PageRank Algorithm Implementation

Project: Numerical Linear Algebra
Indian Institute of Information Technology, Vadodara

November 16, 2015

Contents

1 Updates:

1. Nonunique Ranking(section).
2. Solution to Shortcomings(section).
3. Contribution Table.
4. References added.

2 Project Definition:

Given a few number of interlinked web pages, we need to sort the pages according to relevance with a given query. The whole idea is to create a mini search engine based on Google's Page Rank Algorithm in this project.

3 The project would broadly involve five steps

- Making and Storing web Pages
- Searching (Search Engine)
- Adjacency Matrix formation
- Rank Calculation(Eigen Analysis)
- Sorting of web pages and showing the results.

4 Team members

Members' Name	**ID**
Dilip Puri	201351014
Abhijeet Singh	201351005
Chirag Panpalia	201351001
Sonu Patidar	201351016

5 Basic Philosophy:

Just like a research paper is more important because it has lots of citations, a web page is also more important if lots of other web pages refer to it i.e. has outlink to the page. The importance of a scholarly article is more if it has citation from other important articles similarly a page with outlinks from other important pages has higher important than those which does not. Further we say that if a page has n outlinks, it distributes all of its' importance equally to all the pages it is referring to. So it is like a democritic system in which pages are voting for other pages by giving equal part of their vote to all pages which

are being referred by it. The value of vote describe importance of a page.

6 Mathematics:

Now we are all set to give a mathematical notation to the philosophy described above. Suppose a page P_i has inlinks from a subset S of pages in which a page P_j has L_j outlinks, then we can give a recursive definition of rank of page P_i as follow:

$$Rank(P_i) = \sum_{P_j \in S} \frac{Rank(P_j)}{L_j}$$

The above problem is similar to egg and chicken problem because to find out rank of page P_i I need to know rank of page P_j, now if P_i and P_j both have outlinks to each other, it becomes recursive because again to find out rank of P_j I need to know rank of P_i which is yet to be known.
Thanks to Linear Algebra that it allows us to deal with the problem. From the above equation we can always find out the relationship among the web pages to find out their ranks. Although these relations are recursive, we can represent them into matrix($AX = b$) form. Let's first create a matrix, called the hyperlink matrix, $H = [H_{ij}]$ in which the entry in the i^{th} row and j^{th} column is

$$H_{ij} = \begin{cases} 1/L_j & if P_j \in S_i \\ 0 & otherwise \end{cases}$$

H has some special properties. First, its entries are all nonnegative. Also, the sum of the entries in a column is one unless the page corresponding to that column has no links. Matrices in which all the entries are nonnegative and the sum of the entries in every column is one are called stochastic matrix or Markov matrix.
We will also form vector $I = [I(P_i)]$ whose components are PageRanks that is, the importance rankings of all the pages. The condition above defining the PageRank may be expressed as

$$I = HI$$

In other words, the vector I is an eigenvector of the matrix H with eigenvalue 1. We also call this a stationary vector of H.
Let us look at the following figure showing connections between web pages:

The H matrix for the above web is given as:

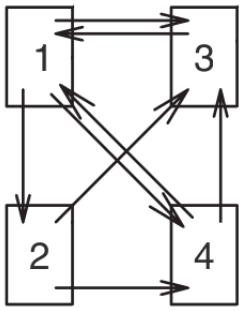

Figure 1: Graph showing connections between web pages

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

Since we know that any square matrix with all column summing to 1 has Eigen value as 1, hence we are sure to get Eigen vector corresponding to Eigen value 1. One more thing to mention here is that for Markov matrix the dominant Eigen value is always 1.
The stationary vector for Markov matrix generated for a graph shows the probability of being at a particular node, now we are looking at our internet as a graph and saying that the the stationary matrix for the matrix H is saying the probability of staying of an internet user at a particular web page. Higher the probability, there are high chances of visiting a web page by random users, since users are visiting a web page with high probability hence definitely the page is more important.
Now our problem is limited to find out the Eigen vector corresponding to Eigen value 1.

7 Shortcomings:

The above description looks fine but internet is so vast and certainly we can not apply easily what we have discussed so for. In this section we discuss two issues: webs with nonunique rankings and webs with dangling nodes.

7.1 Nonunique Rankings

For our rankings, it is desirable that the dimension of $V_1(H)$(Eigen Space corresponding to Eigen value 1) equal 1, so that there is a unique eigenvector x with $\sum_i x_i = 1$ that we can use for importance scores.
Unfortunately, it is not always true that the link matrix A will yield a unique

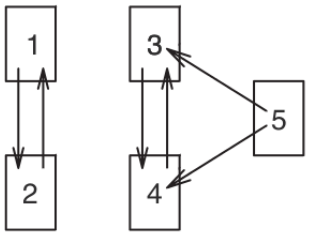

Figure 2: Web containing sub webs

ranking for all webs. Consider the web in Figure 2, the link matrix is:

$$H = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We find here that $V_1(A)$ is two-dimensional; one possible pair of basis vectors is $x = [1/2, 1/2, 0, 0, 0]^T$ and $y = [0, 0, 1/2, 1/2, 0]^T$. It is not clear which, if any, of these eigenvectors we should use for the rankings.

7.2 Dangling Nodes

A web with dangling nodes(node with no outlink) produces a matrix H which contains one or more columns of all zeros. In this case H is column-substochastic, that is, the column sums of H are all less than or equal to 1. Such a matrix must have all eigenvalues less than or equal to 1 in magnitude, but 1 need not actually be an eigenvalue for H.
In summary the matrix H is stochastic, which implies that it has a stationary vector. However, we need H to also be

1. $|\lambda_2| < 1$(for the convergence of Power Method)
2. The stationary vector has all positive entries(strictly greater than zero).

8 Solution to Shortcomings:

To find a new matrix that is both primitive and irreducible, we will modify the way our random surfer moves through the web. As it stands now, the movement of our random surfer is determined by H: either he will follow one of the links on his current page or, if at a page with no links(dangling node), randomly choose any other page to move to. To make our modification, we will first choose a parameter between 0 and 1. Now suppose that our random surfer moves in a

slightly different way. With probability α, he is guided by H. With probability $1-\alpha$, he chooses the next page at random. If we denote $A(n \times n)$ as the matrix whose entries are all one, we obtain the Google matrix:

$$G = \alpha H + (1-\alpha)(1/n)A$$

Notice now that G is stochastic as it is a combination of stochastic matrices and G has a unique stationary vector I that may be found using the power method. For the Google matrix, it has been proven that the magnitude of the second eigenvalue $|\lambda_2| = \alpha$ This means that when α is close to 1 the convergence of the power method will be very slow. As a compromise between these two competing interests, Sergey Brin and Lawrance Page, the creators of PageRank, chose $\alpha = .85$. For any $\alpha \in [0, 1]$, the matrix G is column- stochastic and $V_1(H)$ is always one-dimensional[1].

9 Implementation

In this section we will discuss about the implementation procedure that we followed in environment we worked.

- For implementation of PageRank we are using LAMP and GNU Octave.
- We are storing pages, related keywords and corresponding outlinks.
- It will take keywords from query and find related pages then fetch page id and outlinks.
- Then it will make adjacency matrix.
- Using this adjacent matrix create new link matrix.
- Check for dangling nodes then handle those nodes by using PageRank formula which is given above.
- Then it will find eigenvector correspoing to dominant eigenvalues.
- According to eigenvector it calculate rank vector.
- Show pages accordingly.

10 Contribution:

The following table gives the details of contribution of each member:

Members' Name	ID	Contribution
Dilip Puri	201351014	Implementation(coding) and Presentation
Abhijeet Singh	201351005	Implementation
Chirag Panpalia	201351001	Problem study and algorithm development for implementation.
Sonu Patidar	201351016	Problem study and algorithm development for implementation, report preparation

References

[1] "The $25,000,000,000 Eigenvector: The Linear Algebra behind Google"- Kurt Bryan, Tanya Leise.

[2] "How Google Finds Your Needle in the Web's Haystack"- David Austin, Grand Valley State University.

[3] "Google's PageRank and Beyond: The Science of Search Engine Rankings"- Amy N. Langville & Carl D. Meyer.