Report Homework 3

Tatiana Lopez Guevara

November 6, 2013

1 Problem 1

In the problem we have 2 different groups of people: those who read the paper on Day n (g_1) and those who don't buy it (g_2). Therefore, as the suggestion says, we can define the following vector:

$$v_n = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} \in \mathbb{R}^2$$

where g_1 is the size of the group 1 and g_2 is the size of the group 2.

Since we are dealing with Markov Processes, we know that we are modeling the movement of people from one group to another with no gain or loss. This means, the total population is always conserved.

1.1 Question 1

If we belong to g_1 then, on the next day, we will have two possibilities: not purchasing the newspaper or purchasing and reading it. The problem tells us that the probability of reading the newspaper on the next day if we already purchased it its 0.7, which also means that the other option (move to g_2) will happen with a probability of 0.3. By performing the same analysis for the second group (g_2) we obtain the following Markov state graph:

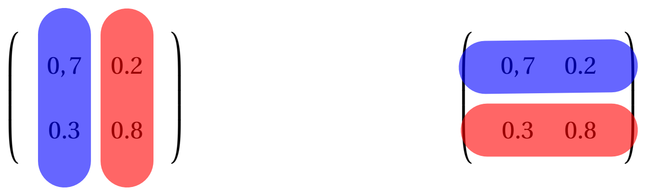


Figure 2: Markov matrix M

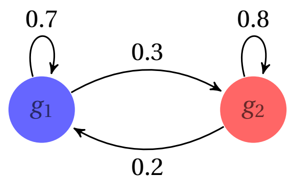


Figure 1: State Transitions

Which in terms of a Markov matrix $\mathrm{M} = (m_{ij})$ becomes figure 2.

As we can see, each column j contains all the possible transitions starting from the state g_j and therefore, since the system is conservative, it must add up to 1 (figure 2 left). Moreover, each row i is the probability to arrive at the state i from any other state (figure 2 rigth).

1.2 Question 2

The matrix M represents which part of the population moves from one state to another or the probability of a single person moving on the next day. This means, the probability to get to state in the day $k+1$ depends on the incoming links to that state from the dat k.

$$\begin{aligned}(g_1)_{n+1} &= 0.7 * (g_1)_n + 0.2 * (g_2)_n \\ (g_2)_{n+1} &= 0.3 * (g_1)_n + 0.8 * (g_2)_n\end{aligned} \tag{1}$$

This can be expressed in a matricial form

$$\begin{aligned}\begin{pmatrix} g_1 \\ g_2 \end{pmatrix}_{n+1} &= \begin{pmatrix} 0,7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}_n \\ v_{n+1} &= \mathrm{M} v_n\end{aligned} \tag{2}$$

1.3 Question 3

As stated before, we are dealing with a $v_{n+1} = \mathrm{M}v_n$ type of problem for which we already know the solution can be expressed as:

$$\begin{aligned} v_{k+1} &= \mathrm{M}^n v_0 \\ v_{k+1} &= \mathrm{S}\Lambda^n \mathrm{S}^{-1} v_0 \end{aligned}$$

We can see that the powers of the matrix M can be calculated using the diagonalized matrix Λ and the eigenvector matrix S. Since this is a Markov matrix we already know that there will be an eigenvalue $\lambda_1 = 1$. To get the other eigenvalue we use the following nice property:

$$\begin{aligned} tr(\mathrm{M}) = \lambda_1 + \lambda_2 &= 0.15 \\ \lambda_2 &= 0.5 \end{aligned}$$

Now, we get the eigenvectors for λ_1:

$$\begin{aligned} \begin{pmatrix} -0.3 & 0.2 \\ 0.3 & -0.2 \end{pmatrix} x_1 &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ x_1 &= \begin{pmatrix} 2 \\ 3 \end{pmatrix} \end{aligned} \tag{3}$$

And the eigenvectors for λ_2:

$$\begin{aligned} \begin{pmatrix} 0.2 & 0.2 \\ 0.3 & 0.3 \end{pmatrix} x_2 &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ x_2 &= \begin{pmatrix} -1 \\ 1 \end{pmatrix} \end{aligned} \tag{4}$$

Packing up everything, we have:

$$\begin{aligned} \Lambda &= \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix} \\ \mathrm{S} &= \begin{pmatrix} 2 & -1 \\ 3 & 1 \end{pmatrix} \\ \mathrm{S}^{-1} &= \begin{pmatrix} \frac{1}{5} & \frac{1}{5} \\ -\frac{3}{5} & \frac{2}{5} \end{pmatrix} \end{aligned}$$

- How likely is to purchase on Day 2? Day 3? Day n?

On the Day n we see that the likelihood of moving from one state to another is given by the n_{th} power of the matrix M.

$$\begin{aligned} v_2 &= \mathrm{M}^2 v_0 \\ v_3 &= \mathrm{M}^3 v_0 \\ v_n &= \mathrm{M}^n v_0 \\ &= \mathrm{S}\Lambda^n \mathrm{S}^{-1} v_0 \end{aligned} \tag{5}$$

The probability to be in the group g_1 on Day n given that today (Day 0) we bought it is the first position of v_n and implies an initial state:

$$v_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

We will apply the equation (5):

$$\begin{aligned} v_2 &= \mathrm{M}^2 v_0 \\ &= (\mathrm{S}\Lambda^2 \mathrm{S}^{-1}) v_0 \\ &= \mathrm{S} \begin{pmatrix} 1 & 0 \\ 0 & 0.25 \end{pmatrix} \mathrm{S}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0.55 \\ 0.45 \end{pmatrix} \end{aligned} \tag{6}$$

Which means there is a 55% of probability that the person buys the paper on Day 2.

For the thrid day we have:

$$\begin{aligned} v_3 &= \mathrm{M}^3 v_0 \\ &= (\mathrm{S}\Lambda^3 \mathrm{S}^{-1}) v_0 \\ &= \mathrm{S} \begin{pmatrix} 1 & 0 \\ 0 & 0.125 \end{pmatrix} \mathrm{S}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0.475 \\ 0.525 \end{pmatrix} \end{aligned} \tag{7}$$

Which means there is a 47.5% of probability that the person buys the paper on Day 3.

$$v_n = (\mathrm{S}\Lambda^n\mathrm{S}^{-1})\begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{8}$$

When n is suficiently large, we get to the stability condition where 40% of the people read the paper.

- What sales figures can *The Computer Visionist* expect on Day 2? Day 3? Day n?

Since we want to know the sales figure, we take into account all the population for the initial state.

$$v_0 = \begin{pmatrix} 750 \\ 250 \end{pmatrix}$$

So we only need to apply again (5)

$$\begin{aligned} v_2 &= \mathrm{M}^2 v_0 \\ &= (\mathrm{S}\Lambda^2\mathrm{S}^{-1})\, v_0 \\ &= \mathrm{S}\begin{pmatrix} 1 & 0 \\ 0 & 0.25 \end{pmatrix}\mathrm{S}^{-1}\begin{pmatrix} 750 \\ 250 \end{pmatrix} \\ &= \begin{pmatrix} 487.5 \\ 512.5 \end{pmatrix} \end{aligned} \tag{9}$$

This means that we expect around 488 people to buy the paper on the second day.

$$\begin{aligned} v_3 &= \mathrm{M}^3 \\ &= (\mathrm{S}\Lambda^3\mathrm{S}^{-1})\, v_0 \\ &= \mathrm{S}\begin{pmatrix} 1 & 0 \\ 0 & 0.125 \end{pmatrix}\mathrm{S}^{-1}\begin{pmatrix} 750 \\ 250 \end{pmatrix} \\ &= \begin{pmatrix} 443.75 \\ 556.25 \end{pmatrix} \end{aligned} \tag{10}$$

Which means we expect around 444 people to buy the paper on Day 3.

$$v_n = (\mathrm{S}\Lambda^n\mathrm{S}^{-1})\begin{pmatrix} 750 \\ 250 \end{pmatrix} \tag{11}$$

This is the expression for a given day n. However if n is very large ($n \to \infty$) then we get to the stability condition described in the following section.

- Will the sales figure fluctuate a great deal from day to day, or are they likely to become stable eventually?

We want to know the stability of the system. Therefore we must analyse what will happen with M in the Day $n \to \infty$.

As we could see in the last equation of (5), if we take $n \to \infty$ the only value that will prevail in Λ will be the eigenvalue 1. The other value is less than one so after every power it will become lower and lower and eventually will turn out to be 0.

$$\begin{aligned} M^{\infty} &= S \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}^{\infty} S^{-1} \\ &= \frac{1}{5} \begin{pmatrix} 2 & 2 \\ 3 & 3 \end{pmatrix} \\ &= \frac{1}{5} [x_1 x_1] \end{aligned} \tag{12}$$

As expected, we obtain a matrix with the corresponding eigenvector of the eigenvalue = 1 and this is called the steady state.

$$\begin{aligned} v_{\infty} &= M^{\infty} v_0 \\ &= \frac{1}{5} \begin{pmatrix} 2 & 2 \\ 3 & 3 \end{pmatrix} \begin{pmatrix} 750 \\ 250 \end{pmatrix} = \begin{pmatrix} 400 \\ 600 \end{pmatrix} \end{aligned}$$

So the sales figure **will** become stable and the expected number of people that will read the newspaper is 400.

2 Google Page Rank

The google page rank algorithm is a nice and powerful application of linear algebra. The stated problem was how to display the obtained results in a given order such that the most important results relative to the user are shown first. The relevance of a website is given by its connection with other websites.

In the first section of [1] a very simple strategy is used: just create and adjacency matrix A_{NxN}, whose entries describe the connection between N different websites. This means, if the cell $a_{ij} = 1$ then there is a link from the website j to the website i and 0 means there is not. The value of a page i is then defined as the sum of all the references it has, which means a sum of the entries of the $row_i(A)$. This, expressed in a matricial form becomes:

$$v_i = \mathrm{P}u\,,\ u \in \mathbb{R}^{\mathrm{N}}$$

Where u is an unitary vector.

The second improvement is to transform the matrix A into a Markov matrix P by dividing each column with its sum. This step is performed in order to assign a relevance to each recommender which is inversely proportional to the number of links it has. So, if one page has a lot of outgoing links, then it is less important than one that has only a few. Also, to avoid a division by zero, the dead ends are assumed to have the lowest possible value because they are modified to have a reference to all web sites.

After that, a weight is given to each recommendation, and this means that the matrix is multiplied by v since this vector, as they describe, is the value of each page.

$$v = \mathrm{P}v$$

By definition, the previous formula is stating that exists an eigenvalue = 1. Of course, since P is a Markov matrix!.

2.1 Eigenvalue Problem

The eigenvalue problem described occurs when we get more than one eigenvalue = 1. This is due to clustering and thus the previous formula will not have a unique solution. Also, since they are dealing with the entire web, there will be a lot of zero entries. Here they apply a very nice trick to get a new Markov matrix whose entries are all >= 0 which is guaranteed to have the following properties for the eigenvector v corresponding to the $\lambda = 1$:

- All entries of v are positive
- The sum of v add up to 1
- There exists an unique eigenvector for the $\lambda = 1$

The performed trick is just to add a "jumping" factor to a random page called r. This value is usually set up to 0.85 for Google.

This means that with an r percent of probability the user will use the links between the pages, but he can also jump (with an $1 - r$ percent of probability) to go to a random site modeled by a Matrix T whose entries all have the same probability to be chosen.

$$\begin{aligned} T &= ones(N,N)/N \\ Q &= rP + (1-r)T \end{aligned} \tag{13}$$

A final nice trick is performed to calculate this eigenvector, that is to say, the rank of the webpages which is related to another cool property of the matrix Q:

- Given any vector w whose entries are positive and add up to 1, then $Q^k w$ converges to v when $k \to \infty$.

This is because, as we saw in the first section, when k is sufficiently large, we reach the stability condition no matter what is the initial condition the result will converge to the eigenvector of λ_1

This allows to avoid the high complexity calculation of the eigenvector by the traditional method, and instead use a numerical iterative method to approximate the vector v. The only thing needed is an initial random vector w that has such property, and sequentialy calculate $Q^k w$ until we find two results that are very similar, which means it has converged.

$$\begin{aligned} Qw &= rPw + (1-r)Tw \\ &= rPw + \frac{1-r}{N}u \end{aligned}$$

Some other simplifications were made: Tw is the average of entries of w so it is replaced by a vector of ones divided by N. This new term does not depend on w which means the calculation can be performed only once (before the iterations). The Pw term is really fast to calculate because P is a sparse matrix, which means it has a lot of zeros.

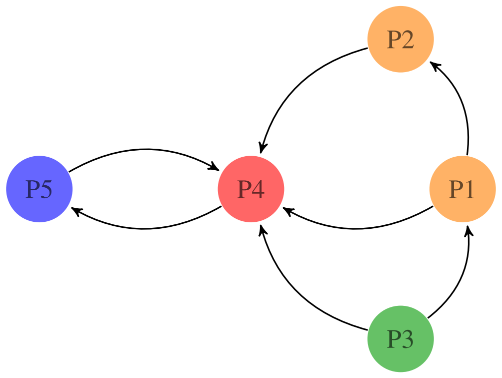
2.2 Example

A matlab code implementation is attached to this report. It takes the adjacency matrix A (any size) and calculates the matrix P, Q and the vector v. It also checks the given approximation of v by actually obtaining the eigenvector of λ_1 with the traditional method. Finally, it shows a graph with each page represented by a circle of a size proportional to the relevance it has.

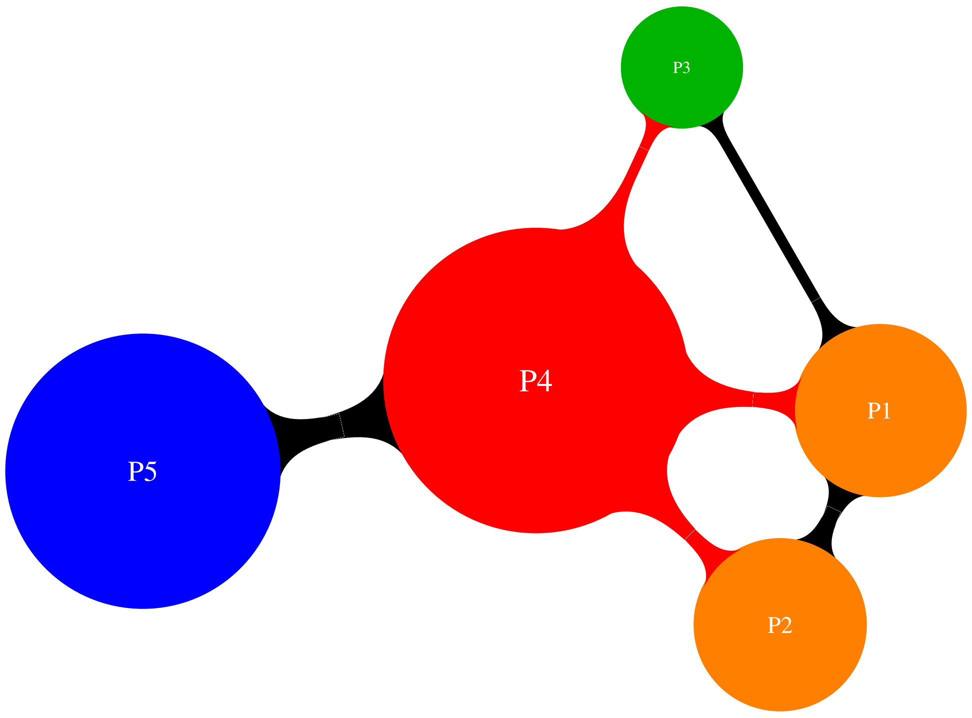
The following matrix was used as the requested 5x5 example:

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

Which is represented by the following graph:



By performing the RankPage algorithm we expect to find that the relevance of the 4th page is greater than all the others and that since the 5th page is referenced by page 4th, then it must also be very relevant. The expected results are shown in the following figure:



The obtained results after running the code are:

$$P = \begin{pmatrix} 0 & 0 & 0.5000 & 0 & 0 \\ 0.5000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.5000 & 1.0000 & 0.5000 & 0 & 1.0000 \\ 0 & 0 & 0 & 1.0000 & 0 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0.0300 & 0.0300 & 0.4550 & 0.0300 & 0.0300 \\ 0.4550 & 0.0300 & 0.0300 & 0.0300 & 0.0300 \\ 0.0300 & 0.0300 & 0.0300 & 0.0300 & 0.0300 \\ 0.4550 & 0.8800 & 0.4550 & 0.0300 & 0.8800 \\ 0.0300 & 0.0300 & 0.0300 & 0.8800 & 0.0300 \end{pmatrix}$$

$$w_{approx} = \begin{pmatrix} 0.0428 \\ 0.0482 \\ 0.0300 \\ 0.4589 \\ 0.4201 \end{pmatrix}$$

$$v_{real} = \begin{pmatrix} 0.0427 \\ 0.0482 \\ 0.0300 \\ 0.4590 \\ 0.4201 \end{pmatrix}$$

We can observe that the two obtained vectors (the approximated through the iteration w_{approx}, and the real eigenvector v_{real}) are very similar. The ϵ factor used as a stop condition was 10^{-4}, that is to say, the algorithm stops until the sum of the individual errors between two consecutive obtained vectors is less or equal than ϵ.

References

[1] Brian White. How google ranks web pages. *Math 51 Lecture Notes*, pages 1–4, 2004.

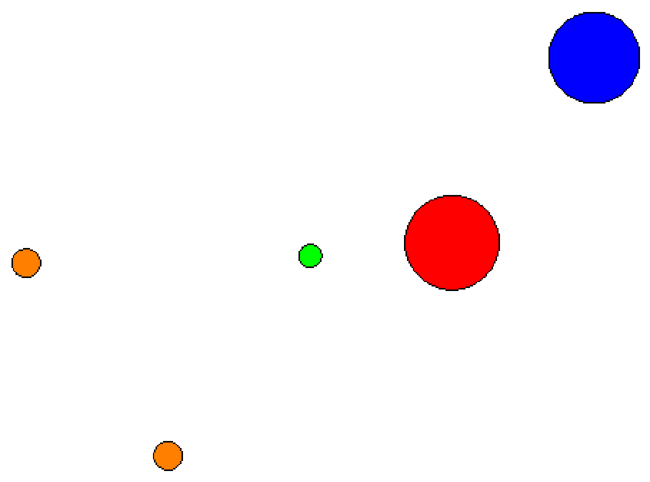


Figure 3: Plotted ranking of Pages ordered from 1 to 5 (left to right)