APPM 2360: Network Approach to Popularity Project 3

December 4, 2014

Souneth Ly (103036031) (L140) (R241)

Zhi Huang (102298433) (L150) (R242)

William Sear (101107069) (L160) (R251)

Contents

1 Adjacency Matrix Properties **2**

2 Example Matrix of Book Citations **2**

3 Ranking Students **3**

4 Gaming the System and Algorithim Verification **6**

5 Examples of Other Adjacency Matrix Applications **7**

6 Appendix A: General Solution Code **7**

7 Appendix B: Power Method **8**

Part 1

Adjacency Matrix Properties

In order to better understand the behaviour of the adjacency matricies we will review important characteristics of those matricies. First and foremost we know that our friendship matrix exhibit the property shown in equation 1 below.

$$M_{i,j} = M_{j,i} \tag{1}$$

This property means that our matrix must be symmetric as the values below the diagonal are a mirror to the values above the diagonal. It makes sense that the friendship matrix exhibits this property because if one person is freinds with another the other person must be freinds with them, facebook does not allow one person to be a friend of someone who does not accept their freind request. This behaviour is characteristic of an undirected graph, where both elements are connected in both directions of travel.

The friendship matrix exhibits significantly different behaviour from the matrix built by Google's page rank algorithim because Google's algorithim relies on counting links between pages. A link from one page to another may not be reciprocated (i.e. the link leads to a page that does not link back to the original site). This is characteristic of an undirected graph, one where an edge allows for travel in one direction but not the other. By this definition Google's algorithim is directed.

Part 2

Example Matrix of Book Citations

In order to better understand the usefulness of eigenvectors and eigenvalues we investigated determining the relative importance of 6 books that cited each other. Our goal was to define which books were the most "valuable" which meant cited by the most books that were cited by the most books. In order to define a matrix for the citations we made the columns correspond to the book and the rows to the books that book cites. As we are dealing with citations the graph is not symmetric as a citation between two books is not necessarily mutual which implies the resulting matrix will not be symmetric. As such this graph is undirected. Based on the data given to us we defined the below table 1 based on these rules.

X	1	2	3	4	5	6
1	0	0	0	0	1	0
2	0	0	0	0	1	1
3	0	1	0	0	0	1
4	1	0	1	0	1	0
5	1	1	0	1	0	0
6	1	1	1	0	0	0

Table 1: Book Citation Matrix

We then used the power method in order to find the eigenvector that results from the maximum eigenvalue. Based on our calculations we find the eigenvalue to be 2.2640 and the resulting eigenvector is shown below:

Book	EigenVector Value
1	-0.2132
2	-0.4072
3	-0.3738
4	-0.4725
5	-0.4827
6	-0.4391

Table 2: Eigenvector for Book Citation Matrix

The eigenvector values given in the above table 2 are the centrality scores for each of the book. In order to rank the scores we take the absolute value of the each eigenvector value and compare them, with larger absolute values corresponding to more influential books. Based on our results book 5 is the most influential followed by book 4, then book 6, then book 2, then book 3, and finally book 1 is the least influential.

Part 3

Ranking Students

In order to make use of the power method algorithim built and used in the previous Part II we performed the same analysis on a larger dataset. The dataset chosen was an undirected graph that defined the freindships of a group of high school students on facebook we were given a 30 by 30 symmetric matrix describing the relationships between 30 students and were then asked to add the information of three other students who were running for class president. This gave us a 33 by 33 symmetric matrix which defined the friendships between different students which was defined such that a 1 for any row and column meant that those two students were friends on Facebook and a 0 for any row and column meant those two students were not freinds on Facebook.

Having defined our matrix we were then able to conduct analysis of it using the power method algorithim. Using relevant data products from this analysis we were able to calculate the rate of convergence using equations 2 and 3.

$$e_k = |\vec{b_k} - \vec{b^*}| \tag{2}$$

Equation 2 allows us to calculate the error between the intermediary vector $\vec{b_k}$ and the final eigenvector $\vec{b^*}$. Using a matrix of all the intermediary vectors from the power method algorithim we could then calculate the error at each stage of the power method.

$$r = \frac{e_{k+1}}{e_k} \tag{3}$$

Once we had established the error at each stage we were then able to use equation 3 to calculate the rate of convergence. Using these two equations we found the rate of convergence to be ~ 0.7327.

In order to better understand the behaviour of the power method model we then ploted the error at each iteration of the method on a semilog plot and the base 10 logarithim of the rate of convergence at each iteration on a standard plot. The resulting plot is shown below in figure 1.

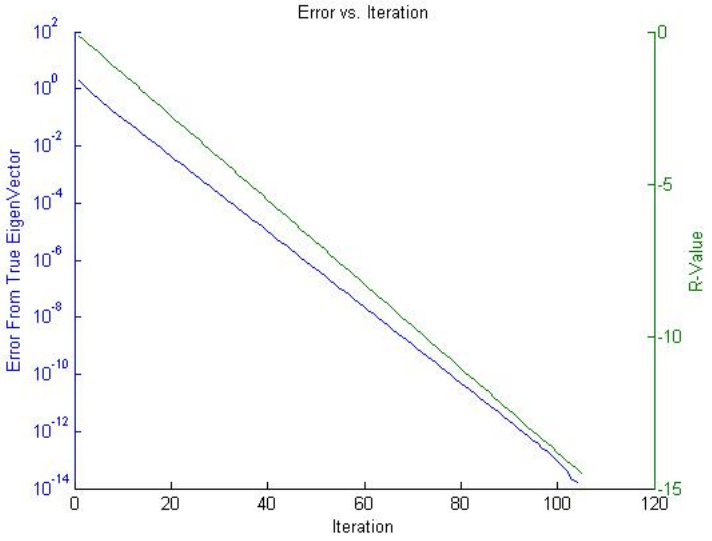


Figure 1: Error and Rate of Convergence Plot

In the plot shown above it can be clearly seen that both the error between the eigenvectors and the rate of increase of the base 10 logarithim of the rate of convergence are linear. The only exception is in later iterations where the larger slope of the rate of convergence forces the error to exhibit an exponential decay due to the rate of increase in the rate of convergence. This makes sense because a linear rate of convergence must necessarily drive an exponential decrease in the error from the true eigenvector which becomes linear when graphed using a semilog plot. Using the power method we found the eigenvector shown in table 3.

Student	EigenVector Value
1	-0.1317
2	-0.2636
3	-0.3106
4	-0.1367
5	-0.2479
6	-0.3466
7	-0.1367
8	-0.2290
9	-0.2525
10	-0.1107
11	-0.0353
12	-0.1328
13	-0.1455
14	-0.1070
15	-0.1051
16	-0.1033
17	-0.1369
18	-0.1011
19	-0.0999
20	-0.0341
21	-0.0572
22	-0.0596
23	-0.0262
24	-0.0267
25	$7.5188*10^{-19}$
26	-0.1153
27	-0.0341
28	-0.1043
29	-0.0088
30	-0.1512
31	-0.5132
32	-0.1327
33	-0.1364

Table 3: Eigenvector for Student Friendship Matrix

Based on table 3 we found that the students 1-30 are ranked in order of greatest to least are shown in the below table:

Rank	Student
1	6
2	3
3	2
4	9
5	5
6	8
7	30
8	13
9	17
10	7
11	4
12	12
13	1
14	26
15	10
16	14
17	15
18	28
19	16
20	18
21	19
22	22
23	21
24	11
25	20
26	27
27	24
28	23
29	29
30	25

Table 4: Students 1-30 Ranked by Popularity

We then consider the special case of the rankings of the candidates. Based on the numbers found in table 3 we can conclude that P31, Mary wins with P33, Veronica and P32, Fred taking second and third respectively.

Part 4

Gaming the System and Algorithim Verification

In order to cheat the system and gain a higher centrality score a student has several options available to them. One would be to create fake accounts and then have those fake accounts make friends with each other,

thus camoflauging them from the real accounts. Another option would be to only make friends with a few very popular people, then the algorithim will consider that person popular, even though the majority of the class may not like that person.

In order to prevent this kind of cheating from occuring two steps must be taken. First, the matrix governing friends must be limited so that only students in the class are considered as in the Part III. After only students in the class are present the final winner is determined not by the centrality score but by their centrality score multiplied by the total number of freinds they have in the class. This means that a student with a few very popular friends who might outscore a student with many less popular friends will not because we are making sure that the student elected is the most popular to the most number of students.

I would not picked the same students to be class president agian, because the centrality score only tells us how popular the student we selected for class president is. It does not take into account that the performance of the student as class president. So in order to make the system more balacned, we could have students rate the capability of each candidate (i.e. their ability to work with students to resolve problems, lead the class cabinet meetings and organize student acitivities and events.) We can then take this new average score and multiply that by the centrality score, compare the magnitude of the product to determine whoever is the most capable and popular.

Part 5

Examples of Other Adjacency Matrix Applications

Adjacency matrices can be used to model a large number of systems. Examples include international trading, international airtravel routes, traders in the stock exchange, and text message. International trade is an example of an adjacency matrix for a directed graph because every country has other countries that it exports to and other countries that it imports from. By creating an adjacency matrix of what countries each nation trades with and in what way we can use the centrality vetor to define which nation does the most trade with other nations. Airtravel routes are an example of an adjacency matrix because we can create a directed graph of every flight that links airports together (i.e. a flight from Denver international airport to the airport in Las Vegas links DIA and that airport). By using the centrality scores found using the power method we can find which airport is most important in moving passengers around the globe. Traders in a stock exchange are an example of an adjacency matrix made from an directed graph because each trader will sell or buy from each other trader. By making the adjacency matrix every trader that a trader has bought from our centrality scores become the trader that handles the most important stock sales. Finally text messages are an example of an adjacency matrix because you can build a matrix of every person in a group of friends who has texted each other. Using the power method on this matrix will create centrality scores that will show which friend has the most text conversations with the group.

Part 6

Appendix A: General Solution Code

Please See Attached

Part 7

Appendix B: Power Method

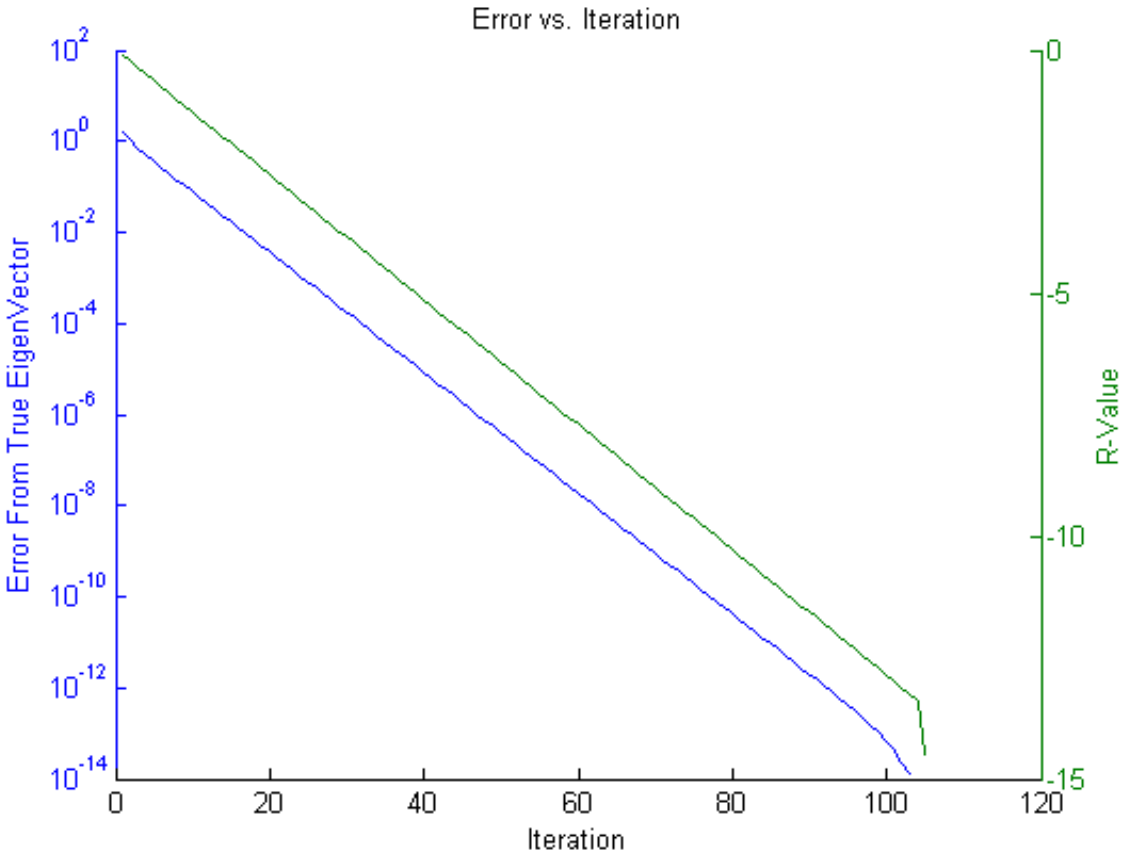
Please See Attached

```
%Question 4
                %1, 2, 3, 4, 5, 6,
bookAdjacency = [0, 0, 0, 0, 1, 0;%1
                 0, 0, 0, 0, 1, 1;%2
                 0, 1, 0, 0, 0, 1;%3
                 1, 0, 1, 0, 1, 0;%4
                 1, 1, 0, 1, 0, 0;%5
                 1, 1, 1, 0, 0, 0]%6
[v,e]=eigs(bookAdjacency, 1)

%Create student matrix
load('student_adjacency.txt')
studentMatrix = zeros(33, 33);
addedZeros = zeros(30,1);
addedZeros2 = zeros(1, 33);
student_adjacency = horzcat(student_adjacency, addedZeros);
student_adjacency = horzcat(student_adjacency, addedZeros);
student_adjacency = horzcat(student_adjacency, addedZeros);
student_adjacency = vertcat(student_adjacency, addedZeros2);
student_adjacency = vertcat(student_adjacency, addedZeros2);
student_adjacency = vertcat(student_adjacency, addedZeros2);
studentMatrix = student_adjacency + studentMatrix;

%add connections
%mary
studentMatrx(1,31)= 1;
studentMatrix(31,1) = 1;
studentMatrix(2,31) = 1;
studentMatrix(31,2) = 1;
studentMatrix(2,31) = 1;
studentMatrix(31,3) = 1;
studentMatrix(3,31) = 1;
studentMatrix(31,4) = 1;
studentMatrix(4,31) = 1;
studentMatrix(31,5) = 1;
studentMatrix(5,31) = 1;
studentMatrix(31,6) = 1;
studentMatrix(6,31) = 1;
studentMatrix(31,7) = 1;
studentMatrix(7,31) = 1;
studentMatrix(31,8) = 1;
studentMatrix(8,31) = 1;
studentMatrix(31,9) = 1;
studentMatrix(9,31) = 1;

%Fred
studentMatrix(32,10) = 1;
studentMatrix(10,32) = 1;
studentMatrix(32,11) = 1;
studentMatrix(11,32) = 1;
studentMatrix(32,12) = 1;
studentMatrix(12,32) = 1;
studentMatrix(32,13) = 1;
studentMatrix(13,32) = 1;
studentMatrix(32,14) = 1;
```

```
studentMatrix(14,32) = 1;

%Veronica
studentMatrix(33,15) = 1;
studentMatrix(15,33) = 1;
studentMatrix(33,16) = 1;
studentMatrix(16,33) = 1;
studentMatrix(33,17) = 1;
studentMatrix(17,33) = 1;
studentMatrix(33,18) = 1;
studentMatrix(18,33) = 1;
studentMatrix(33,19) = 1;
studentMatrix(19,33) = 1;

%Use Power Method
[domV, domGamma, bkVec] = powerMethod(studentMatrix);
%Find error at iteration k
[row, col] = size(bkVec);
for k = 1:1:row;
    ekVec(k,1) = k;
    ekVec(k,2) = norm(transpose(bkVec(k,:))-domV);
end

rateConv = ekVec(51,2)/ekVec(50,2)
for i = 1:1:row;
    rval(i, 1) = (log(rateConv)/log(10))*i;
end
[hAx,hLine1,hLine2] = plotyy(ekVec(:,1),ekVec(:,2),ekVec(:,1),rval,'semilogy','plot');
xlabel('Iteration')
ylabel(hAx(1),'Error From True EigenVector')
ylabel(hAx(2),'R-Value')
title('Error vs. Iteration');
for(i=1:1:33)
    vert(i,1) = i;
end
sortedRows = horzcat(domV, vert);
sortedPeople = sortrows(sortedRows, 1)
```

```
bookAdjacency =

     0     0     0     0     1     0
     0     0     0     0     1     1
     0     1     0     0     0     1
     1     0     1     0     1     0
     1     1     0     1     0     0
     1     1     1     0     0     0


v =

   -0.2132
   -0.4072
   -0.3738
   -0.4725
   -0.4827
```

```
    -0.4391


e =

    2.2640


rateConv =

    0.7437


sortedPeople =

         0   25.0000
    0.0353   29.0000
    0.1051   23.0000
    0.1070   24.0000
    0.1367   20.0000
    0.1367   27.0000
    0.1415   11.0000
    0.2290   21.0000
    0.2387   22.0000
    0.4000   19.0000
    0.4050   18.0000
    0.4136   16.0000
    0.4176   28.0000
    0.4209   15.0000
    0.4283   14.0000
    0.4433   10.0000
    0.4619   26.0000
    0.5276    1.0000
    0.5312   32.0000
    0.5317   12.0000
    0.5463   33.0000
    0.5473    4.0000
    0.5473    7.0000
    0.5482   17.0000
    0.5825   13.0000
    0.6057   30.0000
    0.9169    8.0000
    0.9925    5.0000
    1.0110    9.0000
    1.0554    2.0000
    1.2439    3.0000
    1.3879    6.0000
    2.0551   31.0000
```



Published with MATLAB® R2013b

```
function [maxV, maxGamma, bkVec] = powerMethod(A)

%%%Initial Values%%%
%initial absolute gamma
absGammaCorrected = 1;
%initial x
bx = rand(33,1);
%initial gamma
gamma = 10;

%%%Calculation of EigenVector and EigenValue%%%
index = 1;
bkVec = zeros(1,33);
while (absGammaCorrected > 0)
    bk1 = A*bx;
    absGammaCorrected = abs(norm(bx)-norm(gamma));
    gamma = norm(bx);
    bx = bk1/gamma;
    bkVec = vertcat(bkVec, transpose(bx));
    index = index + 1;
end
bkVec = bkVec(2:end,1:end);
maxV = bx;
maxGamma = gamma;
end
```

```
Error using powerMethod (line 15)
Not enough input arguments.
```

Published with MATLAB® R2013b