

TUFTS UNIVERSITY DEPARTMENT OF MATHEMATICS
Midterm 2

Math 87, Duchin

Spring 2020

Welcome to your second midterm project! Your writeup should aim to be more of a *report* than a problem set, which means that there's an emphasis on providing all the necessary information for the reader to understand what is important. Your job is *critical modeling*: don't just focus on answers but on explanation, exploration, and thinking through the meaning of the parameters and the quality of the answers given by the models we are studying.

You will need to write your own Python code for this midterm, but you should have helpful scaffolded material to draw from the previous HW, as well as a new notebook with web-crawling code. Include pseudo-code and plots where appropriate in your report.

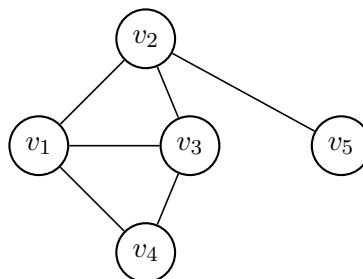
You are welcome to use course notes and written internet resources and to ask questions to the instructors. Please work independently and submit original solutions in your own words **as a single PDF**. Cite sources if you use them (see Affirmation of Integrity statement).

Part I: PageRank on a directed graph

Markov chains formed the original basis of the PageRank algorithm used by Google to rank web-pages by importance. In this project, you'll gather data for a portion of the web, create a series of transition processes modeling someone surfing that portion of the web, and make conclusions about the relative importance of the pages that were visited.

Consider a basic model of surfing the web. You're looking at your favorite website (for example, <https://math.tufts.edu/>) and you follow an outgoing link chosen at random (with equal probability for each page that is linked to, regardless of how many times the link appears). This brings you to a new page, where you do the same thing, and so on. This describes a random walk where the state space is the internet.

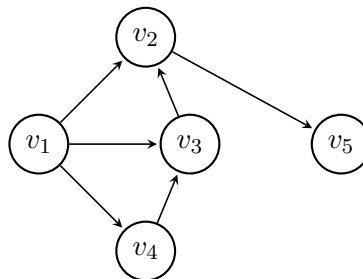
To model this random walk as a Markov Chain, you only need to count the number of distinct outgoing links from each page to assign the probabilities.



1. For the undirected graph above, find the adjacency matrix A , the transition matrix M for simple random walk, and the iteration matrix P .

- Find the *stationary probability vector* w three different ways: using your knowledge of the vertex degrees, using Python's linear algebra package to find eigenvalues and eigenvectors, and using the power method. (Say something about how you chose the starting point for the power method and the number of steps you needed to take.) Once you have w , explain what PageRank would consider to be the two most important websites on this network.

This gets harder with a directed graph. Now you need to deal with the fact that some vertices have no outgoing edges and with the fact that the network could have cycles (like two pages that link each other but have no outgoing links). We will handle that in two ways. First, if a vertex is a dead end, we'll adjust the transitions to make it equally likely for the surfer to transition from there to any vertex (including itself)—call this *teleportation*. Second, we'll introduce a global probability p and declare that a random surfer on a non-dead-end site will teleport with probability p and follow a link with probability $1 - p$. Overall, we'll call this modified walk *random surfing with teleportation*, or PageRank for short.



- Explain why PageRank is an ergodic Markov chain when $p > 0$. Explaining all your steps fully, find the two most important websites in the directed graph using PageRank with $p = .1$, and report the steady-state probability of being at each of them. How do your answers depend on the choice of p ?

Part II: Crawling the web

To generate the adjacency graph of a piece of the web, go to `midterm2.ipynb` on CoLab, which can be accessed from the homework page. This is an updated and Pythonized version of the antique Matlab webscraper

`http://www.mathworks.com/moler/ncm/surfer.m`,

taking into account things like java scripts and rss feeds. WARNING!!: webscraping is a messy business, partly because the web is a messy business. Even with careful customizations this will still find lots of garbage corporate links, to say nothing of other corners of the web you may find yourself in.¹

¹I set it up with an optional blacklist of domains that the crawler should pass by (e.g., anything related to facebook or messenger or linkedin). Try it without the blacklist to see the difference.

Parameters in the webscraping code include: the website that you want to start from and the number of websites that you want to include in your graph. The output includes A , the adjacency matrix of the web network, where each row corresponds to a web page and its links to all other pages in the dataset:

$$A_{ij} = \begin{cases} 1 & \text{if } i \text{ links to } j \\ 0 & \text{if } i \text{ does not link to } j. \end{cases}$$

Output also includes `graph`, the directed graph associated to A . The nodes of the graph correspond to URLs. If you print `graph.nodes[0]`, for instance, you'll get the dictionary entry corresponding to the first URL where you began your web-crawl. If you print `graph.nodes[0]['url']` you'll get just the URL alone.

4. Experiment with the webcrawler by varying the starting location and the number of sites for the crawler to seek to add to the graph. (For instance you should be able to build up to 500 sites in about a minute over standard wifi unless CoLab is acting up.) Explain with pseudo-code (a readable explanation of the algorithmic instructions) how you would modify the adjacency matrix A to build M , P , and w for a given teleportation probability p .
5. Write the code and carry this out! Report on your findings. For some interesting starting website of your choice, explain how many iterations are needed for the power method to converge. Does the list of top ten sites agree between the eigenvalue method and the power method? How does the score of the top website depend on the teleportation probability p ?
6. What are the limitations of PageRank as a model of the *relative importance* of websites? Explain how you could design an improvement that addresses some of these weaknesses.

Avenues for further exploration – strictly optional

- You might find it interesting to read the Wikipedia page about PageRank and/or to read the original paper from 1998 laying out the concepts (link). If you skim the 1998 paper you'll find pointers to how to use PageRank together with a search term. Heads up: as usual, the formulations there will be a little different from here.
- Try to understand how the web-crawler works. (Note: it does NOT behave like the random surfer in PageRank.) Think about the engineering challenges of building a good web-crawler.
- Explore the graph and network tools in the NetworkX library. For instance, you can easily plot a histogram of the degrees in your web graph, which should give you some insight into how the crawler works...

Affirmation of Integrity

In your report, please include 1-2 sentences in your own words indicating what resources you used and what people you interacted with about the contents of this test.