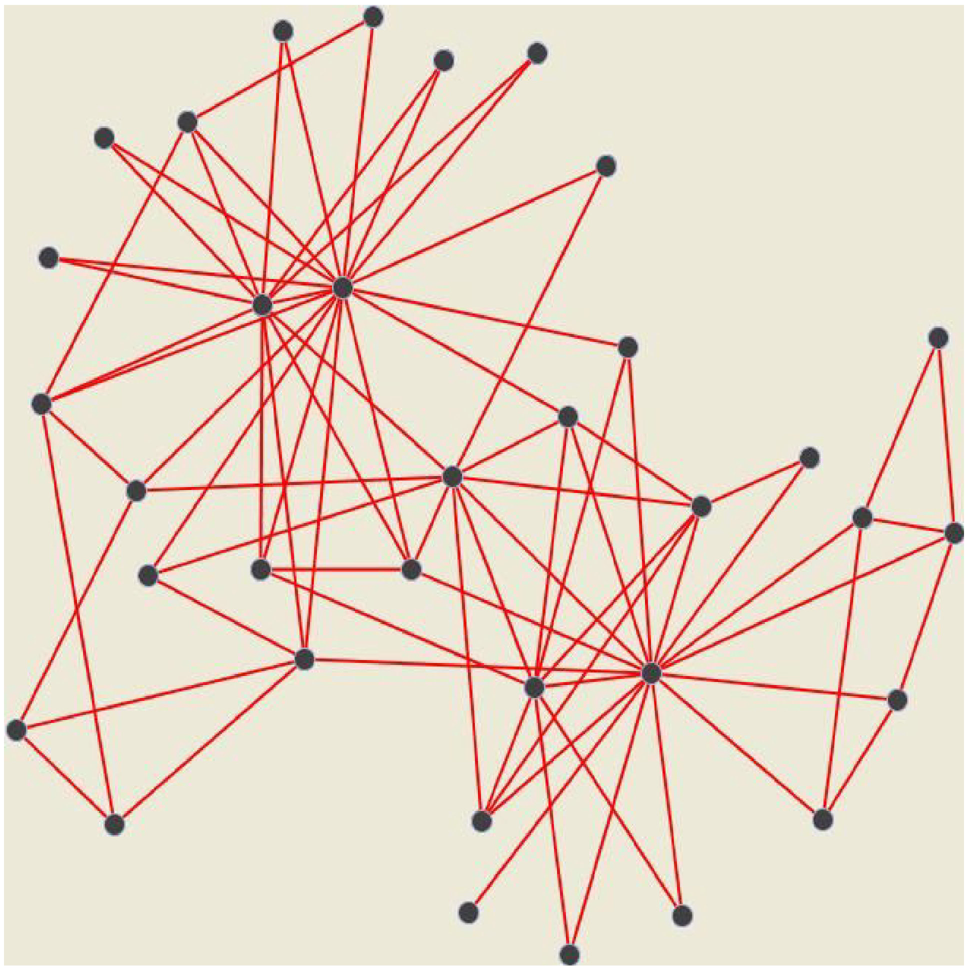
Lab 3: A Network Approach to Popularity
APPM 2360



1 Instructions

Labs may be done in groups of **3** or less. You may use any programming language, but the TAs will only answer coding questions in MATLAB. One report must be turned in for each group and must be in **PDF** format. Labs must include each student's:

- Name
- Student number
- Section number
- Recitation number

This lab is due on **Friday, December 5, 2014** at **5pm**. Each lab must be turned in through D2L on the course page. When you submit the lab please include each group member's information (name, student number, section number, and recitation number) in the **comments**. This allows us to search for a student's report. **Late labs will not be accepted**. Once the labs are graded you need to specifically check that your grade was entered. If you are missing a grade please bring the issue up with your TA within a week of grading.

The report must be typed (including equations). Be sure to label all graph axes and curves so that, independent of the text, it is clear what is in the graph. Simply answering the lab questions will not earn you a good grade. Take time to write your report since up to 20% of your grade may be based on organization, structure, style, grammar, and spelling. Project office hours will be held in ECCR 143 from Monday, December 1 until Friday, December 5.

Introduction

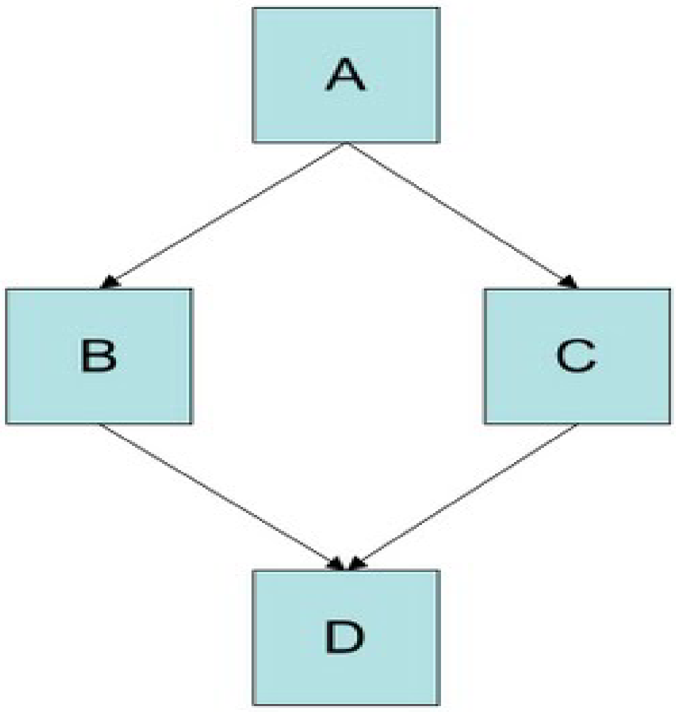
Last year, your high school was shocked when the election for class president was rigged by an unscrupulous student. While this may still bode well for the student's chances in American politics, the school wants to take steps to prevent such a thing from happening again. The principal has decided to move from an election to an objective ranking system. However, the president will still be chosen by popularity alone. How can we rank the students' popularity without voting? By utilizing the power of Facebook, as well as the Page Rank algorithm, we will be able to observe just how popular each candidate is. In this lab, you will learn about the eigenvector centrality method, a tool from network theory used to rank the importance of "nodes" in the network. The method has been used in many different fields, most notably the Page Rank algorithm used to rank Google search results. In our case, it will allow us to impartially rank the popularity of the candidates for class president.

Which friends are most significant?

The first idea to determine popularity may be simply to count the number of Facebook friends each candidate has on Facebook. However, this method has a few weaknesses. It treats each friend as equal. That is, whether the friend is your little brother or the captain of the cheer squad, the impact is the same. It could also be influenced by making fake accounts and adding them as friends. Thus, we want to be able to take into account the impact of each friend. Fortunately, the field of graph theory is able to provide a tool to do this, known as eigenvector centrality. This is the same tool used in the original Google "Page Rank" algorithm.

Centrality

In graph or network theory, there are several types of centrality. Centrality is essentially a determination of the relative importance of a vertex within a graph. An example of a graph is below:



A, B, C, and D are the vertices or nodes of the graph which represents a network. As pictured, the graph is *directed*. This means that there is a connection from A to B, but not from B to A. This describes networks such as power grids, where energy may only be flowing in one direction. If, instead, we ignore the arrows and think of the connections as lines, we have an *undirected* graph. This describes the election scenario, since if I am Facebook friends with you, you must also be Facebook friends with me. For the election, each student is a node, and the connections are Facebook friendships. We wish to find which node is most significant. We will use a method known as Eigenvector Centrality. We can measure the importance of a student in our network by creating an *adjacency matrix*.

Adjacency matrix

In network theory, an adjacency matrix represents which nodes in a network are connected. Assume that the four nodes above are four students, and the connections are not directed. Our adjacency matrix will be a 4x4 matrix. Lets call the matrix M. Node A is student 1. Node B is student 2. Node C is student 3, and node D is student 4. When we look at the indices of M_{ij}, the (i, j) entry will either have a 1 or a 0. If there is a friendship (i.e. connection) between the i^{th} and j^{th} student, then the entry in the (i, j) position will be a 1. If there is no friendship, it will be a 0. The adjacency matrix of the undirected version of the network above is given below.

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

Notice, the matrix has zeros down the diagonal. This should be quite natural since a student is not friends with themselves, which is what the (i, i) entry of 1 represents. However, there should be another important trend you note. If the (i, j) entry has a connection, then the (j, i) entry does as well.

Eigenvector Centrality

Now that we can construct an adjacency matrix, we can finally use it to find the eigenvector centrality. For the i^{th} student, let the centrality score be proportional to the sum of the scores of all students which are connected to it. Therefore, we get

$$x_i = \alpha \sum_{j \epsilon X_i} x_j.$$

Here, x_i is the score for the i^{th} student, and X_i is the set of all students connected to the i^{th} student. Note that this set is also represented by the connections on the i^{th} row of the adjacency matrix. Thus, if we sum each entry on the i^{th} row, we will only add the scores of the students we are connected to. All other students will have their scores multiplied by zero and added for no effect. It looks like this

$$x_i = \alpha \sum_{j=1}^{N} M_{i,j} x_j.$$

Here, N is the number of students in your network. In vector notation, we can rewrite this as $x = \alpha M x$. We can move α to the left hand side, and we get

$$\frac{1}{\alpha} x = Mx.$$

Since α is just a proportionality constant, we can replace $\frac{1}{\alpha}$ with λ. Thus, we see that the vector of the scores of each student is an eigenvector. In fact, it is the eigenvector with the largest eigenvalue, although this fact is not apparent. The larger the magnitude of the value, the more popular the student!

Finding Eigenvectors in Matlab

The `eigs` command in Matlab will calculate the eigenvectors and eigenvalues of a sparse matrix. If you enter

```
[v,e] = eigs(A,1)
```

then `v` will be the eigenvector with the largest eigenvalue, and `e` will be that eigenvalue. You can experiment by increasing the second argument to get more eigenvalues, but we won't use them in this lab.

Now we can do it ourselves!

You didn't think you would be able to get off so easily, just using a Matlab command, did you? We will now use the Power Method to find the eigenvector with the the largest eigenvalue ourselves! This method is pretty magical. If we have a sufficiently "nice" matrix, we can find this eigenvector by starting with a random vector and repeatedly multiplying by the adjacency matrix. This works because the eigenvectors and eigenvalues are a way of "breaking down" what the matrix is doing to each part of a vector. For example,

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

has eigenvalues 1,2,3, with corresponding eigenvectors:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

We can write any vector $\vec{\mathbf{x}}$:

$$\vec{\mathbf{x}} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Thus,

$$\mathbf{A}\vec{\mathbf{x}} = \mathbf{A}\left(a\begin{pmatrix}1\\0\\0\end{pmatrix} + b\begin{pmatrix}0\\1\\0\end{pmatrix} + c\begin{pmatrix}0\\0\\1\end{pmatrix}\right) = a\mathbf{A}\begin{pmatrix}1\\0\\0\end{pmatrix} + b\mathbf{A}\begin{pmatrix}0\\1\\0\end{pmatrix} + c\mathbf{A}\begin{pmatrix}0\\0\\1\end{pmatrix} = a\begin{pmatrix}1\\0\\0\end{pmatrix} + 2b\begin{pmatrix}0\\1\\0\end{pmatrix} + 3c\begin{pmatrix}0\\0\\1\end{pmatrix},$$

$$\mathbf{A}^n\vec{\mathbf{x}} = (1)^n a\begin{pmatrix}1\\0\\0\end{pmatrix} + (2)^n b\begin{pmatrix}0\\1\\0\end{pmatrix} + (3)^n c\begin{pmatrix}0\\0\\1\end{pmatrix}.$$

Notice how we used the definition of an eigenvector. As we increase n, the vector with the 3^n scaling is "taking over" the other two eigenvectors, because it has the largest eigenvalue. We don't want our vectors getting so big, so, after each multiplication, we rescale so that the vector is length 1. We repeat this process until the vector stops changing when multiplied by $\mathbf{A}$. In math symbols, we start with a random $\vec{\mathbf{b}}_0$, then repeat

$$\vec{\mathbf{b}}_{k+1} = \frac{\mathbf{A}\vec{\mathbf{b}}_k}{\left\|\mathbf{A}\vec{\mathbf{b}}_k\right\|},$$

where the norm of a column-vector $\vec{\mathbf{v}}$ is defined by

$$\|\vec{\mathbf{v}}\| = \sqrt{\vec{\mathbf{v}}^T\vec{\mathbf{v}}}.$$

We can calculate this using the built-in Matlab function called `norm`.

Rate of Convergence

When working with a numerical method, we are interested in the rate of convergence. The rate of convergence determines the number of iterations needed to find the solution. To find the rate, we first need to define a *metric*, which is used to measure how far from the solution the current iteration is from the true solution. If $\vec{\mathbf{b}}^*$ is the true eigenvector corresponding to the leading eigenvalue, the error at iteration k is given by:

$$e_k = \left\|\vec{\mathbf{b}}_k - \vec{\mathbf{b}}^*\right\|.$$

The rate of convergence is the ratio:

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

Suprisingly, for this method, and many others, r will stay about the same during each iteration. This property is known as *linear convergence*

Warming up

1) What kind of matrix has the following property?

$$M_{i,j} = M_{j,i}$$

2) Does the friendship adjacency matrix have the property described in 1)? Why or why not?

3) When Google uses the Page Rank algorithm, it creates a graph of the network of websites, with web links serving as the connections. Is this graph directed or undirected? Why?

4) Construct an adjacency matrix from the following network. Does it have the property described in problem #1? There are 6 books. They are simply 1, 2, 3, 4, 5, 6. They are connected as follows.

Book 1 cites 4, 6, and 5.
Book 2 cites 3, 5, and 6.
Book 3 cites 4, and 6.
Book 4 cites 5.
Book 5 cites 4, 1, and 2.
Book 6 cites 2, and 3.

5) Give the centrality scores of these books. What does this tell you about the books?

Ranking the Students

The class president will be the student with the highest centrality score. The candidates are:

Mary (also known as P31)

Friends with P1, P2, P3, P4, P5, P6, P7, P8, P9.

Fred (also known as P32)

Friends with P10, P11, P12, P13, P14.

Veronica (also known as P33)

Friends with P15, P16, P17, P18, P19.

Facebook data

Besides the three candidates, there are 30 other students in the school (P1-30), for a total of 33 students. The adjacency matrix for the 30 non-candidates is posted as `student_adjacency.txt`. Note that, before computing the eigenvectors, you must add the three candidates into the matrix. Use the power method to find the eigenvector corresponding to the largest eigenvalue.

6) What is the rate of convergence for this problem?

7) Plot the error, e_k, against k on a semi-log plot, i.e., plot $\log e_k$ against k. Also plot the line $\log(r)k$ against k on the same plot where r is the rate you found in the previous problem. How do the two curves compare?

8) Rank the students P1-P30, and the candidates, with scores based on Facebook adjacency.

9) How could someone cheat to make their score higher with this system?

10) What are some different flaws in this system of ranking the students' popularity? How could it be changed to correct that problem?

11) Would you have picked the same student to be class president? Why or why not? Are there other factors that you would take into consideration to make the system more balanced?

12) What are some examples of other networks that could be modeled using an adjacency matrix?