Week 6: Markov chains
Ranking nodes in a network

Popular algorithms to rank pages in a web search (e.g., Google's PageRank) are stochastic. They are typically based on the following process: consider a web surfer that visits a website and clicks on one of its links at random; now suppose this process is repeated forever. What fraction of time will the surfer spend on a given page? This answer to this question can then be used to rank that page. This idea is not, however, restricted to the website ranking and can be used to understand the structure of networks in different settings. For instance, this algorithm can be used to extract connectivity information from a social graph (known as *sociograms* in sociometry). Say we ask a student to direct us to a randomly selected friend. We then go to this friend and repeat the process, being directed to another student. This is no different from the random web surfer model. Repeating this process forever allows us to infer the degree of connectivity of each student in the class from the average number of visits to each of them. As futile as this may seem, it is far from a pointless exercise. When marketing products, for example, it is worthwhile to concentrate efforts on the individuals that are most connected as they are likely to influence more people. The crucial insight here is that the network individuals belong to carries information that the individuals in isolation do not. For this homework, we will use a collaboration matrix that you can download from the course's website.

Let us now formalize the model we discussed using the language of graphs. Consider a network with J individuals. We describe this network by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, J\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. An edge $e = (i, j)$ is an ordered pair representing a link from node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$. Also define $\mathcal{N}(i)$ to be the neighborhood of the i-th node, i.e., all nodes $j \in \mathcal{V}$ to which i is pointing. Explicitly, $\mathcal{N}(i) = \{j : (i, j) \in \mathcal{E}\}$. Let $D_i = |\mathcal{N}(i)|$ be the number of nodes in the neighborhood of i. This is often referred to as the degree of i. Similarly, define the incoming neighborhood of i as the set of nodes that point to i, i.e., $\mathcal{N}^{-1}(i) = \{j : (j, i) \in \mathcal{E}\}$.

Suppose now that an outside agent approaches an arbitrary node A_0 at time $n = 0$ and starts the random neighbor jumping process we described earlier. Hence, if the agent is at node A_n at time n, it randomly and equiprobably choose a node in its neighborhood $\mathcal{N}(A_n)$ to visit at time $n+1$. Thus, the probability P_{ij} of the agent transitioning from node i to node j is

$$P_{ij} = \mathbb{P}\left[A_{n+1} = j \mid A_n = i\right] = \frac{1}{D_i}, \quad \text{for } j \in \mathcal{N}(i). \tag{1}$$

Recall that D_i is the number of nodes in the neighborhood of node i. The movement of the agent through the nodes in the graphs is called an equiprobable random walk on the graph.

Finally, we let the fraction of time the agent spends visiting node $i \in \mathcal{V}$ be the i-th node's rank. To express this quantity mathematically, define the indicator function $\mathbb{I}\left[A_n = i\right]$ to be one if the agent visits node i at time n and 0 otherwise. The rank r_i of node i is then given by

$$r_i(A_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(A_n = i). \tag{2}$$

Notice that r_i may depend on the initial node A_0.

A Markov chain model. The stochastic process describing the nodes the agent visits $A_{\mathbb{N}}$ is a Markov chain (MC). Explain why. Moreover, give conditions for the following statements to be true:

1. State i of this MC, i.e., visits of the agent to node i, is transient.
2. All states of this MC are transient.
3. All the states of this MC are recurrent.
4. All states of this MC are aperiodic.
5. All states are positive recurrent.
6. All states are ergodic.
7. The MC is irreducible.

Some of the above statements may always be true whereas others may never be true.

Before proceeding, notice that the agent in this homework behaves differently than the one we covered in class. Indeed, the ranking problem we considered in class assumed that the random walk was recurrent, aperiodic, ergodic, and irreducible. For the rest of this homework, 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, we are going to study the extent to which we can deal with lack of irreducibility. So before you continue, make sure you understand correctly the graph aspects of reducible MCs.

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

$$r_i(A_0) \approx \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left(A_n = i\right). \tag{3}$$

Write a MATLAB script that implements this random walk on the graph underlying the collaboration matrix available on the course website. Use this code to compute the rankings as per (3) for $A_0 = 1$. What condition needs to be satisfied for the ranks in (2) to be independent of the initial state A_0?

If this condition is *not* satisfied, 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 professor who knows—or should know—all students. Call these modified ranks r_i and approximate them as

$$r_i \approx \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left(A_n = i\right) \tag{4}$$

The definitions in (3) and (4) look the same, but recall that the random walk occurs in different graphs. Compute the modified ranks.

Disclaimer: convergence of this algorithm is very slow. Rough numerical approximations are acceptable as the answer to this question.

C Probability update. Although the implementation in part B works, we can obtain a faster version by exploiting the fact that the random visits can be modeled as a MC. Let $p_i(n) = \mathbb{P}[A_n = i]$ denote the probability that the agent is at node i at time n. The probability $p_j(n+1)$ can be expressed in terms of the probabilities at time n of those nodes that can transition into j. Explicitly, the memoryless property allows us to write $p_j(n+1)$ as a function of the $p_i(n)$ for $i \in \mathcal{N}^{-1}(j)$, i.e., the probabilities of the nodes that can transition into j. Write this probability update expression. Using the vector $\boldsymbol{p}(n) = \begin{bmatrix} p_1(n) & \cdots & p_J(n) \end{bmatrix}^T$, write this update equation in matrix form.

D Probability update implementation. An important property of some MCs is the existence of limit probabilities $\lim_{n\to\infty} p_i(n)$. State conditions under they exist. Still, these limit probabilities might depend on the initial probability distribution $\boldsymbol{p}(0)$. Choosing this vector such that all the initial probability is on A_0, i.e., $p_{A_0}(0) = 1$ and $p_i(0) = 0$ for all $i \neq A_0$, the i-th node rank can be computed as

$$r_i(A_0) = \lim_{n\to\infty} p_i(n). \tag{5}$$

Explain why this is the case. Write a MATLAB script that computes ranks using the property stated in (5) (recall the update you derived in part C) and evaluate the ranks for $A_0 = 1$. The ranks in (4) obtained from the modified graph can be computed as the limit

$$r_i = \lim_{n\to\infty} p_i(n) \tag{6}$$

for any initial probability distribution. Explain why this is the case. Modify your function to compute the ranks r_i and rank the nodes once more. A convenient choice of initial probability distribution is $\boldsymbol{p}(0) = (1/J)\mathbb{1}$.

E Recast as a system of linear equations. Focus now only on the modified graph containing the fully connected node. As you have already seen and explained, ranks in this case are independent of the initial state A_0. Use your knowledge of MCs to recast the node ranking problem as the solution of a system of linear equations. Write a MATLAB script to solve this system and compare with the results of parts B and D.

F Recast as an eigenvalue problem. Focus again only on the modified graph containing the fully connected node. Use your knowledge of MCs to recast the node ranking problem as an eigenvalue problem. Write a MATLAB script that computes the desired eigenvector and compare with the results of parts B, D, and E.

Hint: You can use the MATLAB function `eig` to solve eigenvalue problems. Use `help eig` to find out more.

G Compare implementations. All four methods yield the same results, but have specific advantages that make them suitable for different applications. Once again focusing on the modified graph containing the fully connected node, discuss these advantages. Most of them were touted 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.