

APPLIED NUMERICAL METHODS - MATH 151B
Homework 8

Erik Lamb

May 23, 2016

In this assignment, we were to simulate the PageRank algorithm implemented by the Google search engine. This was accomplished by applying three types of Power methods, namely the Power method scaled with the l_∞ -norm, the Power method scaled with the l_2 -norm, and the Inverse Power method, to a simplified graph that models the World Wide Web, given in Figure 1 of the homework. From the graph we constructed an adjacency matrix B , where a matrix element is 1 if a link connects from one node to another and is zero otherwise. We found the following adjacency matrix

$$B = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

Notice that the adjacency matrix is 15×15 due to the fact that there are 15 nodes in Figure 1. In order to find a matrix of probabilities for visiting certain webpages starting from an initial page, we can construct the modified adjacency matrix M , given by the equation

$$m_{ij} = p \left(\frac{b_{ij}}{\sum_{k=1}^n b_{ik}} \right) + \frac{1-p}{n}, \quad (2)$$

where p is the probability of jumping from one webpage to another and is known as the damping factor. In our computations p is set to the typical value of $p = 0.85$. According to Markov's theorem, the left eigenvector of the modified adjacency matrix associated with eigenvalue 1 will be a vector \vec{v} where v_i is the probability that a user will visit page i . The left eigenvector of M is thus given by $\vec{v}^T M = \vec{v}^T$, or after taking the transpose of both sides,

$$\vec{v}^T M^T = \vec{v}. \quad (3)$$

Thus, according to equation (3), we can use the aforementioned power methods to solve for the dominant eigenvector of the transpose of the modified adjacency matrix M , given by equation (2).

This is exactly what we did, using a vector \vec{x} of length 15 contained with random numbers generated by the MATLAB function `rand(n,1)`. The power method using the l_∞ -norm was implemented in the function `power1.m`, the power method using the l_2 was implemented in the function `power2.m`, and the inverse power method was implemented in the function `invpower.m`. For a randomly generated initial approximation \vec{x} , the function `power1.m` found the eigenvector, or webpage ranking,

$$\vec{v} = \begin{pmatrix} 0.108235217965463 \\ 0.116991665925904 \\ 0.137595072891646 \\ 0.0455951376210030 \\ 0.177207515409485 \\ 0.120908331670034 \\ 0.0798524982608460 \\ 0.155165692309877 \\ 0.172081110140075 \\ 0.213672541014263 \\ 0.653991340594072 \\ 0.323539878355427 \\ 1 \\ 0.784082902463547 \\ 0.470594857478657 \end{pmatrix}, \quad (4)$$

with eigenvalue $\lambda = 1$ in 74 iterations. The function `power2.m` found the following eigenvector,

$$\vec{v} = \begin{pmatrix} 0.0675575511528557 \\ 0.0730230936271505 \\ 0.0858831935666677 \\ 0.0284592750867403 \\ 0.110608229114176 \\ 0.0754677724602652 \\ 0.0498418106171488 \\ 0.0968503079906738 \\ 0.107408462968476 \\ 0.133368730537762 \\ 0.408204408394565 \\ 0.201944543800088 \\ 0.624174587047270 \\ 0.489402352454468 \\ 0.293733189855725 \end{pmatrix}, \quad (5)$$

with eigenvalue $\lambda = 1$ and in 71 iterations. Finally, the function `invpower.m` found the eigenvector,

$$\vec{v} = \begin{pmatrix} 0.999996793096402 \\ 0.999995032020625 \\ 0.999994108215595 \\ 0.999993553668276 \\ 0.999996209091863 \\ 0.999995680855634 \\ 0.999994703124169 \\ 0.999994021886903 \\ 0.999994671176961 \\ 0.999995849181133 \\ 1 \\ 0.999991722886313 \\ 0.999994206893762 \\ 0.999991722886313 \\ 1 \end{pmatrix}, \quad (6)$$

with eigenvalue $\lambda = 0.0746$ and in 40 iterations.

From the rankings (4) and (5), we see that page 13 is the most likely page a user will end up on, followed by page 14, page 11, then page 15 and so on. Simply comparing the values of each element and assorting the indices according to the magnitude of each element will give a ranking. Notice that the `power1.m` method has a highest probability of 1 because the vector is normalized by the largest element in it, in accordance with the l_∞ -norm. The `power2.m` method, on the other hand, has a largest probability of 0.624 since the vector is normalized l_2 . Because of this, (5) provides probabilities relative to all elements in the vector, likely giving more realistic results. Both `power1.m` and `power2.m` are stable methods, converging in about the same amount of iterations, around 70. In fact, both methods always converge for a random \vec{x} , and are relatively stable.

The `invpower.m` method, on the other hand, is not stable at all. In fact, the results given are not typical, and the both the eigenvector and eigenvalue approximated by `invpower.m` can change wildly depending on the initial approximation \vec{x} . Sometimes, the Inverse Power method will not even finish after 100000 iterations, indicating that it does not converge. Recall from lecture 22 that the inverse power method does well for symmetric matrices, converging to the $2k$ power. For general cases, such as M , the inverse power method only converges to the k power, and requires a good approximation for q and apparently \vec{x} . Failure to meet these good approximations gives extremely unstable results.

In conclusion, methods `power1.m` and `power2.m` are equally stable and merely provided results with different normalizations. The `invpower.m` is not stable at all and requires both a good q and a good \vec{x} approximation in order to converge to a result. Additionally, `invpower.m` does better with symmetric matrices which M is not. Finally, we notice that the most probable webpages a user will land on are those which have either 3 or 4 links pointing to them, which makes sense logically as there are more paths to those sites.