Mathematics and Applications of PageRank

Davin Tjia, Ulysses Foote-McNabb, Nathaniel Reid

Spring 2021

Contents

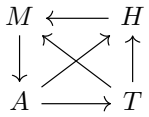
1 Introduction

In January of 1998, two Stanford graduate students published a paper titled "The PageRank Citation Ranking: Bringing Order to the Web"[1]. In this paper, the two computer science PhD candidates outlined a method to take advantage of the hyperlink structure of the World Wide Web to rank every web page using Markov Chains and Stochastic Matrices. They named this method PageRank, and, based on this model, they built a company to provide an internet search engine service from their dorm. By August of the same year, their business grew so quickly that they needed to take a leave of absence from Stanford. Since then, they never returned. Those two students, Larry Page and Sergey Brin, now have net worths of around $100 billion each, and their business has become the tech giant Google. In this paper, we are going to discuss the eigenvector worth billions. It turns out that the mathematical idea behind the algorithm that set apart Google from other fierce competitors in the early 2000s is relatively straightforward and provides some good insights for an introductory linear algebra course. We will cover the basics of the mathematics and programming of the PageRank algorithm. Once we are comfortable with the theoretical aspects of PageRank, we are going to present our application of the PageRank algorithm by ranking all 50 states of America using data on state-to-state migration flows.

2 Mathematics of PageRank

2.1 The Idea

Here we present the general and simplified version of the PageRank algorithm, as there is no update on the formula and code for the algorithm after the initial publication by the founders of Google (for obvious reasons) and some advanced modifications of the algorithm available publicly are beyond our scope. Consider each web page on the internet as a single node, and picture one web page linking to another as one node pointing an arrow at its destination. Such a link is called the *backlink* of the destination node. Let's assume the entire internet is composed of four web pages: M, A, T, and H. The following graph is one possible hyperlink structure formed by these four web pages:



Now we want to figure out which of these four web pages are relatively more important than the others, so that we can put the more relevant pages on the front page of the search result.

The main task of the PageRank algorithm is thus assigning a score to each web page for us, or Google, to rank them. For the PageRank algorithm, this

score depends on how many links are pointed to a given webpage and where those links are from. For example, a fashion blog might link to clothing stores, who then receive a boost in their rankings. The more popular the fashion blog is, the bigger the boost in rankings that the clothing retailers will receive. This is the innovation of PageRank: rather than simply tabulating the total number of links pointing to a given web page, PageRank manages to factor in the quality of the linking websites as well; the endorsement from a more influential website is worth more than a link from a random, unknown website.

2.1.1 Other real-life application

We can also apply the PageRank algorithm to other situations that can be modeled as individual nodes and weighted links between them. An example is sports, where teams are the nodes, games are the links, and results of the games give the weight to each link. In this case, the PageRank algorithm gives advantage to teams that win against other teams with better current records: given two teams with the same number of wins, PageRank would give preference to the rank of a team that won against better opponents. Furthermore, we can also incorporate margin of victory: when two teams have won against the same opponents, the better team is the one that won by a larger margin. For those who are interested in this application, the authors in [2] applied the PageRank algorithm to rank the NFL teams and provided an statistical analysis on the comparison between their ranking and the actual NFL ranking.

2.2 The Matrix of the Nodes Structure

Recall the node graph that we gave as a possible web structure of a four-page internet. Here is a simple approach to turn that link structure into a matrix: Let $m_{ij} = 1$ indicate that there is a backlink from page j to page i, and let $m_{ij} = 0$ if there is none. For our structure, let m_{1j} represent M node, m_{2j} represents A node, m_{3j} represents T node, and m_{4j} represents H node, so we have:

$$M = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

As you can see, $m_{11} = 0$, $m_{12} = 0$, $m_{13} = 1$, and $m_{14} = 1$ because page M is only cited by page T and page H but not A (and a page is not allowed to cite itself). Now, to avoid one page from becoming more influential by simply adding links to many other websites, we weight each website's *outgoing* links by the total number of outgoing links it has. If page j contains n_j links, one of which links to page k, then we will weight the contribution of the link from j to page k's score by $\frac{m_j}{n_j}$. Let $S_k \subset \{1, 2, ..., n\}$ denote the set of page k's backlinks, where n is the number of pages in this web. We have,

$$x_k = \sum_{j \in S_k} \frac{x_j}{n_j}$$

For example, page M is cited by T and H, so $S_k = \{3, 4\}$, and $n_3 = 2$, $n_4 = 1$. Therefore, $m_1 = 0 + 0 + \frac{1}{2} + 1$. Applying this changes, our matrix is now

$$M' = Q = \begin{pmatrix} 0 & 0 & \frac{1}{2} & 1 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Conveniently, implementing this strategy turns our matrix into a left stochastic matrix, meaning a real square matrix where each column sums to 1, which describes a Markov chain.

2.2.1 The Brouwer Fixed Point Theorem, the Perron-Frobenius Theorem, Stationary probability vector, and PageRank

In [3], it is proven that given a transition probability matrix P and the mapping of $v \mapsto vP$, a continuous mapping of P to itself, the Brouwer Fixed Point Theorem implies that there is a fixed point π, which is a stationary probability distribution vector, satisfying

$$\pi^T P = \pi^T$$

This result does not rule out the possibility that there is more than one such π. However, in [4], the following two propositions are proven using the Perron-Frobenius Theorem: let M be a regular stochastic matrix. (1) The matrix M has 1 as an eigenvalue of multiplicity one. (2) All the other eigenvalues λ_j have $|\lambda_j| < 1$. We therefore know that there is one, and only one, stationary probability vector π that does not change under the application of Q:

$$\boldsymbol{\pi}^T Q = \boldsymbol{\pi}^T$$

Recall the definition of eigenvalue and eigenvector. Here, $\boldsymbol{\pi}$ is just the eigenvector of Q^T with eigenvalue of 1, which is a standard problem that students who successfully complete their linear algebra course should be fluent with.

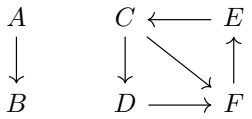
Since $\boldsymbol{\pi}$ is the probability distribution of the set of our web page $\{1, 2, ..., n\}$ ($n = 4$ in our case), the entries of $\boldsymbol{\pi}$ will provide us insight for our rankings as we will immediately see. Calculation of $\boldsymbol{\pi}$ gives that:

$$\boldsymbol{\pi} = \begin{pmatrix} 4 \\ 4 \\ 2 \\ 3 \end{pmatrix}$$

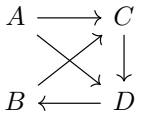
So the ranking scores for web page M, A, T, H are 4, 4, 2, 3, respectively. We then can conclude that the "importance rank" among them is: T < H < A = M. Notice how M has 2 backlinks and A has only 1 but they are tied in the ranking. This is because A receives its one backlink from an important node, M, whereas M receives its backlinks from the two least important nodes. This demonstrates how PageRank also considers where the backlinks come from, instead of just the count of backlinks.

2.2.2 Nonunique Ranking and Dangling Node

Ideally, to obtain a unique ranking, we would want $dim(E_{\lambda=1}(Q)) = 1$, where Q is the left stochastic matrix we introduced earlier. However, that is not always the case. It is not merely a coincidence that $dim(E_{\lambda=1}(Q)) \neq 1$, it turns out that we will have a nonunique ranking only if we have *disconnected* web structures such as:



Another potential issue is that sometimes the sum of our column entries is 0 instead of 1, which is called a left-*sub*stochastic matrix. Similarly, this issue is not due to chance. We will only face such difficulty when we have a *dangling node* such as:



The proper mathematical treatments dealing with these problems is beyond our scope. Fortunately, in our example of state migration flow, all of the 50 states are interconnected, so there will be no disconnected sub-structure nor dangling nodes. A unique solution to our ranking is thus guaranteed. However, if the reader is interested in exploring these two issues, please refer to [5].

2.3 Solving for Eigenvalue using Power Iteration

Now we know that to obtain a vector whose entries can be used to rank the n nodes of our web structure, we need to solve for the eigenvector with eigenvalue of 1 of the corresponding $n \times n$ matrix. This is a rather simple task if n is small, but it quickly becomes a nightmare as n gets larger, even for a computer. The PageRank algorithm is designed to rank all the web pages on the internet, so its matrix is most likely going to have millions, if not billions, of columns and rows. For this reason, we can no longer use the conventional way to solve for the eigenvector. Fortunately, there is an alternative method called *Power Iteration* to *approximate* the eignenvector that still gives a satisfactory result.

2.3.1 Dominant Eigenvector and Power Method

Let $\lambda_1, \lambda_2, ..., \lambda_n$ be the eigenvalues of a $n \times n$ matrix A. For a λ_d such that

$$|\lambda_d| > |\lambda_i|, \quad \forall i \neq d$$

we call such λ_d the dominant eigenvalue and the corresponding eigenvector the *dominant eigenvector* of A.

Next, assume $A \in R^{n \times n}$ is a diagonalizable matrix, by defintion we have

$$A = VDV^{-1}$$
$$A^k = VD^kV^{-1}$$
$$A^kV = VD^k$$

where the columns of V are the eigenvectors of A, and D is a diagonal matrix with the eigenvalues of A in its diagonal entries. Let's choose a nonzero vector $\mathbf{x} = V\mathbf{x}' \in R^n$, where $\mathbf{x}' \neq 0$, and set

$$A^k\mathbf{x} = VD^k\mathbf{x}' = \sum_{j=1}^{n} \mathbf{v}_j \lambda_j^k \mathbf{x}'_j$$

Now we let λ_1 denote our dominant eigenvector and factor it out

$$A^k\mathbf{x} = \lambda_1^k (\sum_{j=1}^{n} \mathbf{v}_j (\frac{\lambda_j}{\lambda_1})^k \mathbf{x}'_j)$$

Since $|\lambda_1| > |\lambda_j|, \forall j \neq 1$, for each $j > 1$ we have

$$\lim_{k \to \infty} (\frac{\lambda_j}{\lambda_1})^k = 0$$

It follows that

$$\lim_{k \to \infty} A^k\mathbf{x} = \lim_{k \to \infty} \lambda_1^k (\sum_{j=1}^{n} \mathbf{v}_j (\frac{\lambda_j}{\lambda_1})^k \mathbf{x}'_j)$$
$$= \lambda_1^k (\mathbf{v}_1\mathbf{x}' + \sum_{j=1}^{n} \mathbf{v}_j (0)^k \mathbf{x}')$$
$$\lim_{k \to \infty} A^k\mathbf{x} = \lambda_1^k \mathbf{v}_1\mathbf{x}'$$

As we can see, for large k, $A^k x$ is close, and nearly parallel, to $\mathbf{v}_1$. Since $\mathbf{v}_1$ is a vector in the matrix V, by definition, it is the eigenvector associated with λ_1, thus $\mathbf{v}_1$ is the dominant eigenvector.

Following from this idea, we can write code of the following form:

```
for j = 1,2,... do
    v_j = Ax_(j-1)
    x_j = v_j/||v_j||
while x_j - x_(j-1) >= epsilon
return x_j
```

to run a loop until $\mathbf{x}_j - \mathbf{x}_{j-1} < \epsilon$ for some ϵ of our choice. The resulting vector, the last $\mathbf{x}_j$ being returned, will then be our approximation for the dominant eigenvector of A. Notice that instead of $x_j = v_j$, we do $x_j = \frac{v_j}{\|v_j\|}$, i.e. we scale v_j by multiplying it with the reciprocal of its norm. This is a common practice for the power method because power iteration tends to produce vectors, v_j in

the above code example, with excessively large entries. We resolve this issue by unit-normalizing the current vector before going into the next iteration. In our code example, instead of using v_j directly, it will be the unit-normalized x_j that goes to the next iteration. In section 2.2.1, we conclude that for a stochastic matrix, there will be one and only one stationary probability vector, $\boldsymbol{\pi}$, which is given by the (dominant) eigenvector associated with the (dominant) eigenvalue of 1. Therefore, the vector computed by the power method is the approximation of $\boldsymbol{\pi}$ that we use for ranking. That is why the power method is convenient for computing PageRank.

2.4 Randomness and Modification to the Matrix

While the previous calculation seems to get the job done, it left out one crucial consideration: the average internet user's activity could be random. Think about your experience interacting with a search engine. Do you always know which specific website you are going to? Or sometimes do you just type a key word in and hope that there will be a result that interests you? Also, you might go from one website to another, and another, and eventually you end up on a web page that is completely unrelated to where you started with. To model this random behavior, the two Google founders introduced the following modification to the matrix Q in the previous section:

$$P = \alpha Q + (1 - \alpha)Y$$

where $0 \leq \alpha \leq 1$. Note that this modification is only for web structures without dangling nodes.

Here, α is a scalar quantity called the *damping factor*, which is quite important as we will discuss next. The value of α originally used by Page and Brin is 0.85. The square matrix Y is called the personalization matrix. The columns of Y are probability distribution vectors, whose entries values and calculation method are currently unknown. The purpose of this matrix is most likely to create personalized search results for each user based on their previous search activities. In the early stages of the development of Google, Page and Brin used $\frac{1}{n}$ for all entries of Y, where n is number of nodes in the web structure. Since we have little knowledge about its true mechanism and will not adjust it in our application example, we will not discuss Y more in depth here.

Now let's see how this modification affects our previous ranking. We update our matrix from Q to P using the same α and Y as Page and Brin:

$$P = 0.85\begin{pmatrix} 0 & 0 & \frac{1}{2} & 1 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} + (1 - 0.85)\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

and solve for $\boldsymbol{\pi}^T P = \boldsymbol{\pi}^T$

$$\boldsymbol{\pi} = \begin{pmatrix} \frac{106613}{81453} \\ \frac{103706}{81453} \\ \frac{40}{57} \\ 1 \end{pmatrix}$$

Using the new $\boldsymbol{\pi}$, our ranking updates to T $<$ H $<$ A $<$ M. In our case, the new ranking gives node M an ever so slight edge that breaks the tie between M and A in the previous ranking.

2.4.1 Analysis on Damping Factor

In the modification above, the most crucial component is the damping factor α. If $\alpha \in (0, 1]$, then the matrix P above is still left stochastic and $dim(V(P)) = 1$, as proven in [5]. The purpose of α is to take the randomness of the web surfer's behavior into consideration. The two extremes, where $\alpha = 0$ or $\alpha = 1$, yield $P = Y$ or $P = Q$, respectively. This implies the larger α is, the closer P is to the original structure of the nodes. As α gets smaller, randomness is being taken into consideration more until α reaches 0 where we assume total randomness and neglect our web structure. Therefore, changing α will potentially give very different rankings. Although the main purpose of α is to model random behavior, it turns out that it also has significant impact on the power method's convergence time, the time it takes to make the differences between two iterations be less than our choice of ϵ. Here, let ϵ be formally defined as the tolerance level measured by the residual:

$$\boldsymbol{\pi}^{(k)T} P - \boldsymbol{\pi}^{(k)T} = \boldsymbol{\pi}^{(k+1)T} - \boldsymbol{\pi}^{(k)T}$$

The rough estimate of the number of iterations, provided by [6], needed to converge to ϵ is

$$\frac{ln(\epsilon)}{ln(\alpha)}$$

Brin and Page reportedly claimed to successfully use only 50 to 100 iterations [6]. Using the equation above, the tolerance level ϵ set by the Google founders is estimated to range from 3×10^{-3} to 9×10^{-9}.

Moreover, in addition to affecting the convergence time of the power method, α also affects the *sensitivity* of the PageRank algorithm. That is, when perturbations to matrix P occur (possibly because the node structure is updated), to what degree do they affect $\boldsymbol{\pi}$, whose entries are used to rank the nodes? The following theorem provides a mathematical analysis on this issue:
Let $\boldsymbol{\pi}(\alpha)$ be a vector function in terms of α given by

$$\boldsymbol{\pi}^T(\alpha) = \frac{1}{\sum_{i=1}^{n} D_i(\alpha)} < D_1(\alpha), D_2(\alpha), ..., D_n(\alpha) >$$

where $D_i(\alpha)$ is the ith principal minor determinant of order $n-1$ of matrix $I-\alpha P$, where I is the $n \times n$ identiy matrix. Then,

$$\| \frac{d\boldsymbol{\pi}^T(\alpha)}{d\alpha} \| \leq \frac{2}{1-\alpha}$$

as proven in [6]. Later, the author of [7] simplified and updated the relationship and the bound to

$$\frac{\|\boldsymbol{\pi}^T - \boldsymbol{\pi}^{*T}\|_1}{\sum \boldsymbol{\pi}_i} \leq \frac{2\alpha}{1-\alpha}$$

where $\boldsymbol{\pi}^*$ is the new stationary probability vector after change of P. Also note that the $\boldsymbol{\pi}$ in the new bound is the probability vector not the function $\boldsymbol{\pi}(\alpha)$ in the old bound. For $\alpha = 0.99$, the new bound gives: $\frac{\|\boldsymbol{\pi}^T - \boldsymbol{\pi}^{*T}\|_1}{\sum \boldsymbol{\pi}_i} \leq 198$, which means that the 1-norm of difference between the previous probability vector and the updated probability vector divided by the sum of all the entries (the ranking scores) in the previous vector is less than or equal to 198. But for $\alpha = 0.85$, $\frac{\|\boldsymbol{\pi}^T - \boldsymbol{\pi}^{*T}\|_1}{\sum \boldsymbol{\pi}_i} \leq 11.3$, which is a significant improvement from $\alpha = 0.99$ even though we only decrease α value by 0.14. This makes sense because if we let the right side of equality for the new bound be a function, f, in terms of α:

$$f(\alpha) = \frac{2\alpha}{1-\alpha}$$

then the derivative of f is given by

$$\frac{df}{d\alpha} = \frac{2}{(1-\alpha)^2}$$

We can see that as $\alpha \to 1$, f' increases rapidly. So for large α, the ranking result is extremely sensitive to the perturbation of P. This is not ideal, because we do not want small changes of a relatively unimportant node to have large impact on the overall ranking. For example, say a previously unknown website receives one or two more backlinks. This does not suddenly make it a lot more popular in a global context. Therefore, although mathematically speaking, $\alpha = 0.99$ is a better modeling of the structure of the nodes than $\alpha = 0.85$, when we take the sensivity of the PageRank algorithm into consideration, $\alpha = 0.85$ actually provides a more stable and better ranking in the long run. It is thus important to consider the balance between random behavior modeling, the true structure of the nodes, required convergence time, and sensitivity of the ranking when adjusting the value of α for one's own ranking purposes.

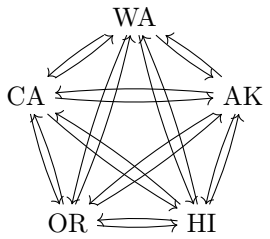
3 Application of PageRank: Ranking the Fifty States

3.1 Objective

So far, we have discussed the theoretical ideas of the PageRank algorithm. Now, we want to present our own application of the algorithm by applying the math-

ematical and programmatic framework on state-to-state migration data to rank the fifty states of the United States. Recall the sport example that we mentioned earlier. The structure of state-to-state migration data is similar to the sport team match-up results in the example. For a given year, the U.S. Census Bureau tracks the number of people who move from one state to another. If we treat this migration flow like a hyperlink graph, each state can be represented by a node and each connection represents the migration weighted by the number of people who moved.

The PageRank algorithm is advantageous because simply using the raw number of people moving will potentially leave out other important considerations. A presumably better indicator of the draw of a state is how many people are moving out from other states that are relatively more "popular". This is analogous to how a link from a more prominent website gives its target a higher weighting. We will eventually apply this model to all 50 states, but as of now, for simplicity, we will focus on states that border the Pacific: Alaska, California, Hawaii, Oregon, and Washington. This subset is a suitable representative as it includes states of different sizes and characteristics. The following graph is the node structure formed by these five states, analogous to node graph formed by the web pages in the example of section 2:



3.2 Source and Structure of the Data

Our data comes from the U.S. Census Bureau's American Community Survey (ACS) [8]. The ACS surveys millions of people each year, and uses the data it gathers in combination with the decennial official Census data to produce estimates on a variety of topics. In this case, we are focusing on state-to-state migration database, which is published annually (most recently in 2019). The relevant portion of the database is arranged into 50 columns and 50 rows, corresponding to the 50 U.S. states. Conveniently, the starting database closely matches the desired matrix structure for PageRank.

3.3 An Example using five states

To start, let's look at the smaller example with the five states subset that we picked earlier. Because we noticed that the rankings would be skewed towards states with larger populations, instead of using raw immigration numbers, we decided it would be beneficial to weight each immigration number by the sum

of the populations of the emigration and immigration states, giving us this pre-normalized M:

$$M = \begin{pmatrix} 0 & 0.00012722 & 0.00042805 & 0.00042198 & 0.00051579 \\ 0.00006398 & 0 & 0.00027059 & 0.00087666 & 0.00100385 \\ 0.00010241 & 0.00029606 & 0 & 0.00050184 & 0.00065608 \\ 0.00015304 & 0.00039907 & 0.00028930 & 0 & 0.00287990 \\ 0.00017395 & 0.00068399 & 0.00048430 & 0.00182292 & 0 \end{pmatrix}$$

where m_{1j} to m_{5j} represent Alaska, California, Hawaii, Oregon, and Washington, respectively in alphabetical order. Dividing each column of M by by its sum gives us the column stochastic matrix Q:

$$Q = \begin{pmatrix} 0 & 0.12968466 & 0.20756894 & 0.31018285 & 0.35256355 \\ 0.08445345 & 0 & 0.19654486 & 0.26492758 & 0.45407411 \\ 0.2907472 & 0.1837968 & 0 & 0.19650499 & 0.32895101 \\ 0.11646051 & 0.24194501 & 0.13850102 & 0 & 0.50309346 \\ 0.10202348 & 0.19856126 & 0.12977254 & 0.56964272 & 0 \end{pmatrix}$$

Finally we obtain our Markov chain matrix, P, that we use to calculate the ranking by using the same α, whose value is set to 0.85, and Y, a matrix with all the entries $\frac{1}{n}$ where $n = 5$ in this example, as in section 2.4:

$$P = \alpha Q + (1 - \alpha)Y$$
$$= \begin{pmatrix} 0.03 & 0.14023196 & 0.2064336 & 0.29365542 & 0.32967902 \\ 0.10178543 & 0.03 & 0.19706313 & 0.25518844 & 0.41596299 \\ 0.27713512 & 0.18622728 & 0.03 & 0.19702924 & 0.30960836 \\ 0.12899144 & 0.23565326 & 0.14772587 & 0.03 & 0.45762944 \\ 0.11671996 & 0.19877707 & 0.14030666 & 0.51419631 & 0.03 \end{pmatrix}$$

Now we can use the power iteration until convergence with a tolerance level of $\epsilon = 10^{-7}$ to determine the dominant eigenvector, whose entries will then be used to rank the five states. It takes 19 iterations to converge within our choice of ϵ:

$$x_0 = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}^T$$

$$x_{19} = \begin{pmatrix} 0.27511247 \\ 0.3623454 \\ 0.30733007 \\ 0.56670133 \\ 0.61434255 \end{pmatrix}^T = x_0 P^{19} \qquad x_{18} = \begin{pmatrix} 0.27511247 \\ 0.36234539 \\ 0.30733007 \\ 0.56670138 \\ 0.61434251 \end{pmatrix}^T = x_0 P^{18}$$

$$\|x_{18} - x_{19}\| < \epsilon \implies x_{18} \approx x_{19} \approx \boldsymbol{\pi} = \boldsymbol{\pi} P$$

Same as what we did in section 2.4, we can interpret the entries of x_{19} as the ranking scores for us determine the ranking of our five states subset in terms of immigration preferences: 1. Washington, 2. Oregon, 3. California, 4. Hawaii, 5. Alaska. So it seems that among people moving within the west coast, Washington is the most popular spot!

3.4 Programming

The five states subset example was calculated using this python code for power iteration:

```
def power_multiply(stochastic_matrix, tol):
    i = 0
    v = None
    # empty dummy vector (x_-1 is meaningless)
    x_cur = np.zeros((1,n))
    # unit column vector with all entries 1/n for x_0
    x_next = np.ones((1,n))/n
    # multiply x_i by A while the norm of the difference
    # between current and
    # previous iteration is greater error tolerance
    while norm(x_next - x_cur) >= tol:
        x_cur = x_next
        v = np.dot(x_cur,stochastic_matrix)
        x_next = v/norm(v)
        i+=1
    # returns approximate eigenvector
    # and number of iterations it took to get below tolerance
    return x_next, i
```

With this code, we can scale the method up to the 2019 immigration data for all 50 states. This also gives us an opportunity to programatically test α values to verify its effects on the computational cost, accuracy, and stability of the power method.

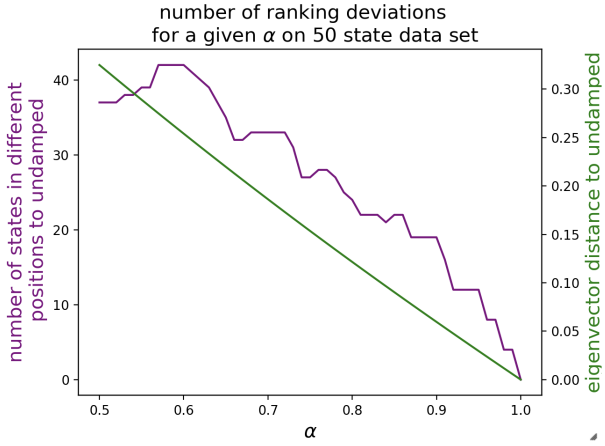
Following is the code used to generate stochastic matrices with arbitrary alpha values:

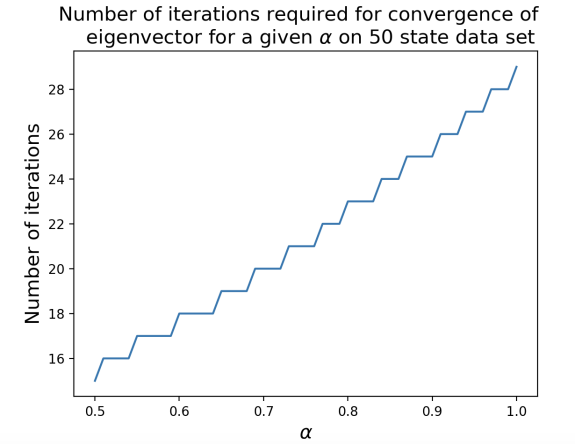
```
def normalize(matrix, alpha):
    # divides each row (we are operating on the matrices in transpose)
    # by its sum, making it stochastic
    stochastic = np.array(\[row/sum(row) for row in matrix\])
    # normalizing with the uniform stochastic matrix
    # multiplied by given alpha value
    damped_stochastic = alpha*stochastic + np.ones((n,n))*((1-alpha)/n)
    return damped_stochastic
```

3.5 Result and Discussion

3.5.1 Effects of the Damping Factor: Theoretical vs. Experimental

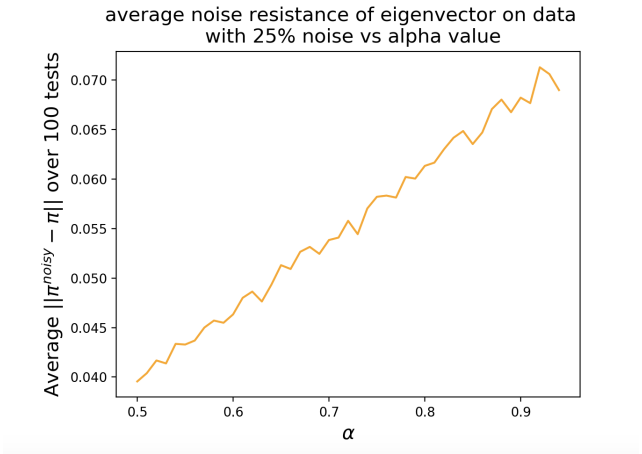
Running experiments with α values from 0.5 to 1 (no damping) on the full data set gives the following results:



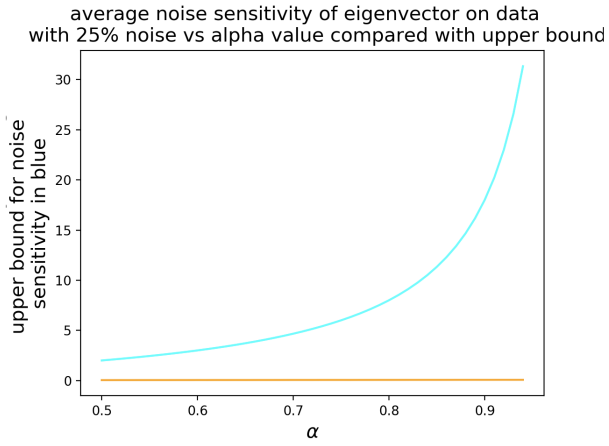


As we can see in the first graph, changing the α value indeed affects the final ranking, confirming our previous claim. For example, changing α value from 1 to 0.6 will cause more than 40, or 80% of, states to switch positions in the two ranking outcomes. Furthermore, because the number of iterations required for convergence for a 50 by 50 matrix is so insignificant, the rough estimate for number of required iterations, $\frac{ln(\epsilon)}{ln(\alpha)}$, is mostly inaccurate in our case. Referring to the second graph, you can see that if we double the α value from .5 to 1, it only increases the convergence time by 86.7%, while the theoretical estimation predicts a massive 6796.8% surge in run time.

We also ran experiments to test the effects of the different choices of α on noise resistence. To do this, for a given alpha value, we computed the dominant eigenvector of 100 matrices where each was multiplied by a random factor between 0.875 and 1.125. To calculate the sensitivity to this perturbation, we took the one norm of difference between the true eigenvector (where the matrix is multiplied by 1) for a given alpha, and the other 100 eigenvectors calculated with the noisy matrices. The resulting errors were as follows:



This outcome matches with our previous prediction in section 2.4.1 where we claimed that the algorithm is more sensitive (the average $\|\pi^{noisy} - \pi\|$ increases) as the α approaches 1. In the same section, we also stated that the sensitivity to the perturbation of PageRank algorithm is bounded by $\frac{2\alpha}{1-\alpha}$. The following graph confirms the validity of the bound:



where the function in the color cyan represents the bound as $\alpha \to 1$ and the

function in color orange is the changes, $\frac{\|\pi^{noisy} - \pi\|}{\sum \pi_i}$, due to senstivity.

We want to note that the actual relationship between alpha value and resistance to noises seems to be linear for our relatively smaller data set, so a lower alpha value (below 0.85) would not be particularly advantageous in this respect.

3.5.2 The Ranking

Finally, we are to present the culmination of this project: the ranking of the fifty states based on state-to-state migration data using PageRank algorithm.

the Ranking of Fifty States			
Rank	State Name	Rank	State Name
1	New York	26	Iowa
2	Colorado	27	Idaho
3	Virginia	28	South Carolina
4	Florida	29	New Hampshire
5	Massachusetts	30	Alaska
6	Illinois	31	Montana
7	California	32	Wisconsin
8	North Carolina	33	Michigan
9	Washington	34	Kentucky
10	Pennsylvania	35	Nebraska
11	Texas	36	Hawaii
12	New Jersey	37	Louisiana
13	Maryland	38	Oklahoma
14	Georgia	39	Alabama
15	Arizona	40	New Mexico
16	Missouri	41	Wyoming
17	Tennessee	42	Maine
18	Minnesota	43	South Dakota
19	Connecticut	44	North Dakota
20	Nevada	45	Arkansas
21	Oregon	46	Mississippi
22	Utah	47	Rhode Island
23	Ohio	48	West Virginia
24	Indiana	49	Vermont
25	Kansas	50	Delaware

References

[1] Larry Page and Sergey Brin. *The PageRank Citation Ranking: Bringing Order to the Web.* Stanford, California, 1998.

[2] Laurie Zach, Ron Lamb, and Sarah Ball. *An application of Google's PageRank to NFL rankings.* Involve- a journal of mathematics, Vol. 5, No.3: 463-471, 2012

[3] Markov Chains: Basic Theory
http://galton.uchicago.edu/ lalley/Courses/383/MarkovChains.pdf

[4] R. Clark Robinson. *Perron-Frobenius Theorem.* Northwestern, Illinois, 2002

[5] Kurt Bryan and Tanya Leise. *The $25,000 Eigenvector: The Linear Algebra behind Google.* Society for Industrial and Applied Mathematics, 2006.

[6] Amy N. Langville and Carl D. Meyer. *Deeper Inside PageRank* International Mathematics Vol. 1, NO.3 : 335-380, 2003/2004

[7] Monica Bianchini, Marco Gori, Franco Scarselli. *Inside PageRank.* University of Siena, Italy, 2003

[8] *State-to-State Migration Flows.* U.S. Census Bureau, 2019.
https://www.census.gov/data/tables/time-series/demo/geographic-mobility/state-to-state-migration.html