Project #2: The PageRank Algorithm

1 Mathematical Foundation

1.1 Introduction

The first search engines used text based ranking systems to decide which pages are most relevant to a given query: For example, if we search a particular keyword, all the engine does is to count the number of occurrences of the keyword on the page. The page with the highes occurence would be displayed first.

There where however a number of problems with this approach. A search about a common term such as "Internet" was problematic. The first page displayed by one of the early search engines was written in Chinese, with repeated occurrences of the word "Internet" and containing no other information about the Internet. Moreover, suppose we wanted to find some information about UTSA. Suppose we decided to write a web site that contains the word "UTSA" a billion times and nothing else. Would it then make sense for our web site to be the first one displayed by a search engine? The answer is obviously no. However, if all a search engine does is to count occurrences of the words given in the query, this is exactly what might happen.

The usefulness of a search engine depends on the relevance of the result set it gives back. There may of course be millions of web pages that include a particular word or phrase; however some of them will be more relevant, popular, or authoritative than others. A user does not have the ability or patience to scan through all pages that contain the given query words. One expects the relevant pages to be displayed within the top 20-30 pages returned by the search engine.

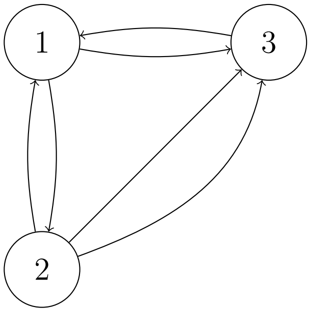
Modern search engines employ methods of ranking the results to provide the "best" results first that are more elaborate than just plain text ranking. One of the most known and influential algorithms for computing the relevance of web pages is the Page Rank algorithm used by the Google search engine. It was invented by Larry Page and Sergey Brin while they were graduate students at Stanford, and it became a Google trademark in 1998.

Page Rank is based on the following idea: the importance of any web page can be judged by looking at the pages that link to it. If we create a web page i and include a hyperlink to the web page j, this means that we consider j important and relevant for our topic. If there are a lot of pages that link to j, this means that the common belief is that page j is important. If on the other hand, j has only one backlink, but that comes from an authoritative site k, (like www.google.com, www.cnn.com, www.cornell.edu) we say that k transfers its authority to j; in other words, k asserts that j is important. Whether we talk about popularity or authority, we can iteratively assign a rank to each web page, based on the ranks of the pages that point to it.

1.2 Directed graphs

We begin by picturing the Web net as a directed graph, with nodes represented by web pages and edges represented by the links between them. For example, in the following graph, website 2 has three outgoing links (one to webpage 1 and 2 to webpage 3).

Figure 1: A directed graph.



1.3 Stochastic processes

Consider webpages $1, 2, 3$. An internet user, say Alice, starts randomly on one of the webpages. Her initial probability distribution can be described by a vector $x_0 = (p_i)_{i=1,\ldots,3}$ such that $\sum_i p_i = 1$.

Alice will then randomly choose a link from the website and click on it. To capture this information, we form a **transition matrix**:

$$A = \begin{bmatrix} 0 & \frac{L_{12}}{N_2} & \frac{L_{13}}{N_3} \\ \frac{L_{21}}{N_1} & 0 & \frac{L_{23}}{N_3} \\ \frac{L_{31}}{N_1} & \frac{L_{32}}{N_2} & 0 \end{bmatrix} \tag{1}$$

where

- L_{ij} the number of links from the j-th website to the i-th website.
- $N_j = \sum_i L_{ij}$.

Note that A has the property that **all entries of A are nonnegative and each column sums to 1.** Consider the following sequence of probability vectors:

$$x_{n+1} = Ax_n \quad (n = 0, 1, 2, \ldots) \tag{2}$$

$(x_{n+1})_i$ is the probability that Alice finds herself on the i-th webpage after n steps of random clicking.

Assuming the limit $x_\infty = \lim_{n\to\infty} x_n$ exists, the vector x_∞ "ranks" the webpages according to their importance: If the i-th webpage has the largest entry x_i, it will be the most important one. A webpage j with the next largest entry will be the second most important one etc.

2 Project Assignment

1. **Handwritten part:**

   (a) Determine the transition matrix A corresponding to the graph depicted in Figure 1.

   (b) Let $x \in \mathbb{R}^n$ be a vector $x = [p_1, \dots, p_n]^T$ with $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$. We call such a vector a vector of probabilities. Let A be a matrix with nonnegative entries and whose columns sum to 1, i.e.

   $$\sum_{i=1}^{n} a_{ij} = 1. \tag{3}$$

   Show that for any such vector, Ax is again a vector of probabilities.

   Hint: $Ax = [y_1, \dots, y_n]^T$ where $y_i = \sum_j a_{ij} p_j$. Compute $\sum_i y_i$.

2. **Coding:**

   (a) Your code should give the user the opportunity to specify the 3×3 matrix A and a starting vector x_0.

   (b) Write code that check if the user has indeed entered a valid transition matrix (nonnegative entries, columns sum to 1).

   (c) Write code that computes the sequence of of vectors

   $$x_{n+1} = Ax_n \quad (n = 0, \dots, N) \tag{4}$$

   Write a routine to verify that each of the x_n is a probability vector.

   (d) Determine x_∞ approximately and write code that compute Ax_∞. Demonstrate graphically that

   $$\lim_{n\to\infty} x_n = x_\infty \tag{5}$$

   for example by plotting $\|x_\infty - x_n\|$. **Hint:** For most A and x_0, Ax_∞ should be approximately x_∞.

(e) Write code that computes the eigenvalues and eigenvectors of A.

(f) Test your code with several transition matrices A (including the A from Figure 1) and different starting vectors. Form a hypothesis about the relationship between the vector x_∞ and the eigenvalues of A.

(g) **Optional:** Google uses the following modified transition matrix:

$$M = (1-p)A + pB \tag{6}$$

where $0 < p < 1$ is a parameter and

$$B = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{7}$$

Explain what the modification achieves (see e.g. `https://pi.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html`).

3 Deliverables

To receive credit, submit: A pdf file containing your handwritten work as well as screenshots of your code including output.