Markov chains (week 6)

1 Rank of nodes in graphs. The most popular algorithms to rank pages in a web search are stochastic. Consider a web surfer that visits a page and clicks on any of the page’s links at random. She repeats this process forever. What fraction of his time will be spent on a given page? The answer to this question is the rank assigned to the page. The same idea can be used to understand the structure of networks in different settings. For example, we can use this algorithm to extract connectivity information from a social graph. Say we choose a student and ask her to direct us to any of her friends selected randomly. We then go to this friend repeat the question and are directed to this new student. This is no different from the random web surfer model. Repeating this process forever we can therefore infer the degree of connectedness of students in the class from the average number of visits to each of them. This is not a pointless exercise. To market products, for example, it is worthwhile to concentrate the effort in the individuals that are most connected to other persons. The important insight here is that the network possesses knowledge that individuals do not. For this homework we use a collaboration matrix that you were asked to complete in class that you can download from the class’s webpage.

For a formal problem definition consider a network with \( J \) nodes. Describe connectivity by a directed graph \( G(V, E) \) with \( V \) denoting the set of nodes and \( E \) the set of edges. An edge is an ordered pair \( e := (i, j) \) representing a link from node \( i \) to node \( j \). Further define \( n(i) \) as the \( i \)-th node’s neighborhood containing the indexes \( j \) to which \( i \) is pointing, i.e., \( n(i) := \{ j : (i, j) \in E \} \). Let \( N_i \) be the number of nodes in the neighborhood of \( i \). Similarly, define the incoming neighborhood of \( i \) as the set of nodes that point to \( i \), i.e., \( n^{-1}(i) := \{ j : (j, i) \in E \} \).

An outside agent approaches an arbitrary node \( A_0 \) at time \( n = 0 \). From there it jumps to one of the neighbors \( n(A_0) \) of \( A_0 \) at time \( t = 1 \), then to one of the neighbors of this neighbor and so on. If the agent is visiting node \( A_n \) at time \( n \) it will visit a node in the neighborhood \( n(A_n) \) at time \( n+1 \) with all neighbors chosen equiprobably. The fraction of time the agent spends visiting node \( i \), is defined as the node’s rank. To express this mathematically define the indicator function \( I \{ A_n = i \} \) with value 1 when the agent visits \( i \) at time \( n \) and 0 otherwise. The rank \( r_i \) of node \( i \) is then given by

\[
r_i(A_0) := \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} I(A_n = i).\tag{1}
\]

Since we are considering equal probabilities of jumping to any neighbor, the probability \( P_{ij} \) of the agent transitioning from node \( i \) to node \( j \) is

\[
P_{ij} := P \{ A_{n+1} = j \mid A_n = i \} = \frac{1}{N_i}, \quad j \in n(i)\tag{2}
\]

where, we recall, \( N_i \) is the number of nodes in the neighborhood of \( i \). The movement of the agent through the nodes in the graphs is called an equiprobable random walk in a graph.

A Markov chain model. The stochastic process of agent visits \( A_n \) is a Markov chain (MC). Explain. Give conditions for the following statements to be true:

- State \( i \), meaning visit of agent to node \( i \), of this MC is transient.
- All states of this MC are transient.
- All states of this MC are recurrent.
- All states of this MC are aperiodic.
- All states are positive recurrent.
- All states are ergodic.
- The MC is irreducible.

Some of the above statements might be always true and some might be never true. Also notice that the agent here behaves somewhat different from the agent we covered in class. In class, we covered the ranking problem assuming that the random walk was recurrent, aperiodic, ergodic and irreducible. In the rest of the exercise you can assume that all states in this random walk are recurrent, aperiodic and ergodic, but you cannot assume that the MC is irreducible unless otherwise explicitly stated. If you wish, you are going to study the extent to which we can deal with lack of irreducibility. Thus, before proceeding make sure that you understand correctly the graph’s aspect when the MS is not irreducible.

B Implement random walk. We are now ready to build an algorithm to compute \( r_i \). We start with a randomly chosen node \( A_0 \) and jump equiprobably to any of its neighboring nodes \( j \in n(i) \). The probability of selecting any
of this nodes is \( P_{ij} = 1/N_i \). We repeat this process a large number of times \( n \) and keep track of the number of visits to each node. The rank \( r_i \) is then approximated as the ratio between the number of visits to node \( i \) and the total time \( n \), i.e.,

\[
 r_i(A_0) \approx \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(A_n = i).
\]  

(3)

Write Matlab code to implement this random walk. Use this code to compute the rankings as defined in (3) for \( A_0 = 1 \). What condition needs to be satisfied for the ranks in (1) to be independent of the initial state \( A_0 \)? If this condition is not satisfied we can modify the algorithm by introducing an artificial node that is connected to all members of the graph. You can think of this node as the class’s professor that knows, or should know, all students. Call these modified ranks \( r_i \) and approximate them as

\[
 r_i \approx \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(A_n = i).
\]  

(4)

The definitions in (3) and (4) look the same but recall that the agent visits are in different graphs. Compute your modified ranks. Convergence of this algorithm is very slow. Rough numerical approximations are acceptable as the answer to this question.

C Probability update. The algorithm in part B is certainly a possibility, but we can obtain a faster version by exploiting the fact that these random visits can be modeled as a MC. Let \( p_i(n) := \mathbb{P}\{A_n = i\} \) denote the probability that the outside agent is at node \( i \) at time \( n \). It is possible to express \( p_i(n+1) \) in terms of the probabilities at time \( n \) of those nodes that can transition into \( i \). Write this probability update. Define the vector \( \mathbf{p}(n) := [p_1(n), \ldots, p_J(n)]^T \) and write this update equation in matrix form.

D Find ranks using the probability update. An interesting property of MCs is the existence of limit probabilities \( \lim_{n \to \infty} p_i(n) \) under some conditions. State these conditions. These limit probabilities might depend on the initial probability distribution \( \mathbf{p}(0) \). If this vector is chosen such that all the initial probability is on \( A_0 \), i.e., \( p_{A_0}(0) = 1 \) the rank can be equally computed as

\[
 r_i(A_0) = \lim_{n \to \infty} p_i(n)
\]  

(5)

Explain why this is the case. Write Matlab code to compute ranks using the property stated in (5). Use this code to compute the rankings as defined in (3) for \( A_0 = 1 \). The ranks in (4) obtained from the modified graph can be computed as the limit

\[
 r_i = \lim_{n \to \infty} p_i(n)
\]  

(6)

for any initial probability distribution. Explain why this is the case. Modify your function to compute your ranks \( r_i \). Run you function and show the rankings. A convenient initial probability distribution is \( \mathbf{p}(0) = (1/J)1 \).

E Recast as system of linear equations. Restrict attention to the modified graph containing the fully connected node. As you have already seen and explained, ranks are independent of the initial state \( A_0 \). Use your knowledge of MCs to recast the ranking problem as the solution of a system of linear equations. Solve this system of linear equations and compare with the results of parts B and D.

F Recast as eigenvalue problem. Restrict attention to the modified graph containing the fully connected node. Use your knowledge of MCs to recast the ranking problem as an eigenvector problem for a certain matrix. Compute this eigenvector and compare with the results of parts B, D and E. The Matlab function to compute eigenvalues and eigenvectors is \texttt{eig()}.

G Discuss advantages of each method. Restrict attention to the modified graph containing the fully connected node. All four methods yield the same results but have particular advantages that make them suitable for different applications. Discuss. Most of these advantages were discussed in class. But there is a particular advantage of the method in part D that we did not discuss and you should now be able to appreciate.