PAGERANK: AN APPLICATION OF PROBABILITY AND LINEAR ALGEBRA

1. PageRank: The Math behind Google Search

PageRank is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. It is a way of measuring the importance of website pages. PageRank works by counting the number and quality of links to a webpage to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

PageRank can be viewed from two completely equivalent perspectives – both resulting in the same set of equations. In the first formulation, the PageRank of a webpage is the sum of all the PageRanks of the pages pointing to it, together with a small decay factor.

In the original formulation of the algorithm, the PageRank of a page P_i, denoted by $r(P_i)$, is the sum of the PageRanks of all pages pointing into P_i.

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|} \tag{1}$$

Where B_{P_i} is the set of pages pointing into P_i and $|P_i|$ is the number of outlinks from P_i.

As we can see, this is a recursive definition. In order to solve this, we initiate the process with all pages having equal PageRanks and repeat this process till it converges to some final stable value. Let's consider a small example web with only 6 URLs. The bi-directional arrows indicate that these pages have links with each other. That is, Page 1 has a link to Page 3 and Page 3 links back to Page 1, in this example.

Table 1 shows the PageRank results after first few iterations of the calculation.

Table 1. First few iterations of PageRank calculations on a tiny web with 6 Urls

Iteration 0	Iteration 1	Iteration 2	Rank at Iteration 2
$r_0(P_1) = \frac{1}{6}$	$r_0(P_1) = \frac{1}{18}$	$r_0(P_1) = \frac{1}{36}$	5
$r_0(P_2) = \frac{1}{6}$	$r_0(P_1) = \frac{5}{36}$	$r_0(P_1) = \frac{1}{18}$	4
$r_0(P_3) = \frac{1}{6}$	$r_0(P_1) = \frac{1}{12}$	$r_0(P_1) = \frac{1}{36}$	5
$r_0(P_4) = \frac{1}{6}$	$r_0(P_1) = \frac{1}{4}$	$r_0(P_1) = \frac{17}{72}$	1
$r_0(P_5) = \frac{1}{6}$	$r_0(P_1) = \frac{5}{36}$	$r_0(P_1) = \frac{11}{72}$	3
$r_0(P_6) = \frac{1}{6}$	$r_0(P_1) = \frac{1}{6}$	$r_0(P_1) = \frac{14}{72}$	2

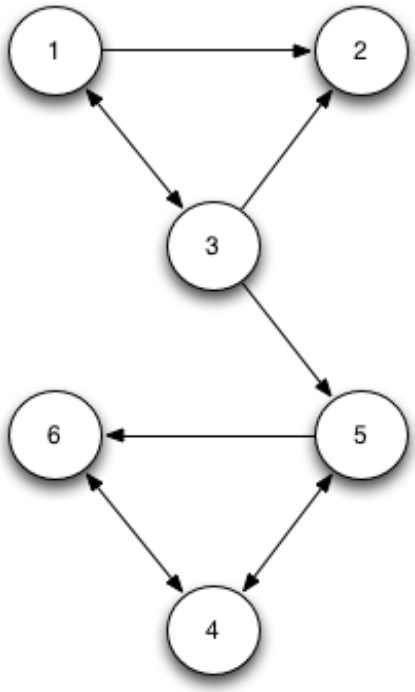

FIGURE 1. Toy Example with 6 URLs.

1.1. **PageRank - Transition Matrix.** The above iterative approach calculates the PageRank one page at a time. Using matrices, we can perform all these calculations in parallel and each iteration, compute the PageRank vector. In order to accomplish this, we introduce an $n \times n$ matrix A and an $1 \times n$ vector r which represents the PageRank of all the n pages. The A matrix is row normalized such that each element $A_{ij} = \frac{1}{|P_i|}$ if there is a link from page i to page j and 0 otherwise. Note that this can be interpreted as the probability of clicking one of the links in page i at random. For our example, the A matrix is given below

$$\mathbf{A} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2}$$

In the above matrix, the non-zero row elements correspond to the outlinks from the corresponding page whereas inlinks are given in the corresponding columns. We can now perform the iterative calculation for PageRank using matrix algebra.

Let r_i be the initial rank of the URLs. Let's set $r_i = (.167, .167, .167, .167, .167, .167)$. One can also read this as the initial probability that a user will chose one of these pages. After one click (i.e. following a link), the probability of finding a user on a page is simply

$$r_f = A \times r_i = \begin{bmatrix} 0.0556 \\ 0.139 \\ 0.0835 \\ 0.25 \\ 0.139 \\ 0.167 \end{bmatrix} \tag{3}$$

After n steps, r_f becomes:

$$r_f = (A)^n \times r_i \tag{4}$$

You can see that as we run through these iterations, the ranks of the pages converge and stabilize. The final PageRank of webpages are the solution to the equation $r = A \times r$. The PageRank of a page can be seen as simply the probability of finding a user randomly landing on a page based on the link structure of the web.

Let $r = (r_1, r_2, r_3, r_4, r_5, r_6)$ be the final probability vector. We want r such that $r = A \times r$. If you notice, this resembles an eigenvalue decomposition with $\lambda = 1$. That is, if the matrix A has an eigenvalue of 1 then we can find a solution. Does the A matrix have an eigenvalue of 1? Is it guaranteed to always have it? Under what conditions do we find an eigenvalue of 1 and is the resulting eigenvector(s) unique? If so, we can claim that there is a unique PageRank for web pages. Furthermore, we have dealt with situations where all the pages are connected. What happens if the pages are disjoint? Do we still get a PageRank? Can we even run the power iteration calculations in such cases?

1.2. **What happens if the web-pages are disconnected?** When the link structure of the web is such that pages are disconnected into disjoint sets, then the solution we get from running the power iterations (or eigen value calculations) is equivalent to considering independent universes of web graphs, each with its own PageRank ordering and no connections between the disjoint groups. We can easily verify that in a disjoint system show below.

In order to handle these types of disjoint graphs, Page and Brin came up with the concept of decay, shown in Equation 5 below. This was the insight of Page and Brin to handle the degenerate cases. It makes the transition matrix well-behaved and have rows that sum to 1. Each row represents the probability of transferring from one URL to another URL. They introduce a new matrix B which is a decayed version of the original matrix A (meaning each element of A is multiplied by a number smaller than unity) together with a uniform vector of length n which assigns some finite probability for reaching any web page at random. To make every row add up to unity, the uniform probability is properly adjusted to be $(1 - d)$ where d is the decay applied to the original transition matrix A. In the classic version by Page and Brin, they chose $d = 0.85$.

$$\mathbf{B} = 0.85 \times \mathbf{A} + \frac{0.15}{n} \tag{5}$$

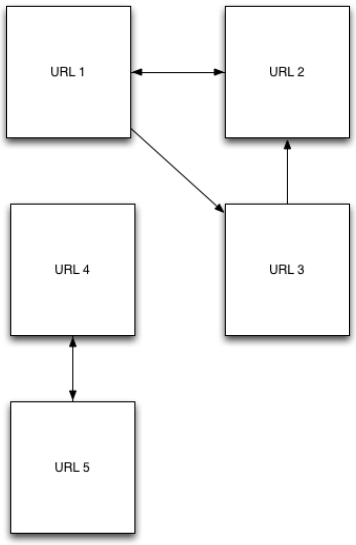

FIGURE 2. Example with 5 URLs.

This new matrix B has no breakages in the graph and it introduces weak links between all pages and running power iteration on this matrix results in a single PageRank vector for all web pages.

2. PageRank Algorithm User Model

This new matrix B can be interpreted as follows

(1) 85% of the time, users will randomly follow a link on a URL.
(2) 15% of the time, they'll randomly jump to some new URL that is not

This approach makes intuitive sense and more importantly, makes the matrix well-behaved. Now, let's revisit the eigenvalue interpretation. Does the matrix B have an eigenvalue of 1? Under what conditions is it guaranteed to have an eigenvalue of 1. If it does not have a unity eigenvalue, then our power iterations are not going to converge and we cannot calculate a PageRank.

2.1. **Perron Frobenius Theorem.**

Perron and Frobenius proved that if a square matrix M has positive entries then it has a unique largest real eigenvalue and the corresponding eigenvector has strictly positive components. All other eigenvalues of M are smaller than this largest eigenvalue. Further if M is a positive, row stochastic matrix, then:

- 1 is an eigenvalue of multiplicity one.
- 1 is the largest eigenvalue: all the other eigenvalues are in modulus smaller than 1.
- The eigenvector corresponding to eigenvalue 1 has all positive entries. In particular, for the eigenvalue 1 there exists a unique eigenvector with the sum of its entries equal to 1.

A row stochastic matrix is a matrix where any row sums to 1. The way we constructed matrix B, we can rely on Perron Frobenius theorem that B is guaranteed to have a unique eigenvalue of 1 with all positive eigenvectors which sum to 1, neatly completing the picture for us. This gives a unique ranking of web pages, or the PageRank of the web.

2.2. **PageRank Random Walk model - from Wikipedia.** If web surfers who start on a random page have an 85% likelihood of choosing a random link from the page they are currently visiting, and a 15% likelihood of jumping to a page chosen at random from the entire web, they will reach Page E 8.1% of the time. (The 15% likelihood of jumping to an arbitrary page corresponds to a damping factor of 85%.) Without damping, all web surfers would eventually end up on Pages A, B, or C, and all other pages would have PageRank zero. In the presence of damping, Page A effectively links to all pages in the web, even though it has no outgoing links of its own.

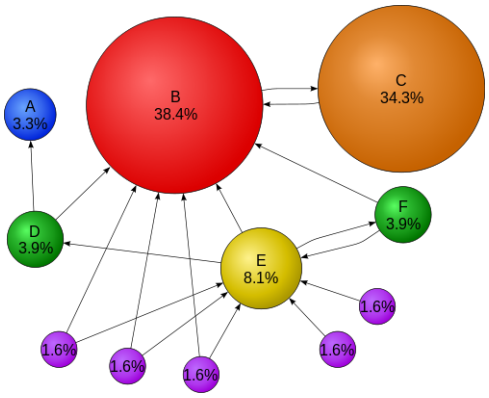

FIGURE 3. Mathematical PageRanks for a simple network, expressed as percentages. (Google uses a logarithmic scale.) Page C has a higher PageRank than Page E, even though there are fewer links to C; the one link to C comes from an important page and hence is of high value.

Final, Problem 1. 30 Points

IS 605 FUNDAMENTALS OF COMPUTATIONAL MATHEMATICS - 2014

1. Playing with PageRank

You'll verify for yourself that PageRank works by performing calculations on a small universe of web pages.

Let's use the 6 page universe that we had in the previous discussion For this directed graph, perform the following calculations in R.

- Form the A matrix. Then, introduce decay and form the B matrix as we did in the course notes. (5 Points)
- Start with a uniform rank vector r and perform power iterations on B till convergence. That is, compute the solution $r = B^n \times r$. Attempt this for a sufficiently large n so that r actually converges. (5 Points)
- Compute the eigen-decomposition of B and verify that you indeed get an eigenvalue of 1 as the largest eigenvalue and that its corresponding eigenvector is the same vector that you obtained in the previous power iteration method. Further, this eigenvector has all positive entries and it sums to 1.(10 points)
- Use the *graph* package in R and its *page.rank* method to compute the Page Rank of the graph as given in A. Note that you don't need to apply decay. The package starts with a connected graph and applies decay internally. Verify that you do get the same PageRank vector as the two approaches above. (10 points)

Final Problem 2. 40 points.

1. Go to Kaggle.com and build an account if you do not already have one. It is free.

2. Go to https://www.kaggle.com/c/digit-recognizer/overview, accept the rules of the competition, and download the data. You will not be required to submit work to Kaggle, but you do need the data.
   'MNIST ("Modified National Institute of Standards and Technology") is the de facto "hello world" dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike."

3. Using the training.csv file, plot representations of the first 10 images to understand the data format. Go ahead and divide all pixels by 255 to produce values between 0 and 1. (This is equivalent to min-max scaling.) (5 points)
4. What is the frequency distribution of the numbers in the dataset? (5 points)
5. For each number, provide the mean pixel intensity. What does this tell you? (5 points)
6. Reduce the data by using principal components that account for 95% of the variance. How many components did you generate? Use PCA to generate all possible components (100% of the variance). How many components are possible? Why? (5 points)
7. Plot the first 10 images generated by PCA. They will appear to be noise. Why? (5 points)
8. Now, select only those images that have labels that are 8's. Re-run PCA that accounts for all of the variance (100%). Plot the first 10 images. What do you see? (5 points)
9. An incorrect approach to predicting the images would be to build a linear regression model with y as the digit values and X as the pixel matrix. Instead, we can build a multinomial model that classifies the digits. Build a multinomial model on the entirety of the training set. Then provide its classification accuracy (percent correctly identified) as well as a matrix of observed versus forecast values (confusion matrix). This matrix will be a 10 x 10, and correct classifications will be on the diagonal. (10 points)

F inal Problem 3. 30 points

You are to *compete* in the House Prices: Advanced Regression Techniques competition https://www.kaggle.com/c/house-prices-advanced-regression-techniques . I want you to do the following.

Descriptive and Inferential Statistics. Provide univariate descriptive statistics and appropriate plots for the training data set. Provide a scatterplot matrix for at least two of the independent variables and the dependent variable. Derive a correlation matrix for *any* three quantitative variables in the dataset. Test the hypotheses that the correlations between each pairwise set of variables is 0 and provide an 80% confidence interval. Discuss the meaning of your analysis. Would you be worried about familywise error? Why or why not? 5 points

Linear Algebra and Correlation. Invert your correlation matrix from above. (This is known as the precision matrix and contains variance inflation factors on the diagonal.) Multiply the correlation matrix by the precision matrix, and then multiply the precision matrix by the correlation matrix. Conduct LU decomposition on the matrix. 5 points

Calculus-Based Probability & Statistics. Many times, it makes sense to fit a closed form distribution to data. Select a variable in the Kaggle.com training dataset that is skewed to the right, shift it so that the minimum value is absolutely above zero if necessary. Then load the MASS package and run fitdistr to fit an exponential probability density function. (See https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/fitdistr.html). Find the optimal value of λ for this distribution, and then take 1000 samples from this exponential distribution using this value (e.g., rexp(1000, λ)). Plot a histogram and compare it with a histogram of your original variable. Using the exponential pdf, find the 5^{th} and 95^{th} percentiles using the cumulative distribution function (CDF). Also generate a 95% confidence interval from the empirical data, assuming normality. Finally, provide the empirical 5^{th} percentile and 95^{th} percentile of the data. Discuss. 10 points

Modeling. Build some type of *multiple* regression model and **submit your model** to the competition board. Provide your complete model summary and results with analysis. **Report your Kaggle.com user name and score.** 10 points