EE141L: Lab 3
PageRank

September 16, 2018

1 Web search

In this lab you will learn about how matrix vector multiplication can be applied to model and analyze complex network systems. We will use as an example PageRank[1], the algorithm behind early versions of Google search.

1.1 Network model

We will consider network models, where

- Each website is a node/vertex in the network/graph,
- A link from website i to website j is denoted by an arrow, $i \longrightarrow j$.

See Figure 1 for an example network with 6 websites.

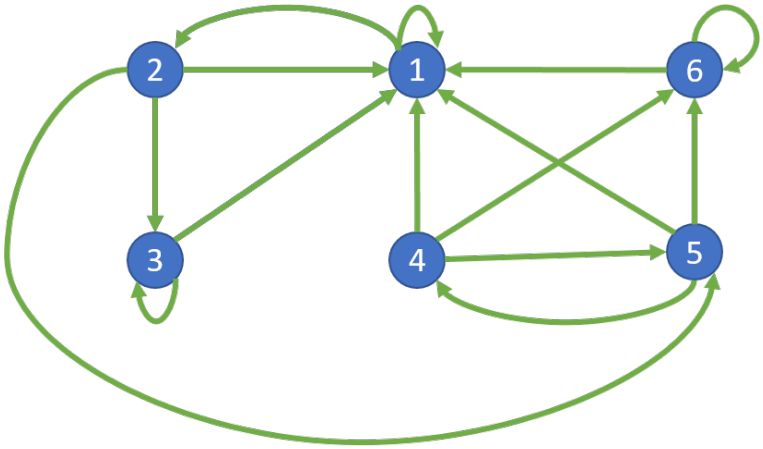


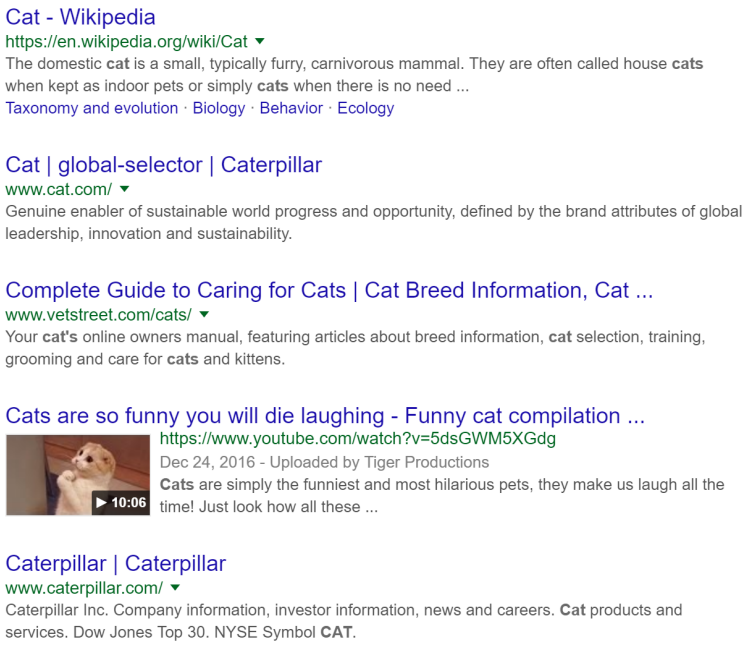
Figure 1: Network representing links among 6 websites

[1]Page, L., Brin, S., Motwani, R., Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.

1.2 Search Algorithm

A simple search algorithm can be described as follows

1. Input a seach query, e.g. the word **cat**
2. Construct the network of all websites containing the word **cat**
3. Create a ranking of websites, from most to least relevant.
4. Display results.



2 PageRank: ranking of websites

PageRank is an algorithm that produces a ranking of websites. It is based on two simple principles

1. Relevant websites have many incoming links.
2. Relevant websites are linked from other relevant websites.

2.1 Random surfer

The network from figure 1 can be described by its adjacency matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{1}$$

Where $\mathbf{A}_{ij} = 1$ when $j \longrightarrow i$, there is a link from website j to website i. The key insight behind the ranking algorithm is to imagine a "random surfer" who constantly navigates surfs the web by following links. We will see how understanding random surfing reveals information about the relative importance of different nodes. A linear algebra formulation allows to represent mathematically the random surfing process and then extract conclusions via elementary matrix operations. Later this semester we will study other tools to understand a system of this nature.

A random surfing session starts at a random website (any node in the network) and then iterates the following steps:

1. Pick one link on that website at random.
2. Click on the link, go back to step 1).

During this procedure, we can count the number of times each website is visited. The rank of a website is the average number of times the website was visited by the random surfer

$$\text{rank}(i) = \frac{\#\text{times website } i \text{ was visited}}{\text{number of visited websites}}. \tag{2}$$

A large value of $\text{rank}(i)$ indicates the i-th website is more relevant.

Note that in this model a simulation would have to traverse the network many times in order to provide a result, because each time the simulation is in a specific node the random surfer will follow only one link. This can be done much more easily by assigning equal probability weights to each outgoing link of a node. For example, Figure 1 there are 3 outgoing links coming from node 2 and so we can assign a probability of $1/3$. Thus, instead of following only one link chosen at random, we follow all 3, but with the corresponding probability weights. This allows ranks to be computed very easily using matrix vector multiplications. First we define the probability transition matrix

$$\mathbf{S} = \begin{bmatrix} 1/2 & 1/3 & 1/2 & 1/3 & 1/3 & 1/2 \\ 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/2 \end{bmatrix}. \tag{3}$$

where each column corresponds to one of the nodes, and each of the entries in one column corresponds to the outgoing probabilities of edges in that node (note that each of the columns adds to 1). More formally, $\mathbf{S}_{ij}$ is the probability that the random surfer will go from website j to website i. The random surfer equation is given by

$$\mathbf{u}^{(k+1)} = \mathbf{S}\mathbf{u}^{(k)}, \tag{4}$$

where $\mathbf{u}^{(0)}$ is any non negative vector with entries adding up to 1. You can think of $\mathbf{u}^{(k)}$ as the state of the system after k iterations, given the initial state $\mathbf{u}^{(0)}$. The i-th entry of $\mathbf{u}^{(k)}$ is an approximation of rank(i).

Problem 1. In this problem we will compute an approximate value for the random surfer ranking

1. Start Matlab and make sure the file network1.mat is in your current folder. Put this line of code at the beginning of your script load('network1.mat')
2. Starting with $\mathbf{u}^{(0)} = [0, 0, 1, 0, 0, 0]^\top$, compute $\mathbf{u}^{(k)}$ for $k = 1, \cdots, 10$. Describe the behavior of $\mathbf{u}^{(k)}$ as k increases. Report the value of $\mathbf{u}^{(10)}$.
3. Repeat part 2, for $\mathbf{u}^{(0)} = [1, 0, 0, 0, 0, 0]^\top$
4. Repeat part 2, for $\mathbf{u}^{(0)} = [0, 0, 0, 0, 0, 1]^\top$
5. What can you say about the value of $\mathbf{u}^{(10)}$ computed in parts 2, 3, and 4?.
6. What is the rank for each website?. Order the websites from most to least relevant (hint: use the values of $u^{(10)}$ found before).

2.2 Random Teleportation

A potential problem with the random surfer model comes from pages that do not link to any other pages. Once the surfer arrives to one of these, there is nowhere to go!

As a solution we will consider a teleportation model. This corresponds to random surfing that completely ignores the links in the network and so can jump from any page to any other page. Equivalently, this is as if the random surfer was navigating through a network where all pages are connected to each other. The matrix $\mathbf{T}$ is analogous to $\mathbf{S}$ from the previous section, but now all probabilities are equal. You can see how this means that all nodes to each other, since we have all $1's$ in each column:

$$\mathbf{T} = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{5}$$

Problem 2. Repeat Problem 1 replacing $\mathbf{S}$ by $\mathbf{T}$.

2.3 PageRank

Random surfer and random teleportation methods are not perfect.

- The random surfer method might get stuck in websites without outgoing links. For example note that if we remove the link $3 \longrightarrow 1$ in the network from Figure 1, a random surfer might get stuck in website 3 indefinitely.
- Obviously, the random teleportation method does not have this problem, but it will completely ignore the network structure.

The PageRank method combines both the random surfer, and the random teleportation as follows. Define the PageRank transition matrix as

$$\mathbf{P} = \alpha \mathbf{S} + (1 - \alpha)\mathbf{T}, \tag{6}$$

where $\alpha \in (0, 1)$.

Problem 3. Repeat Problem 1 using PageRank matrix $\mathbf{P}$, for $\alpha = 0.85$.

3 PageRank in a large network

In this section you will implement PageRank in a real network with $\sim$ 800 nodes, taken from `https://snap.stanford.edu/data/email-Eu-core.html`. You can get the data by putting the file network2.mat in your current folder, and loading it with load('network2.mat'). The random surfer matrix $\mathbf{S}$ is included in the mat file. The random teleportation matrix $\mathbf{T}$ can be constructed using the matlab function ones(n,n)/n, where n is the number of websites in the network (use function size). For problem 4 use $\alpha = 0.85$ and $\mathbf{u_0} = [1, 0, 0, \ldots, 0]^T \in \mathbb{R}^n$

Problem 4. PageRank in large nework

1. Construct PageRank matrix $\mathbf{P}$.
2. Implement PageRank, starting from $\mathbf{u}^{(0)}$, compute $\mathbf{u}^{(1)}$, $\mathbf{u}^{(2)}$, ...$\mathbf{u}^{(N)}$. **You have to implement this with a for loop**.
3. Experiment with the value of N that will give you a good solution $\mathbf{u}^{(N)}$. Explain your final choice of N.
4. Use the resulting rank and report the 10 most relevant websites/nodes of the network. For this you might want to investigate the matlab function sort