UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

LEONARDO RANGEL DE ALBUQUERQUE

Pagerank: uma visão de álgebra linear

Campinas
24/11/2023

LEONARDO RANGEL DE ALBUQUERQUE

Pagerank: uma visão de álgebra linear

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. Paulo José da Silva e Silva.

Resumo

Ao final da década de 90, a Google obteve grande sucesso na Internet como um site de busca graças a um algoritmo, chamado de PageRank, que se baseia, surpreendentemente, nos conceitos de autovalor e autovetor de matrizes. Por outro lado, observa-se que os alunos de graduação da Unicamp não valorizam suficientemente esses conceitos introduzidos no curso de Álgebra Linear. Este trabalho visa então apresentar os conceitos matemáticos fundamentais do PageRank para alunos de graduação por meio de um artigo expositivo e iterativo para aumentar o seu interesse por conceitos tão importantes. Inicialmente faremos uma breve introdução a cadeias de Markov que serão usadas para modelar o comportamento de usuários na Internet por uma matriz especial. Em seguida, veremos que esse comportamento leva naturalmente a um algoritmo cômputo de autovalores e autovetores dessa matriz que revelam naturalmente o Rank, ou importância, das páginas visitadas.

Abstract

By the end of the 90's, *Google* had a great success as a search site on the Internet thanks to an algorithm called PageRank which bases itself, surprisingly, on the concepts of eigenvalues and eigenvectors of matrices. On the other hand, it's observed that Unicamp undergraduates students don't give the necessary value to these concepts in their Linear Algebra course. This project has as it's main objective to present the mathematical fundamental concepts of PageRank to undergraduates via an expositive and interactive article to strengthen their interest in these important concepts. Firstly, we'll make a brief introduction of Markov chains that will be used to model the behavior of an Internet user according to a special matrix. Following that, we'll see that this behavior leads naturally to an algorithm for computing eigenvalues and eigenvectors of this matrix that naturally reveals the Rank, or importance, of visited pages.

Conteúdo

1 Introdução

O PageRank é um algoritmo que, dado um conjunto de páginas, utiliza a estrutura dos *links* entre elas. O sucesso de sua implementação na Internet do fim da década de 90 ao começo dos anos 2000, foi um dos grandes responsáveis por fazer da *Google* uma das grandes empresas tecnológicas do mundo. O PageRank, por sua vez, utiliza a álgebra linear, com um enfoque maior no conceito de autovalor e autovetor, tanto para calcular *ranks* de páginas da Internet, quanto para justificar teoricamente porque o algoritmo "funciona". Devido a isso, esta pesquisa foi feita com o intuito de introduzir e explicar, de maneira intuitiva, o PageRank para alunos de graduação em cursos de exatas, observarem uma aplicação prática de álgebra linear, e, principalmente do problema de autovalor-autovetor, gerando, assim, um maior interesse pela matéria e seu conteúdo.

2 Metodologia

Para, inicialmente, aprender a respeito do PageRank de uma forma matematicamente aprofundada, foi utilizado o livro *Google's PageRank and Beyond: The Science of Search Engine Rankings* de Carl D. Meyer e Amy Langville Carl D. Meyer [2011]. Após o estudo completo do livro, foi também realizada uma leitura complementar do artigo científico *The $25,000,000,000 Eigenvector: The Linear Algebra Behind Google* criado por Kurt Bryan e Tanya Leise Kurt Bryan [2006] para ter a experiência de conhecer o PageRank por outra abordagem.

Após concluir o estudo do assunto, foi desenvolvido um artigo expositivo e interativo, no formato *notebook*, para atingir o objetivo principal da pesquisa. Nele, introduzimos o PageRank de maneira intuitiva para o leitor, abordando os principais conceitos, muitos da área da álgebra linear, para que o leitor possa adquirir uma noção matemática básica por trás do algoritmo.

No final do artigo, há códigos interativos na linguagem Python. Neles, o leitor pode aplicar o método, abordado previamente, para calcular o PageRank de um conjunto de páginas. Além disso, há a possibilidade da criação de um conjunto de páginas a partir de uma página inicial da Internet fornecida pelo próprio leitor.

Os códigos desenvolvidos no texto foram traduzidos a partir do livro Carl D. Meyer [2011]. Nele, os algoritmos são originalmente escritos na linguagem MATLAB. Assim, conforme previsto no cronograma de atividades, houve a tradução dos mesmos da linguagem MATLAB para Python.

3 Discussão

Criar um texto "totalmente novo"com o objetivo de dissertar a respeito do PageRank nesta monografia seria redundante, visto que o artigo expositivo já cumpre esta função. Portanto, para abordar o problema do PageRank, está escrito aqui uma versão adaptada, para a monografia, do artigo criado na pesquisa. Ele, por sua vez, pode ser encontrado em **https://github.com/rangelalbuq/PageRank**.

3.1 Cadeias de Markov

Inicialmente, discutiremos a respeito de cadeias de Markov, visto que utilizaremos seus conceitos ao longo do texto.

De uma forma breve, uma variável aleatória é uma função que associa resultados de um evento a valores. Definindo de uma forma um pouco mais formal, seja Ω o conjunto de todos os possíveis resultados que podem acontecer em um evento, uma variável aletória X é uma função $X : \Omega \to E$, em que E é um espaço mensurável. A definição precisa do que é um espaço mensurável, porém, vai além dos propósitos deste texto. Para nós, basta saber que ele é um espaço no qual conseguimos definir uma noção de probabilidade de seus subconjuntos. Como, para os fins deste texto, E será sempre um conjunto finito, isso é sempre possível de ser feito.

Por exemplo, o lançamento de uma moeda possuí dois possíveis resultados, pode sair cara, ou sair coroa. Assim, temos que $\Omega = \{\text{cara}, \text{coroa}\}$. Podemos definir uma variável aleatória $X : \Omega \to \{0, 1\}$ de tal modo que caso o resultado do lançamento da moeda for cara, $X(\text{cara}) = 1$, e se caso sair coroa, $X(\text{coroa}) = 0$. Deste modo, $P(X = 1) = \frac{1}{2}$ se refere ao fato da probabilidade de sair cara do lançamento ser $\frac{1}{2}$ (o significado de $P(X = 0) = \frac{1}{2}$ é análogo).

Variáveis aleatórias podem também ser usadas para afirmar a "localização" de algo em um espaço. Por exemplo, seja A um espaço que contém vários estados A_i, $i = 1, \ldots, n$. A equação $X = A_i$ pode ser usada em certos contextos para dizer que X *está* em A_i e, consequentemente, $P(X = A_i)$ se referir à probabilidade de X *estar* em A_i. Podemos, ainda por cima, falar de *quando* alguém está em A_i introduzindo um índice que indique tempo na variável aleatória X. Assim, por exemplo, a equação $X_t = A_i$ pode indicar que X está em A_i no tempo t e $P(X_t = A_i)$ se referir a probabilidade de X estar em A_i no tempo t. Caso fossemos, nesses casos, definir X formalmente, poderíamos dizer que X é uma função identidade de modo que $X : A \to A$, com $X(A_i) = A_i$. Porém, como é "chato" escrever $X(A_i) = A_i$, dizemos apenas $X = A_i$.

Um *processo estocástico* é um conjunto de variáveis aleatórias $\{X_t\}_{t=0}^{\infty}$ que possuem um conjunto de possíveis valores em comum $S = \{S_1, S_2, \ldots, S_n\}$, que é chamado de *espaço de estados* do processo. O parâmetro t é geralmente pensado como tempo, e X_t representa o estado do processo no tempo t. Por exemplo, considere o processo de navegar na Internet clicando em links que vão de uma página a outra. O espaço de estados S é o conjunto de todas a páginas da Internet e a variável aleatória X_t é a página visitada no tempo t.

Uma *cadeia de Markov* é um processo estocástico com a propriedade de ser um processo "sem memória". A probabilidade da cadeia ir para algum estado depende somente do estado atual que ela se encontra e não dos anteriores. Lidaremos neste texto com o tipo específico de uma *cadeia de Markov discreta*, isto é, uma cadeia o qual "anda" com passos discretos através do tempo. Escrevendo isso em forma de equação temos,

$$P\left(X_{t+1} = S_j | X_t = S_{i_t}, X_{t-1} = S_{i_{t-1}}, \ldots, X_0 = S_{i_0}\right) = P\left(X_{t+1} = S_j | X_t = S_{i_t}\right)$$

para cada $t = 0, 1, 2, \ldots$. A notação $P(E|F)$ representa a probabilidade condicional do evento E ocorrer dado que F ocorreu. Neste texto, iremos considerar que o processo de um usuário navegar na Internet é uma cadeia de Markov, ou seja, o fato dele ir de uma página a outra dependerá somente da página que se encontra.

A *probabilidade de transição* $p_{ij}(t) = P\left(X_{t+1} = S_j | X_t = S_i\right)$ é a probabilidade da cadeia ir ao estado S_j dado que ela se encontra, no tempo t, no estado S_i. Por exemplo, pensando no contexto de páginas da Internet, $p_{7,1}(2)$ é a probabilidade de um usuário ir à página 1 dado o fato dele estar na página 7 no tempo $t = 2$. Escrevemos p_{ij} em função de t visto que a probabilidade de ir, a partir de S_i, para S_j pode variar dependendo do tempo t em que a cadeia se encontra. Caso uma cadeia fosse de Markov e as probabilidades de transição não variassem com o tempo, isto é $p_{ij}(t) = p_{ij}$, diríamos que essa cadeia é uma *cadeia de Markov estacionária*.

Um *vetor de distribuição de probabilidade* (ou apenas "vetor de probabilidade") é um vetor cujas componentes são probabilidades, ou seja, são números não negativos que somam 1. Assim, $p^t(t) = (p_1(t), p_2(t), \ldots, p_n(t))$ é não-negativo e $\|p(t)\|_1 = 1$. Analogamente a uma cadeia de Markov estacionária, um *vetor de distribuição de probabilidade estacionário* é um vetor de probabilidade que não varia com o tempo, isto é, $p(t) = p$.

Uma *matriz de probabilidade de transição* $P(t)$ é definida como sendo uma matriz que tem como linhas, vetores de probabilidade $p(t)$. Assim, ela é não-negativa, e a soma dos elementos de cada linha deve ser 1. Matrizes que satisfazem essas duas propriedades, são chamadas de *matrizes estocásticas*. Portanto, dado um tempo t específico, $P(t)$ é uma matriz estocástica.

Agora que abordamos alguns novos conceitos de cadeias de Markov, vamos começar a trabalhar no problema do *PageRank*. Para conseguir dar uma classificação a páginas da Internet, precisamos, primeiro, saber como um usuário "da vida real" navega. Assim, nosso primeiro objetivo será desenvolver um modelo que simule a *caminhada* de um usuário fictício num conjunto dado de páginas da Internet.

3.2 A Matriz Google

Para começarmos, suponha um conjunto P com n páginas da Internet dadas por P_i $(i = 1, 2, \cdots, n)$. Suponha também que as páginas desse conjunto possuem *links* que vão para páginas do mesmo conjunto. Uma forma interessante de visualizar as páginas de P e as ligações entre elas é por meio de um grafo. Por motivos didáticos, usaremos um conjunto P com $n = 7$ páginas dadas pelo grafo abaixo.

Na figura 1, os nós (círculos) representam as páginas e as arestas (setas) representam as ligações entre as páginas.

Podemos representar esse grafo em um formato matricial. Seja A uma matriz tal que $a_{ij} = 1$, se a página i possui um link para a página j, e $a_{ij} = 0$ caso contrário (a página i não possui um *link* para a página j). Acabamos de criar uma chamada matriz de *Adjacência* do grafo. Assim, a matriz A do grafo do conjunto de páginas P é dada por

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Agora, vamos olhar para a matriz A de uma forma diferente. E se o elemento

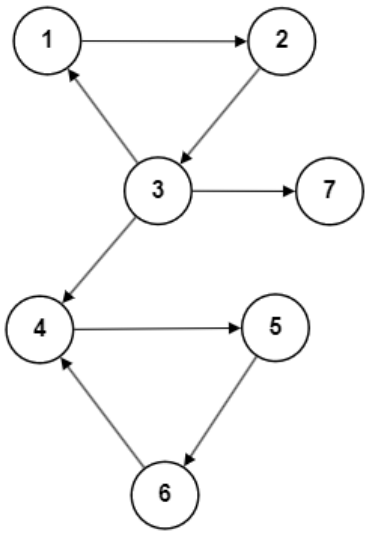


Figura 1: Grafo do conjunto P

a_{ij} representasse a probabilidade de um usuário da Internet ir à página j considerando o fato dele estar, no momento, na página i? Observando A, vemos que essa interpretação nova não está consoante com a matriz e que um problema já visível está em sua linha 3. Segundo nossa interpretação, se um usuário estiver na página 3, a probabilidade dele ir para página 1 é igual à 1. Porém, a chance dele ir para as páginas 4 e 7 também é 1, algo que não faz sentido. Contornaremos esse problema criando uma *nova* matriz que tenha a mesma "cara" de A e que também, corresponda com essa nova interpretação probabilística.

Um modo de criar essa nova matriz, digamos H, de forma que, as probabilidades sejam "justas" ou "sem viés" é pela seguinte definição: o elemento h_{ij} é igual a $(\sum_{k=1}^{n} h_{ik})^{-1}$ se a página i possui um *link* para a página j e $h_{ij} = 0$ caso contrário. Embora pareça um pouco complicado essa nova definição, saiba que a única diferença entre ela e a de A é que estamos "normalizando" as linhas não-nulas para que a soma de seus elementos seja igual a 1 e assim, faça sentido pensar nela probabilisticamente. Deste modo, H é dada por,

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

A matriz H é uma chamada *matriz subestocástica*, visto que quase todas suas linhas somam a 1, exceto as nulas. Com essa modificação, temos que cada elemento de H, com excessão dos da linha nula, é uma *probabilidade de transição* p_{ij} entre páginas, fazendo

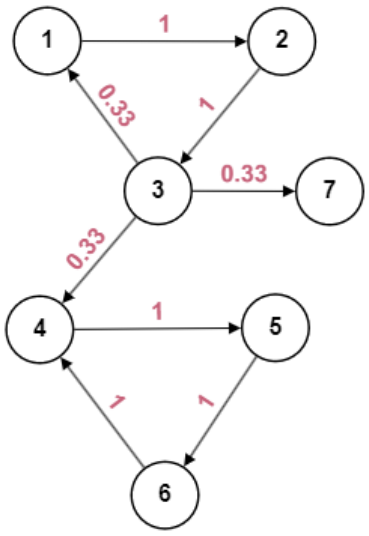


Figura 2: Grafo representando as probabilidades de transição de H

com que H seja "quase" uma *matriz de probabilidade de transição*. O grafo da figura 2 representa visualmente as probabilidades de transição entre as páginas de H.

Para facilitar a interpretação, vamos dar o nome de Paulinho ao "usuário fictício" que caminha nas páginas de P. Assim, no momento, ele está seguindo as probabilidades da matriz H para caminhar entre as páginas.

Observando a matriz ou seu grafo, é fácil notar um comportamento de "parada" de Paulinho. Caso ele chegue na página 7, não há a possibilidade dele sair dela, fazendo com que Paulinho fique nela **para sempre**. É razoável inferir que isso é algo indesejado em nosso modelo, que a página 7 seja um "buraco negro" para Paulinho, em que, se chegar lá, viverá para sempre.

Esse comportamento se deve ao fato de H ter a linha 7 completamente nula. Portanto, faremos o seguinte para contornar esse fato: se uma linha contiver apenas zeros, ela será alterada de forma que, todos seus elementos sejam iguais à $\frac{1}{n}$, em que n é o número de páginas de P. O que isso interpretativamente faz é que caso Paulinho chegue a uma página que não possua ligação alguma com outra, ele se direcionará a alguma página qualquer de P. É como se ele escolhesse uma página do seu histórico de navegação "aleatoriamente".

Assim, vamos atualizar a matriz H dando um novo nome a ela: S. Aplicando a modificação, temos que S é dada por

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \end{bmatrix}.$$

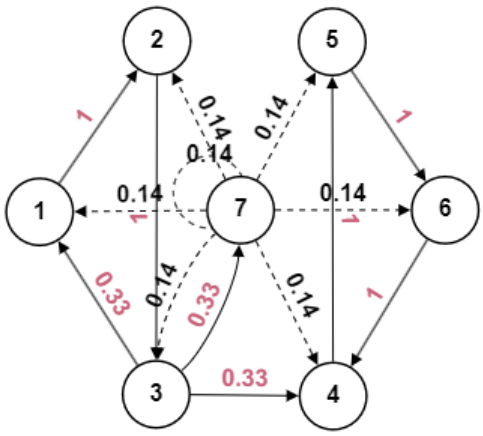


Figura 3: Grafo representando as probabilidades de transição de S

Com nossa atualização, S agora é "totalmente" uma *matriz de probabilidade de transição estocástica* com cada elemento seu s_{ij} sendo uma *probabilidade de transição* entre páginas. É possível visualizar as probabilidades de transição entre as páginas de S pelo grafo da figura 3.

Porém, ainda há mais um problema com nosso modelo. Se, você leitor, observasse o comportamento de Paulinho na matriz S por um tempo suficientemente grande, provavelmente iria notar a seguinte propriedade: se Paulinho estiver na página 4, ele irá para a página 5. Se estiver na página 5, ele irá para a página 6. E se estiver na página 6, ele irá para a página 4. E assim por diante, para sempre. Criando assim um *loop* eterno de navegação. E com isso, a partir do momento em que entra pela primeira vez em uma dessas páginas, as outras (P_i com $i = 1, 2, 3, 7$) efetivamente "não existirão" mais para Paulinho. De novo, é razoável de se pensar que esse comportamento observado é indesejado para nosso modelo, visto que as pessoas da "vida real" na Internet não navegam em ciclos periódicos eternos entre páginas.

Para que o fato discutido não ocorra, consideraremos mais uma, e última modificação no comportamento de Paulinho. Agora, antes de simplesmente selecionar um *link* na página que atualmente se encontra, ele terá uma probabilidade $1 - \alpha$ (com $\alpha \in (0, 1)$) de ir para uma página qualquer de P. Isso, além trazer propriedades que garantirão o êxito de nosso modelo, algo que veremos posteriormente, ela também traz a ele um comportamento esperado de qualquer um que navega a Internet. É razoável esperar de uma pessoa que ela não somente vá sendo levada site a site seguindo apenas os *links* da página em que se encontra. Ela também pode, por exemplo, entrar em algum site em que a aba está aberta em seu navegador, ou também, abrir um de seu histórico por livre e espontânea vontade.

Seja o vetor $e \in \mathbb{R}^n$ um vetor coluna com todas entradas iguais a 1

$$e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

A nova matriz criada a partir de S será a chamada *matriz Google* G que é dada pela seguinte equação:

$$G = \alpha S + (1 - \alpha)\,\frac{1}{n} e e^T.$$

Em que $\frac{1}{n} e e^T \in \mathbb{R}^{n \times n}$ é uma matriz de "teleportação aleatória", o qual todos seus elementos são iguais à $\frac{1}{n}$. Em nosso exemplo, escolhendo $\alpha = 0.85$, a matriz G é

$$G = 0.85 \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \end{bmatrix} + 0.15 \begin{bmatrix} 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \end{bmatrix}.$$

Note que, que nem a matriz S, G representa uma *matriz de probabilidade de transição estocástica*, com cada elemento seu g_{ij} sendo uma *probabilidade de transição* entre páginas. Caso fosse criado um grafo para visualizar as probabilidades de transição de G, ele seria um tanto quanto "poluído", visto que segundo a matriz G, para cada página de P há uma ligação para todas páginas. O grafo de S já possui um grau de poluição visual notável, agora imagine se tivesse um para G!

Agora que desenvolvemos uma *matriz de probabilidade de transição* que simula, de forma razoável, o comportamento de um usuário qualquer da Internet, vamos encontrar um jeito de medir a *importância* de cada página utilizando esse modelo.

3.3 O Vetor do PageRank

A filosofia geral por trás do algoritmo do PageRank pode ser resumida na seguinte frase: *uma página é importante se páginas importantes direcionam a ela.* Vamos, até o final dessa subseção, chegar na equação o qual define o problema geral e, por si só, representa esta filosofia.

Primeiramente, começaremos "chutando"uma fórmula, e, a partir dela, chegaremos no nosso objetivo. Uma possível fórmula poderia simplesmente ser a soma dos PageRanks das outras páginas que apontam para ela. Assim sendo,

$$r(P_i) = \sum_{P_j \in B_{P_i}} r(P_j),$$

em que $r(P_i)$ é o PageRank da página i, $r(P_j)$ o da página j e B_{P_i} é o conjunto das páginas que apontam para a página i. Como cada $r(P_i)$ é um *rank*, temos que $r(P_i) > 0$.

Para que o valor de algum $r(P_i)$ não "exploda", será também imposto a condição de que $\sum_{i=1}^{n} r(P_i) = 1$.

Porém, se questione do seguinte: imagine duas páginas que possuem o mesmo PageRank. Uma dessas páginas possui apenas um link para uma página qualquer, enquanto a outra possui links para cem páginas. A "importância" do link da primeira página deve ser o mesmo que a "importância" de cada link da segunda?

Para os criadores do Google, a resposta é não. Quanto mais links uma página possui para outras páginas, a "importância" dada a cada um desses links deve valer menos. Portanto, para a fórmula do PageRank, é preciso ter um fator de peso que mede o quão "expressivo" é um link.

Deste modo, vamos utilizar a seguinte fórmula:

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

em que $|P_j|$ é o número de links de P_j.

O problema óbvio com essa fórmula é que simplesmente não sabemos os valores de $r(P_j)$ para calcular $r(P_i)$. A forma utilizada para lidar com isso será aplicar a fórmula sucessivamente para as páginas de P, utilizando, a cada nova iteração, os valores obtidos para $r(P_j)$ da iteração prévia e "torcer" para que os valores de $r(P_j)$ convirjam para algo *estável* após "várias" iterações. Assim, introduzindo uma nova notação, temos

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|},$$

em que $r_{k+1}(P_i)$ representa a $k+1$-ésima iteração do *rank* de P_i e $r_k(P_j)$ representa a k-ésima iteração do *rank* de P_j.

Para que a fórmula acima "funcione", é preciso saber quem são os $r_0(P_i)$, $i = 1, \ldots, n$. Como inicialmente não conhecemos nada sobre as páginas P_i, é razoável pensar que todas, no começo do processo, valem o mesmo. Portanto a condição inicial dos $r_k(P_i)$ será $r_0(P_i) = \frac{1}{n}$ para $i = 1, \ldots, n$.

Vamos ilustrar o funcionamento dessa fórmula utilizando o mesmo conjunto P de páginas da figura 1 da subseção "A Matriz Google".

Calculando apenas a primeira iteração para cada página, temos:

$$r_1(P_1) = \frac{r_0(P_3)}{|P_3|} = \frac{\frac{1}{7}}{3} = \frac{1}{21};$$

$$r_1(P_2) = \frac{r_0(P_1)}{|P_1|} = \frac{\frac{1}{7}}{1} = \frac{1}{7};$$

$$r_1(P_3) = \frac{r_0(P_2)}{|P_2|} = \frac{\frac{1}{7}}{1} = \frac{1}{7};$$

$$r_1(P_4) = \frac{r_0(P_3)}{|P_3|} + \frac{r_0(P_6)}{|P_6|} = \frac{\frac{1}{7}}{3} + \frac{\frac{1}{7}}{1} = \frac{4}{21};$$

$$r_1(P_5) = \frac{r_0(P_4)}{|P_4|} = \frac{\frac{1}{7}}{1} = \frac{1}{7};$$

$$r_1(P_6) = \frac{r_0(P_5)}{|P_5|} = \frac{\frac{1}{7}}{1} = \frac{1}{7};$$

$$r_1(P_7) = \frac{r_0(P_3)}{|P_3|} = \frac{\frac{1}{7}}{3} = \frac{1}{21}.$$

Veja que, a partir da primeira iteração, o *PageRank* de algumas páginas já começam se sobressair sobre outras. Caso continuássemos, utilizariamos os $r_1(P_i)$, $i = 1, 2, \ldots, 7$, como sendo os ranks "atuais" das páginas para o cálculo dos $r_2(P_i)$. Depois usaríamos os $r_2(P_i)$ para o cálculo dos $r_3(P_i)$, e assim por diante, até, possívelmente, atingir valores estáveis.

Voltando ao caso geral de um conjunto de n páginas P, por

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|},$$

temos n fórmulas que calculam $r_{k+1}(P_1)$, $r_{k+1}(P_2)$, ..., $r_{k+1}(P_n)$. No cálculo de cada uma, sempre há uma soma de n termos (considerando soma por 0). Cada um desses termos é formado por uma multiplicação entre um *rank* $r_k(P_j)$ e um peso, sendo o mesmo 0 ou $\frac{1}{|P_j|}$. Desse modo, salta aos olhos que esse sistema de equações está parecendo muito uma multiplicação matriz-vetor. E de fato, ele realmente pode ser representado dessa forma.

Os $r_k(P_i)$, $i = 1, \ldots, n$ serão as componentes de um vetor coluna π^k, assim $r_k(P_1) = \pi_1^k$, $r_k(P_2) = \pi_2^k$, etc, e os pesos $\frac{1}{|P_i|}$ farão parte da matriz de coeficientes.

Agora você leitor, consegue lembrar de alguma matriz que era composta por pesos, que juntos somavam a 1 e que dependiam somente do quanto de *links* uma página P_i possuía para outras páginas? Essa é justo a matriz H. Para lembrança, a matriz H do exemplo dado na seção "A Matriz Google" é

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

Como agora sabemos a matriz e o vetor que estamos lidando, falta se decidir se a multiplicação será dada pela direita, $\pi^{k+1} = H\pi^k$, ou pela esquerda, $(\pi^t)^{k+1} = (\pi^t)^k H$ (π^t representa o vetor transposto de π). Utilizando o mesmo exemplo do grafo anterior, vemos que a multiplicação entre π^0 e H deve ser feita pela esquerda para "bater" com o sistema de equações dos $r_1(P_i)$ calculado acima.

$$\left(\pi^t\right)^1 = \left(\pi^t\right)^0 H$$

$$\left(\pi^t\right)^1 = \begin{bmatrix} \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{21} & \frac{1}{7} & \frac{1}{7} & \frac{4}{21} & \frac{1}{7} & \frac{1}{7} & \frac{1}{21} \end{bmatrix}.$$

Assim, temos que a fórmula iterativa

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}, \quad \text{para } i = 1, 2, \ldots, n,$$

pode ser representa pela seguinte forma mais compacta:

$$(\pi^t)^{k+1} = (\pi^t)^k H.$$

Contudo, ainda sim temos problemas no cálculo do *PageRank*. Como já foi discutido na seção "A Matriz Google" (3.2), sabemos que a utilização da matriz H trás problemas para o modelo. Portanto, em vez de π^k ser multiplicado por H, ele será multiplicado por G. Desse modo tem-se,

$$(\pi^t)^{k+1} = (\pi^t)^k G.$$

Uma implicação "ruim" do uso da G na equação é que ela não nos dará o "real" ranqueamento de cada página. Isso se deve ao fato de G ser o resultado de uma soma ponderada entre S e ee^t. A matriz S preserva a estrutura de links das páginas da Internet e lida de forma "especial" com páginas sem links. Enquanto ee^t, que é uma matriz de teleportação aleatória, não preserva a estrutura. Ainda assim é necessário a utilização de G para, pelo menos, obter uma aproximação do *rank* "real" das páginas.

Não é possível de se calcular este *rank* "real", utilizando somente S. Apenas o uso dela pode acarretar no fenômeno do usuário fictício seguir um caminho periódico previsível entre as páginas, algo visto na seção anterior. Ou, de forma mais geral, o usuário pode navegar somente num conglomerado de páginas sem ter a possibilidade do mesmo conseguir "chegar" em alguma página fora desse conglomerado. Em qualquer um desses casos, o *PageRank* das páginas que não pertencem a esse conglomerado iria tender a zero conforme o cálculo dos *ranks* for sendo executado. O que implica na falha do modelo para o cálculo do *rank* dessas páginas, algo não desejado.

Esperamos que os elementos de π^k, $\pi_i^k = r_k(P_i)$, convirjam para um valor estável depois de várias iterações, isso significa que para um k "grande", $\pi^k \approx \pi$, para certo π. Assim, podemos escrever que para $k \to \infty$, $\pi^k = \pi$ em que π, que é chamado de o *vetor do PageRank*, satisfaz

$$\pi^t = \pi^t G.$$

Visto que há condições impostas aos todos $r_k(P_i)$ $i = 1, \ldots, n$, o vetor π^k também as terá. Primeiramente, temos que $r_k(P_i) = \pi_i^k > 0$ o que implica em $\pi^k > 0$.

E também que como $\sum_{i=1}^{n} r_k(P_i) = \sum_{i=1}^{n} |r_k(P_i)| = 1$ isso significa que a norma 1 de π^k, $\|\pi^k\|_1$, é igual a 1. Note que π^k se encaixa na definição de um *vetor de distribuição de probabilidade* enunciada na seção "Cadeias de Markov" (3.1).

Deste modo, o problema do *PageRank* se reduz a encontrar o *vetor de probabilidade estacionário* π que satisfaz as condições $\pi > 0$, $\|\pi\|_1 = 1$ e também $\pi^t = \pi^t G$.

Agora que formulamos o problema e sabemos as condições que π deve satisfazer já podemos, então, dado uma matriz G de um conjunto de páginas P, encontrar o vetor π que representa o *rank* de cada uma das páginas utilizando a fórmula $(\pi^t)^{k+1} = (\pi^t)^k G$. Contudo, até o momento nada nos garante que o método proposto irá funcionar, isto é, convergir para π. Ainda mais, caso funcione, poderia haver o caso de existir mais de um vetor que satisfazesse as condições para ser classificado como *vetor do PageRank*, implicando na existência de mais de um *rank* para a mesma página, algo não desejado. E, além do mais, pode ainda haver o caso do vetor π simplesmente não existir.

Deste modo ficamos com os seguintes questionamentos a respeito do *vetor do PageRank* π: Dado uma matriz G, o vetor π existe? ele é único? o método proposto para seu cálculo sempre converge?

Antes de poder começar a responder essas perguntas, precisamos fazer uma pequena pausa no raciocínio para falar sobre autovalores e autovetores.

3.4 Autovalor e Autovetor

Você provavelmente está acostumado com a seguinte definição de autovalor e autovetor: dada uma matriz qualquer $A \in \mathbb{R}^{n \times n}$ e um vetor $v \in \mathbb{R}^n$, com $v \neq 0$, v é autovetor de A associado ao autovalor λ se

$$Av = \lambda v,$$

para algum $\lambda \in \mathbb{R}$. Uma forma de se calcular os autovalores λ é pelo *polinômio característico* de A, $p(A) = det(A - \lambda I)$.

Porém, mesmo A possuindo apenas entradas reais, as raízes de $p(A)$ podem assumir valores complexos. Assim, se nos restringirmos a valores reais para os autovalores λ, é possível que haja a "perda" de certos λ. Isso implicaria que a "quantidade", multiplicidade algébrica, de autovalores de A seja menor do que a própria dimensão de A, o que implicaria em problemas para nossa análise. Portanto, para que nenhum λ fique de fora, será adotada a seguinte definição de autovalor e autovetor:

Dada uma matriz $A \in \mathbb{C}^{n \times n}$ e um vetor $v \in \mathbb{C}^n$, com $v \neq 0$, v é autovetor de A associado ao autovalor λ se

$$Av = \lambda v,$$

para algum $\lambda \in \mathbb{C}$.

O conjunto de autovalores λ que uma matriz A possui será chamado de *espectro* de A, denotado por $\sigma(A)$. O valor absoluto dos maiores λ em módulo será chamado de *raio espectral*, denotado por $\rho(A)$.

Uma observação importante é que existem tanto autovetor à direita, $Ax = \lambda x$, quanto à esquerda, $y^t A = \lambda y^t$. Os autovetores à direita são os que estamos mais habituados a lidar. Mas não se preocupe, se você entende autovetores à direita você também entende o à esquerda. É praticamente a mesma ideia. Algo não difícil de provar

é que o espectro $\sigma(A)$ de uma matriz A, o conjunto dos autovalores de uma matriz, dos autovetores à direita e dos à esquerda de A é o mesmo, assim, se existem autovetores à direita para um certo λ irá também existir autovetores à esquerda para o mesmo λ. Porém há sim, certas diferenças entre ambos, como por exemplo que se um vetor v é autovetor à direita associado a um autovalor λ isso não implica que v^t também será autovetor à esquerda associado a λ.

Agora que abordamos o conceito de autovalores e autovetores, vamos voltar a discussão a respeito do problema do PageRank. Relembrando, queremos encontrar um *vetor de densidade de probabilidade estacionário* π, chamado de *vetor do PageRank*, tal que $\pi^t = \pi^t G$, π seja positivo e tenha norma 1 igual a 1. Observando novamente sob novos olhares as condições de π, mais especificamente a primeira mencionada, vemos que π nada mais é que um autovetor a esquerda de G associado ao autovalor $\lambda = 1$. Deste modo, o problema do PageRank também é um problema de autovalor e autovetor.

Agora que conhecemos mais uma característica de π, podemos começar a encontrar respostas para os questionamentos a respeito do mesmo. Primeiramente, para discutir sobre a existência e unicidade, utilizaremos um teorema importante na área de álgebra linear, o chamado Teorema de Perron.

3.5 Perron

O Teorema de Perron infere propriedades de um autovalor específico de uma matriz A que satisfaz apenas duas condições: A é quadrada, $A \in \mathbb{R}^{n\times n}$, e que todos seus elementos sejam positivos, $a_{ij} > 0, i, j = 1, 2, \ldots, n$.

Dentre todas as afirmações que o teorema diz, as mais importantes, para nós, são as seguintes: Seja A uma matriz quadrada com apenas entradas positivas. Existe um único autovetor p à direita de A, com $p > 0$ e $\|p\|_1 = 1$, chamado *vetor de Perron*, associado a um autovalor $\lambda = r > 0$, chamado de *raiz de Perron*, com multiplicidade algébrica igual à 1. Além de p ser único, ele ainda possui a "unicidade" de, exceto de seus múltiplos positivos, não haver outros autovetores positivos à direita de A, independentemente do autovalor. E, além de tudo isso, r está no raio espectral de A, ou seja, $r = \rho(A)$.

No teorema está sendo apenas descrito o *autovetor de Perron* à direita tal que $Ap = rp$. Porém ainda assim existe o *autovetor de Perron* à esquerda, digamos q, com as mesmas condições e propriedades de p associado também a *raiz de Perron* r em que $q^t A = rq^t$.

A *matriz Google* G satisfaz as duas condições iniciais do teorema, assim deve existir um autovalor $\lambda = r$ de G que possui as propriedades descritas acima. O que nos resta agora é encontrá-lo.

Note que a multiplicação entre G e e, o vetor coluna com todas entradas iguais a 1, representa a soma dos elementos das linhas de G, que por construção é igual à 1. Deste modo temos que,

$$Ge = e.$$

Esse resultado implica que o vetor e é um autovetor à direita de G com autovalor $\lambda = 1$. Sabemos pelo Teorema de Perron que autovetores positivos devem ter como autovalor associado $\lambda = r$. Como $e > 0$ e tem como autovalor associado $\lambda = 1$, concluímos que $r = 1$.

Agora que sabemos que o *autovalor de Perron* de G é 1, podemos nos apropriar das propriedades do teorema. Como o *vetor do PageRank* π satisfaz

$$\pi^t = \pi^t G,\ \pi > 0\ e\ \|\pi\|_1 = 1,$$

temos que π é o *vetor de Perron* à esquerda de G. Assim, utilizando G para se calcular o vetor π, sua existência e unicidade são garantidos graças ao Teorema de Perron.

Como agora sabemos que o *vetor de probabilidade estacionário* π existe e é único, só nos resta ainda um questionamento: é possível calcular π pelo método iterativo proposto?

3.6 Método da Potência

O método iterativo proposto na seção O Vetor do PageRank (3.3) é um caso específico do Método da Potência. O Método da Potência é uma técnica iterativa usada para determinar o autovalor dominante de uma matriz, isto é, o maior autovalor em magnitude. O método, além de calcular um autovalor, obtém também um autovetor associado.

Para poder aplicar o Método da Potência, devemos ter uma matriz $A_{n\times n}$ diagonalizável (possui n autovetores linearmente independentes) com um maior autovalor em magnitude $\rho(A) = |\lambda_1|$, de tal forma que

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

Observe que o fato de $|\lambda_1| > |\lambda_i|$ para $i = 2, \ldots, n$ impõe que $\lambda_1 \in \mathbb{R}$. Caso a parte imaginária de λ_1 fosse diferente de zero, existiria um autovalor conjugado $\bar{\lambda}_1 \neq \lambda_1$ de tal forma que $|\bar{\lambda}_1| = |\lambda_1|$, que quebraria a hipótese.

Seja $\{\nu^1, \nu^2, \nu^3, \ldots, \nu^n\}$ um conjunto de n autovetores linearmente independentes associados aos seus respectivos autovalores λ_i. Podemos representar um vetor $x \in \mathbb{R}^n$ qualquer utilizando os autovetores ν^i's como base, ou seja,

$$x = \sum_{i=1}^{n} \alpha_i \nu^i,$$

com $\alpha_i \in \mathbb{R}$, $i = 1, 2, \ldots, n$. Aplicando a matriz A a x obtemos,

$$Ax = \sum_{i=1}^{n} \alpha_i A\nu^i = \sum_{i=1}^{n} \lambda_i \alpha_i \nu^i.$$

Aplicando a matriz A a Ax obtemos,

$$AAx = A^2x = \sum_{i=1}^{n} \lambda_i \alpha_i A\nu^i = \sum_{i=1}^{n} \lambda_i^2 \alpha_i \nu^i.$$

Fazendo esse mesmo processo k vezes chegamos a,

$$AA^{k-1}x = A^k x = \sum_{i=1}^{n} \lambda_i^{k-1} \alpha_i A\nu^i = \sum_{i=1}^{n} \lambda_i^k \alpha_i \nu^i.$$

Fatorando λ_1^k do lado direito da última equação vemos que,

$$A^k x = \lambda_1^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k \alpha_i \nu^i,$$

e assim, chegamos a,

$$A^k x = \lambda_1^k \left(\alpha_1\nu^1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k \alpha_2\nu^2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k \alpha_n\nu^n\right).$$

Observe que como $|\lambda_1| > |\lambda_i|$ para $i = 2, \ldots, n$, temos que $(\lambda_i/\lambda_1) < 1$ e assim,

$$\lim_{k\to\infty} \left(\frac{\lambda_i}{\lambda_1}\right)^k = 0, \quad \text{para } i = 2, \ldots, n.$$

Portanto conforme aumenta o número de iterações k, os termos de $\lambda_1^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k \alpha_i\nu^i$ vão tendendo cada vez mais a $\lambda_1^k\alpha_1\nu^1$. Para que possamos tomar o limite de k tendendo ao infinito, devemos tomar o cuidado para que λ_1^k convirja. Para isso, só nos resta a opção $\lambda_1 = 1$. Caso $|\lambda_1|$ fosse menor do que 1, teríamos $\lim_{k\to\infty} \lambda_1^k = 0$, que não nos interessa. Caso $|\lambda_1| > 1$, o limite $\lim_{k\to\infty} \lambda_1^k$ iria divergir para o infinito. E se caso $\lambda_1 = -1$, λ_1^k não convergiria para um valor fixo conforme k aumenta. Assim, assumindo $\lambda_1 = 1$ e tomando o limite de k tendendo ao infinito, vemos que,

$$\lim_{k\to\infty} A^k x = \lim_{k\to\infty} \lambda_1^k \left(\alpha_1\nu^1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k \alpha_2\nu^2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k \alpha_n\nu^n\right) = \lim_{k\to\infty} \lambda_1^k\alpha_1\nu^1$$

e assim,

$$\lim_{k\to\infty} A^k x = \alpha_1\nu^1.$$

Portanto, chegamos à conclusão que, pela equação acima, $A^k x$ tende a um par de autovalor-autovetor, com autovalor $\lambda = 1$ e autovetor associado $\nu = \alpha_1\nu^1$ sempre que A for diagonalizável e tiver $\lambda_1 = 1$ como o maior autovalor em módulo. A única forma do limite dar um resultado "insatisfatório", é caso $\alpha_1 = 0$, isto é, a contribuição de ν^1 na representação de x pela base $\{\nu^1, \nu^2, \ldots, \nu^n\}$ for zero. Contudo, se isso acontecesse, o que é raro, poderíamos apenas usar algum outro x_0 que tenha $\alpha_1 \neq 0$ em $A^k x_0$ para obter um par autovalor-autovetor com $\lambda = 1$ e o autovetor associado.

Observe que, a velocidade de convergência de A^k para $\alpha_1\nu^1$ dependerá do valor de $\left(\frac{\lambda_2}{\lambda_1}\right)$. Caso $|\lambda_2| \approx |\lambda_1|$, teremos $\left(\frac{\lambda_2}{\lambda_1}\right) \approx 1$, o que implica em uma convergência lenta para $\alpha_1\nu^1$. E se caso $|\lambda_2|$ for bem menor que $|\lambda_1|$, teremos $\left(\frac{\lambda_2}{\lambda_1}\right) \ll 1$, o que implica em uma convergência rápida do método.

Desenvolvemos até aqui o Método da Potência para $\lambda_1 = 1$, que é o caso que justamente nos interessa. No caso que $\lambda_1 \neq 1$, o Método da Potência é levemente diferente. Nesse, $A^k x$ é normalizada a cada iteração, com o intuito de não permitir que tende a zero ou divirja. Todavia, a ideia por trás e o resultado final obtido será análogo ao que foi desenvolvido. Para mais informações, consulte Burden et al. [2015].

Olhando de volta para a *matriz Google* G, podemos assumir a propriedade da mesma ser diagonalizável. O fenômeno de uma matriz, relacionada a algum problema, não ser diagonalizável é algo raro e isso sempre está relacionado com uma característica do problema em si. Por sua vez, a suposição da matriz G ser diagonalizável é algo nada "problemática", visto que não há nenhuma característica do problema do PageRank que nos diria algo ao contrário.

O que nos resta agora é analisar os maiores autovalores em magnitude de G para se certificar que o Método da Potência aplicada a ela converge. Já sabemos da seção Perron que $\lambda_1 = 1 = \rho(G)$, que implica que $1 \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$. Porém, deve-se ter $\lambda_1 = 1$ como o único autovalor no raio espectral, isto é, $1 > |\lambda_i|$ para $i = 2, 3, \ldots, n$. É possível provar, no entanto não o faremos aqui, que o parâmetro $\alpha \in (0, 1)$ da equação que define G,

$$G = \alpha S + (1 - \alpha)\frac{1}{n} e e^t,$$

é o segundo maior autovalor em magnitude de G. Como $\lambda_2 = \alpha < 1$, temos que $1 > \alpha \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$.

Como G satisfaz ambas as hipóteses para a convergência do Método da Potência, temos que o limite $\lim_{k\to\infty} G^k x$ converge, com uma velocidade dependendo de $\left(\frac{\alpha}{1}\right)$, para um autovetor ν associado ao autovalor $\lambda = 1$. Assim,

$$\lim_{k\to\infty} G^k x = \nu.$$

Como π e ν são autovetores associados a $\lambda = 1$ que, devido ao Teorema de Perron, possui multiplicidade algébrica igual a 1, sabemos que ν é um múltiplo de π. Devido ao fato de $\|\pi\|_1 = 1$, $\pi > 0$ e ν ser $\nu > 0$ ou $\nu < 0$, temos que

$$\pi = \text{sgn}(\nu)\frac{\nu}{\|\nu\|_1},$$

em que $\text{sgn}(\nu) = 1$ caso $\nu > 0$ e $\text{sgn}(\nu) = $ -1 caso $\nu < 0$. Assim, chegamos na conclusão que o *vetor do PageRank* π é calculável pelo Método da Potência aplicado a G.

Como, agora, conseguimos responder nossas dúvidas a respeito da existência e unicidade de π e concluímos que ele é calculável pelo método proposto, vamos calcular o *vetor do PageRank* para alguns exemplos.

3.7 Exemplos

Nesta seção, será calculado o *vetor do PageRank* π para alguns casos. O cálculo será feito pelo seguinte código em Python:

```
1 import numpy as np
2
3 def Vetor_PageRank(H, alpha, epsilon):
```

```
4       H = H.astype(float)
5       n = np.size(H, 0)
6       a = np.sum(H, axis = 1)
7       for i in range(n):
8           if a[i] == 0:
9               a[i] = 1
10          else:
11              H[i] = H[i]/a[i]
12              a[i] = 0
13      pi      = np.ones(n)/n
14      k       = 0
15      residuo = 1
16      while residuo >= epsilon:
17          prevpi  = np.array(pi)
18          k       = k + 1
19          pi      = np.array(alpha*pi@H + (alpha*(pi@a)+1-alpha
               )*(np.ones(n)/n))
20          residuo = np.linalg.norm(pi - prevpi, ord = 1)
21      print("numero de iteracoes: ",k)
22      print("vetor PageRank:")
23      print(pi)
24      print(np.linalg.norm(pi, ord=1))
```

No código, calculamos π por um método ligeiramente "diferente" do que vimos na teoria. Pelas seções O Vetor PageRank (3.3) e Método da Potência (3.6), desenvolvemos e concluímos que era possível calcular π multiplicando ele, a cada iteração, pela matriz G. Porém, a *matriz Google* é densa, no sentido que todas suas entradas são números diferentes de 0. Por essa razão, cada multiplicação entre π^k e G é extremamente cara computacionalmente, fazendo com que demore uma quantidade de tempo inviavelmente grande caso fossemos calcular o *PageRank* para um conjunto com um grande número de páginas. Assim, fazemos a multiplicação entre π^k e G de forma "indireta", escrevendo G em termos da matriz H, vista na seção A Matriz Google (3.2). Ela, por sua vez, preserva a estrutura de *links* do conjunto de páginas que calcularemos o *PageRank*, e, como normalmente páginas da Internet possuem poucos *links* para outras páginas, isso faz com que H seja uma matriz esparsa, no sentido de possuir poucas entradas não-nulas. E, portanto, como no cálculo envolvem muitos zeros, cada multiplicação matriz-vetor é consideravelmente mais barata.

Vamos deduzir a fórmula utilizada no código. Primeiramente, para poder escrever G em termos de H, precisamos saber escrever S em termos de H. Conseguimos calcular S pela seguinte maneira: seja a um vetor tal que $a_i = 1$ caso a linha i de H for nula e $a_i = 0$ caso contrário. A multiplicação $\frac{1}{n}ae^t$, em que e é o vetor com todas entradas iguais a 1, gera uma matriz em que a linha i é o vetor $\frac{1}{n}e^t$ caso $a_i = 1$ ou é o vetor nulo caso $a_i = 0$. Como exemplo, caso $a \in \mathbb{R}^3$ for dada por

$$a = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

a matriz $\frac{1}{3}ae^t$ será

$$\frac{1}{3}ae^t = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}.$$

Como S é justamente a matriz H só que com linhas nulas alteradas para $\frac{1}{n}e^t$, S é dada por

$$S = H + \frac{1}{n}ae^t.$$

Substituindo a fórmula de S na fórmula de G, chegamos na seguinte equação:

$$\begin{aligned} G &= \alpha S + (1-\alpha)\frac{1}{n}ee^t; \\ &= \alpha\left(H + \frac{1}{n}ae^t\right) + (1-\alpha)\frac{1}{n}ee^t; \\ &= \alpha H + \frac{\alpha}{n}ae^t + (1-\alpha)\frac{1}{n}ee^t; \\ &= \alpha H + (\alpha a + (1-\alpha)e)\frac{1}{n}e^t. \end{aligned}$$

Multiplicando essa equação por $(\pi^t)^k$,

$$\begin{aligned} (\pi^t)^k G &= (\pi^t)^k \left(\alpha H + (\alpha a + (1-\alpha)e)\,\frac{1}{n}e^t\right); \\ &= \alpha(\pi^t)^k H + (\pi^t)^k\,(\alpha a + (1-\alpha)e)\,\frac{1}{n}e^t. \end{aligned}$$

Utilizando a relação $(\pi^t)^k e = 1$, chegamos na equação iterativa

$$(\pi^t)^{k+1} = \alpha(\pi^t)^k H + \left(\alpha(\pi^t)^k a + 1 - \alpha\right)\frac{1}{n}e^t,$$

que é a utilizada no código.

Para calcular o *vetor do PageRank* π de um conjunto de páginas P, a função *Vetor_PageRank* do código requer 3 entradas: a matriz de adjacência A do grafo de P, o parâmetro $\alpha \in (0,1)$ e o resíduo mínimo ϵ desejado, isto é, a norma 1 mínimia desejada da diferença entre duas iterações subsequentes de $(\pi^t)^k$, $\|\pi^{k+1} - \pi^k\|_1 \leq \epsilon$, para que o código pare.

Agora que sabemos a fórmula de $(\pi^t)^{k+1}$ utilizada e as entradas requeridas, vamos calcular o *vetor do PageRank* para alguns conjuntos de páginas.

Voltando ao conjunto P "clássico" de nosso texto, cujo grafo, para recordação, é

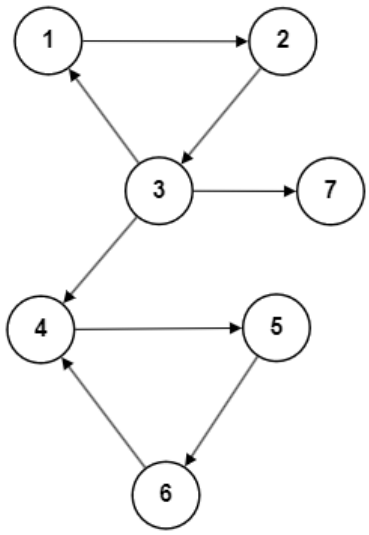


Figura 4: Grafo do conjunto P

vamos calcular o *rank* de cada página. A matriz de adjacência de P é

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Para a escolha do parâmetro α, vamos usufruir do valor, supostamente, utilizado pela Google, $\alpha = 0.85$. E, como temos um conjunto pequeno de páginas, vamos impor um $\epsilon = 10^{-6}$.

```
A = np.array([[0,1,0,0,0,0,0],
              [0,0,1,0,0,0,0],
              [1,0,0,1,0,0,1],
              [0,0,0,0,1,0,0],
              [0,0,0,0,0,1,0],
              [0,0,0,1,0,0,0],
              [0,0,0,0,0,0,0]])
Vetor_PageRank(A, 0.85, 10**(-6))
```

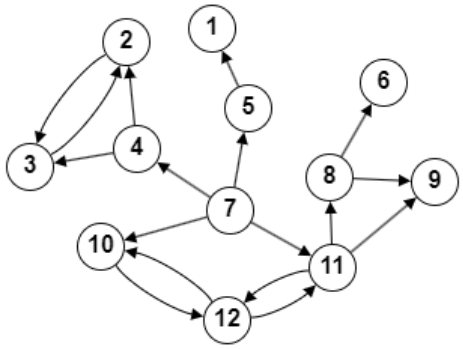
Caso o código fosse executado, teríamos o seguinte output:

```
Numero de iteracoes:  33
Vetor PageRank calculado:
[0.05352352 0.07342292 0.09033744 0.25251642 0.24256672
   0.23410946
```

```
4  0.05352352]
```

Assim, temos que, segundo o algoritmo, a página mais importante desse conjunto é a página P_5, e as menos importantes são as páginas P_1 e P_7. É possível observar que, o motivo de P_5 ficar em primeiro lugar, vem do fato de P_4, que é uma página que possui dois *link* para si, ter uma ligação apenas para P_5, fazendo com que seu "voto de confiança" tenha um peso "imenso". Podemos argumentar também que, como P_3 possui o maior número de *links* para outras páginas, o peso de cada ligação de P_3 tem a menor relevância do conjunto. Assim, isso justifica o fato de P_1 e P_7 terem ficado em último lugar no placar de importância, visto que, a única página que conecta a P_1 e P_7 é P_3.

Para o próximo exemplo, vamos calcular o *PageRank* do conjunto de páginas do seguinte grafo,



em que sua matriz de adjacência é dada por

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Escolhendo $\alpha = 0.85$ e $\epsilon = 10^{-10}$ visto que agora precisamos de uma melhor precisão numérica para diferenciar o *rank* de um número maior de páginas, o *vetor do PageRank* é dado ao executar a linha de comando abaixo.

```
1 A = np.array([
2     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
3     [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
4     [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
    [0, 1, 1, 0, 0, 0, 0, 0, 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, 0, 0],
    [0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]])
Vetor_PageRank(A, 0.85, 10**(-10))
```

E assim, temos como saída:

```
Numero de iteracoes:  89
Vetor PageRank calculado:
[0.04726832 0.23515349 0.23515349 0.02822424 0.02822424
   0.04202597
 0.02327772 0.04411353 0.06286178 0.07353814 0.07353814
    0.10662095]
```

Portanto, segundo o algoritmo, as páginas mais relevantes desse conjunto são P_2 e P_3, e a menos relevante é a P_7. Salta aos olhos que, mesmo P_7 estando "no centro de tudo", fazendo um papel importante de conectar a diferentes conglomerados de páginas, ela é vista pelo *PageRank* como sendo a "pior" página de todas, já que ninguém possui um *link* para ela.

Agora que foi visto dois exemplos de grafos criados "artificialmente", vamos agora calcular o PageRank para conjuntos de páginas da própria Internet. Para auxiliar nessa tarefa, a função *Crawl* abaixo, dada uma página inicial, retorna a matriz de adjacência de um conjunto das páginas que são "linkadas"pela página inicial. Ou seja, se uma página da Internet está nesse conjunto é devido ao fato da página inicial possuir um *link* para ela.

O código que defini a função *Crawl* se encontra abaixo:

```
import re
from urllib.request import urlopen

#inserir link inicial sem "/" no final

def Crawl(link_inicial, n): # n e o numero de paginas do
   conjunto P
    n = n + 1
    m = 0
    U = [m] * n
    H = np.zeros((n,n), dtype = int)
    U[m] = link_inicial
    link_acabou = False
    for i in range(n):
        print("entrando em:", U[i])
        if U[i] == 0:
```

```
16              print("Acabaram os links da pagina inicial")
17              link_acabou = True
18              break
19          page = str(urlopen(U[i]).read())
20          posicao_https = 0
21          e = 0
22          for k in range(len(re.findall('http',page))):
23              if re.search('http', page[e:]) == None:
24                  break
25              posicao_https = e + re.search('http', page[e:]).
                   start()
26              if re.search('\"', page[posicao_https:]) is None:
27                  print("Skipando")
28                  continue
29              e = posicao_https + re.search('\"', page[
                   posicao_https:]).start()
30              url = page[posicao_https:e]
31              skips ={'.gif','.jpg','.pdf','.css','lmscadsi','
                   cybernet',
32              'search.cgi','.ram','www.w3.org', 'scripts',
33              'netscape','shockwave','webex','fansonly','.js',
                   '.png',
34              'creativecommons.org', '.net'}
35              url_inutil = False
36              for skip in skips:
37                  if re.search(skip, url):
38                      url_inutil = True
39                      break
40              if url_inutil:
41                  continue
42              if url.endswith('/'):
43                  url = url[:-1]
44              j = 0
45              url_ja_pertence = False
46              for q in range(m + 1):
47                  if url == U[q]:
48                      j = q
49                      url_ja_pertence = True
50                      break
51              if i == 0:
52                  try:
53                      urlopen(url)
54                  except:
55                      continue
56              if m < n-1 and url_ja_pertence == False and i ==
                   0:
```

```
57                  m = m + 1
58                  j = m
59                  U[m] = url
60                  print("link adicionado:", url)
61                  url_ja_pertence = True
62              if url_ja_pertence and i != j: #link da pagina
                   para si mesma nao conta
63                  H[i][j] = 1
64      if link_acabou:
65          U = np.delete(U,np.s_[i:n+1],0)
66          H = np.delete(np.delete(H,np.s_[i:n+1],0),np.s_[i:n
               +1],1)
67      H = np.delete(np.delete(H, 0, axis=0), 0, axis=1) #
           exclusao da primeira linha e coluna
68      print(H)
69      return H
```

A função tem como parâmetros o link da página inicial, no formato de string, e o número de páginas que se deseja no conjunto formado. Para utilizar o código, é preciso que o link da página inicial não tenha '/' no final. Ou seja, como exemplo, caso queiramos colocar a página da coordenação de graduação do curso de Matemática Aplicada e Computacional da UNICAMP como argumento na função *Crawl*, devemos inserir **'https://www.ime.unicamp.br/~mac'** e não **'https://www.ime.unicamp.br/~mac/'**. Outro ponto importante de se notar é que caso o número de páginas desejado for maior do que o número de *links* válidos na página inical, o conjunto formado terá todas as páginas "linkadas"pela página inicial.

Resumidamente, o algoritmo age da seguinte forma: ele entra na página inicial para buscar e guardar *links* viáveis. Após encontrar n páginas ou encontrar todos os *links* possíveis, o algoritmo entra em cada página para anotar se há ou não um *link* para as outras. Após estes passos, a função retorna a matriz de adjacência do conjunto. E, além do mais, durante todo processo, ocorre "prints"na tela informando qual página está sendo visitada e quais estão sendo adicionadas no conjunto.

Uma característica importante da função *Crawl* de saber é que caso a página inicial entrar no conjunto durante a execução, ela certifica-se de retirá-la ao final. Isso é feito devido ao fato de que caso calculássemos o PageRank do conjunto com a página inicial, teríamos um resultado "enviesado", visto que, por construção, a página inicial direciona para todas as páginas do conjunto.

Para vermos, na prática, o que de fato *Crawl* faz, vamos executar o código abaixo,

```
1 Crawl('https://www.ime.unicamp.br/~mac', 6)
```

Após a execução, temos a saída

```
1 entrando em: https://www.ime.unicamp.br/~mac
2 link adicionado: https://macunicamp.slack.com
3 link adicionado: https://bit.ly/slackMAC2
4 link adicionado: https://play.google.com/store/apps/details?
    id=com.Slack
```

```
link adicionado: https://apps.apple.com/br/app/slack/
    id618783545
link adicionado: https://www.linkedin.com/groups/12292620
link adicionado: https://www.ime.unicamp.br/~mac/album.html
entrando em: https://macunicamp.slack.com
entrando em: https://bit.ly/slackMAC2
entrando em: https://play.google.com/store/apps/details?id=
    com.Slack
entrando em: https://apps.apple.com/br/app/slack/id618783545
entrando em: https://www.linkedin.com/groups/12292620
entrando em: https://www.ime.unicamp.br/~mac/album.html
[[0 0 0 0 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]
 [1 1 1 1 1 0]]
```

Calculando o PageRank desse conjunto de páginas por meio do comando,

```
Vetor_PageRank(Crawl('https://www.ime.unicamp.br/~mac', 6),
    0.85, 10**(-10))
```

recebemos a seguinte saída:

```
entrando em: https://www.ime.unicamp.br/~mac
link adicionado: https://macunicamp.slack.com
link adicionado: https://bit.ly/slackMAC2
link adicionado: https://play.google.com/store/apps/details?
    id=com.Slack
link adicionado: https://apps.apple.com/br/app/slack/
    id618783545
link adicionado: https://www.linkedin.com/groups/12292620
link adicionado: https://www.ime.unicamp.br/~mac/album.html
entrando em: https://macunicamp.slack.com
entrando em: https://bit.ly/slackMAC2
entrando em: https://play.google.com/store/apps/details?id=
    com.Slack
entrando em: https://apps.apple.com/br/app/slack/id618783545
entrando em: https://www.linkedin.com/groups/12292620
entrando em: https://www.ime.unicamp.br/~mac/album.html
[[0 0 0 0 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]
 [1 1 1 1 1 0]]
Numero de iteracoes:  13
Vetor PageRank calculado:
```

```
[0.14914909 0.14914909 0.27592581 0.14914909 0.14914909
 0.12747785]
```

E assim, após ver o PageRank agir na prática, finalizamos a discussão a respeito do algoritmo.

4 Conclusão

Ao final deste trabalho, é visível que foi possível abordar ao leitor os conceitos matemáticos fundamentais do PageRank de maneira intuitiva e não rigorosa. Assim, o objetivo principal do projeto de criar um artigo adequado a alunos de graduação da UNICAMP de tal maneira que demonstrasse a importância do conceito de autovalores e autovetores foi atingido.

Sinto pessoalmente que o desenvolvimento deste projeto, e principalmente, graças as conversas que tive em reuniões semanais com o Prof. Paulo José da Silva e Silva me fez mudar como aluno. Fez com que mudasse minha atitude, para uma mais ativa, em relação a pesquisar e estudar um assunto específico. Em outras palavras, aprendi mais a aprender. Sou grato ao Prof. Paulo e à UNICAMP, visto que graças a ambos tive a oportunidade de praticar e conhecer o ser pesquisador, algo que por conta própria não faria.

Referências

R.L. Burden, J.D. Faires, and A.M. Burden. *Numerical Analysis*. Cengage Learning, 2015. ISBN 9781305465350.

Amy Langville Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, July 2011.

Tanya Leise Kurt Bryan. The $25,000,000,000 eigenvector: The linear algebra behind google. *SIAM Review*, 48(3):569–581, January 2006. ISSN 0036-1445. doi: 10.1137/050623280.