CS134: Network Science **Due:** Wednesday, October 24th 2018, 11:59am

Profs. Singer and Mitzenmacher **Problem set 6**

1. Cascade Behavior in Trees (20 points). Consider a variant of the branching process discussed in lecture. The base graph is a rooted, infinite k-ary tree, with $k \geq 2$ an integer (each node has k children and one parent, except for the root which has no parents). We define level n to be the set of nodes in this tree at distance n from the origin.

Let $A_v \in \{0, 1\}$ indicate whether node v is active. If $A_v = 1$, v is active, and if $A_v = 0$, v is inactive. The root node r (the sole node at level $n = 0$) is assumed to be active. For any active node v at level n, the number of its active children at level $n + 1$ is a random variable Z_v (and all sets of active children that have the same size are equally likely). An inactive node will have no children.

All the Z_v are mutually independent and identically distributed, according to a probability distribution function P, so that $\mathbb{P}(Z_v = \ell) = P(\ell)$. Assume that $0 < P(0) < 1$ and $0 < P(1) < 1$.

Given these data, the whole (random) tree of active nodes can be constructed inductively. Let X_n be the number of active nodes at level n.

a. (5 points) What is $\mathbb{E}[X_n]$ in terms of P? (The expectation is taken across all realizations of the random tree.)

b. (5 points) Let q_n be the probability that $X_n \neq 0$. For $n > 0$, we have $q_n = f(q_{n-1})$. Write down the formula for f and explain your answer. (The function f will likely be in terms of P and k.)

c. (5 points) Recall the function $f : [0, 1] \to [0, 1]$ you described above. Give the sign of its first derivative and the sign of the second derivative. (Here P is again arbitrary, subject to the assumptions given in the statement.)

d. (5 points) Define $q_\infty = \lim_{n\to\infty} q_n$ when this limit exists. Describe how you can find q_∞ by looking at a plot. Give a verbal statement of the meaning of q_∞ (this is required to get full points on this part).

2. Stochastic Matrices (20 points). Let P be an $n \times n$ matrix with every entry a nonnegative real number, and the (i, j) entry denoted by p_{ij}. Assume that for every $i \in \{1, \ldots, n\}$ we have $\sum_{j=1}^{n} p_{ij} = 1$. A matrix having these properties is called *row-stochastic*. Assume Q is another row-stochastic matrix.

a. (2 points) Write down a formula for the (i, j) entry of the matrix product $R = QP$.

b. (6 points) For every $i \in \{1, \ldots, n\}$, compute $\sum_{j=1}^{n} r_{ij}$ (give a number) and justify your answer. Note that this is the sum of all entries in row i of R.

c. (6 points) Compute the row sum of all the entries in row i of P^k, where $k \geq 1$, and justify your answer.

d. (6 points) For each of the three different 5×5 matrices P below, compute $L = P^{100}$ to three decimal digits of precision using your favorite technology. Also compute LP for each case. What do you notice?

$$P_1 = \begin{bmatrix} 0.1 & 0.4 & 0 & 0.3 & 0.2 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0.6 & 0 & 0.2 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.3 & 0.3 & 0.1 & 0.1 & 0.2 \end{bmatrix}, P_2 = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.3 & 0.4 & 0 & 0.2 \\ 0.3 & 0.1 & 0.1 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.3 & 0.2 \\ 0.2 & 0.1 & 0.3 & 0.2 & 0.2 \end{bmatrix}, P_3 = \begin{bmatrix} 0.1 & 0.3 & 0.4 & 0.2 & 0 \\ 0 & 0 & 0.5 & 0.4 & 0.1 \\ 0.2 & 0.2 & 0.3 & 0.2 & 0.1 \\ 0 & 0.3 & 0.5 & 0.1 & 0.1 \\ 0.6 & 0.2 & 0.1 & 0 & 0.1 \end{bmatrix}$$

3. Submodularity (20 points). A set function $f : 2^N \to \mathbb{R}$ is a function whose domain is the set of all subsets from some set N (i.e. the power set of N) and its range is the set of real numbers $\mathbb{R}$. We will be interested in a special class of set functions, namely those that are monotone, normalized, and most importantly *submodular*:

- A function $f : 2^N \to \mathbb{R}$ is *monotone* if for any $S, T \subseteq N$ we have that $S \subseteq T$ implies that $f(S) \leq f(T)$, i.e. if we add more elements into the set S, the value of the function will increase;
- A function $f : 2^N \to \mathbb{R}$ is *normalized* if $f(\emptyset) = 0$, i.e. if we choose no items we will get 0 value;
- A function $f : 2^N \to \mathbb{R}$ is *submodular* if it has a diminishing returns property. That is, for any $S \subseteq T \subseteq N$ and $a \notin T$ we have that:

$$f(S \cup \{a\}) - f(S) \geq f(T \cup \{a\}) - f(T)$$

Intuitively the property of diminishing returns says "the more I have, the less happy an extra item a makes me" (or the more I already have, the less important acquiring a becomes). Many natural functions are submodular as you will see both in this problem and in the class.

a. (10 points) A function $f(S)$ is called a *cover* function if it can be defined in the following manner. Given a set $S \subseteq \{1, 2, \ldots, n\}$ for some $n \geq 1$, there exists a collection of subsets $C_1, \ldots, C_n$, where all $C_i \subseteq N$ for a set N, and $f(S) = |\cup_{i \in S} C_i|$. Intuitively, the value $f(S)$ is the number of elements in N that are "covered" by the set S. Show that any cover function is normalized, monotone, and submodular.

b. (10 points) Given an undirected graph $G = (V, E)$, and some subset of nodes $S \subseteq V$, the **neighborhood** of S, denoted $\mathcal{N}(S)$, is the set of all nodes in V that have at least one neighbor in S. Show that the function $f(S) = |\mathcal{N}(S)|$ is a cover function (assume the set of nodes V can be denoted as a set of integers from 1 to n, much like what you've done in previous coding problems).

4. Learning Influence (20 points). Assume that we have a graph $G = (V, E)$ with n nodes and m edges. Each node $v \in V$ corresponds to an individual and each edge $(u, v) \in E$ to a friendship between its two endpoints. Each edge is also associated with a probability $p_{(u,v)} \in (0, 1)$ that denotes the *influence probability* between nodes u and v as discussed in class. These probabilities are unknown to us and we would like to estimate them from data (e.g. cascades). Towards this end, we are presented with k samples of the form (e, y) where $e = \{u, v\} \in E$ denotes an edge and $y \in \{0, 1\}$ denotes whether node u managed to influence node v or not.

a. **(4 points)** What would be a good estimation, $\hat{p}_e$, of the real probability, p_e of an edge e?

b. **(8 points)** How many samples are required to guarantee that for a specific edge $e \in E$, $\hat{p}_e$ and p_e are in absolute distance of at most ϵ away with probability at least $1-\delta$, for any $\epsilon, \delta \in (0,1)$?

c. **(8 points)** How many samples per edge are required to guarantee that for all edges $e \in E$, $\hat{p}_e$ and p_e are in absolute distance of at most ϵ away with probability at least $1-\delta$, for any $\epsilon, \delta \in (0,1)$?

5. Learning Influence Experimentally (20 points) Let's say we have a graph $G = (V, E)$ and some opinion data for each node $v \in V$. Again, each edge $(u, v) \in E$ is associated with a probability $p_{(u,v)} \in (0, 1)$ that denotes the *influence probability* between nodes u and v, as discussed in class. Assume each node has an opinion that changes over time based on the influence of its neighbors for a particular observed cascade. Remember that the opinion of a node is either 0 (node is inactive) or 1 (node is active).

Let's focus on a particular observed cascade c, that starts from a given root node r. The cascade proceeds in discrete time steps t where t ranges from 1 to $n-1$ (every cascade can survive for at most $n-1$ time steps). In every time-step $t+1$ each node u that became active during time-step t, will try activate each of its neighbors $v \in N(u)$ independently and will succeed with probability $p_{(u,v)}$. Node u will only try to activate each of its neighbors once in each cascade. If u is activated at time step t, on time step $t+1$ it will try to activate each of its neighbors. If u fails to activate v, it will not try again, but another one of v's neighbors may activate it in the future. Moreover, once a node is active, it cannot become inactive.

In the traditional IC model it is assumed that in every time step t we observe the nodes that became active at time t. Hence, a cascade will be a set of sets of the form: $C = \{\{r\}, X_1, X_2, \ldots\}$, where r is the root node and X_t are the new nodes that became active on time-step t. However, for the purpose of this assignment we will make a slightly stronger assumption: a cascade will consist of the root node r and then for every time step we will be observing all the edges that became active in this time step[1]. So if at some time step t node u influences node v instead of having $v \in X_t$ we will have $(u, v) \in X_t$.

In reality, we cannot a priori know the influence probabilities. However, we can learn them from data! This is the purpose of this assignment. Towards this end you are required to:

a. **(4 points)** Write a function `load_graph`, that takes as input a .txt file containing the edges of a graph, together with the influence probabilities and returns the adjacency list of this graph (in the form of list of lists) and the list of the influence probabilities (also in the form of list of lists). You can assume that the graph in the .txt file will be undirected and each line will be either a comment (starting with '#'), or it will contain 3 numbers "u v p" where the first two are integers and correspond to the endpoints of the edge and the last one is a float that corresponds to the influence probability of this edge. For example when we invoke `load_graph(blah.txt)`, where the file `blah.txt` is the following:

```
#This is an example
0 1 0.5
1 2 0.25
1 3 0.75
```

[1] this simplifies the problem because in the former case if we see a node becomes active in time $t+1$ while having two neighbors that are active in time t, there is no way to know which of the two neighbors actually influenced it. In the latter case this is not a problem since we will observe the activation of all the edges, so we know whether the node was influenced by its first parent, its second one or both of them.

we should get the adjacency list: `[[1], [0, 2, 3], [1], [1]]` and the list of the probabilities: `[[0.5], [0.5, 0.25, 0.75], [0.25], [0.75]]`.

b. **(4 points)** Write a function `load_cascades`, that takes as input a .txt file containing a set of cascades, and returns a list of these cascades. Every cascade should also be a list of the form: `[<root>, ` $[X_1, X_2, \ldots]$ `]`, where X_t is the list of edges that got activated in time step t. The form of this .txt file will be:

```
#New Cascade
<root>
#Time Step
<edges activated in this time step>
#Time Step
<edges activated in this time step>
...
#Time Step
#EOF
```

The last line of the file will always be `#EOF`.

For example when we invoke `load_cascades(casc_examp.txt)`, where the file `casc_examp.txt` is the following:

```
#New Cascade
5
#Time Step
5 1
5 7
5 9
#Time Step
1 2
7 8
7 3
#Time Step
3 15
#Time Step
#New Cascade
1
#Time Step
1 2
#Time Step
2 7
#Time Step
#EOF
```

we should get the cascade list: `[[5, [[[5,1], [5,7], [5,9]], [[1,2], [7,8], [7,3]], [[3, 15]], []]], [1, [[[1,2]], [[2,7]], []]]]`.

c. **(4 points)** Write a function `estimate_probabilities`, that takes as input a graph (in the form of 5a), a list of cascades (in the form of 5b) and estimates the empirical probability of each edge in the graph. The estimate for every edge should simply be $\frac{n^+}{n}$, where n^+ is the number of activations of the edge while n the total number that the edge was accessed (*remember that the graph is undirected*). When an edge is accessed 0 times, this implies that we don't have

any information about the influence probability of this edge (you can denote this by setting the probability to be -1 for example).

d. **(4 points)** Write a function `avg_error`, that takes as input a graph (in the form of 5a), a list of cascades (in the form of 5b) and the list of the real probabilities in the edges (in the form of 5a) and computes the average learning error in the network, i.e. the average absolute difference between the real influence probability of an edge and your empirical estimate. That is: avg_error $= \frac{1}{|E|}\sum_{e\in E}|\hat{p}_e - p_e|$. Again, when an edge is accessed 0 times, this implies that we don't have any information about the influence probability of this edge and thus we will define the error between our prediction and the real value to be 1 (the maximum possible).

e. **(4 points)** Use the file "cascades1.txt" that contains 500 cascades, to learn the influence probabilities in the graph "graph1.txt" (contains a graph on 500 nodes). Plot the average error as a function of the number of cascades (your estimation of the edge probabilities should improve with the more cascades that you observe). Do the same for the files "graph2.txt" (1000 nodes) and "cascades2.txt" (1000 cascades).