CS224W: Analysis of Networks **Fall 2017**

Problem Set 2

Due 11:59pm PDT October 26, 2017

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible. You are allowed to take a maximum of 1 late period (see the information sheet at the end of this document for the definition of a late period).

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

Submitting answers: Prepare answers to your homework in a single PDF file and submit it via GradeScope. Make sure that the answer to each sub-question is on a *separate, single page*. The number of the question should be at the top of each page. We recommend that you use the submission template latex file to prepare your submission.

Fill out the information sheet located at the end of this problem set or at the end of the template file and sign it in order to acknowledge the Honor Code (if typesetting the homework, you may type your name instead of signing). This should be the last page of your submission. Failure to fill out the information sheet will result in a reduction of 2 points from your homework score.

Starter Code: For problems 2 and 3, starter code is available at `http://web.stanford.edu/class/cs224w/homeworks/hw2/starter_code/`. Note that the starter code is a nudge towards *one* way of solving the problem, amongst many. Feel free to cannibalize it in any way possible to construct your own solution.

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

1 The Configuration Model [25 points – Praty, Ziyi]

A common method of analyzing the properties of real world networks is to analyze their behavior in comparison with a generated theoretical model referred to as a"null model." We have previously discussed some null models in class (Lecture 3), such as the Erdős-Rényi model and the configuration model. While the Erdős-Rényi model has many nice theoretical properties, the configuration model is useful because it generates random networks with a specified degree sequence. In other words, given a real network, the configuration model allows you to sample from the space of networks that have the exact same sequence of degrees (i.e., the sampled random networks have the same degree distribution as the network you are studying).

In this problem, we will analyze the configuration model in detail. For more background the on the configuration model, check out these lecture notes by Aaron Clauset: `http://tuvalu.santafe.edu/~aaronc/courses/5352/fall2013/csci5352_2013_L11.pdf`

We will use the configuration model to examine the properties of the power grid of the western US (as of 1998). In this network, the nodes represent transformers, substations, and generators and the

(undirected) edges represent transmission lines. The dataset for this problem should be downloaded from http://web.stanford.edu/class/cs224w/homeworks/hw2/data_sets/USpowergrid_n4941.txt. The first two columns of the file are the two nodes comprising an undirected edge; ignore the third column.

1.1 Simulating the configuration model through stub-matching [15 points]

First, we will generate a configuration model by the stub matching method that was briefly mentioned during the "random graphs" lecture (Lecture 3). The intuition behind this algorithm is that we first break the network apart into a bunch of "stubs", which are basically nodes with dangling edges; then we generate a random network by randomly pairing up these stubs and connecting them.

A random network is generated from the stub-matching algorithm as follows:

1. First, calculate the degree sequence of the real world network by creating a vector $\overrightarrow{k} = \{k_1, k_2, ...k_n\}$ where k_i is the degree of node i and n is the number of nodes in the graph.
2. Then create an array v and fill it by writing the index i exactly k_i times for each element in the degree sequence. Each of these i's represents a stub of the i^{th} node in the graph.
3. Next, randomly permute the elements of v.
4. Finally, create a random network by connecting adjacent pairs in v with an edge. In other words, connect up the nodes corresponding to the first and second elements, the third and fourth, and so on. Formally, v is of length $2m$ and the m undirected edges of the graph are $(v_1, v_2), (v_3, v_4), \ldots, (v_{2m-1}, v_{2m})$.

If this was confusing, don't worry! Figure 1 provides a visual example of the algorithm that should clarify things. One import note is that this algorithm can sometimes create improper (i.e., non-simple) networks with self-loops or multiple edges between two nodes; if this happens, then you must reject this sampled network and try again. In other words, you must make sure the sampled network is simple (no self-loops or multi-edges) after you construct it and draw a new sample if the constructed network is non-simple.

(a) **[10 points]** Draw 100 random (simple) network samples using the stub-matching algorithm using the degree sequence of the power grid network. Calculate the average clustering coefficient of each of these samples and report the mean value across samples.

(b) **[2 points]** Consider node i with degree k_i, and node j with degree k_j. Under a random matching on the half-edges, what is the probability that node i and j are connected? What does the result imply? For simplicity, self-loops and multi-edges are allowed during stub-matching for this subproblem.

(c) **[3 points]** We artificially reject self-loops when building the graph, but for large graphs, self-loops won't be a big trouble. Consider a graph with a very large number of edges m. What is the expected number of self-loops if we build the graph by stub-matching? What does this result imply about the fraction of self-loops in the graph? (Hint: write the expected number

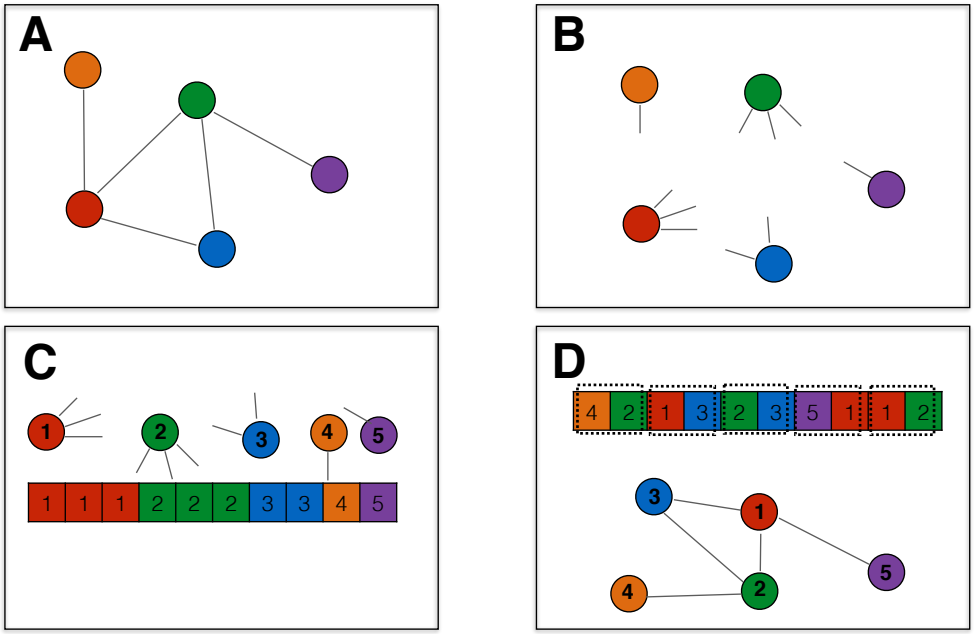


Figure 1: **Example run of the stub matching algorithm. (A)** A toy example network where the nodes are labeled with different colors. **(B)** We break the toy network into "stubs", where each node of degree k has k half-edges coming out of it. **(C)**, We label the nodes and make a vector v where each node's label is repeated in the vector k times (where k is that node's degree). **(D)**, We take a random permutation of v and then connect adjacent pairs (dashed boxes) to create a random network.

as function of $\langle k^2 \rangle$ and $\langle k \rangle$, where $\langle k^m \rangle = \frac{1}{n} \sum_i k_i^m$ and n is the number of nodes)

1.2 Simulating the configuration model through rewiring [10 points]

A second (and more popular) approach to sampling from the configuration model is to do "edge rewiring". The idea with this algorithm is to start with an empirical network and then randomly rewire edges until it is essentially random.

Implement "edge rewiring" for the US power grid graph. To do this we iteratively repeat the following process:

1. Randomly select two distinct edges $e_1 = (a, b)$ and $e_2 = (c, d)$ from the graph. We will try to re-wire these edges.
2. Randomly select one of endpoint of edge e_1 and call it u. Let v be the other endpoint in e_1. At this point, either $u = a$, $v = b$ or $u = b$, $v = a$. Do the same for edge e_2. Call the randomly selected endpoint w and the other endpoint x.
3. Perform the rewiring. In the graph, replace the undirected edges $e_1 = (a, b)$ and $e_2 = (c, d)$ with the undirected edges (u, w) and (v, x) *as long as this results in a simple network (no self-loops or multi-edges)*. If the result is not a simple network, reject this rewiring and return to step 1; otherwise, keep the newly swapped edges and return to step 1.

Run your edge rewiring implementation for 10000 iterations on the power grid network. Every 100 iterations, calculate the average clustering coefficient of the rewired network. Then plot the average clustering coefficient as a function of the number of iterations. Briefly comment on the curve that represents the model being rewired by both explaining its shape and how it relates to the clustering coefficient generated by the stub-matching algorithm (hint: they should be pretty similar).

What to submit

Page 1:
- The average clustering coefficient generated from the stub-matching algorithm (averaged over 100 samples).
- Expression for the probability that nodes i and j with degrees k_i and k_j are connected. 2-3 sentences explaining the expression and what it implies.
- The expected number of self-loops. 2-3 sentences explaining what the results implies about the fraction of self-loops in the graph?

Page 2:
- A plot of the average clustering coefficient as a function of the iteration number for the rewiring algorithm. A brief comment on this plot (1–2 sentences).

2 Signed Triad Analysis [45 points – Poorvi, Anunay]

In this question you will explore the notion of balance in signed networks. The first part will explore the distribution of signed triads in an empirical network. The remaining parts will consider different generative models that could potentially produce this sort of triad distribution.

2.1 Signed Triads in Epinions [15 points]

Download the following files.

Epinions dataset: http://web.stanford.edu/class/cs224w/homeworks/hw2/data_sets/epinions-signed.txt
Starter code: http://web.stanford.edu/class/cs224w/homeworks/hw2/starter_code/q2_starter.py

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.

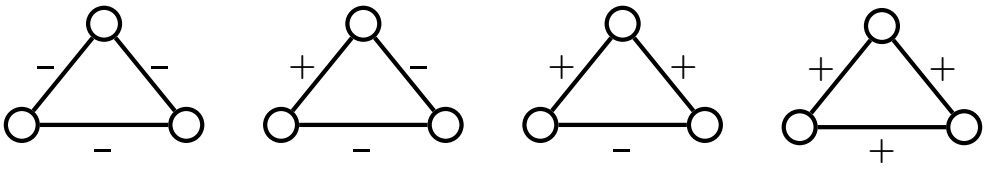
Since a member can either trust or distrust other members, we have a special type of network called signed network where every edge has a sign. A positive sign on an edge indicates trust between the two users whereas a negative edge indicates distrust between the two users.

We will consider the graph as undirected and study the various forms of triads as shown in Figure 2.

(a) **[5 points]** Calculate the count and fraction of triads of each type in the Epinions network. *(Count each triad only once and do not count self edges)*

(b) **[5 points]** Calculate the fraction of positive and negative edges in the graph. Let the fraction of positive edges be p. Assuming that each edge of a triad will independently be assigned a positive sign with probability p and a negative sign with probability $1 - p$, calculate the probability of each type of triad.

Figure 2: The four different types of signed triads.



(a) t_0 (b) t_1 (c) t_2 (d) t_3

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

2.2 Balance in a Random Signed Network [8 points]

Now that we have seen how signed triads are distributed in a real network, we will analyze a simple generative model of signed networks to see how *surprising* this empirical distribution really is. In particular, we will analyze how likely it is for a balanced triad to occur in this random model.

Consider the following simple model for constructing random signed networks, which we will call the G^+ model. Start with a complete graph on n nodes. For each edge e mark its sign as positive with probability p (and thus negative with probability $1-p$). All edges are undirected.

Let G_B denote the event that a graph G is balanced. In this question, we'll show that $P(G_B) \to 0$ as $n \to \infty$ for graphs generated according to the G^+ model. Assume that $p = 1/2$.

(a) **[2 points]** Let T be a maximum set of ***disjoint-edge*** triangles in G. A *disjoint-edge* set of triangles is one in which every edge is in exactly one triangle. Give a simple lower bound for $|T|$, the number of triangles that don't share any edges in G (the bound should be an increasing function of n).

(b) **[2 points]** For any triangle in G, what is the probability that it is balanced?

(c) **[2 points]** Using the simple lower bound from part (i), give an upper bound on the probability that *all* of the triangles in T are balanced. Show that this probability approaches 0 as $n \to \infty$.

(d) **[2 points]** Explain why the last part implies that $P(G_B) \to 0$ as $n \to \infty$.

2.3 A Dynamic Process Model of Balance [5 points]

If balanced signed networks do not show up by chance, as we showed in the previous part of the question, how do they arise? One class of mechanisms that researchers have proposed and studied are *dynamic processes*, in which signed networks can evolve over time to become more balanced. The following describes a very simple example of such a dynamic process:

(I) Pick a triad at random.

(II) If it's balanced, do nothing.

(III) Otherwise, choose one of the edges uniformly at random and flip its sign (so that the triad becomes balanced).

Consider the following claim: in this process, the number of balanced triads can never decrease. Is this true? If so, give a proof, otherwise give a counterexample.

2.4 Simulation [8 points]

Now let us run simulations of the dynamic process on small networks.

(I) Create a complete network on 10 nodes.

(II) For each edge, choose a sign $(+, -)$ at random $(p = 1/2)$.

(III) Run the dynamic process described in the previous part for 1,000,000 iterations.

Repeat this process 100 times (so that you run the dynamic process described in the previous part for 100 different graphs generated from the G^+ model). What fraction of the networks end up balanced? [Hint: To check whether a network is balanced or not, remember that a graph is balanced if and only if it's possible to separate the nodes into two "factions" such that every pair of nodes in the same faction is linked by a positive edge and every pair of nodes in different factions is linked by a negative edge. To speed things up, you can stop the process once the network is balanced, because once the network becomes balanced it will never change.]

2.5 The Role of New Nodes I [4 points]

Another way signed networks can evolve over time is if new nodes join the network and create new signed edges to nodes already in the network. Consider the network shown in Figure 3. Is it possible to add a node D such that it forms signed edges with all existing nodes (A, B, and C), but isn't itself part of any unbalanced triangles? Justify your answer.

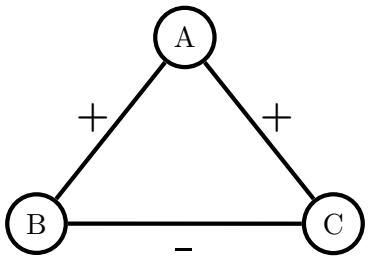


Figure 3: An unbalanced network with one triangle.

2.6 The Role of New Nodes II [5 points]

Using your answer to the previous sub-problem, consider the following question.

Take any complete signed network, on any number of nodes, that is unbalanced. When (if ever) is it possible for a new node X able to join the network and form edges to all existing nodes in such a way that it does not become involved in any unbalanced triangles? If it is possible, give an example. If it is not, give an argument explaining why.

What to submit

Page 3:
- (part a) the count and fraction of triads in each type of opinion network;
- (part b) the fraction of positive and negative edges in the network and the probability of each triad
- (part c) which triads appear more/less than the baseline values and a brief (2–3 sentence) explanation.

Page 4:
- (part a) a simple lower bound for $|T|$.
- (part b) the probability that any triangle in G is balanced.
- (part c) an upper bound on the probability that all triangles in T are balanced, and show that the probability approaches 0 as $n \to \infty$.
- (part d) 1–2 sentences explaining why $P(G_B) \to 0$ as $n \to \infty$.

Page 5:
- Whether the number of balanced triads never decreases.
- A proof (if true) or a counterexample (if false).

Page 6:
- The fraction of networks that end up balanced.

Page 7:
- Whether it is possible to add a node D, and a justification.

Page 8:
- Whether it is possible for a new node X to join.
- An example (if possible) or argument (if impossible).

3 Decision-based Cascades: A Local Election [30 points – Anthony, Yokila]

It's election season and two candidates, Candidate A and Candidate B, are in a hotly contested city council race in sunny New Suburb Town. You are a strategic advisor for Candidate A in charge of election forecasting and voter acquisition tactics.

Based on careful modeling, you've created two possible versions of the social graph of voters. Each graph has 10,000 nodes, where nodes are denoted by an integer ID between 0 and 9999. The edge lists of the graphs may be downloaded from:

- Graph 1: http://web.stanford.edu/class/cs224w/homeworks/hw2/data_sets/graph1.txt
- Graph 2: http://web.stanford.edu/class/cs224w/homeworks/hw2/data_sets/graph2.txt

Both graphs are **undirected**.

Given the hyper-partisan political climate of New Suburb Town, most voters have already made up their minds: 40% know they will vote for A, 40% know they will vote for B, and the remaining 20% are undecided. Each voter's support is determined by the last digit of their node id. If the last digit is 0–3, the node supports A. If the last digit is 4–7, the node supports B. And if the last digit is 8 or 9, the node is undecided.

The undecided voters will go through a 10-day decision period where they choose a candidate based on the majority of their friends. The **decision period** works as follows:

1. The graphs are initialized with every voter's initial state (A, B, or undecided).
2. In each iteration, every undecided voter is decides on a candidate. Voters are processed in increasing order of node ID. For every undecided voter, if the majority of their friends support A, they now support A. If the majority of their friends support B, they now support B. "Majority" for A means that strictly more of their friends support A than the number of their friends supporting B, and vice versa for B (ignoring undecided friends).
3. If a voter has an equal number of friends supporting A and B, we assign support for A or B in alternating fashion, starting with A. In other words, as the voters are being processed in increasing order of node ID, the first tie leads to support for A, the second tie leads to support for B, the third for A, the fourth for B, and so on. This alternating assignment happens at a global level for the whole network, across all rounds. (Keep a single global variable that keeps track of whether the current alternating vote is A or B, and initialize it to A in the first round. Then as you iterate over nodes in order of increasing ID, whenever you assign a vote using this alternating variable, change its value afterwards.)
4. When processing the updates, use the values from the current iteration. For example, when updating the votes for node 10, you should use the updated votes for nodes 0–9 from the current iteration, and nodes 11 and onwards from the previous iteration.
5. There are 10 iterations of the process described above.
6. On the 11th day, it's election day, and the votes are counted.

Note that only the undecided voters go through the decision process. The decision process does not change the loyalties of those voters who have already made up their minds.

3.1 Basic setup and forecasting [8 points]

Read in the two graphs and assign the initial vote configurations to the network. Then, perform the 10 iterations of the voting process. Which candidate wins in Graph 1, and by how many votes? Which candidate wins in Graph 2, and by how many votes?

3.2 TV Advertising [8 points]

You have amassed a substantial war chest of \$9000, and you have decided to spend this money by showing ads on the local news. Unfortunately, only 100 New Suburb Townians watch the local news—those with ids 3000–3089. However, your ads are extremely persuasive, so anyone who sees the ad is immediately swayed to vote for candidate A regardless of his/her previous decision. You may spend \$1000 at a time on ads. The first \$1,000 reaches voters 3000–3009, the second \$1000 reaches voters 3010–3019, and so on. In other words, the total of $\$k$ in advertising would reach voters with ids from 3000 to $3000 + \frac{k}{100} - 1$. This advertising happens before the decision period. *After voters are persuaded by your ads, they never change their minds again.*

Simulate the effect of advertising spending on the two possible social graphs. First, read in the two graphs again and assign the initial configurations as before. Now, before the decision process, you purchase $\$k$ of ads and go through the decision process of counting votes.

For each of the two social graphs, plot $\$k$ (the amount you spend) on the x-axis (for values $k = 1000, 2000, \ldots, 9000$) and the number of votes you win by on the y-axis (that is, the number of

votes for A less the number of votes for B). Put these on the same plot. What's the minimum amount you can spend to win the election in each of the two social graphs?

Note that the TV advertising affects all of the voters who see the ads and not just those who are undecided.

3.3 Wining and Dining the High Rollers [8 points]

TV advertising is only one way to spend your campaign war chest. You have another idea to have a very classy \$1000 per plate event for the high rollers of New Suburb Town (the people with the highest degree in the social graph). You invite high rollers in order of how many people they know, and everyone that comes to your dinner is *instantly persuaded to vote for candidate A regardless of his/her previous decision.* This event will happen before the decision period. When there are ties between voters with the same degree, the high roller with lowest node ID get chosen first.

Simulate the effect of the high roller dinner on the two graphs. First, read in the graphs and assign the initial configuration as before. Now, before the decision process, you spend $\$k$ on the fancy dinner and then go through the decision process of counting votes.

For each of the two social graphs, plot $\$k$ (the amount you spend) on the x-axis (for values $k = 1000, 2000, \ldots, 9000$) and the number of votes you win by on the y-axis (that is, the number of votes for A less the number of votes for B). What's the minimum amount you can spend to win the election in each of the two social graphs?

Note that wining and dining sways all the voters to vote for A and not just those who are undecided.

3.4 Analysis [6 points]

Plot the degree distributions on a log-log scale of the two graphs on the same plot (as in Question 1.1 on Problem Set 1). Although both graphs have roughly the same number of edges, one was generated from an Erdős-Rényi Random graph model and the other was generated from a preferential attachment model. Which is which? In 1–2 sentences, briefly summarize how this information explains the results of Question 3.3.

What to submit

Page 9:
- Which candidate wins and by how many votes in each graph.

Page 10:
- Plot of winning margin in each graph as a function of \$k (on the same plot)
- The minimum amount you can spend to win the election in each graph

Page 11:
- Plot of winning margin in each graph as a function of \$k (on the same plot)
- The minimum amount you can spend to win the election in each graph

Page 12:
- Log-log plot of the degree distributions (on the same plot),
- which graph corresponds to which random graph model
- 1–2 sentences on why the random graph models explain the results of Question 3.3.

Information sheet
CS224W: Analysis of Networks

Assignment Submission Fill in and include this information sheet with each of your assignments. This page should be the last page of your submission. Assignments are due at 11:59pm and are always due on a Thursday. All students (SCPD and non-SCPD) must submit their homeworks via GradeScope (http://www.gradescope.com). Students can typeset or scan their homeworks. Make sure that you answer each (sub-)question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code at http://snap.stanford.edu/submit. Put all the code for a single question into a single file and upload it. Please do not put any code in your GradeScope submissions.

Late Homework Policy Each student will have a total of *two* free late periods. *Homeworks are due on Thursdays at 11:59pm PDT and one late period expires on the following Monday at 11:59pm PDT.* Only one late period may be used for an assignment. Any homework received after 11:59pm PDT on the Monday following the homework due date will receive no credit. Once these late periods are exhausted, any assignments turned in late will receive no credit.

Honor Code We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down their solutions independently i.e., each student must understand the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions obtained from the web (github/google/previous year solutions etc.) is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code very seriously and expect students to do the same.

Your name: _____

Email: _____ **SUID:** _____

Discussion Group: _____

I acknowledge and accept the Honor Code.

(Signed) _____