CS224W: Social and Information Network Analysis **Fall 2013**

Problem Set 3

Due 9:30am November 7, 2013

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible. Please fill the cover sheet (http://cs224w.stanford.edu/cover.pdf) and submit it as a front page with your answers. We will subtract 2 points for failing to include the sheet. Include the names of your collaborators (see the course website for the collaboration or honor code policy). You are allowed to take maximum of 1 late period (see course website for a definition of a late period).

Regular (non-SCPD) students should submit hard copies of the answers either in class or in the submission cabinet (see course website for location). You should also include source code and any other files you used in the paper submission.

SCPD students should submit their answers through SCPD. The submission must include the cover sheet, all the answers, the source code and the usual SCPD routing form (http://scpd.stanford.edu/generalInformation/pdf/SCPD_HomeworkRouteForm.pdf).

Additionally, all students (Regular and SCPD) should upload all code to http://snap.stanford.edu/submit/.

Questions

1 Influence Maximization [25 points – Bell, Zhemin]

In class we discussed influence maximization problem and the greedy hill-climbing approach to solving it. In the algorithm, we add nodes to the current set one at a time. At step 0, we have an empty set S_0. At step $i > 0$, we pick the node which maximizes the marginal gain: $S_i = \arg\max_u f(S_{i-1} \cup u) - f(S_{i-1})$, where $f(S)$ denotes the number of nodes influenced by the initially active set S.

As we showed in the class the hill climbing algorithm cannot guarantee an optimal solution. In other words, there might exist a set T with $|T| = i$ and $f(S_i) < f(T)$. Parts (a) and (b) of this problem ask you to construct examples where this is the case. Your answer should consist of: (1) For every node u its influence set X_u (you can describe the set or draw a directed graph where an edge from A to B indicates that node A influences node B with probability 1), (2) S_i, the set of nodes that a greedy hill climbing would choose after i iterations, and (3) T, the optimal set of i nodes.

For all the questions, you can assume: (1) The nodes in S are influencing themselves, i.e., the count of total influence $f(S)$ includes the nodes in S. (2) When several nodes have the same level of marginal gain, we choose one of them at random.

(a) [8 points] For $i = 2$, construct an example where $f(S_i) < f(T)$. That is, hill-climbing will only find a non-optimal solution.

(b) [8 points] For $i = 3$, construct an example where $f(S_i) \leq 0.8f(T)$. That is, hill-climbing will only find a solution that is at 80% of the optimal solution.

(c) [4 points] Give a property of influence sets X_u such that $f(S_i) = f(T)$. By this we mean: what is a sufficient property of influence sets of nodes such that greedy hill-climbing always outputs the optimal solution. The property does not need to be a necessary one. It must be a property of X_u. Note properties like "the network has only i nodes" are not valid as correct answers.

(d) [5 points] Assume that we stop hill-climbing after k steps and $|S_k| = |T| = k$. Recall that in the class we proved a bound in the form of

$$f(T) \leq f(S_k) + \sum_{i=1}^{k} \delta_i, \tag{1}$$

where $\delta_1, .., \delta_k$ are the largest k values of $\{f(S_k \cup u) - f(S_k)|u\}$ (See the course slides). Construct a family of examples for which $f(S_k) + \sum_{i=1}^{k} \delta_i - f(T)$ can be arbitrarily large.

What to submit

1.1) Submit an example: X_u, S, T.

1.2) Submit an example: X_u, S, T.

1.3) Submit a property and a brief explanation.

1.4) Submit a family of examples and a brief explanation.

2 Empirical Power Laws [25 points – Yoni]

In this problem, you will generate a dataset following power-law distribution and test various methods to estimate the α exponent. For convenience, you are free to use any third party

library.

(a) [5 points] Recall from class, the probability density function (pdf) of power-law distribution, where $h(x) \propto x^{-\alpha}$, is:

$$P(X = x) = \frac{\alpha - 1}{x_{\min}} \left(\frac{x}{x_{\min}} \right)^{-\alpha}$$

Please derive the expression of $P(X \geq x)$, the Complementary Cumulative Distribution Function (CCDF), in terms of α.

(b) [5 points] Generate a dataset of 100,000 values following a power-law distribution with exponent $\alpha = 2$ and $x_{\min} = 1$. Plot on a log-log scale the empirical estimate of $P(X = x)$ from your data. To compute this estimate on the data you generated, you may round each point to the nearest integer and compute a normalized histogram over integer values. If you prefer, you may use other density estimation procedures. Do not plot using "bars" but use "points". Also include the actual probability density function as a line of slope $-\alpha$ on the same plot. This will help you verify that you generated the data correctly. Explain briefly what you did with points x for which the estimate of $P(X = x)$ is zero, and why.

(c) [3 points] One way to fit a power law distribution to data is to create a histogram of the frequencies of x. On a log-log plot this becomes $\ln(h(x)) \propto -\alpha \ln(x)$. You can extract the value of α by performing a least-squares linear regression on the log-log histogram data. Again, you may round points to the nearest integer to compute the histogram. Report the value of α you find. Can you improve the accuracy of this method by not using all the data? Explain how, and implement this improvement.

(d) [2 points] Similar to (c), now run a least-squares linear regression on the CCDF. This is the empirical CCDF, and you should estimate it from the data. Report the value of α you find.

(e) [5 points] Run a maximum likelihood estimate of α on the original data, with $x_{\min}$ fixed to 1. Report the value you find.

(f) [5 points] Repeat the above process 100 times. Each time, generate a new dataset of 100,000 values, and estimate α using each of the three methods above. Compute the mean and variance for each method over these 100 repetitions. Based on these values, which method gives the best estimate? Which one the worst? Explain why.

Notes

- All estimates are "reasonably" close to the true value, but not all methods are equally good at estimating. Part of the goal of this question is to understand the differences between the methods, so please do not ask us to verify that the values you got are reasonable.
- You may use any software package that you like, including of course other packages for Python like numpy and scipy.
- If you are not sure how to solve least-squares regression in Python, one possible solution you may use is the *lstsq* function in the *linalg* package, which is part of numpy.

What to submit

- For (a), submit the expression for the CCDF, and the derivation of this expression.
- For (b), submit the plot of the empirical and theoretical pdfs, and your code. Also submit a brief explanation about points where the estimate is zero.
- For (c)-(e), submit the values of α using each of the methods, and your code. For (c) also submit your suggestion for a way to improve the accuracy, with an explanation.
- For (f), submit the mean and variance for the estimates of α using each of the methods, and a short analysis of these results. Also submit your code.

3 Combination of exponentials for generating power laws [25 points – Peter, Justin]

In class we've already seen how the process of preferential attachment can generate power-law distributions. In this question you will explore a different power-law generating process: the *combination of exponentials.* This approach was developed in order to explain why word frequency follows a power-law distribution.

3.1 Empirical power laws in word-frequency distributions [10 points]

Download the files

- http://www.stanford.edu/class/cs224w/homeworks/hw3/mobydick.txt and
- http://www.stanford.edu/class/cs224w/homeworks/hw3/donquijote.txt,

which contain Herman Melville's *Moby Dick* (in English) and Miguel de Cervantes Saavedra's *Don Quijote* (in Spanish), respectively, with one single word on each line.

(i) [5 points]

For each of the two files, plot the distribution of word frequencies on a log-log scale. That is, the x-axis should show the number of times a word can appear (i.e., *word frequency*); the y-axis should show the fraction of distinct words with frequency x.

(ii) [5 points]

Using the maximum-likelihood technique fit power-law coefficients α to both plots (give one pair of coefficients per plot). List $x_{\min}$ and α for both texts (Moby Dick and Don Quixote).

What do you conclude about the statistical properties of human language?

3.2 Theoretical power laws in monkey novels [15 points]

It might seem surprising that the word-frequency distribution of natural language should follow a power law as strictly as it does, and there has been much scholarly dispute about the processes that make all human writers sample words from exactly the same type of distribution. In this sub-question you will prove that even a randomly typing monkey produces words according to a power-law frequency distribution, and that the power law observed in humans is maybe not that surprising after all.

Assume our monkey hits the space bar with probability q_s and each of m letters with probability $(1 - q_s)/m$. In this setting, we define a *word* as a sequence of letters separated by spaces.

(iii) [4 points]

How many distinct words of length y are there? Thus, show that $p(y)$, the probability of a word having length y, is proportional to e^{ay} (where the word is picked uniformly at random from the set of *distinct* words). Give an expression for a in terms of m.

(iv) [5 points]

Consider a particular word of length y. Let x be the *relative frequency* of this word (i.e., the probability of the word, followed by a space, in the monkey's output). Derive an expression for x in terms of y, q_s, and m. Conclude that the frequency x of a word is proportional to e^{by}. Give an expression for b in terms of q_s and m.

(v) [6 points]

Using parts (iii) and (iv), show that $p(x)$, the probability of a word having relative frequency x (where the word is picked uniformly at random from the set of *distinct* words of all possible lengths), is proportional to $x^{-\alpha}$. (*Hint: don't compute* $p(x)$ *exactly.*)

- ⋆ Although y can take on only discrete values, you may pretend the function $p(y) = e^{ay}$ you derived in (iii) is defined over continuous rather than integral values of y, and that $p(x)$ is also a continuous function in x. You might also want to use the fact that if x is an invertible function of y, then $|p(y)\ dy| = |p(x)\ dx|$.

What is α in terms of q_s and m?

To which value does α converge as m gets large and q_s gets small? Does it converge from above or below? (E.g. $\lim_{x\to\infty} \frac{1}{x}$ approaches 0 from above while $\lim_{x\to\infty} -\frac{1}{x}$ approaches 0 from below.)

Compare this value to the value of α you empirically found in (ii).

What to submit

3.1) **(i)** Plot the distribution of word frequencies for both texts. Print and submit your code online. **(ii)** Give values of α and $x_{\min}$ for both texts. Make a conclusion about the statistical properties of human language.

3.2) **(iii)** Show that $p(y)$ is proportional to e^{ay}. Give a in terms of m. **(iv)** Derive x in terms of y, q_s, m. Show that x is proportional to e^{by}. Give b in terms of q_s, m. **(v)** Show $p(x)$ is proportional to $x^{-\alpha}$. Give α in terms of q_s, m. The value to which α converges (from above or below?). Compare this value of α to that from (ii).

4 Exploring Robustness of Networks [25 Points - Ashley, Ashwin]

In this problem we will examine the relationship between connectivity and robustness in several different networks. Our analysis will be closely related to a paper by Albert, Jeong and Barabasi, *Error and Attack Tolerance of Complex Networks* (Nature, 2000), which may serve as useful background reading.

While working through this problem, bear in mind that there are a number of ways to formalize the notion of network robustness. Depending on the application, we may be interested in the mere possibility of reaching other nodes, or the average shortest path length between pairs of nodes, or the worst case shortest path length (i.e., the diameter). For part 4.1, we will be using the diameter and for part 4.2, we will be using the fraction of nodes in the

largest connected component.

We will use the following three *undirected* networks for this problem. (Note: Duplicate edges are ok when randomly generating edges. Their effect on the final plots is trivial.)

- *G(n,m) Random network (G_{nm})*: Pick $n = 10670$ nodes and $m = 22002$ edges at random to construct this network. Generate the network yoursefl (don't use SNAP functions).

- *Autonomous System Network (AS)* - You can download this network from `http://snap.stanford.edu/data/oregon1_010331.txt.gz`. It is an undirected network for the Internet network with a total of 10,670 nodes and 22,002 edges.

- *Graph with Preferential Attachment (PA)*: Generate an undirected graph using a variation of the preferential attachment method described in Problem 2. Begin with a complete graph of 40 nodes. At each time step, introduce a new node to the network. Add two edges for the new node in the following fashion: select an edge from the current network uniformly at random and add an edge between the new node and one of the edge's endpoints. Pick the edge's endpoint uniformly at random. Continue doing this till your graph has 10,670 nodes. Your graph should ideally have 22,040 edges (the actual number may be lesser because of the random number generation but it is expected to be around 22,000). Generate the network yourself (don't use SNAP functions).

Tip : Because graph generation is time-consuming, consider using the SNAP functions `SaveEdgeList()` and `LoadEdgeList()` to avoid constantly recreating the networks. Also, for a given network, you should use identical realizations of the network to do all sub-parts of this problem, e.g., once you generate a $G(n, m)$ network, use a replica of it for all sub parts.

You will be deleting nodes from these networks and computing a given property of the network after each deletion phase. Since it would be computationally too intensive to do this after the deletion of every node, delete nodes in batches of X. First delete X nodes, compute the network property, delete additional X nodes, compute the property again and so on till $Y\%$ of the original nodes have been deleted.

You will be considering two node deletion policies in this question. The first, called *Failure*, samples nodes uniformly at random and deletes them from the graph. As the name suggests, this corresponds to a real life scenario where every node is equally probable to break down. The second policy, called *Attack*, corresponds to a scenario where an adversary selectively targets a specific node. In this case, always delete the nodes in decreasing order of their degree, i.e. highest degree node first. You may use the SNAP `GetMxDegNId()` function to

find the highest degree node in a graph. As you will learn from your analysis, the internet network is fairly easy to attack!

4.1 [13 Points]

The property you will be computing in this part is the diameter of the graph. Since computing the diameter can be very expensive, you will sample 20 nodes uniformly at random from the graph, find the shortest path length from these sampled nodes to all other nodes (by doing a BFS for example) and take the maximum value. This logic is already implemented in the `GetBfsFullDiam()` function in SNAP that you may use directly.

$$diam = \max_{i \in sampleSet} (\max_{0 \leq j \leq |N|} d(u_i, u_j))$$

For each of the three networks, plot the diameter of the network against the percentage of nodes removed from the network for both the *Failure* and the *Attack* scenario, i.e., your plot should have a total of six lines, two lines each for the three networks, one for the *Failure* scenario and one for *Attack*. Do this for the following two scenarios:

- $X = |N|/100$ and $Y = 50$
- $X = |N|/1000$ and $Y = 2$

where N is the number of nodes in the original network (in this case 10670).

Analyze the *Failure* vs *Attack* scenarios for each of the networks, as well as, comment on the difference (if any) between the behavior of different networks under the two deletion policies.

4.2 [12 Points]

In this part, the property being used is the fraction of nodes which are part of the largest connected component. The denominator of the fraction is the current graph size (as opposed to the original graph size). Plot this fraction versus the percentage of the deleted nodes of all three networks for the two deletion policies in the same plot (again 6 lines in the plot). Use $X = |N|/100$ and $Y = 50$. Analyze the *Failure* vs *Attack* scenarios for each of the networks. Comment on the robustness of the networks using the plot.

What to submit

4.1)
- Graph showing Diameter for following plots for $X = |N|/100$, $Y = 50$ (all on the same graph): G_{nm} Failure, G_{nm} Attack, AS Failure, AS Attack, PA Failure, PA Attack

- Graph showing Diameter for following plots for $X = |N|/1000$, $Y = 2$ (all on the same graph): G_{nm} Failure, G_{nm} Attack, AS Failure, AS Attack, PA Failure, PA Attack
- Analysis of *Failure* vs. *Attack* scenarios for each of the networks
- Comparison of behavior of the different networks under the two deletion policies
- Print out your code and also submit it online

4.2)
- Graph showing Fraction of Nodes in LCC ($|LCC|/|CurrentGraph|$) for following plots for $X = |N|/100$, $Y = 50$ (all on the same graph): G_{nm} Failure, G_{nm} Attack, AS Failure, AS Attack, PA Failure, PA Attack
- Analysis of *Failure* vs *Attack* scenarios for each of the networks
- Comment on robustness of the three networks.
- Print out your code and also submit it online