数学软件——短学期课程
Matlab 第四次作业

姓名：汪利军
学号：3140105707
班级：统计 1401

2016.07.10

目 录

1 迭代法

1.1 算法思路

首先给定一个 PageRank 的初始值 PR0，利用迭代的方式，利用当前的 PR 值通过平摊计算后得到新的 PR 值，若两次的 PR 值差异小于误差限，迭代停止。根据最终的 PR 值对网页进行重要性评比。首先我们采取不归一化的方式来进行迭代。

1.2 代码

```
function [iterations, ranking] = IterationSolvePageRank(G,error)
N = size(G, 1); % number of website
c = sum(G, 1);
ranking = [];
[~, index] = find(c > 0);
for i = index
    G(:, i) = G(:, i)./c(i);
end
iterations = 0;
PR0 = ones(N, 1);
PR = G * PR0;
while norm(PR - PR0,2) > error
    PR0 = PR;
    PR = G * PR0;
    iterations = iterations + 1;
    if iterations > 10000
        disp('10000 步迭代后无法达到指定误差限');
        return;
    end
end
message = [num2str(iterations), '步后达到指定误差限'];
```

```
disp(message);
ranking = sortrows([PR,(1:N)'], -1);
ranking = ranking(:,2);
end
```

1.3 运行结果

运行结果如图 (1) 所示，计算效率如图 (2) 所示

```
>> [iters, ranking] = IterationSolvePageRank(Problem.A',3);
>> ranking(1:20)'

ans =

  Columns 1 through 13

      8226      8059      7741      8057      8225      6837      6839      6840      6838      8227      8060      6197      5287

  Columns 14 through 20

      5253      5636      7494      1572      5637      2674      1609
```

Figure 1: 迭代法运行结果

```
>> tic;IterationSolvePageRank(Problem.A',3);toc;
Elapsed time is 0.060262 seconds.
>> tic;for i = 1:50 IterationSolvePageRank(Problem.A',3);end;toc;
Elapsed time is 2.312845 seconds.
```

Figure 2: 运算效率

从图 (3) 中可以看出, 在一定误差范围内，迭代法的收敛速度还是很快的，但是到达一定下限后就几乎不会收敛了。

```
>> [iters, ranking] = IterationSolvePageRank(Problem.A', 3);
116步后达到指定误差限
>> [iters, ranking] = IterationSolvePageRank(Problem.A', 2.9);
132步后达到指定误差限
>> [iters, ranking] = IterationSolvePageRank(Problem.A', 2.8);
10000步迭代后无法达到指定误差限
>> [iters, ranking] = IterationSolvePageRank(Problem.A', 2.85);
148步后达到指定误差限
>> [iters, ranking] = IterationSolvePageRank(Problem.A', 2.89);
134步后达到指定误差限
>> [iters, ranking] = IterationSolvePageRank(Problem.A', 3);
116步后达到指定误差限
```

Figure 3: 迭代速度

2 迭代法 (归一化)

2.1 算法思路

如果对上述迭代进行归一化，则会保证有唯一解，但运算结果表明不归一化与归一化结果是一致的。

2.2 代码

```
function [iterations, ranking] = IterationSolvePageRank2(G,error)
N = size(G, 1); % number of website
c = sum(G, 1);
ranking = [];
[~, index] = find(c > 0);
for i = index
    G(:, i) = G(:, i)./c(i);
end
iterations = 0;
PR0 = ones(N, 1)./N;
```

```matlab
PR = G * PR0;
PR = PR./sum(PR);
while norm(PR - PR0,2) > error
    PR0 = PR;
    PR = G * PR0;
    PR = PR./sum(PR);
    iterations = iterations + 1;
    if iterations > 100000
        disp('100000 步迭代后无法达到指定误差限');
        return;
    end
end
message = [num2str(iterations), '步后达到指定误差限'];
disp(message);
ranking = sortrows([PR,(1:N)'], -1);
ranking = ranking(:,2);
end
```

2.3 运行结果

可见归一化与没有归一化的结果是一致的。

```
>> ranking(1:20)'

ans =

  Columns 1 through 13

        8226        8059        7741        8057        8225        6837        6839        6840        6838        8227        8060        6197        5287

  Columns 14 through 20

        5253        5636        7494        1572        5637        2674        1609
```

```
>> tic;IterationSolvePageRank(Problem.A',0.0001);toc;
100000步迭代后无法达到指定误差限
Elapsed time is 17.002001 seconds.
>> tic;IterationSolvePageRank(Problem.A',0.005);toc;
20步后达到指定误差限
Elapsed time is 0.039257 seconds.
>> tic;IterationSolvePageRank(Problem.A',0.001);toc;
104步后达到指定误差限
Elapsed time is 0.056712 seconds.
```

3 线性方程组法

3.1 算法思路

考虑矩阵 $G=(g_{ij})$，若从网页 j 到 i 有一个链接，则 $g_{ij}=1$，否则 $g_{ij}=0$. 记第 j 个网页的出度为 $c_j=\sum\limits_i g_{ij}$. 将浏览网页视作马尔科夫过程，在浏览网页 j 时，有一定概率选择点击 j 上的链接浏览网页 i，也有可能从地址栏或者收藏夹直接切换到网页 i. 设浏览 j 时点击其上链接的概率记作 ρ （阻尼因子），网页总量为 N，则从网页 j 到网页 i 的转移概率为

$$a_{ij}=\begin{cases} g_{ij}[\rho\cdot c_j+(1-\rho)\cdot\frac{1}{N}]+(1-g_{ij})[(1-\rho)\cdot\frac{1}{N}]=\frac{\rho g_{ij}}{c_j}+\frac{1-\rho}{N} & 若\quad c_j\neq 0\\ \frac{1}{N} & 若\quad c_j=0\end{cases}\tag{1}$$

记 $A=(a_{ij})$，则

$$A=\rho GD+ez^T\tag{2}$$

其中 D 为对角阵 $diag\{d_1,d_2,\cdots,d_N\}$，$e$ 为行向量 $[1,1,\cdots,1]$,z 为列向量 $[z_1,z_2,\cdots,z_N]^T$

$$d_j=\begin{cases}0 & c_j=0\\ \frac{1}{c)_j} & c_j\neq 0\end{cases}\qquad z_j=\begin{cases}\frac{1}{N} & c_j=0\\ \frac{1-\rho}{N} & c_j\neq 0\end{cases}\tag{3}$$

于是则有

$$Ax=\rho GDx+ez^Tx=x\tag{4}$$

又 $z^T x$ 的值为定值 k 则

$$(I - \rho GD)x = ke \tag{5}$$

但我们只需要相对大小，所以我们可以将上述方程简化为

$$(I - \rho GD)x = e \tag{6}$$

3.2 代码

```
function ranking = LinearEqsSolvePageRank(G)

N = size(G, 1);
c = sum(G, 1);
rho = 0.85;
d = zeros(N, 1);
[~,index] = find(c > 0);
d(index) = 1./c(index);

D = diag(d);
e = ones(N,1);
I = eye(N);
PR = (I - rho*G*D)\e;

ranking = sortrows([PR, (1:N)'], -1);
ranking = ranking(:,2);
end
```

3.3 运算结果

```
>> tic;ranking = LinearEqsSolvePageRank(Problem.A');toc;
Elapsed time is 7.738506 seconds.
>> ranking(1:20)'

ans =

  Columns 1 through 13

       2264        8226        8059        8057        4485        5707        8225        6837        6839        6840        6838        7261        7429

  Columns 14 through 20

       7611        7032        7035        7034        7033        7741        6517
```

Figure 4: 线性方程组法运行结果

前十名排序如下图

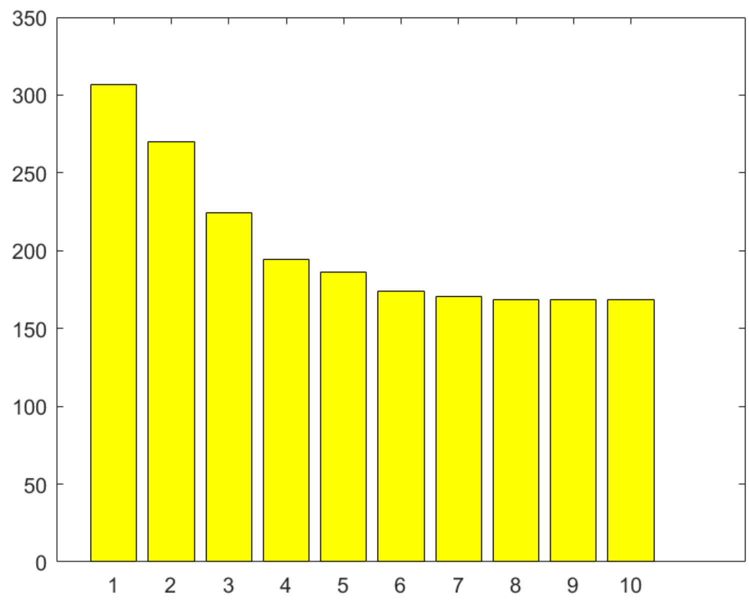

Figure 5: 前十名排名

4 稀疏矩阵存储

4.1 算法思路

对于上述的线性方程组解法，注意到题目提供的矩阵为稀疏矩阵，所以对上述线性方程组的解法中的矩阵用稀疏矩阵存储

4.2 代码

```
function ranking = SpLinearEqsSolvePageRank(G)

N = size(G, 1);
c = sum(G, 1);
[I,J,K] = find(c);
K = 1./K;
d = sparse(I, J, K);
D = diag(d);

rho = 0.85;
e = ones(N,1);
I = speye(N);
PR = (I-rho*G*D)\e;

ranking = sortrows([PR, (1:N)'], -1);
ranking = ranking(:,2);
end
```

4.3 运行结果

运行结果可以发现，运算效率远高于不用稀疏矩阵存储的算法，可见，使用稀疏矩阵不但能够节约内存，还能够加快运算速度。

```
>> tic;SpLinearEqsSolvePageRank(Problem.A');toc;
Elapsed time is 0.391859 seconds.
>> tic;LinearEqsSolvePageRank(Problem.A');toc;
Elapsed time is 7.407177 seconds.
>> tic;for i = 1:50 SpLinearEqsSolvePageRank(Problem.A');end;toc;
Elapsed time is 2.431815 seconds.
>> tic;for i = 1:50 LinearEqsSolvePageRank(Problem.A');end;toc;
Elapsed time is 375.018653 seconds.
```

Figure 6: 稀疏矩阵存储运算效率

5 比较与分析

对于迭代法和线性方程组法进行 PageRank 排序，前 20 基本相同，但次序不完全一样。但迭代法的收敛速度较小，另外如果进行稀疏存储，则会大大加快运算速度，这在更大规模的网页 PageRank 的计算中是很有必要的。

排名	迭代法	迭代法 (归一化)	线性方程组法	排名	迭代法	迭代法 (归一化)	线性方程组法
1	8226	8226	2264	11	8060	8060	6838
2	8059	8059	8226	12	6197	6197	7261
3	7741	7741	8059	13	5287	5287	7429
4	8057	8057	8057	14	5253	5253	7611
5	8225	8225	4485	15	5636	5636	7032
6	6837	6837	5707	16	7494	7494	7035
7	6839	6839	8225	17	1572	1572	7034
8	6840	6840	6839	18	5637	5637	7033
9	6838	6838	6840	19	2674	2674	7741
10	8227	8227	6837	20	1609	1609	6517

Table 2: 结果比较