General Instructions

Submission instructions: You should submit your answers via GradeScope and your code via Snap submission site.

Submitting answers: Prepare answers to your homework into a single PDF file and submit it via http://gradescope.com. Make sure that answer to each question is on a separate page. This means you should submit a 22-page PDF (1 page for the cover sheet, 5 pages for the answers to question 1, 3 pages for answers to question 2, 8 pages for question 3, and 5 pages for question 4). On top of each page write the number of the question you are answering. Please find the cover sheet and the recommended templates located here:

http://cs224w.stanford.edu/homework/hw2/submission_template_hw2.tex
http://cs224w.stanford.edu/homework/hw2/submission_template_hw2.pdf
http://cs224w.stanford.edu/homework/hw2/submission_template_hw2.docx

Not including the cover sheet in your submission will result in a 2-point penalty.

Submitting code: Upload your code at http://snap.stanford.edu/submit. Put all the code for a single question into a single file and upload it.

Questions

We strongly encourage you to use Snap.py for Python. However, you can use any other graph analysis tool or package you want (SNAP for C++, NetworkX for Python, JUNG for Java, etc.). See http://cs224w.stanford.edu/resources.html for more details.

1 Broadcasting in the Network [25 points - Paris]

A very fruitful avenue to design centrality measures is to consider a stochastic (random) process on the graph and extract the importance of nodes by optimizing some “natural” objective. Here we consider that there is an undirected graph $G(V,E)$ with adjacency matrix $A \in \{0,1\}^{n \times n}$ and that during each round $k$:

- **Graph process:** each vertex sends a (different) message to each of its neighbors independently at a uniformly random time between $[k,k+1)$, i.e., for each edge $(i,j)$, $T_{ij}$ (respectively $T_{ji}$) is the time $i$ (resp. $j$) sends a message to $j$ (resp. $i$) and is uniformly distributed in $[k,k+1)$ independently from all other edges.

- **Action:** at any time $t \in [k,k+1)$ we can “bug” any (more than one) node in the network and intercept the incoming messages. Let $t_i$ be the total time we “bugged” node $i$ during this round. The probability of us being caught by node $i$ at the end of the round is $t_i^\beta$, where $\beta > 0$ is known.
Our objective is to specify the total times \( t = (t_1, \ldots, t_n) \) that we are going to monitor each node, in order to maximize the expected number of messages intercepted during a single round, while making sure that the probability that we get caught at the end of the round is smaller than a number \( \gamma < 1 \). We assume that messages from previous rounds are not retransmitted.

(a) **[3 points]** Give an expression for the expected number of messages \( m_i(t_i) \) intercepted from node \( i \), if we monitor it for \( t_i \) time in total.

(b) **[4 points]** Let \( A_i(t_i) \) be the event that we get caught at node \( i \) at the end of the round if we monitor it for \( t_i \) total time. Using the fact that for any set of events \( A_1, A_2: \Pr(A_1 \cup A_2) \leq \Pr(A_1) + \Pr(A_2) \), prove that the probability of being caught by at least one node at the end of the round is less than \( r(t) := \sum_{i=1}^n t_i^\beta \).

(c) **[9 points]** For \( \beta = 2 \) and any \( \gamma < 1 \) find a strategy \( (t_1, \ldots, t_n) \) that maximizes the total expected number of intercepted messages \( \sum_i m_i(t_i) \) subject to the fact that \( r(t) \leq \gamma \). Comment on the resulting centrality measure. *Hint: Cauchy-Schwarz inequality.*

(d) **[9 points]** Repeat part (c) for \( \beta = 1 \) and assuming that all degrees are distinct. What changed? Comment on the solution when the degrees are not necessarily distinct.

(e) **[+3 points, Bonus]** What is the expression for any \( \beta \neq 1 \)?

**What to submit**

(a) Write the expression and a short explanation.

(b) Write a short proof.

(c) Write the expression and a short proof. Compare your result to known centrality measures.

(d) Write the expression and a short proof. Compare with part (c). Say what happens when degrees are not necessarily distinct.

(e) Write the expression and a short proof.

---

2 **Signed Triad Analysis [15 points - Sameep, Nihit]**

Download the Epinions dataset from here: [http://cs224w.stanford.edu/homework/hw2/epinion_signed.txt](http://cs224w.stanford.edu/homework/hw2/epinion_signed.txt)

Epinions is a consumer review site where members can decide if they trust each other or not. This leads to a web of trust which you will analyze in this question.

Since a member can either trust or distrust other members, we have a special type of network called signed network where an edge has a sign. Positive sign on edge indicates trust relationship whereas negative edge indicates distrust relationship.

We will consider the graph as undirected and study the various forms of triads as shown in Figure 1.

(a) **[5 points]** Calculate the count and fraction of triads of each type in the opinion network.
Figure 1: Triads

(b) [5 points] Calculate the fraction of positive and negative edges in the graph. Let this fraction of positive edges be \( p \). Assuming that an edge of a triad will independently take positive sign with probability \( p \) and negative sign with probability \( (1 - p) \), calculate the probability for a triad to form each combination of signs.

(c) [5 points] Compare the probabilities from part (b) with the fractions calculated in part (a). Which type of triads do you see more in data as compared to baseline values? Which type of triads do you see less? Provide an explanation for this observation.

What to submit

(a) Write the count and fraction of triads of each type

(b) Write the fraction of positive and negative edges. Write the probability of formation for each type of triad

(c) Give the triads which have more occurrence in part a as compared to baseline calculated in part b. Give the triads which have less occurrence in comparison to baseline calculated in part b. Write a short explanation for the observation.

3 Decentralized Search [40 points – Caroline]

In class, we saw a decentralized search algorithm based on geography that seeks a path between two nodes on a grid by successively taking the edge towards the node closest to the destination.

Here we will examine an example of a decentralized search algorithm on a network whose nodes reside in the leaves of a tree. The tree may, for instance, be interpreted as representing the hierarchical organization of a university where one is more likely to have friends inside the same department, a bit less likely in the same school, and the least likely across schools.

Let us organize students at Stanford into a tree hierarchy, where the root is Stanford University and the second level contains the different schools (engineering, humanities, etc.). The third level represents the departments and the final level (i.e., the leaves) are the Stanford students. Tom, a student from the computer science department, wants to hang out with Mary, who is in sociology. If Tom does not know Mary, he could ask a friend in the sociology department to introduce them. If Tom does not know anybody in the sociology department, he may seek a friend in the Stanford
Figure 2: Illustration for Question 3 of the hierarchical graph. Black nodes and edges are used to illustrate the hierarchy and structure, but are not part of our network. Red nodes (leaf nodes) and red edges are the ones in our network. The lowest common ancestor of $s$ and $t$ is the root of the tree. The decentralized search proceeds as follows. Denote the starting node by $s$ and the destination by $t$. At each step, the algorithm looks at the neighbors of the current node $s$ and moves to the one “closest” to $t$, that is, the algorithm moves to the node with the lowest common ancestor with $t$. In this graph, from $s$ we move to $u$.

humanities school instead. In general, he will try to find a friend who is “close” to Mary in the tree.

There are three parts in this problem. The first two parts explore an effective decentralized search algorithm on the hierarchical model in a specific setting. The third part involves simulation experiments on the model under a more general setting. There are many subproblems, but do not panic. A good understanding of the math involved in the lattice model we covered in class will make this problem easy and shorten each subproblem to just a few line proofs.

Basic Tree Properties [13 points]

This part covers some basic facts of the setting of the hierarchical model and defines the notion of tree based “distance” between the nodes.

Consider a complete, perfectly balanced $b$-ary tree (each non-leaf node has $b$ children and $b \geq 2$) $T$, and a network whose nodes are the leaves of $T$ (the red nodes in the picture). Let the number of network nodes (equivalently, the number of leaves in the tree) be $N$ and let $h(T)$ denote the height of $T$.

One important thing to keep in mind: In this problem, the network nodes are only the leaf nodes in the tree. Other nodes in the tree are virtual and only there to determine the edge creation probabilities between the nodes of the network.

(a) [4 points] Write $h(T)$ in terms of $N$.

(b) [4 points] Next we want to define the notion of “tree distance.” The intuition we want to capture is that students that share the same department are closer than for example students
sharing schools. For instance, in the tree in Figure 2 nodes $u$ and $t$ are “closer” than nodes $u$ and $s$. We formalize the notion of “distance” as follows:

Given two network nodes (leaf nodes) $v$ and $w$, let $L(v, w)$ denote the subtree of $T$ rooted at the lowest common ancestor of $v$ and $w$, and $h(v, w)$ denote its height (that is, $h(L(v, w))$). In Figure 2, $L(u, t)$ is the tree in the circle and $h(u, t) = 2$. Note that we can think of $h(v, w)$ as a “distance” between nodes $v$ and $w$.

For a given node $v$, what is the maximum possible value of $h(v, w)$?

(c) [5 points] Given a value $d$ and a network node $v$, show that there are $b^d - b^{d-1}$ nodes satisfying $h(v, w) = d$.

**Network Path Properties [20 points]**

This part helps you design a decentralized search algorithm in the network.

We will generate a random network on the leaf nodes in a way that models the observation that a node is more likely to know “close” nodes than “distant” nodes according to our university organizational hierarchy captured by the tree $T$. For a node $v$, we define a probability distribution of node $v$ creating an edge to any other node $w$:

$$p_v(w) = \frac{1}{Z} b^{-h(v,w)}$$

where $Z = \sum_{w \neq v} b^{-h(v,w)}$ is a normalizing constant. By symmetry, all nodes $v$ have the same normalizing constant.

Next, we set some parameter $k$ and ensure that every node $v$ has exactly $k$ outgoing edges. We do this with the following procedure. For each node $v$, we repeatedly sample a random node $w$ according to $p_v$ and create edge $(v, w)$ in the network. We continue this until $v$ has exactly $k$ neighbors. Equivalently, after we add an edge from $v$ to $w$, we can set $p_v(w)$ to 0 and renormalize with a new $Z$ to ensure that $\sum_w p(w) = 1$. This results in a $k$-regular directed network.

(d) [5 points] Show that $Z \leq \log_b N$. (Hint: use the results in (ii) and (iii).)

(e) [5 points] For two leaf nodes $v$ and $t$, let $T'$ be the subtree of $L(v, t)$ satisfying:

- $T'$ is of height $h(v, t) - 1$,
- $T'$ contains $t$,
- $T'$ does not contain $v$.

For instance, in Fig. 2, $T'$ of $L(s, t)$ is the tree in the circle.

Let us consider node $v$ and an edge $e$ from $v$ to a random node $u$ sampled from $p_u$. We say that $e$ points to $T'$ if $u$ is a leaf node of $T'$. Show that the probability of $e$ pointing to $T'$ is no less than $\frac{1}{\log_b N}$.

(f) [5 points] Let the out-degree $k$ for each node be $c \cdot (\log_b N)^2$, where $c$ and $b$ are constants. Show that when $N$ grows very large, the probability of $v$ not having any edge pointing to $T'$ is asymptotically no more than $N^{-\theta}$, where $\theta$ is a positive constant which you need to compute.
(Hints: Use the result in (v) and recall that each of the $k$ edges is independently created. Also, use $\lim_{x \to \infty} (1 - \frac{1}{x})^x = \frac{1}{e}$.)

Argue why the above result indicates that for any node $v$, we can, with high probability, find an edge to a (leaf) node $u$ satisfying $h(u, t) < h(v, t)$.

(g) [5 points] Show that starting from any (leaf) node $s$, within $O(\log_b N)$ steps, we can reach any (leaf) node $t$. You do not need to prove it in a strict probabilistic argument. You can just assume that for any (leaf) node $v$, you can always get to a (leaf) node $u$ satisfying $h(u, t) < h(v, t)$ and argue why you can reach $t$ in $O(\log_b N)$ steps.

Simulation [7 points]

In (i) to (vii), we have set the theory to find an efficient decentralized search algorithm, assuming that for each edge of $v$, the probability of it going to $w$ is proportional to $b^{-h(v, w)}$. Now we experimentally investigate a more general case where the edge probability is proportional to $b^{-\alpha h(v, w)}$. Here $\alpha > 0$ is a parameter in our experiments.

In the experiments below, we consider a network with the setting $h(T) = 10$, $b = 2$, $k = 4$, and a given $\alpha$. That is, the network consists of all the leaves in a binary tree of height 10; the out degree of each node is 4. Given $\alpha$, we create edges according to the distribution described above.

(h) [7 points] Create random networks for $\alpha = 0.1, 0.2, \ldots, 10$. For each of these networks, sample 1,000 random $(s, t)$ pairs ($s \neq t$). Then do a decentralized search starting from $s$ as follows. Assuming that we are currently at (leaf) node $s$, we pick its neighbor $u$ (also a leaf node) with smallest $h(u, t)$ (break ties arbitrarily). If $u = t$, the search succeeds. If $h(s, t) > h(u, t)$, we set $s$ to $u$ and repeat. If $h(s, t) \leq h(u, t)$, the search fails.

For each $\alpha$, pick 1,000 pairs of nodes and compute the average path length for the searches that succeeded. Then draw a plot of the average search time (number of steps it takes to reach $t$) as a function of $\alpha$. Also, plot the search success probability as a function of $\alpha$.

Briefly comment on the plots and explain the shape of the curve.

What to submit

(a) Write the expression and a short explanation.

(b) Write the expression and a short explanation.

(c) Write a short proof.

(d) Write a short proof.

(e) Write a short proof.

(f) Write a short proof, give an expression for $\theta$ and brief argument.

(g) Write a short proof.

(h) Provide both plots and a brief comment. Upload code to http://snap.stanford.edu/submit
4 Variations on a Theme of PageRank [25 points – Omid]

Personalized PageRank

Personalizing PageRank is a very important real-world problem: different users find different pages relevant, so search engines can provide better results if they tailor their page relevance estimates to the users they are serving. Recall from class that PageRank can be specialized with clever modifications of the teleport vector. In this question, we’ll explore how this can be applied to personalize the PageRank algorithm.

Assume that people’s interests are represented by a set of representative pages. For example, if Zuzanna is interested in sports and food, then we could represent her interests with the set of pages \{www.espn.com, www.epicurious.com\}. For notational convenience, we’ll use integers as names for webpages.

(a) [7 points] Suppose you have already computed the personalized PageRank vectors for the following users:

- Agatha, whose interests are represented by the teleport set \{1, 2, 3\},
- Bertha, whose interests are represented by the teleport set \{3, 4, 5\},
- Clementine, whose interests are represented by the teleport set \{1, 4, 5\}, and
- DeShawn, whose interests are represented by the teleport set \{1\}.

Without looking at the graph, can you compute the personalized PageRank vectors for the following users? If so, how? If not, why not? Assume a fixed teleport parameter $\beta$.

i. [2 points] Eloise, whose interests are represented by the teleport set \{2\}.

ii. [2 points] Felicity, whose interests are represented by the teleport set \{5\}.

iii. [3 points] Glynnis, whose interests are represented by the teleport set \{1, 2, 3, 4, 5\} with weights 0.1, 0.2, 0.3, 0.2, 0.2, respectively.

(b) [3 points] Suppose that you’ve already computed the personalized PageRank vectors of a set of users (denote the computed vectors $V$). What is the set of all personalized PageRank vectors that you can compute from $V$ without accessing the web graph?

Spam Farms

The staggering number of people who use search engines to find information every day makes having a high PageRank score a valuable asset, which creates an incentive for people to game the system and artificially inflate their website’s PageRank score. Since the PageRank algorithm is based on link structure, many PageRank spam attacks use special network configurations to inflate a target page’s PageRank score. We’ll explore these configurations, called spam farms, in this part of the question.

(c) [5 points] Consider the spam farm shown in Figure 3. The spammer controls a set of boosting pages $1, \ldots, k$ (where page $i$ has PageRank score $p_i$) and is trying to increase the PageRank of the target page $p_0$. The target page receives $\lambda$ amount of PageRank from the rest of the
Figure 3: A spam farm.

Figure 4: Linked spam farms.

graph (represented by the dotted arrow). This means that \( \lambda = \sum_{i \in S} r_i d_i \), where \( S \) is the set of nodes in the rest of the network that link to \( p_0 \) and \( r_i \) and \( d_i \) represent, respectively, node \( i \)'s PageRank score and outdegree. Let \( N \) denote the number of pages in the entire web, including the boosting pages and target page. Calculate the PageRank of the target page \( p_0 \) with this configuration as a function of \( \lambda \), \( k \), \( \beta \), and \( N \). Your solution should not include other parameters (such as \( p_i \)).

Hint: You can write an expression for \( p_0 \) in terms of \( p_1, \ldots, p_k \), \( \beta \), \( k \), and \( N \). You can also write the PageRanks of the boosting pages in terms of \( p_0 \), \( \beta \), \( k \), and \( N \).

(d) [5 points] It turns out that the structure in Figure 3 is optimal, in the sense that it maximizes the target’s PageRank \( p_0 \) with the resources available. However, it may still be possible to do better by joining forces with other spammers. It also turns out that the \( \lambda \) contribution from the rest of the graph is mathematically equivalent to having some extra number of boosting pages, so for the rest of this question we’ll ignore \( \lambda \). Consider the case where two spammers link their spam farms by each linking to the other’s target page as well as their own, as shown in Figure 4. Let \( p'_0 \) and \( q'_0 \) denote the PageRank values of the target pages if the spammers use the individual spam configuration that we discussed in the previous question (without \( \lambda \) this time). Calculate the PageRanks of the target pages \( \overline{p_0} \) and \( \overline{q_0} \) with the new configuration as a function of \( k, m, \beta, \) and \( N \) (where again \( N \) denotes the number of pages in the web graph). What are \( \overline{p_0} - p'_0 \) and \( \overline{q_0} - q'_0 \)? Are the spammers better off than they would be if they operated independently (i.e. is \( \overline{p_0} + \overline{q_0} > p'_0 + q'_0 \))? 

(e) [5 points] There are other ways spammers can form alliances. Consider the setup shown in
Figure 5, where the spammers only link their target pages together. Again let $p'_0$ and $q'_0$ denote the PageRank values of the target pages that the spammers would get on their own. Calculate the PageRank of the target pages $\overline{p_0}$ and $\overline{q_0}$ with this configuration as a function of $k$, $m$, $\beta$, and $N$ (where again $N$ denotes the number of pages in the web graph). What are $\overline{p_0} - p'_0$ and $\overline{q_0} - q'_0$? Are the spammers better off than they would be if they operated independently (i.e. is $\overline{p_0} + \overline{q_0} > p'_0 + q'_0$)?

![Figure 5: Another way of linking two spam farms.](image)

**What to submit**

(a) For each of (i),(ii), and (iii), 'yes' or 'no' and an explanation of why or why not.

(b) A mathematical expression for the set in terms of $V$.

(c) An expression for $p_0$ in terms of $k, \beta,$ and $N$. Also show how you derived the expression.

(d) (i) Expressions for $p'_0 + q'_0$ and $\overline{p_0} + \overline{q_0}$ in terms of $k, m, \beta$ and $N$. Show how you derived these expressions.

(ii) A yes/no answer to whether spammers are better off and a brief explanation.

(e) (i) Expressions for $p'_0 + q'_0$ and $\overline{p_0} + \overline{q_0}$ in terms of $k, m, \beta$ and $N$. Show how you derived the expressions.

(ii) A yes/no answer to whether spammers are better off and a brief explanation.