Modelagem Matemática: Como o Google ordena páginas

Esta é uma tarefa computacional em que usaremos conceitos de Vetores e Sistemas Lineares em uma aplicação interessante. Ela pode ser realizada em grupo, com grupos de até 4 alunos. A constituição dos grupos deve ser informada assim que possível, bem antes da entrega da tarefa. A programação pode ser feita em C, Python (3.7) ou Fortran. Além deste enunciado, está disponível no Moodle um artigo, no qual ele se baseia. Procure ler o artigo, que contém uma série de perguntas e exercícios interessantes. As partes relevantes para esta tarefa estão explicadas aqui. A data de entrega será por volta do final de agosto. Procurem iniciar logo, para que eventuais dúvidas possam ser sanadas.

1 Introdução

Quando fazemos uma procura em um site de busca na internet, como o Google, (que dominou a internet, mas havia Yahoo, Altavista ou outros ainda menos usados), desejamos obter as páginas que contêm um determinado assunto ou palavra-chave, ordenadas em ordem decrescente de prioridade, apesar de não haver uma definição muito clara do que isso significa. O aparecimento do Google no final dos anos 90 foi uma espécie de divisor de águas no que se refere à procura de assuntos na rede. Isto porque o Google parece sempre colocar os sites mais relevantes primeiro. Com outros sites de procura, muitas vezes era necessário olhar páginas e mais páginas até que os resultados interessantes aparecessem.

O objetivo desta tarefa é entender como funciona um site de procura. Em particular, descreveremos o algoritmo utilizado pelo Google para ordenar as páginas em ordem decrescente de importância.

Um site de procura como o Google basicamente faz 3 coisas:

i) varre a rede e localiza todas as páginas públicas;

ii) indexa os dados de i) em um banco de dados de forma que uma procura por palavra-chave possa ser feita de uma maneira eficiente;

iii) atribui uma importância a cada página do banco de dados de ii), de forma que quando um usuário faz uma procura e o subconjunto das páginas que contêm um determinado termo é encontrado, elas podem ser listadas em ordem decrescente de importância;

Como foi dito, descreveremos como o passo iii) acima é feito e um algoritmo para realizar tal atribuição de importâncias.

A idéia básica é atribuir pesos positivos às diversas páginas, sendo as com maior peso, as mais importantes.

Para isso, é preciso armazenar a rede como um grafo orientado, onde os vértices são as diversas páginas e as arestas orientadas (setas) representam um link de uma página para outra. Aqui apareceram alguns termos, possivelmente desconhecidos para vocês. Vai aqui uma breve introdução ao que são grafos:

Um **grafo** é composto por um conjunto de n **vértices** e um conjunto de **arestas**, que conectam vértices. Em um grafo orientado as arestas têm direção, ou seja, distinguimos a aresta (i, j) (conectando o vértice i ao j) da aresta (j, i) que tem a outra "mão de direção". Veja que cada aresta pode ser descrita como um par ordenado de vértices. Esta estrutura

de vértices e arestas é extremamente útil na modelagem de muitos problemas (por exemplo, vértices podem ser cidades e arestas estradas, ou vértices lojas ou centros de distribuição de produtos e as arestas as conecções entre eles. A cada aresta ainda pode se atribuir um peso, representando por exemplo distâncias ou custos de transporte, etc). A descrição de um grafo orientado pode ser facilmente armazenada em uma matriz $n \times n$. O elemento $a_{i,j}$ da matriz será não nulo caso exista a conecção do vértice i ao vértice j e zero caso esta aresta não faça parte do grafo. O valor que armazenamos nesta posição da matriz pode ser um peso atribuído à aresta correspondente. Em vária aplicações, conexões de um vértice a ele mesmo são irrelevantes e podemos colocar a diagonal da matriz como nula.

Em um primeiro exemplo, consideramos uma rede de apenas 4 páginas (veja a figura 1). A página 1 possui links para as páginas 2,3 e 4. A página 2 para 3 e 4, a página 3 para 1 e a página 4 para 1 e 3.

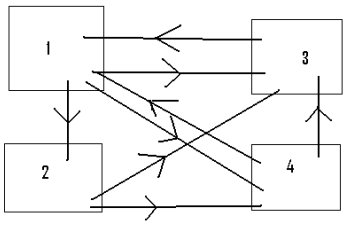

Figure 1: Uma pequena rede de conexões

A importância da página i, onde $i \in \{1, 2, 3, 4\}$ no exemplo anterior, será representada por um número real positivo x_i, o peso da página i.

A questão a se pensar agora é, como atribuir os $x_i's$?

Existem alguns fatores a se levar em conta, mais precisamente:

1. o fato de uma página ter muitos links apontando para ela a torna mais importante;
2. apesar do que foi dito acima, os links que apontam para uma certa página fixada P, em geral têm importâncias diferentes: um link vindo do Yahoo apontando diretamente para P tem muito mais importância que muitos links vindos de páginas de amigos de P; em outras palavras, quando uma página que possui muitos links apontando para ela aponta para P, isto deve pesar mais na importância de P que links vindos de páginas pouco apontadas;
3. por fim, um link vindo para P de uma página que aponta para muitas outras páginas deve ter menor importância que um link vindo de páginas que apontam para poucas;

A motivação para os fatores acima vem do seguinte: uma página P será importante, se um usuário clicando aleatoriamente nos links das páginas que ele encontra, tiver alta probabilidade de acessar P.

Vamos agora, através do exemplo 1, descrever como atribuir os pesos de acordo com os fatores qualitativos acima descritos.

$$\begin{cases} x_1 = x_3/1 + x_4/2 \\ x_2 = x_1/3 \\ x_3 = x_1/3 + x_2/2 + x_4/2 \\ x_4 = x_1/3 + x_2/2 \end{cases}$$

Observe que definimos o peso de uma página como a soma dos pesos das páginas que apontam para ela, divididos pelo número de links que saem de cada página. Obtemos um sistema linear que, em forma matricial, é dado por:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \tag{1}$$

ou seja transformamos o problema de ranqueamento da importância das páginas na rede num problema de encontrar uma solução para (1).

A equação que queremos resolver tem a forma:

$$Mx = x, \text{ ou equivalentemente: } (M - I)x = 0 \tag{2}$$

onde M é uma matriz $n \times n$ e $x \in R^n$.

É claro que tal problema só tem solução não trivial se $M - I$ for uma matriz singular, ou seja, se $det(M - I) = 0$.

Assim, no nosso exemplo, o sistema (1) terá solução se a matriz (chamada matriz de ligação)

$$M = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix}$$

obedecer à condição $det(M - I) = 0$.

Além disso, para que não haja ambiguidade na atribuição dos pesos relativos às diversas páginas, o espaço de soluções do sistema homogêneo $(M - I)x = 0$ deve ter dimensão 1, ou seja, o posto desta matriz deve ser $n - 1$.

Para a rede acima, observamos que o espaço de soluções é unidimensional, dado pelos múltiplos de (12 4 9 6). Iremos normalizar a solução de forma que a soma das suas entradas seja 1. Isto será sempre possível, tomando um múltiplo correto do vetor encontrado.

No exemplo, obtemos os pesos normalizados:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.387 \\ 0.129 \\ 0.290 \\ 0.194 \end{pmatrix}$$

Note que a página 3 apesar de muito apontada, não é a mais importante. Isto porque ela aponta somente para a 1, que ainda é apontada pela 4, o que a torna a mais vista por um clicador de links aleatório.

Uma outra propriedade desejada é que os pesos sejam todos positivos. Veremos adiante que a matriz de ligação pode ser construída de maneira a garantir que isto ocorra.

2 Um pouco de teoria

Inicialmente vamos mostrar que o sistema $(M - I)x = 0$, onde M é uma matriz de ligação, sempre possui solução não trivial, desde que a rede não possua páginas que não apontam para lugar nenhum (essa hipótese simplificadora será sempre usada), chamadas páginas mortas. Uma tal página geraria uma coluna de zeros na matriz de ligação e por diversas razões, isto não é interessante (procure explicar porque ...) .

Lema: Se M é uma matriz $n \times n$, tal que a soma dos elementos de cada coluna é igual a 1, então $M - I$ é singular.

Demonstração: Note que para a matriz transposta de M temos:

$$M^t . \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} .$$

Como consequência o vetor com todas componentes iguais a 1 é uma solução não nula de $(M^t - I)x = 0$ e portanto $det(M^t - I) = det(M - I)^t = det(M - I) = 0$.

Como já dissemos, dois fatos importantes nessa análise são os seguintes: o espaço de soluções do sistema $(M - I)x = 0$ deve ser unidimensional e deve ser possível escolher uma solução positiva com a soma das suas componentes igual a 1. Por exemplo, se a rede não for conexa, como no exemplo 2, onde a página 1 aponta para a 2, a 2 aponta para 1, a 3 aponta para a 4, a 4 aponta para 3 e a 5 aponta para 3 e 4. Neste caso o espaço de soluções do sistema $(M - I)x = 0$ tem dimensão 2 (verifique!). Isto tem sentido prático, pois não é tão simples comparar as importâncias de páginas em redes separadas.

Vamos agora enunciar um resultado que será fundamental para que obtenhamos a distribuição de pesos para as páginas conforme desejado. Este resultado é parte do Teorema de Perron-Frobenius (veja por exemplo [2]).

Teorema Seja M uma matriz $n \times n$ com todos os elementos positivos ($m_{i,j} > 0, \forall i, j$) e tal que a soma dos elementos de cada coluna de M seja 1. Então $(M - I)$ é singular, a dimensão do espaço de soluções de $(M - I)x = 0$ é um e todas as componentes das soluções têm mesmo sinal, podendo portanto ser escolhidas estritamente positivas.

Demonstração: Já vimos que a matriz $(M - I)$ tem determinante nulo. Vamos agora mostrar que toda solução x não trivial de $(M - I)x = 0$ tem todas componentes de mesmo sinal. Como $Mx = x$, temos que cada componente do vetor x é dada por:

$$x_i = \sum_{j=1}^{n} m_{i,j} x_j$$

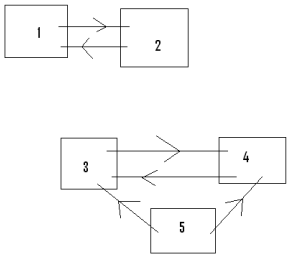

Figure 2: Uma rede desconexa.

e portanto

$$|x_i| = |(\sum_{j=1}^{n} m_{i,j}x_j)| \leq \sum_{j=1}^{n} m_{i,j}|x_j| \ ,$$

sendo que na última passagem usamos que $m_{i,j} > 0$ e vale a igualdade se e somente se todos os x_j que forem não nulos tiverem mesmo sinal. Vamos supor que este não seja o caso: Então temos:

$$|x_i| < \sum_{j=1}^{n} m_{i,j}|x_j| \ .$$

Segue que

$$\sum_{i=1}^{n} |x_i| < \sum_{i=1}^{n}\sum_{j=1}^{n} m_{i,j}|x_j|, \text{ mas}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} m_{i,j}|x_j| = \sum_{j=1}^{n}\sum_{i=1}^{n} m_{i,j}|x_j| = \sum_{j=1}^{n} |x_j|(\sum_{i=1}^{n} m_{i,j}) = \sum_{j=1}^{n} |x_j| \ .$$

Na última passagem usamos o fato de que as colunas de M têm soma 1. Obtivemos então que

$$\sum_{i=1}^{n} |x_i| < \sum_{j=1}^{n} |x_j|$$

o que é uma evidente contradição. Logo, necessariamente todas as componentes x_i não nulas têm mesmo sinal. Agora, do fato de que

$$|x_i| = |(\sum_{j=1}^{n} m_{i,j}x_j)|$$

a única possibilidade de termos uma componente nula é quando a solução x é a solução trivial, e portanto as soluções não triviais possuem todas componentes de mesmo sinal.

Resta agora mostrar que o espaço de soluções de $(M-I)x = 0$ tem dimensão 1. Suponha que existam duas soluções x e y linearmente independentes (e portanto não nulas). Note inicialmente que qualquer combinação linear delas é também solução. Caso a soma das

componentes de alguma delas fosse 0, teríamos uma solução com componentes de sinais opostos, o que já vimos que não ocorre. Seja então $d = \sum_{i=1}^{n} x_i \neq 0$. Defina $s = -\frac{\sum_{i=1}^{n} y_i}{d}$ e $w = sx + y$. Temos que a soma das componentes de w assim definido é igual a zero. Mas isto só pode ocorrer caso w seja a solução trivial, contrariando o fato de x e y serem linearmente independentes. Isto mostra que não podemos ter duas soluções L.I. e portanto que o espaço de soluções de $(M - I)x = 0$ é unidimensional, concluindo a demonstração do teorema.

Uma matriz de ligação M, em geral, não satisfaz as hipóteses do teorema acima, uma vez que pode ter entradas nulas.

Iremos então modificar ligeiramente a matriz M, redefinindo-a como

$$M = (1 - \alpha)M + \alpha S_n,$$

onde S_n é a matriz $n \times n$ com todos elementos iguais a $1/n$ e α é positivo e menor que 1.

Esta é a estratégia usada pelo Google, que adota $\alpha = 0.15$ e esse será o valor sugerido para uso nesta tarefa. Vocês podem naturalmente estudar os efeitos de modificar esse parâmetro.

Essa modificação da matriz M torna todos seus elementos positivos, sem alterar a soma dos elementos de cada coluna, que continua sendo 1 (independendo da escolha de $0 < \alpha < 1$). Assim, esta matriz M modificada satisfaz as hipóteses do teorema.

Logo, dada uma rede sem páginas mortas, o processo de ranquear as páginas consiste em obter a matriz M, modificá-la como descrito acima e achar x.

Para obter a solução vocês devem implementar o algoritmo de escalonamento (eliminação Gaussiana). Nesta implementação é importante notar que a Matriz $M - I$ é singular. Do fato de que sabemos que as soluções têm todas as entradas com mesmo sinal (e portanto não nulas), qualquer uma das variáveis pode ser tomada como a variável dependente (vamos escolher x_n). Para o algoritmo ser mais robusto computacionalmente é conveniente adotar uma estratégia de pivotamento. Isto significa que faremos trocas de linhas não apenas quando um elemento que ocuparia a posição de pivot é nulo, mas iremos trocar de modo a colocar na posição de pivot o elemento de maior modulo dentre os candidatos. O escalonamento de $A = M - I$ segue então o seguinte algoritmo:

Para k de 1 a $n - 1$ faça

Determine i_k tal que $|a_{i_k,k}| = \max_{i \geq k} |a_{i,k}|$

Se $i_k \neq k$ troque as linhas k e i_k da matriz A

Para i de $k + 1$ a n faça

$\alpha_i = a_{i,k}/a_{k,k}$

para j de k a n faça

$a_{i,j} = a_{i,j} - \alpha_i * a_{k,j}$

Fim do para

Fim do para

Fim do para

Ao final deste processo, a última linha da matriz deveria ser nula. Porém, devemos levar em conta que estamos lidando com uma implementação computacional, em que as operações sofrem erros de arredondamento. Assim, em posições da matriz em que teoricamente deveríamos ter zeros, possivelmente teremos números muito pequenos. Para todos os efeitos eles devem ser considerados como nulos. Para a obtenção da solução após o escalonamento, proceda da seguinte forma. Atribua valor arbitrário positivo a x_n (1 por exemplo) e a partir disso calcule:

para k de $n-1$ a 1 com passo -1 faça

$x_k = (-\sum_{j=k+1}^{n} a_{k,j}x_j)/a_{k,k}$

Ao final, resta apenas fazer a normalização da solução para que as componentes tenhm soma 1.

3 Algumas tarefas

Tarefa 1

Você deve escrever um programa computacional que gere o ranking de importância das páginas de uma dada rede. Uma das possíveis entradas para uma rede seria a matriz representando o grafo da rede. Seu programa deve construir a matriz de ligação M, representando a rede e modificá-la como descrito para que satisfaça as propriedades necessárias e então montar a matriz $A = M - I$. O cálculo da solução deve então ser feito através do processo de escalonamento. Como um primeiro teste calcule o ranking de importância para a rede mostrada na figura 3, com 8 páginas:

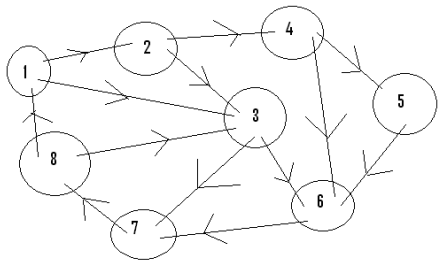

Figure 3: Tarefa 1

A saída do programa deve ser um ranking das páginas desta rede e a importância de cada uma (dada pelos pesos calculados).

Tarefa 2

Uma rede tem uma arquitetura do tipo **Cacique-tribo** se podemos dividir as páginas em k grupos (ou tribos), onde o grupo i possuiu n_i páginas. As páginas de um grupo apontam para todas as outras páginas do mesmo grupo (e logo são apontadas por todas as páginas da tribo) mas apenas uma delas, o cacique do grupo, aponta para páginas fora deste grupo ou é apontada por páginas de fora da tribo. Os caciques de cada um dos grupos apontam também para todos os outros caciques.

Um exemplo de rede com esta arquitetura é dado na figura 4, onde temos 3 grupos com 2, 2 e 3 páginas respectivamente. As páginas 1, 3, e 5 são os caciques dos grupos.

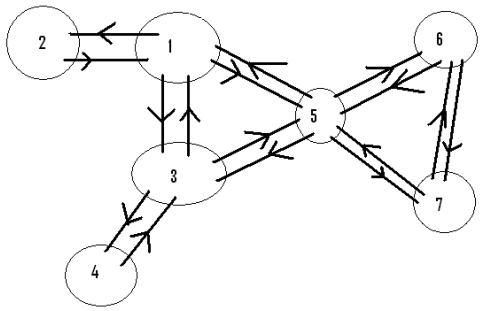

Figure 4: Tarefa 2

Podemos ver que, numa rede com esta arquitetura, cada página do grupo i que não é o cacique aponta para $n_i - 1$ páginas, enquanto que o cacique do grupo i aponta para $n_i - 1 + k - 1$ páginas. O total m de links (ou de entradas não nulas da matriz de ligação M) é de

$$m = \sum_{i=1}^{k} (n_i - 1)^2 + \sum_{i=1}^{k} (n_i - 1 + k - 1),$$

onde a primeira somatória descreve o total de páginas apontadas pelos índios, e a segunda descreve o total apontado pelos caciques.

Nesta tarefa você deve calcular o ranking de importância das páginas de uma rede com 20 grupos com esta arquitetura, onde o primeiro grupo tem 2 páginas (a página 1, cacique do grupo, e a página 2, índia), o segundo grupo tem 3 páginas (a página 3, cacique, e os índios 4 e 5), o terceiro grupo tem 4 páginas e assim por diante, até o vigésimo grupo que possui 21 páginas (a página 210 é o cacique, e os índios são as páginas 211 a 230). O total de páginas nesta rede é $n = 230$.

Teste também o programa com 40 grupos construídos desta mesma forma.

4 Outro método para determinar o vetor x

Vamos descrever agora um outro algoritmo para o cálculo de x, de natureza iterativa, gerando uma sequência de vetores convergindo para a solução do problema. O algoritmo é relativamente simples, a partir de um vetor inicial $x^{(0)}$ (normalizado, por exemplo com $x_i^{(0)} = 1/n$) computa-se a sequência $x^{(k+1)} = Mx^{(k)}$ até se obter convergência, dentro de uma certa tolerância de erro. Antes, definimos a chamada norma 1 de um vetor v de R^n:

$$||v||_1 = \sum_{i=1}^{n} |v_i|$$

e a partir de M com entradas positivas e soma das colunas igual a 1, a seguinte constante

$$c = \max_{1 \le j \le n} \left| 1 - 2 \min_{1 \le i \le n} m_{ij} \right| .$$

Note que $0 < c < 1$, se $n > 2$.

Mostremos agora o seguinte resultado.

Teorema Seja M uma matriz $n \times n$ com todos os elementos positivos e tal que a soma dos elementos de cada coluna de M seja 1. Defina o espaço V dos vetores $v \in R^n$ tal the $\sum_{i=1}^{n} v_i = 0$. Então $Mv \in V$ para todo $v \in V$ e além disso $||Mv||_1 \le c||v||_1$.

Demonstração: Seja $w = Mv$. Então

$$w_i = \sum_{j=1}^{n} m_{i,j} v_j \text{ e portanto}$$

$$\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} \sum_{j=1}^{n} m_{i,j} v_j = \sum_{j=1}^{n} v_j (\sum_{i=1}^{n} m_{i,j}) = \sum_{j=1}^{n} v_j = 0 \ ,$$

e $w \in V$. Agora temos que

$$||w||_1 = \sum_{i=1}^{n} sgn(w_i) w_i = \sum_{i=1}^{n} sgn(w_i) \sum_{j=1}^{n} m_{i,j} v_j$$

onde $sgn(w_i)$ é o sinal de w_i, ou seja, $sgn(w_i) = 1$ se $w_i \ge 0$ e $sgn(w_i) = -1$ se $w_i < 0$. Note que como $w \in V$ nem todas suas componentes têm mesmo sinal. Defina

$$\lambda_j = \sum_{i=1}^{n} sgn(w_i) m_{i,j} \ .$$

Obtemos então que

$$||w||_1 = \sum_{i=1}^{n} sgn(w_i) \sum_{j=1}^{n} m_{i,j} v_j = \sum_{j=1}^{n} v_j (\sum_{i=1}^{n} sgn(w_i) m_{i,j}) = \sum_{j=1}^{n} \lambda_j v_j \ .$$

Como $\sum_{i=1}^{n} m_{i,j} = 1, 0 < m_{i,j} < 1$ e os sinais de w_i não são todos iguais, segue que

$$|\lambda_j| \le 1 - 2 \min_{1,..,n} m_{i,j} < 1 \ \text{ e } c = \max_{j=1,..,n} |\lambda_j| < 1 \ .$$

Obtemos portanto que

$$||w||_1 = \sum_{j=1}^{n} \lambda_j v_j = |\sum_{j=1}^{n} \lambda_j v_j| \leq \sum_{j=1}^{n} |\lambda_j||v_j| \leq c \sum_{j=1}^{n} |v_j| = c||v||_1 \ ,$$

completando a demonstração.

Estamos agora em condição de mostrar que o processo iterativo para o cálculo de x converge. Consideremos $x^{(0)}$ qualquer vetor com componentes positivas e soma 1 (e portanto, $||x^{(0)}||_1 = 1$). A solução de $(M - I)x = 0$ com soma de componentes igual a um é unica, como já vimos. Seja $\bar{x}$ esta solução, a qual desejamos calcular.

Temos que $\bar{x} - x^{(0)} = v \in V$ e que

$$M^k x^{(0)} = M^k(\bar{x} - v) = M^k \bar{x} - M^k v = \bar{x} - M^k v \ ,$$

usando o fato de que $M\bar{x} = \bar{x}$. Do teorema anterior, temos que $||M^k v||_1 \leq c^k ||v||_1$ e como $c < 1$ obtemos que $\lim_{k\to\infty} M^k v = 0$. Segue portanto que

$$\lim_{k\to\infty} M^k x^{(0)} = \bar{x} \ .$$

Podemos ainda delimitar o erro em relação à solução a cada iteração, dado por $e^{(k)} = \bar{x} - x^{(k)}$, da seguinte forma. Como

$$||e^{(k+1)}||_1 = ||\bar{x} - x^{(k+1)}||_1 \leq c||\bar{x} - x^{(k)}||_1 \leq c(||\bar{x} - x^{(k+1)}||_1 + ||x^{(k+1)} - x^{(k)}||_1)$$

obtemos

$$||e^{(k+1)}||_1 \leq \frac{c}{1-c}||x^{(k+1)} - x^{(k)}||_1$$

.

Esta estimativa nos permite delimitar o erro em relação à solução exata a partir da norma da diferença entre duas iteradas consecutivas e do valor de c.

A príncipio pode parecer que calcular $M^k x^{(0)}$ será computacionalmente caro se n é grande, mas veremos que este processo tem algumas vantagens.

Note que se y é um vetor com todas entradas positivas e $\|y\|_1 = 1$, então

$$z = My = (1 - \alpha)M_o y + \alpha S_n y = (1 - \alpha)M_o y + \alpha s$$

onde M_o é a matriz de ligação original, antes de ser modificada e

$$s = \begin{pmatrix} 1/n \\ 1/n \\ . \\ . \\ . \\ 1/n \end{pmatrix}.$$

Mais ainda, z tem todas as entradas positivas e $\|z\|_1 = 1$. Assim, de fato o que é preciso fazer repetidas vezes, é o produto de M_o por um vetor e isto é simples, sendo computacionalmente viável quando M_o é uma matriz esparsa (com relativamente poucas entradas

não-nulas, o que é típico para uma rede de páginas web). A maneira como isto deve ser feito para economizar tempo e memória é explicada a seguir.

Podemos armazenar matrizes esparsas guardando apenas as informações sobre os elementos não nulos da matriz. Para tal podemos utilizar três vetores, cujo tamanho é igual ao número de elementos não nulos. Em um vetor V armazenamos o valor dos elementos não nulos da matriz. Nos outros dois vetores, L e C guardamos a informação sobre a que linha e coluna o respectivo elemento da matriz pertence. Exemplificamos aqui este armazenamento, com a matriz de ligação da rede da figura 1:

$$\begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix}$$

Os três vetores correspondentes à matriz (que percorremos linha por linha) serão:

$$V = (1, 1/2, 1/3, 1/3, 1/2, 1/2, 1/3, 1/2) \ ,$$

$$L = (1, 1, 2, 3, 3, 3, 4, 4) \text{ e}$$

$$C = (3, 4, 1, 1, 2, 4, 1, 2) \ .$$

Veja que no exemplo cacique / tribo com 20 grupos como descrito anteriormente, o número total de ligações (elementos não nulos em M_o) é igual a $m = (\frac{20\times21\times41}{6}) + (210 + 190) = 3270$ enquanto a dimensão da matriz incluindo os zeros seria $(230)^2 = 52900$. Numa rede de grandes dimensões, economiza-se muita memória. Não apenas isso, mas o custo da multiplicação $M_o y$ também será significativamente reduzido.

Para fazer o produto $M_o y$, evitando as multiplicações por zero e armazenando o resultado em um vetor z, executa-se:

para i de 1 a n faça $z(i) = 0$

para k de 1 a m faça

$z(L_k) = z(L_k) + V_k \ y(C_k)$

Inicie as iterações a partir de $x^{(0)} = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})$ e pare o cálculo quando $||e^{(k)}||_1 < 10^{-5}$.

Você deve comparar os resultados e tempo de execução para as duas formas de resolver o problema, com o método de escalonamento e o método iterativo.

Referências

[1] ver o artigo "The 25,000,000,000 eigenvector: the linear algebra behind Google."

[2] Richard Bellman, Introduction to Matrix analysis traz o teorema de Perron e trocentas outras coisas interessantes ...