| \|_F^{\frac{1}{m}} = \|A^m\|_F^{\frac{1}{m}}$. Hence,

$\lim_{m\to\infty} \|A^m\|_F^{\frac{1}{m}} \leq \lim_{m\to\infty} \| \, |A|^m \, \|_F^{\frac{1}{m}} \leq \lim_{m\to\infty} \|B^m\|_F^{\frac{1}{m}}$.

The result follows from Theorem 2.2. ■

Corollary 2.4 Let $A, B \in M_n(R)$. If $0 \leq A \leq B$, then $\rho(A) \leq \rho(B)$.

Proof: If $0 \leq A \leq B$, then $|A| \leq B$.

The result follows from Theorem 2.3. ■

Theorem 2.5 If $A \in M_n(R)$ and $A \geq 0$, then $\rho(A)$ is an eigenvalue of A and there is a nonnegative vector $\mathbf{x} \geq 0, \mathbf{x} \neq 0$, such that $A\mathbf{x} = \rho(A)\mathbf{x}$.

Proof: For any $\epsilon > 0$, define $A(\epsilon) \equiv [a_{ij} + \epsilon] > 0$. Denote by $\mathbf{x}(\epsilon)$ the Perron vector (see [HJ06], page 497) of $A(\epsilon)$, so that $\mathbf{x}(\epsilon) > 0$ and $\sum_{i=1}^{n} \mathbf{x}(\epsilon)_i = 1$. Since the set of vectors $\mathbf{x}(\epsilon) : \epsilon > 0$ is contained in the compact set $\mathbf{x} : \mathbf{x} \in R^n, \|\mathbf{x}\|_1 \leq 1$, there is a monotone decreasing sequence $\epsilon_1, \epsilon_2, \cdots$ with $\lim_{k\to\infty} \epsilon_k = 0$ such that $\lim_{k\to\infty} \mathbf{x}(\epsilon_k) \equiv \mathbf{x}$ exists. Since $\mathbf{x}(\epsilon_k) > 0$ for all $k = 1, 2, \cdots$, it must be that $\mathbf{x} = \lim_{k\to\infty} \mathbf{x}(\epsilon_k) \geq 0; \mathbf{x} = 0$ is impossible because

$$\sum_{i=1}^{n} \mathbf{x_i} = \lim_{k\to\infty} \sum_{i=1}^{n} \mathbf{x}(\epsilon_k)_i = 1$$

By Corollary 2.4 , $\rho(A(\epsilon_k)) \geq \rho(A(\epsilon_{k+1})) \geq \cdots \geq \rho(A)$ for all $k = 1, 2, \cdots$, so the sequence of real numbers $\rho(A(\epsilon_k))_{k=1,2,\cdots}$ is a monotone decreasing sequence. Thus, $\rho \equiv \lim_{k\to\infty} \rho(A(\epsilon_k))$ exists and $\rho \geq \rho(A)$. But from the fact that

$$A\mathbf{x} = \lim_{k\to\infty} A(\epsilon_k)\mathbf{x}(\epsilon_k) = \lim_{k\to\infty} \rho(A(\epsilon_k))\mathbf{x}(\epsilon_k) = \lim_{k\to\infty} \rho(A\epsilon_k)) \lim_{k\to\infty} \mathbf{x}(\epsilon_k) = \rho\mathbf{x}$$

and the fact that $\mathbf{x} \neq 0$, we deduce that ρ is an eigenvalue of A. But then $\rho \geq \rho(A)$, so it must be that $\rho = \rho(A)$. ■

The matrices generated by the information retrieval methods such as HITS are nonnegative. Thus, for such matrices, Theorem 2.5 guarantees that there are nonnegative eigenvectors associated with the eigenvalue $\rho(A)$.

For nonnegative irreducible matrices we have the celebrated Perron-Frobenius theorem. For a proof, see chapter 8 in [HJ06].

Theorem 2.6 Let $A \in M_n(R)$ and suppose that A is irreducible and nonnegative. Then
(a) $\rho(A) > 0$;
(b) $\rho(A)$ is an eigenvalue of A;
(c) There is a positive vector $\mathbf{x}$ such that $A\mathbf{x} = \rho(A)\mathbf{x}$; and
(d) $\rho(A)$ is an algebraically (and hence geometrically) simple eigenvalue of A.

In particular, for the nonnegative PageRank Google matrices (see chapter 5), irreducibility is imposed and the Perron-Frobenius theorem is applicable.

3. Latent Semantic Indexing

The main goal of information retrieval is to retrieve the documents from a given document database that are most relevant to a given user query. When we try to literally match the user query word-by-word we run into the problems of synonymy and polysemy. Synonymy refers to the concept of multiple words having the same meaning. Polysemy refers to the concept of the same word having multiple meanings. For example, the word "bark" both refers to a tree bark and a dog's bark. Sometimes when the user query may not contain all the terms, it is necessary to understand the underlying concept or meaning of the user query. Also some documents that are relevant to the user query may not contain the exact terms specified in the user's query.

The techniques of Linear Algebra are used for simple lexical matching and also for sophisticated matching techniques such as LSI (Latent Semantic Indexing). The following illustrates how we can do a simple lexical matching using Linear Algebra [L06].

Suppose there are are m terms and n documents in a document database. The common terms such as prepositions, articles etc., are stripped away from each document. The m terms are the set of the remaining terms in all the documents. The term-by-document matrix A is of size $m \times n$ and is formed by considering the terms as rows and the documents as columns. The (i, j) entry of the matrix A is 1 if the term i is present in the document j, 0 otherwise. An $m \times 1$ query vector $\mathbf{q}$ is formed by considering the terms which interest the user. The ith entry of $\mathbf{q}$ is 1 if the user is interested in term i, 0 otherwise.

Let the vector $\mathbf{y} = A^T\mathbf{q}$. The ith entry of the vector $\mathbf{y}$ represents the number of query terms present in the document i. Thus we can find the documents that have the most number of query terms using this method.

As mentioned above, by doing an exact query match we face the problems of synonymy, polysemy and other problems. If we consider how closely each of the document vectors is related to the query vector, we can overcome some of the problems associated with exact

matching of the query. The cosine of the angle between a document vector and the query vector gives a measure of how closely they are related.

We now represent the entries of the term-by-document matrix A by relative frequencies. The matrix P is formed by scaling each column of A so that all the column vectors are unit vectors (using the 2-norm). The columns of the database matrix P are determined by setting

$$\mathbf{p_j} = \frac{1}{\|\mathbf{a_j}\|}\mathbf{a_j} \quad j = 1, 2, \cdots\cdots .$$

The query vector $\mathbf{q}$ is also scaled so that it becomes a unit vector. If we set $\mathbf{y} = P^T\mathbf{q}$, then $y_i = \mathbf{p_i}^T\mathbf{q} = \cos\theta_i$, where θ_i is the angle between the unit vectors $\mathbf{q}$ and $\mathbf{p_i}$. The closer to 1 that $\cos\theta_i$ is, the better the match of the query vector and the ith document.

LSI also calculates the cosine of the angle between the document vector and the query vector. LSI computes the cosine of the angle in the reduced rank document vector space. It uses Singular Value Decomposition [L06] to reduce the rank of the document vector space. A detailed description of LSI along with an example can be found in [DDF90] and [BDO95].

LSI emphasizes on identifying the implicit higher order structure in the association of terms with documents [BDO95]. The high computational cost of LSI restricts it to a relatively smaller number of documents, say around few hundred thousand. It cannot be applied to huge document databases such as the World Wide Web. As mentioned above, the simple lexical retrieval methods restrict the scope of the search. They also can be inaccurate due to the problems of synonymy and polysemy. LSI helps to overcome these problems. It also helps to identify related documents which may not contain the actual search term.

LSI assumes that there is some underlying latent semantic structure in the data. Singular Value Decomposition allows the arrangement of the document space to reflect the major associative patterns in the data, and to ignore the smaller, less important influences. As a result, terms that did not actually appear in a document may still end up close to the document, if that is consistent with the major patterns of associations in the data. The position in the space then serves as the new kind of semantic indexing, and the retrieval proceeds by using the terms in a query to identify a point in the space, and documents in

the neighborhood are returned to the user. The following theorems describe the Singular Value Decomposition.

Singular Value Decomposition

Theorem 3.1. If $A \in M_{m,n}(R)$ has rank r, then it can factored in the form

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

where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and $\Sigma = [\sigma_{i,j}]$ is an $m \times n$ matrix whose off-diagonal entries are all 0, and

$$\sigma_{11} \geq \sigma_{22} \geq \cdots \geq \sigma_{rr} > \sigma_{r+1,r+1} = \cdots = \sigma_{qq} = 0,$$

where $q = \min\{m, n\}$.

Note:

1) The numbers $\sigma_{11}, \sigma_{22}, \cdots, \sigma_{rr}$ are the positive square roots of the positive eigenvalues of $A^T A$ and thus are unique. These numbers are called the *singular values* of A.

2) The columns of V are eigenvectors of $A^T A$ and the columns of U are eigenvectors of AA^T.

3) The factorization $A = U\Sigma V^T$ is known as the *singular value decomposition* (SVD) of A.

4) Partitioning $U = [\mathbf{u_1}, \cdots, \mathbf{u_m}]$ and $V = [\mathbf{v_1}, \cdots, \mathbf{v_n}]$ into columns, we can expand $A = U\Sigma V^T$ and then represent A by an outer product expansion

$$A = \sigma_{11}\mathbf{u_1}\mathbf{v_1}^T + \sigma_{22}\mathbf{u_2}\mathbf{v_2}^T + \cdots + \sigma_{rr}\mathbf{u_r}\mathbf{v_r}^T.$$

Theorem 3.2. Let $A \in M_{m,n}(R)$, $r = \text{rank}(A)$, $A = U\Sigma V^T$ be the singular value decomposition of A, and M_k denote the set of all matrices in $M_{m,n}(R)$ with rank k or less, where $0 < k < r$. Further, let the $m \times n$ matrix $A_k = U_k \Sigma_k V_k^T$, where U_k is the $m \times k$ matrix whose columns are the first k columns of U, V_k is the $n \times k$ matrix whose columns are the

first k columns of V, and Σ_k is the $k \times k$ diagonal matrix whose diagonal entries are the k largest singular values of A. Then,

$$\|A - A_k\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2} = \min_{S \in M_k} \|A - S\|_F .$$

The proofs of Theorem 3.1 and Theorem 3.2 can be found in [L06], Chapter 6. We point out that the matrix A_k in Theorem 3.2 is the following rank k approximation of the rank r matrix A:

$$A_k = \sigma_{11}\mathbf{u_1}\mathbf{v_1}^T + \sigma_{22}\mathbf{u_2}\mathbf{v_2}^T + \cdots + \sigma_{kk}\mathbf{u_k}\mathbf{v_k}^T.$$

This is obtained by truncating the last $r-k$ terms from the outer expansion of A. There are many applications of the singular value decomposition, including digital image processing and principal component analysis. We next consider the application to information retrieval.

Initially the term-by-document matrix A is constructed. The matrix A is factored into the product of three matrices U, V, and Σ using SVD. We find that Theorem 3.1 ensures that any term-by-document matrix A has a SVD decomposition. The SVD derives the latent semantic structure model from the three factor matrices U, Σ, V. These matrices reflect a breakdown of the original relationships into linearly independent vectors or factor values. Then the matrix A_k is formed by using the k largest singular values as described in Theorem 3.2.

The use of k largest singular values approximates the original term-by-document matrix A by A_k. The truncated matrix A_k is supposed to capture most of the underlying structure in the association of terms and documents, and also removes the noise or variability. The smaller associative patterns in the data do not reflect the underlying semantic structure. These are eliminated by reducing the dimension of the original document vector space. Terms that occur in similar documents will be near each other in the k-dimensional space even if they never co-occur in the same document. Some documents which do not share any words with a user's query may be near it in k-space.

For example, consider the words physician, doctor, treatment and waste-water. The terms physician and doctor are synonyms, treatment is a related concept to doctors, while waste-water is an unrelated concept to doctors. The words physician and doctor will occur with many of the same words such as health, disease, medicine, prescription, hospital, diagnosis etc. Hence these two words will have the similar representation in the document space. Thus, when we search for the phrase "doctor treatment", the documents that use the synonym physician will also be retrieved even though they may not contain the word doctor. The documents related to treatment, diseases etc., will also have a great overlap with the documents related to doctor. Hence, the corresponding document vectors will have a smaller angle with the query vector. However, documents that are related to waste-water treatment will have lesser overlap with the documents related to doctors. The reduction of dimension of the document space will reduce the interference from these lesser related concepts.

The query vector $\tilde{\mathbf{q}}$ in the reduced k-space is computed in the following way:

$$\tilde{\mathbf{q}} = \mathbf{q}^{\mathbf{T}} U_k \Sigma_k^{-1}.$$

The vector $\mathbf{q}$ is the vector of words in the user's query. The sum of the k-dimensional term vectors is reflected by the $\mathbf{q}^{\mathbf{T}} U_k$ term in the above equation, and the right multiplication by Σ_k^{-1} differentially weights the separate dimensions. Thus, the query vector is located at the weighted sum of its constituent term vectors. The query vector $\tilde{\mathbf{q}}$ can then be compared to all the document vectors. In order to be compared with the query vector $\tilde{\mathbf{q}}$, we need to compute the document vectors too in the reduced k-space. The jth column of A_k is given by $A_k \mathbf{e_j}$, where the vector $\mathbf{e_j}$ is the jth column of $n \times n$ identity matrix. The j^{th}document vector $\tilde{\mathbf{d_j}}$ in the reduced k-space is computed in the following way:

$$\tilde{\mathbf{d_j}} = (A_k \mathbf{e_j})^T U_k \Sigma_k^{-1}$$

for $j = 1, \cdots, n$, which can be easily simplified to $\mathbf{e_j}^T V_k$.

The cosine of the angle between the query vetor $\tilde{\mathbf{q}}$ and the document vector $\tilde{\mathbf{d}}_\mathbf{j}$ gives the measure of how closely they are related. The cosines are computed by

$$\cos\theta_j = \frac{\tilde{\mathbf{d}}_\mathbf{j}^{\,T}\,\tilde{\mathbf{q}}}{\|\tilde{\mathbf{d}}_\mathbf{j}\|_2\ \|\tilde{\mathbf{q}}\|_2}$$

for $j = 1, \cdots, n$.

Example

The MATLAB program in this example orders the given documents according to their relevance to the query vector. The program takes the term-by-document matrix and the query vector as the input. The output is a list of $\cos\theta$ values for the list of documents. The list of documents followed by the query is given below.

D1: http://www.medicinenet.com/script/main/hp.asp

Description: MedicineNet provides reliable doctor produced health and medical information. Learn about diseases and conditions, symptoms and signs, procedures and tests, medications.

Terms: diseases, tests, symptoms

D2: http://www.ihmspt.com

Description: Provides physical therapy, rehabilitation services and workplace solutions.

Terms: therapy, workplace

D3: http://health.allrefer.com

Description: Medical encyclopedia with information about medical topics, surgeries, symptoms for diseases, tests and examinations, diet and nutrition, injuries and wounds, poisons and overdoses.

Terms: encyclopedia, symptoms, diseases, medical

D4: http://www.wrongdiagnosis.com/crtop/aboutus.htm

Description: One of the worlds leading providers of online medical health information. An

independent, objective source of factual, mainstream health information for both consumers and health professionals. Provides a free health-information service to help people understand their health better, offering crucial and factual health information that is otherwise difficult to find. The objective of the site is to encourage consumers to be informed and interested in managing their health, and to know what questions to ask their doctors to help ensure they are getting the best healthcare possible.
Terms: free, information, professionals

D5: http://northsidepediatrics.com/pages.meta
Description: A group of Board Certified Pediatricians committed to providing the highest quality of comprehensive care to infants, children and adolescents of the greater Atlanta area.
Terms: infants, children, Atlanta, pediatrician

D6: http://www.keepkidshealthy.com/welcome/welcome.html
Description: Pediatrician's guide to your children's health and safety. We supplement the information that you receive from your child's physician, with a special emphasis on better health through preventive care.
Terms: children, pediatrician, safety, preventive

The query is "information on children's diseases". The query terms are: information, children, disease.

The term-by-document matrix is given below:

$No.$	$Term$	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$
$T1$	$adolescent$	0	0	0	0	1	0
$T2$	$Atlanta$	0	0	0	0	1	0
$T3$	$children$	0	0	0	0	1	1
$T4$	$condition$	1	0	0	0	0	0
$T5$	$consumers$	0	0	0	1	0	0
$T6$	$dictionary$	1	0	0	0	0	0
$T7$	$diet$	0	0	1	0	0	0
$T8$	$disease$	1	0	1	0	0	0
$T9$	$encyclopedia$	0	0	1	0	0	0
$T10$	$examination$	0	0	1	0	0	0
$T11$	$free$	0	0	0	1	0	0
$T12$	$infant$	0	0	0	0	1	0
$T13$	$information$	1	0	1	1	0	0
$T14$	$injuries$	0	0	1	0	0	0
$T15$	$medical$	1	0	1	1	0	0
$T16$	$medications$	1	0	0	0	0	0
$T17$	$medicine$	1	0	0	0	0	0
$T18$	$nutrition$	0	0	1	0	0	0
$T19$	$overdose$	0	0	1	0	0	0
$T20$	$pediatrician$	0	0	0	0	1	1
$T21$	$poisons$	0	0	1	0	0	0
$T22$	$preventive$	0	0	0	0	0	1
$T23$	$procedure$	1	0	0	0	0	0
$T24$	$professional$	0	0	0	1	0	0
$T25$	$promotion$	0	1	0	0	0	0
$T26$	$providers$	0	1	0	0	0	0
$T27$	$safety$	0	0	0	0	0	1
$T28$	$surgery$	0	0	1	0	0	0
$T29$	$symptom$	1	0	1	0	0	0
$T30$	$terms$	1	0	0	0	0	0
$T31$	$tests$	1	0	1	0	0	0
$T32$	$therapy$	0	1	0	0	0	0
$T33$	$workplace$	0	1	0	0	0	0
$T34$	$wounds$	0	0	1	0	0	0

The query vector is: $[\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ]^T$

The MATLAB program is:

```
% The dimensions of the term-by-document matrix
m = 35; % The number of rows of term-by-document matrix
n = 6; % The number of columns of term-by-document matrix
k = 4; % The number of dimensions we would choose after SVD has been computed.
        % We would also compute the cosine values for k = 2, k = 3.
% tdmatrix holds the term-by-matrix
[U,Sigma,V] = svd(tdmatrix); % Compute the SVD of the tdmatrix
Uk = zeros(m, k);
Sigmak = zeros(k, k);
Vk = zeros(n, k);
for i = 1 : k
Uk(:, i) =U(:, i);
end
for i = 1 : k
for j = 1 : k
Sigmak(i, j) = Sigma(i, j);
end
end
for i = 1 : k
Vk(:, i) = V(:, i);
end
Sigmakinv = zeros(k, k);
for i = 1 : k
Sigmakinv(i, i) = 1/(Sigmak(i, i));
end
% queryvec holds the query vector.
```

```
queryvectrans = queryvec';
% The query vector in the k-space is stored in queryvecktrans
queryvecktrans = (queryvectrans) * (Uk) * (Sigmakinv);
% Finding the document co-ordinates in the k-space
% Computing the cosines between the query vector and document vectors
cosinematrix = zeros(n, 1);
normquery = norm(queryvecktrans, 2);
for i = 1 : n
dotproduct = 0;
for j = 1 : k
dotproduct = dotproduct + (queryvecktrans(1, j) * Vk(i, j));
end
normdoc = norm(Vk(i, :), 2);
% The computation of cosines for each of the document vector
cosinematrix(i, 1) = dotproduct/(normquery * normdoc);
end
%End of the program
```

As noted above, the document vector space is approximated by the k-space. The order of relevance for the documents, for various test queries, is compared for different values of k and the best approximation is chosen.

In this example, we find the order of relevance for the documents D1 to D6 for the values of $k = 2, 3, 4$.

For $k = 2$:

The $\cos\theta$ values for the documents D1 to D6 are: $0.8316, -0.0575, 0.5391, 0.9716, -0.0575, 0.0575$.

Hence the order of the documents according to their relevance is:

$$D4, D1, D3, D6, D2, D5.$$

For $k = 3$:

The $\cos\theta$ values for the documents D1 to D6 are: $0.6312, -0.4098, 0.4091, 0.7374, 0.6512, 0.6512$.

Hence the order of the documents according to their relevance is:

$$D4, D5, D6, D1, D3, D2.$$

For $k = 4$:

The $\cos\theta$ values for the documents D1 to D6 are: $0.6086, -0.1985, 0.4058, 0.2029, 0.6510, 0.6510$.

Hence the order of the documents according to their relevance is:

$$D5, D6, D1, D3, D4, D2.$$

The documents D5 and D6 which refer to children's diseases are the most relevant for the query "information on children's diseases". Hence, when $k = 4$, we find that the documents are most appropriately ordered according to their releavnce to the user's query. Thus, we get the best approximation of the document vector space when $k = 4$.

In [BDJ99], a new vector space model that is different from LSI but is based on SVD is described. In this model too a term-by-document matrix A and a query vector $\mathbf{q}$ are constructed. Then, as in LSI, the matrix A is factored into the matrices, U, Σ, V using SVD. Also, as in LSI, the matrix A_k is formed by using the k largest singular values as described in Theorem 3.2. But unlike LSI, this model compares the query vector $\mathbf{q}$ directly to the columns of A_k. The cosines of the angles between the query vector $\mathbf{q}$ and approximate document vectors are computed by

$$\cos\theta_j = \frac{(A_k\mathbf{e_j})^T\mathbf{q}}{\|A_k\mathbf{e_j}\|_2\ \|\mathbf{q}\|_2} = \frac{(U_k\Sigma_k V_k^T\mathbf{e_j})^T\mathbf{q}}{\|U_k\Sigma_k V_k^T\mathbf{e_j}\|_2\ \|\mathbf{q}\|_2} = \frac{\mathbf{e_j}^T V_k\Sigma_k(U_k^T\mathbf{q})}{\|\Sigma_k V_k^T\mathbf{e_j}\|_2\ \|\mathbf{q}\|_2}$$

for $j = 1, \cdots, n$. If we define the vector $\mathbf{s_j} = \Sigma_k V_k^T \mathbf{e_j}$, the formula reduces to

$$\cos\theta_j = \frac{\mathbf{s_j}^T (U_k^T \mathbf{q})}{\|\mathbf{s_j}\|_2 \; \|\mathbf{q}\|_2}, \quad j = 1, \cdots, n.$$

The cosines can be computed without explicitly forming the matrix A_k. The norms $\|\mathbf{s_j}\|_2$ are computed once for each term-by-document matrix and subsequently used for all queries.

4. Hyper-text Induced Topic Search

HITS (Hypertext Induced Topic Search) was developed by Jon Kleinberg [K99] in 1997. The HITS algorithm exploits the information embedded in the link structure of the pages in the World Wide Web (WWW). This algorithm is intended to determine the relevancy of the web pages returned by a user's search query.

The World Wide Web (WWW) is a huge growing collection of hyper-linked documents. When users search the WWW for a common topic there are thousands of documents which contain the search term. The challenge of the web search engines is to present to the users the most relevant documents out of these thousands of documents. The earlier search engines did not exploit the information embedded in the link structure and relied mainly on the textual content of the web pages. Henceforth, such search engines are referred to as simple text-based search engines.

The simple text-based search engines first extract all the documents that contain the search term. They then rank the documents based on the textual content of the web pages. They use factors such as how many times the search term appears in the web page, if the search term appears in the headings, or in the bold text etc., to rank the web pages as more relevant. But this approach has certain drawbacks. Some relevant or authoritative pages may not contain the search term at all. Some of the relevant pages may not contain the search term in such a way as to make them rank higher. For example, for a search term like "computer manufacturers" the true authoritative pages like www.ibm.com, www.apple.com, etc., may not be ranked higher by a text-based search engine as the search term might not appear in their home pages. In some cases, the authoritative pages might not have the search term on their home pages.

There is a lot of information embedded in the hyper-linked structure of the web. The hyper-linked structure of the web inherently has the human judgment as to which pages are more important. If a page j contains a link to page i then it is an inherent endorsement that page j gives to page i. If we consider only the number of in links to determine the relevance

of a page then it might result in universally popular pages like www.yahoo.com as ranking highly in terms of relevance.

Kleinberg [K99] suggests the HITS algorithm which exploits the link structure of the WWW and helps to overcome the above disadvantages of simple text-based searches. Initially, the documents are ranked by running a pure simple text-based search. Then the top few web pages, say around 200, are picked from this search. This set is referred to as the root set. The root set might not have the authoritative pages but there exists a high probability that these pages might point to the authoritative pages. The root set is expanded to include the pages that are either pointed by them or point to them. This expanded set is referred to as the base set.

A graph N is constructed from this base set. Each web page is considered as a vertex or a node. If page i has a link to page j then a corresponding directed edge is drawn from node i to node j. As mentioned above if a page j contains a link to page i then it is an inherent endorsement that page j gives to page i. Pages which have many endorsements are deemed to be more popular or more authoritative. But again as mentioned above, some pages are universally popular and will have many in links with respect to many queries, but these universally popular pages may not be most relevant to a "particular" user query. The authoritative pages should not only have many in links, but there should be a considerable overlap in the sets of pages that point to them. There are "hub pages" which point to many authority pages. Kleinberg [K99] suggests that the philosophy of the HITS algorithm is that "good authorities are pointed to by good hubs and good hubs point to good authorities".

This cyclicality is resolved and the relevancy of the web pages is determined in the following way:

Each page is assigned both a hub score and an authority score. The higher the authority score of a page, the more relevant it is.

Let a_i be the authority score of page i and h_i be the hub score of page i.

The adjacency matrix L of the graph N is defined as follows:

$L_{ij} = 1$, if there exists an edge from node i to node j, 0 otherwise.

The authority score of page i is the sum of the hub scores of all pages that point to page i. Letting $P_j \to P_i$ indicate that there exists a link from page j to page i, we have for all $i = 1, \ldots, n$ that

$$a_i = \sum_{j:P_j \to P_i} h_j,$$

or

$$a_i = (\text{row } i \text{ of } L^T)\mathbf{h_{k-1}}$$

so that

$$\mathbf{a_k} = L^T \mathbf{h_{k-1}}, \tag{1}$$

for all k, where $\mathbf{a_k}$ ($\mathbf{h_k}$) denotes the authority (hub) vector at the kth stage.

Similarly, for hub scores, we have for all $i = 1, \ldots, n$ that

$$h_i = \sum_{j:P_i \to P_j} a_j$$

or

$$h_i = (\text{row } i \text{ of } L)\mathbf{a_k}$$

so that

$$\mathbf{h_k} = L\mathbf{a_k}, \tag{2}$$

and correspondingly

$$\mathbf{h_{k-1}} = L\mathbf{a_{k-1}} \tag{3}$$

for all k.

Substitute (1) into (2):

$$\mathbf{h_k} = LL^T \mathbf{h_{k-1}}. \tag{4}$$

Substitute (3) into (1):

$$\mathbf{a_k} = L^T L\mathbf{a_{k-1}}. \tag{5}$$

The matrix $L^T L$ determines the authority scores, and hence it is called the *authority matrix*. The matrix LL^T is called the *hub matrix*. The equations (4) and (5) can be used to implement the power method for computing the limiting HITS vectors, **a** and **h**, [LM05]. The power method [M00] is explained below.

Power Method

For a diagonalizable marix $A \in M_n(R)$ with distinct eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_k$, there exist matrices $G_1, G_2, \cdots, G_k$ such that

$$A = \lambda_1 G_1 + \lambda_2 G_2 + \cdots\cdots + \lambda_k G_k, \tag{6}$$

where the G_i's have the following properties:

- G_i is the projector onto $N(A - \lambda_i I)$ along $R(A - \lambda_i I)$.
- $G_i G_j = 0$ whenever $i \neq j$.
- $G_1 + G_2 + \cdots + G_k = I$.

The expansion of matrix A in equation (6) is known as the *spectral decomposition* of A, and the G_i's are called the *spectral projectors* associated with A.

A function on the above diagonalizable matrix A can be written as

$$f(A) = \sum_{i=1}^{k} f(\lambda_i) G_i. \tag{7}$$

Consider a diagonalizable matrix A with eigenvalues

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_k|.$$

Consider the function $f(z) = (z/\lambda_1)^n$. From equation (7), we have

$$\left(\frac{A}{\lambda_1}\right)^n = f(A) = f(\lambda_1) G_1 + f(\lambda_2) G_2 + \cdots + f(\lambda_k) G_k.$$

Hence,

$$\left(\frac{A}{\lambda_1}\right)^n = G_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^n G_2 + \cdots + \left(\frac{\lambda_k}{\lambda_1}\right)^n G_k \to G_1.$$

as $n \to \infty$. Thus, $(A^n\mathbf{x_0}/\lambda_1^n) \to G_1\mathbf{x_0} \in N(A - \lambda_1 I)$ for all $\mathbf{x_0}$.

So if $G_1\mathbf{x_0} \neq \mathbf{0}$, or equivalently, $\mathbf{x_0} \notin R(A - \lambda_1 I)$, then $A^n\mathbf{x_0}/\lambda_1^n$ converges to an eigenvector associated wih λ_1 (or equivalently, to a dominant eigenvector of A). Thus, $A^n\mathbf{x_0}$ converges to the dominant eigenvector of A, as λ_1^n is only a scaling factor.

Let $m(A^n\mathbf{x_0})$ be a normalizing scalar derived from $A^n\mathbf{x_0}$. Suppose $m(A^n\mathbf{x_0}/\lambda_1^n) \to \gamma$. Since $(A^n/\lambda_1^n) \to G_1$, we have

$$\frac{A^n\mathbf{x_0}}{m(A^n\mathbf{x_0})} = \frac{(A^n/\lambda_1^n)\mathbf{x_0}}{m(A^n\mathbf{x_0}/\lambda_1^n)} \to \frac{G_1\mathbf{x_0}}{\gamma}$$

as $n \to \infty$. $\frac{G_1\mathbf{x_0}}{\gamma} = \mathbf{x}$ is an eigenvector associated with λ_1 or the dominant eigenvector of matrix A.

Rather than successively powering A, the sequence $A^n\mathbf{x_0}/m(A^n\mathbf{x_0})$ is more efficiently generated by starting with $\mathbf{x_0} \notin R(A - \lambda_1 I)$ and setting

$$\mathbf{y_n} = A\mathbf{x_n}, \quad \nu_n = m(\mathbf{y_n}), \quad \mathbf{x_{n+1}} = \frac{\mathbf{y_n}}{\nu_n}, \quad n = 0, 1, 2, \cdots$$

Then $\mathbf{x_n} \to \mathbf{x}$. And, if $m(\mathbf{v})$ is the first maximal component of $\mathbf{v}$, then $\nu_n \to \lambda_1$, the dominant eigenvalue of A.

HITS Convergence

The equations (4) and (5) can be used to apply the power method to compute the HITS vectors $\mathbf{a}$ and $\mathbf{h}$. To find the HITS authority vector we can set the following equations:

$$\mathbf{y_n} = L^T L\mathbf{a_n}, \quad \nu_n = m(\mathbf{y_n}), \quad \mathbf{a_{n+1}} = \frac{\mathbf{y_n}}{\nu_n}, \quad n = 0, 1, 2, \cdots$$

We can set the initial vector $\mathbf{a_0}$ to random values and normalize it (can use any norm such as the 1-norm) and generate the above sequence (using the same norm for $m(\mathbf{y_n})$). The authority vector $\mathbf{a}$ results from the convergence of the above sequence. Once we get the authority vector we can use the equation $\mathbf{h} = L\mathbf{a}$ to compute the hub vector.

Or, conversely, we can generate the sequence of hub vectors using a random normalized initial hub vector and use the equation $\mathbf{a} = L^T\mathbf{h}$ to compute the authority vector.

The matrices L^TL and LL^T are symmetric, positive semidefinite, and nonnegative. Hence, their distinct eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_k$ are real and non-negative with

$$\lambda_1 > \lambda_2 > \cdots > \lambda_k \geq 0.$$

This ensures that HITS with normalization always converges. This inequality $\lambda_1 > \lambda_2$ is true in all cases and hence it is not possible to have multiple dominant eigenvalues on the spectral circle. However, λ_1 can be a repeated root of the characteristic polynomial, in which case the dominant eigenspace can be multi-dimensional. In such a case, the different initial vectors produce different limiting authority or hub vectors. The article [LM05] has an example that shows how different limiting authority and hub vectors can be produced by using different initial vectors.

If the matrices L^TL and LL^T are irreducible then the Perron-Frobenius Theorem ensures that there exists a unique dominant eigenvector. But HITS does not impose irreducibility on L^TL and LL^T. Hence, the HITS algorithm can converge to non-unique solutions.

The power method also requires that the initial vector $\mathbf{a_0} \notin R(L^TL - \lambda_1 I)$. We randomly generate the initial vector and hence there is a high probability that this condition holds in practice [LM05].

Example

The following MATLAB program computes the authority vector for a given adjacency matrix, L. It takes the initial vector as input.

% Input the initial vector X.

$X = [1; 1; 1; 1; 1; 1]$

limitval = 0.0001; numiter = 500;

% Input the matrix L.

```
L = [
 0  1  0  1  0  0  ;
 1  0  0  1  0  0  ;
 0  1  0  1  0  0  ;
 0  0  1  0  1  1  ;
 0  0  1  0  0  0  ;
 0  0  1  0  0  0  ;
]

B = L' * L;

% Normalizing X

tempnm = norm(X, 1);

tempc = 1/tempnm;

X = tempc * X;

%Initializing XPREV

XPREV = X;

count = 0;

% While Loop

while ( count < numiter)

count = count +1 ;

X = B * X ;

% Normalizing X

nm = norm(X, 1);

c = 1/nm;

X = c * X;

diff = X - XPREV;

nm2 = norm(diff, 2);

if (nm2 < limitval)

break;

end

XPREV = X;
```

end; % End of while loop

The above program generates the following authority vector in 20 iterations.

$\mathbf{a} = [\ 1.3639e-012, 0.0001, 0.4999, 0.0001, 0.2500, 0.2500\ ]$

This vector represents the authority ratings for the above example.

The hub ratings $\mathbf{h}$ are calculated by using the equation $\mathbf{h} = L\mathbf{a}$.

The hub ratings for the above example are given in the vector

$\mathbf{h} = [\ 0.0002, 0.0001, 0.0002, 0.9999, 0.4999, 0.4999\ ]$.

The list of the web pages in the order of authority rating scores (highest to lowest) is:

$$3, (5, 6), (2, 4), 1.$$

The list of the web pages in the order of hub rating scores (highest to lowest) is:

$$4, (5, 6), (1, 3), 2.$$

Modified HITS

Definitions

If A is an $n \times n$ matrix and σ is a permutation of $\{1, \cdots, n\}$, then the sequence of elements $a_{1,\sigma(1)}, \cdots, a_{n,\sigma(n)}$ is called the diagonal of A corresponding to σ. If σ is the identity, the diagonal is called the main diagonal.

If A is a nonnegative square matrix, A is said to have total support if $A \neq 0$ and if every positive element of A lies on a positive diagonal.

A nonnegative matrix that contains a positive diagonal is said to have support.

R. Sinkhorn and P. Knopp [SK67] prove the following theorem.

Theorem 4.1 Let A be a nonnegative $n \times n$ matrix. A necessary and sufficient condition that there exists a doubly stochastic matrix B of the form $D_1 A D_2$ where D_1 and D_2 are diagonal matrices with positive main diagonals is that A has total support. If B exists then

it is unique. Also D_1 and D_2 are unique up to a scalar multiple if and only if A is fully indecomposable.

A necessary and sufficient condition that the iterative process of alternately normalizing the rows and columns of A will converge to a doubly stochastic limit is that A has support. If A has total support, this limit is the described matrix D_1AD_2. If A has support which is not total, this limit cannot be of the form D_1AD_2.

As pointed out in [K06], the SK algorithm is perhaps the simplest method for finding a doubly stochastic scaling of a nonnegative matrix, A. It does this by generating a sequence of matrices whose rows and columns are normalized alternately. The algorithm can be thought of in terms of matrices

$$A_0 = A = D_0AE_0, A_1 = D_1AE_1, A_2 = D_2AE_2, \cdots$$

whose limit is the doubly stochastic matrix we are looking for, or in terms of pairs of diagonal matrices

$$(D_0, E_0), (D_1, E_1), (D_2, E_2), \cdots$$

whose limit gives the desired scaling of A.

Not surprisingly, the simplicity of the method has led to its repeated discovery. It is claimed to have been first used in the 1930's for calculating traffic flow [B67] and appeared in 1937 as a method for predicting telephone traffic distribution [K37]. In the numerical analysis community it is most usually named after Sinkhorn and Knopp, who proved convergence results for the method in the 1960's [SK67], but it is also known by other names, such as the RAS method [B70] and the Bregman's balancing method [LS81].

Perhaps the simplest representation of the method is given in [KK96]. Let $\mathcal{D}(\mathbf{x}) = \text{diag}(\mathbf{x})$ and suppose that $P = \mathcal{D}(\mathbf{r})A\mathcal{D}(\mathbf{c})$ is doubly stochastic. The vector $\mathbf{e}$ is a vector where all elements are equal to 1. Manipulation of the identities $P\mathbf{e} = \mathbf{e}$ and $P^T\mathbf{e} = \mathbf{e}$ gives

$$\mathbf{c} = \mathcal{D}(A^T\mathbf{r})^{-1}\mathbf{e}, \quad \mathbf{r} = \mathcal{D}(A\mathbf{c})^{-1}\mathbf{e},$$

which suggests the fixed point iteration

$$\mathbf{c_{k+1}} = \mathcal{D}(A^T\mathbf{r_k})^{-1}\mathbf{e}, \quad \mathbf{r_{k+1}} = \mathcal{D}(A\mathbf{c_{k+1}})^{-1}\mathbf{e}.$$

One can show that this iteration is precisely the SK algorithm when $\mathbf{r_0} = \mathbf{e}$. Note that this can be achieved by repeatedly issuing the commands

$$c = 1./(A' * r), \quad r = 1./(A * c)$$

in MATLAB.

The following example illustrates the application of SK-algorithm to obtain a doubly stochastic matrix. Let

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 1 & 2 \\ 2 & 5 & 1 \end{bmatrix}.$$

The initial vector $\mathbf{r}$ is set to $[1 \;\; 1 \;\; 1]$. By repeated application (in this case, only 3 iterations are needed) of the above MATLAB commands, convergence of the $\mathbf{r}$ and $\mathbf{c}$ vectors has been obtained:

$$\mathbf{r} = [1.5049 \;\; 0.8872 \;\; 0.8265]$$

$$\mathbf{c} = [0.1718 \;\; 0.1245 \;\; 0.2436]$$

Then $\mathcal{D}(\mathbf{r})A\mathcal{D}(\mathbf{c}) = \begin{bmatrix} 0.2586 & 0.3749 & 0.3665 \\ 0.4574 & 0.1105 & 0.4322 \\ 0.2840 & 0.5147 & 0.2013 \end{bmatrix}$ is the doubly stochastic matrix.

In [K06], a simple heuristic for application of the SK algorithm to the page ranking problem is presented. As in the case of HITS, two measures indicating a hub score and an authority score for each page are given by the SK algorithm. Consider the matrix L which represents the in links and out links of each web page as in the case of HITS. If we think in terms of traffic flowing around the network represented by L, then we need to balance the flow through each node. That is we need to obtain a doubly stochastic scaling of L. If node i in the unweighted graph draws traffic in disproportionately then this will have to be compensated for by r_i being relatively small. If a node has a tendency to emit traffic then

c_i will need to be relatively small. The stronger authorities will have a higher tendency to draw the traffic in and stronger hubs will have a higher tendency to emit the traffic out. The reciprocal of each of the entries in $\mathbf{r}$ gives the authority score of the corresponding node. Similarly the reciprocal of each of the entries in $\mathbf{c}$ gives the hub score of the corresponding node.

If we consider the matrix A in the above example to represent in links and out links of a network then the ordering of the authorities is:

$$3, 2, 1.$$

The ordering of the hubs is:

$$2, 1, 3.$$

5. PageRank

As mentioned in [LM05], the PageRank concept was developed by Larry Page and Sergy Brin in 1998. PageRank is central to the software for their popular search engine Google. PageRank is also a link analysis model like HITS, but it avoids some of the weaknesses of HITS.

In the search engine Google, many activities pertaining to search and ranking of retrieved pages are performed prior to query time. Initially the web pages in the World Wide Web (WWW) are retrieved by the robot crawlers. All the pages in the WWW are not retrieved by robot crawlers. It is possible to retrieve the pages from only a small portion of the WWW due to the enormous size of the WWW. An inverted file is constructed from the pages returned from the robot crawlers. The inverted file contains the list of all the possible terms occurring in the WWW. Each term is followed by the page(s) in which it occurs. The PageRank values are calculated and stored for each indexed web page. The PageRank value indicates how important or how popular a particular web page is in relation to the other web pages. When a user enters a search term the relevant pages are retrieved from the indexed file and they are ranked according to their PageRank values.

The central idea behind the PageRank is that "Important pages refer to other Important pages". As in the case of the HITS algorithm, if a web page contains a hyper-link to another page then it indicates that it endorses or recommends the page to which it links to. A page with more in links is more "important" than a page with fewer in links. However, the status of the recommending page also needs to be considered. An "endorsement" (indicated in the form of an in link) from a page having a greater number of in links has a higher value than a page with fewer in links. However if a page has many out links then it indicates that it is quite "generous" in giving endorsements. Hence, an "endorsement" from a page having more out links is of a lower value than a page with fewer out links. These ideas are encapsulated in the following equations.

Let n be the total number of web pages whose PageRank values need to be computed. Let r_i be the PageRank value of the ith web page. Then r_i is defined as:

$$r_i = \sum_{k \in B_i} \frac{r_k}{m_k} \tag{1}$$

where $B_i = \{$indices of all pages pointing to page $i\}$, and $m_k =$ (number of out links from page k). This equation is computed iteratively. Each of the n pages are assigned an initial PageRank of $\frac{1}{n}$. Let r_i^j denote the PageRank of page i at the jth iteration. That is, $r_i^0 = \frac{1}{n}$. The PageRank is then successively computed by

$$r_i^j = \sum_{k \in B_i} \frac{r_k^{j-1}}{m_k} \quad j = 1, 2, 3 \cdots \tag{2}$$

Let the vector $\boldsymbol{\pi}^T$ contain the PageRank values of the n pages successively. We define $\boldsymbol{\pi}_j^T = (r_1^j, r_2^j, \cdots, r_n^j)$. We then define a matrix P such that
$P(i,j) = \frac{1}{m_i}$ if page i links to page j, 0 otherwise, where
$m_i =$ number of out links from page i.
The PageRank defined in the equation (2) can be calculated by iteratively computing the equation

$$\boldsymbol{\pi}_j^T = \boldsymbol{\pi}_{j-1}^T P. \tag{3}$$

The PageRank vector $\boldsymbol{\pi}^T$ arising from equation (3) can be computed by using the Power Method described in Chapter 4. To find the PageRank vector $\boldsymbol{\pi}^T$ we set the following equations:

$$\mathbf{y_m} = P^T \boldsymbol{\pi}_m, \quad \nu_m = m(\mathbf{y_m}), \quad \boldsymbol{\pi}_{m+1} = \frac{\mathbf{y_m}}{\nu_m}, \quad m = 0, 1, 2, \cdots$$

This sequence converges to the right-hand dominant eigenvector of P^T. Taking the transpose of the above set of equations results in the following set of equations:

$$\mathbf{y}_m^T = \boldsymbol{\pi}_m^T P, \quad \nu_m = m(\mathbf{y_m}), \quad \boldsymbol{\pi}_{m+1}^T = \frac{\mathbf{y}_m^T}{\nu_m}, \quad m = 0, 1, 2, \cdots$$

We set the initial vector $\boldsymbol{\pi}_0^T = \{\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\}$ where n is the number of indexed web pages. $\boldsymbol{\pi}_0^T$ is in a normalized form (according to 1-norm). The above sequence is generated (using 1-norm for $m(\mathbf{y_m})$). The PageRank vector $\boldsymbol{\pi}^T$ results from the convergence of the above sequence.

According to the Power Method the following conditions need to be satisfied for the sequence to converge.

1. $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_k|$ where $\lambda_1, \lambda_2, \cdots, \lambda_k$ are the distinct eigenvalues of the matrix P. That is there should be only one eigenvalue on the dominant spectral circle.

2. $\boldsymbol{\pi}_0^T \notin R(P^T - \lambda_1 I)$, where λ_1 is the dominant eigenvalue. This condition is usually satisfied.

But, even if these conditions are satisfied, the dominant eigenvalue might have its geometric multiplicity greater than 1 and hence the sequence might not converge to a unique dominant eigenvector as in the case of HITS algorithm. The following sections explain the conditions under which the sequence converges to a unique dominant eigenvector. Most often, the matrix P must be adjusted in order to achieve this. Adjustments to the matrix P are also done to customize the rankings and adjust the convergence rate.

Markov Model and Stochastic Matrix $\overline{P}$

The "raw" matrix P is nonnegative with row sums equal to one or zero. Zero row sums correspond to pages that have no out links. These pages are also referred to as "dangling nodes". If a user reaches a dangling node, then he cannot follow any out link to get to another page. In such a case Google assumes that the user tends to randomly jump to a new page by entering its URL on the command line. Google refers to this tendency as "teleportation". Google also assumes that each URL on WWW has an equal likelihood of being selected. Hence each zero row is replaced by $\frac{\mathbf{e^T}}{n}$ where n is the number of web pages and also the order of P. After this adjustment the resultant matrix is referred to as $\overline{P}$. The matrix $\overline{P}$ is a row stochastic matrix where the sum of the elements of each row is exactly

one. Replacing the matrix P by the matrix $\overline{P}$ in equation (3) results in:

$$\boldsymbol{\pi}_j^T = \boldsymbol{\pi}_{j-1}^T \overline{P}. \tag{4}$$

The PageRank iteration in the equation (4) represents the evolution of a Markov chain on the hyper-link structure of the WWW.

A Markov chain [M00], page 687 is a stochastic process (a set of random variables $\{X_t\}_{t=0}^{\infty}$ in which each X_t has the same range $\{S_1, S_2, \cdots, S_n\}$, called the state space) that satisfies the Markov property

$$P(X_{t+1} = S_j | X_t = S_{i_t}, X_{t-1} = S_{i_{t-1}}, \cdots, X_0 = S_{i_0}) = P(X_{t+1} = S_j | X_t = S_{i_t}) \quad t = 0, 1, 2, \cdots$$

Thus the Markov property asserts that the state of the chain at the next time period depends only on the current state and not the past history of the chain. If the time is considered discretely rather than continuously, then the Markov chain is called discrete Markov chain. The state space can be finite or infinite.

Every Markov chain defines a square row stochastic matrix. The value $q_{ij}(t) = P(X_t = S_j | X_{t-1} = S_i)$ is the probability of being in state S_j at time t given that the chain is in state S_i at time $t-1$, or $q_{ij}(t)$ is called the transition probability of moving from S_i to S_j at time t. The matrix of transition probabilities $Q(t) = [q_{ij}(t)]$, where $Q(t) \in M_n(R)$, is called a transition matrix and the probabilities $\{q_{ij}(t)\}$, $0 < i < n-1$, $0 < j < n-1$ are called transition probabilities. If the transition probabilities do not vary with time then the chain is said to be stationary and the transition matrix will be the constant stochastic matrix $Q = [q_{ij}]$, where $Q \in M_n(R)$. Conversely, every stochastic matrix $Q \in M_n(R)$ defines an n-state Markov chain because the entries $q_{ij} \in Q$ define a set of transition probabilities, which can be interpreted as a stationary Markov chain on n states.

A probability distribution vector is defined to be a nonnegative vector $\mathbf{q^T} = (q_1, q_2, \cdots, q_n)$ such that $\sum_k q_k = 1$. For an n-state Markov chain, the kth step probability distribution vector is defined to be

$$\mathbf{q^T}(\mathbf{k}) = (q_1(k), q_2(k), \cdots, q_n(k)), \quad k = 1, 2, \cdots, \quad q_j(k) = P(X_k = S_j).$$

That is $q_j(k)$ is the probability of being in the jth state after the kth step, but before the $(k+1)^{st}$ step. The initial distribution vector is

$$\mathbf{q^T(0)} = (q_1(0), q_2(0), \cdots, q_n(0)), \quad q_j(0) = P(X_0 = S_j)$$

is the probability that the chain starts in S_j.

The hyper-link structure of the WWW consists of pages (or nodes) and the links among the pages (or nodes). The probability of moving from page i to page j is equal to $\frac{1}{m_i}$, where m_i is the number of out links from page i. This assumes that the user is equally likely to move to each of the pages linked by page i. The n pages can be considered as n states and p_{ij} is the probability of moving from state (page) i to state (page) j. We find that $\overline{P}$ is a transition probability matrix. It contains probabilities of moving between different states (pages) in one click.

The kth step probability distribution vector is denoted by $\mathbf{p^T(k)} = (p_1(k), p_2(k), \cdots, p_n(k))$. The ith component of $\mathbf{p^T(k)}$ denotes the probability that the user is at page i after the kth step, or after k clicks. Starting with $\mathbf{p^T(1)} = \mathbf{p^T(0)}P$ and multiplying by P at each step, it is easy to see iteratively that the kth step distribution vector is given by $\mathbf{p^T(k)} = \mathbf{p^T(0)}P^k$. The ith component of the limiting distribution vector, $\lim_{k\to\infty} \mathbf{p^T(k)}$, indicates the long-run probability that the user is at page i. If the limit $\lim_{k\to\infty} \mathbf{p^T(k)}$ exists, then it converges to the left-hand dominant eigenvector (probability distribution vector) of P [M00], page 690. That is $\lim_{k\to\infty} \mathbf{p^T(k)} = \boldsymbol{\pi}^T$. It should be noted that whether this limit converges or not, the left-hand dominant eigenvector $\boldsymbol{\pi}^T$ corresponding to the dominant eigenvalue $\rho(1)$ indicates the long-run fraction of time that the Markov chain spends in different states. Hence the ith PageRank value, $\boldsymbol{\pi}_i^T$, indicates the long-run fraction of the time spent at the ith page by a user clicking the links eternally.

PageRank Convergence and the Google matrix $\overline{\overline{P}}$

An irreducible Markov chain is one in which every state is eventually reachable from every other state. That is, there exists a path from node i to node j for all $i, j, i \neq j$.

Reducible chains contain states in which the chain eventually gets trapped. If the underlying Markov chain is reducible then the matrix $\overline{P}$ is also reducible. By reordering the columns and rows of the transition matrix $\overline{P}$, it can be reduced to the following form:

$$\overline{P} = \begin{array}{c|cc} & S_1 & S_2 \\ \hline S_1 & T_{11} & T_{12} \\ S_2 & 0 & T_{22} \end{array}$$

Once a state in S_2 has been reached the chain never reaches the states in S_1. For example: Let a page i have two in links, one from page j and one from page k. Let page i have only one out link to page j. Let page j have only one in link from page i and one out link to page i. If we reach pages i or j, then we get trapped within these two pages as there are no out links to other pages.

As the matrix $\overline{P}$ is row stochastic, $\rho(\overline{P}) = 1$, see Lemma 8.1.21 in [HJ06]. According to the Perron-Frobenius theorem, if the matrix $\overline{P}$ is nonnegative and irreducible, then $\rho(\overline{P})$ is an algebraically (and hence geometrically) simple eigenvalue of $\overline{P}$. Hence the dominant eigenvector corresponding to the dominant eigenvalue $\rho(\overline{P})$ is unique. Also according to this theorem this eigenvector is positive.

A nonnegative matrix is said to be primitive if it is irreducible and has only one eigenvalue on the dominant spectral circle. Hence, if the matrix $\overline{P}$ is primitive then it has a unique dominant eigenvector. Also in this case, the limit $\lim_{k\to\infty} \mathbf{p^T(k)}$ converges to this unique dominant eigenvector.

The matrix $\overline{P}$ is most likely reducible due to the underlying link structure of the WWW. Hence, Google imposes irreducibility on the matrix $\overline{P}$ by making every state directly reachable from every other state. Google founders reasoned that when a user gets trapped in a set of pages (states) he will exhibit "teleportation" tendency by entering a random URL on the command line and come out of the set of trapped pages (states). Each URL has equal probability of being selected. Hence they added a perturbation matrix $E = \frac{\mathbf{ee^T}}{n}$ to $\overline{P}$ to form

$$\overline{\overline{P}} = \alpha\overline{P} + (1-\alpha)E, \tag{5}$$

where α is a scalar between 0 and 1. The matrix $\overline{\overline{P}}$ is an irreducible matrix (in fact, a positive matrix) and is used in the equation (4) to compute the PageRank vector. As the matrix $\overline{\overline{P}}$ is positive, it is also primitive, see Theorem 8.5.2 [HJ06]. Hence $\overline{\overline{P}}$ has a unique dominant eigenvector. This results in a PageRank vector that is unique and positive.

Later Google modified its assumption that each URL has equal probability of being selected by adding a new perturbation matrix $E = \mathbf{e}\mathbf{v}^T$. The vector $\mathbf{v^T}$ is a positive vector and it enables non-uniform probabilities for teleporting to particular pages. This also allows Google to tweak $\mathbf{v^T}$ so that the PageRank values are modified according to commercial considerations. The irreducible, nonnegative, stochastic matrix $\overline{\overline{P}}$ is referred to as the "Google matrix" and its stationary distribution $\boldsymbol{\pi}^T$ is referred to as the real PageRank vector.

Implementation of PageRank

In the implementation of Google, PageRank is just one part of the ranking system. PageRank is combined with other scores to give an overall ranking. PageRank scores of all the documents in the indexed portion of the WWW are calculated before any query is entered. When the user enters the search term, the documents containing the search term are retrieved. This subset of the documents is called the relevancy set. The relevancy set is then sorted according to the PageRank scores combined with the other scores of each document in the set.

The computation of PageRank is a very costly and time-consuming effort as it involves finding the dominant eigenvector of an irreducible, stochastic matrix $\overline{\overline{P}}$ whose size is on the order of billions. After specifying a value for α and setting $\boldsymbol{\pi}_0^T = \frac{\mathbf{e^T}}{n}$, the following equation:

$$\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T(\alpha\overline{P} + (1-\alpha)\mathbf{e}\mathbf{v^T}) = \alpha\boldsymbol{\pi}_k^T\overline{P} + (1-\alpha)\mathbf{v}^T \tag{6}$$

is iterated until the desired degree of convergence is attained.

Let the spectrum of $\overline{P}$ be $\sigma(\overline{P}) = \{1, \mu_2, \cdots, \mu_n\}$ and the spectrum of $\overline{\overline{P}}$ be $\sigma(\overline{\overline{P}}) = \{1, \lambda_2, \cdots, \lambda_n\}$. Then $\lambda_k = \alpha\mu_k$ for $k = 2, 3, \cdots, n$ [M00], page 502, regardless of the vector $\mathbf{v^T}$. The Power Method dictates that the rate of convergence of equation (6) depends on

the rate at which $\lambda_2^k \to 0$. That is, the rate of convergence depends on the rate at which $(\alpha\mu_2)^k \to 0$. Due to the structure of the WWW the value of μ_2 tends to be quite close to 1 . Hence the rate of convergence depends on how fast $\alpha^k \to 0$. The smaller the value of α, the faster the rate of convergence. But if the value of α is small then the hyper-link structure of the web is represented less truly in the computation of PageRank. These factors are considered when the value of α is tuned.

Each of the elements of the PageRank vector is between 0 and 1. The PageRank values need to be of very high accuracy in order to distinguish between the PageRank values of billions of pages of WWW. Also PageRank values might be densely packed in some sections of the PageRank vector. This requires an accuracy of the order of 10^{-12}. But comparisons are made only among a subset of the elements of the PageRank vector. The elements of this subset might not be very highly densely packed, so an accuracy of 10^{-9} is sufficient.

Example

The following MATLAB program computes the PageRank vector for a given "raw" matrix P1. The dangling nodes must be taken care of by replacing the rows representing them with the vector $\frac{\mathbf{e^T}}{n}$. The initial vector is set to be $\frac{\mathbf{e^T}}{n}$.

```
% Input the initial vector pi.
pi = [1/6; 1/6; 1/6; 1/6; 1/6; 1/6];
pitrans =pi';
limitval = 0.0001; numiter = 500;
% Input the stochastic matrix P1.
P1 = [
   0   1/3  1/3   0    0   1/3  ;
   0    0   1/2   0   1/2   0   ;
   0    0    0   1/2  1/2   0   ;
   1    0    0    0    0    0   ;
  1/6  1/6  1/6  1/6  1/6  1/6  ;
   0    0   1/2   0   1/2   0   ;
]
```

```
% Construction of perturbation matrix
alpha = 0.85;
e = [1; 1; 1; 1; 1; 1];
n = 6;
E = ( e * e' )/ n;
P2 = alpha * P1 +(1- alpha)* E ;
% Initializing piPREVtrans
piPREVtrans = pitrans;
count =0;
% While Loop
while ( count < numiter)
count = count +1 ;
pitrans = pitrans * P2 ;
diff = pitrans - piPREVtrans;
nm2 = norm(diff, 2);
if (nm2 < limitval)
break;
end
piPREVtrans = pitrans;
end; % End of while loop
```

The above program generates the following PageRank vector in 6 iterations. $\boldsymbol{\pi}^T = [\ 0.18348, 0.10983, 0.20793, 0.14580, 0.24390, 0.10983\ ]$. The list of the web pages in the order of PageRank scores (highest to lowest) is:

$$5, 3, 1, 4, (2, 6).$$

Strengths and Weaknesses of PageRank

PageRank is a query independent measure. At run-time, the PageRank values need not be calculated as in the case of HITS. They just need to be looked up and hence the user experiences less wait time for the results of his search. The PageRank algorithm is quite spam-resistant unlike HITS. PageRank values are not derived from a local neighborhood graph as in the case of HITS. The PageRank values will increase only when they have in links from the other important pages. It is possible to add mutual links among a set of pages and try to increase the PageRank. By doing so PageRank increases slightly but the increase is inconsequential. The PageRank algorithm has the provision of "personalization" which enables Google to push the PageRank's value of a page up or down.

PageRank score is an importance score but not a relevance score as in the case of HITS. Initially, the relevant set is determined by extracting the pages that contain the search term. If these pages are sorted purely according to the PageRank values then it might be possible that some off-topic pages might be ranked higher. Some globally important pages with higher PageRank values can be part of the relevant set if they contain the search term. But these pages might be less authoritative on the particular topic. These pages get ranked higher than the authoritative but less globally important pages. Hence the PageRank might not be a true relevance measure.

References

[B67] L.M. Bergman, Proof of convergence of Sheleikhovskii's method for a problem with transportation constraints, *U.S.S.R Comput. Math. and Math. Phys.,* **1**:191-204, 1967.

[B70] M. Bacharach, Bi-proportional Matrices and Input-Output Change, *CUP*, 1970.

[BDO95] M.W. Berry, S.T. Dumais and G.W. O'Brien, Using linear algebra for intelligent information retrieval, *SIAM Rev* **37**:573:595, 1995.

[BDJ99] M.W. Berry, Z. Drmac, E.R. Jessup, Matrices, Vector Spaces, and Information Retrieval, *SIAM Rev* **41**:335:362, 1999.

[DDF90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer and R. Harshman, Indexing by latent semantic analysis, *J. American Society for Information Science* **41**:391-407, 1990.

[HJ06] R. A. Horn, C. R. Johnson, *Matrix Analysis*, 20th printing, Cambridge University Press, New York, 2006.

[K06] P. A. Knight, The Sinkhorn-Knopp Algorithm: Convergence and Applications *CERFACS Technical Report* TR/PA/06/42, Strathclyde University of Mathematics Department Technical Report 8-2006.

[K37] R. Kruithof, Telefoonverkeersrekening, *De Ingenieur* **52**:E15-E25, 1937.

[K99] J. M. Kleinberg, Authoritative Sources in a Hyper-linked Environment *Journal of ACM* **46**, 1999.

[KK96] B. Kalantari and L.Khachiyan, On the complexity of non-negative scaling, *Linear Algebra Applications* **240**:87-103, 1996.

[L06] S. J. Leon, *Linear Algebra with Applicaions*, 7th edition, Pearson Prentice Hall, New Jersey, 2006.

[LM05] A. N. Langville and C. D. Meyer, A survey of eigenvector methods of web information retrieval *SIAM Review* **47**:135-161, 2005.

[LS81] B. Lamond and N.F. Stewart, Bregman's balancing method, *Transportation Research* **15B**:239-248, 1981.

[M00] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

[SK67] R. Sinkhorn and P. Knopp, Concerning nonnegative matrices and doubly stochastic matrices *Pacific J. Mathematics* **21**:343-348, 1967.