CSCI 5352, Network Analysis and Modeling **Prof. Aaron Clauset**
Problem Set 4 **Fall 2021**

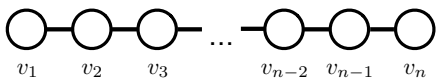
There are 100 regular points and 35 extra points possible on this assignment.

1. (15 pts) A 2015 study of tenure-track Computer Science professors at PhD-granting universities in the U.S. and Canada recorded the geographic region of each professor's doctoral and faculty institutions. As a network, each directed edge $(u \to v)$ represents the movement of a person, from (doctoral) university u to (faculty) university v, and nodes have a label that indicates which geographic region where the corresponding university is located. Using the regions as communities, the mixing matrix $\mathcal{M}$ gives the fraction of edges that point from region r (rows) and region s (columns):

$\mathcal{M}$	Northeast	Midwest	South	West	Canada
Northeast	0.119	0.053	0.074	0.055	0.022
Midwest	0.031	0.067	0.061	0.026	0.011
South	0.025	0.027	0.083	0.024	0.006
West	0.049	0.033	0.043	0.073	0.011
Canada	0.006	0.005	0.005	0.005	0.085

   Calculate and report (i) the values e_{rr} and a_r that appear in the definition of modularity Q [see Eq. (7.58) in *Networks*], and (ii) the contribution Q_r for each region to the total modularity Q. Hence, calculate and report the modularity Q of the network with respect to regional flow of faculty. What do you conclude about overall assortativity in this system, and about the way the different regions are coupled together?

2. (10 pts) Consider an undirected "line graph" with n vertices and $n-1$ edges, like so:

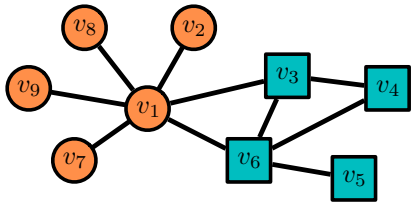


   (a) Show mathematically that if we divide this network into any two contiguous groups, such that one group has g connected vertices and the other has $n-g$, the modularity Q takes the value

$$Q = \frac{3 - 4n + 4gn - 4g^2}{2(n-1)^2} \quad .$$

   (b) (10 pts *extra credit*) Considering the same graph, show that when n is even, the optimal division, in terms of modularity Q, is the division that splits the network exactly down the middle, into two parts of equal size.

3. (12 pts) Use the $n = 9$ node network and $c = 2$ group partition $\vec{z}$ given below to complete the following calculations.



(a) Using the stochastic block model (SBM), (i) tabulate and report the maximum likelihood mixing matrix $\mathcal{M}$, and (ii) calculate and report the likelihood $\mathcal{L}$ of this network under that model.

(b) Using the degree-corrected stochastic block model (DC-SBM), (i) tabulate and report the mixing matrix ω and group degrees κ, and (ii) calculate and report the likelihood $\mathcal{L}$ of this network under that model.

(c) Comment on which model is more likely to generate the observed network.

4. (63 pts total) *Detecting community structure.* In this problem, you will implement and apply the locally greedy heuristic for estimating the DC-SBM, as a way to find communities in a network. Because the algorithm has several parts, we will build it up in pieces.

   Recall that the algorithm runs in phases, and begins with a random initial partition. When a phase begins at $t = 0$, all nodes are "unfrozen." At each step t in a phase, it freezes exactly one node and the phase ends when $t = n$ and all nodes are frozen. At each step of a phase, the algorithm considers *all possible* group-changes for all unfrozen nodes, of which there are $(n - t)(c - 1)$ such single-node moves. Of these possible moves, it makes a greedy choice and then "freezes" the node it moved. At the end of a phase, if the best of the n partitions it explored z_* is better than the initial partition z_0, then it begins a new phase with z_* as the initial partition and we unfreeze all the nodes. Otherwise, it halts and returns the current phase's z_0.

   To implement this algorithm, you will (a) first write the core logic of taking a single step within a phase. Then, you will (b) wrap that logic in a loop over nodes to run a single phase. Next, you will (c) wrap the phase logic in a loop that runs the entire algorithm from start to finish once. Finally, you will (d) wrap your algorithm in a loop that will try multiple random partitions, reporting the best final result of the set. With this function in hand, you'll then (e) apply it to some real-world networks and get some communities!

   (a) (18 pts) Write a function `makeAMove()` that implements the core logic of the algorithm:

   - Input: a graph G, a current partition z_t, a number of groups c, and a binary "isFrozen" vector f (if $f_i = 1$, then i is "frozen")

- This function should loop over the $n-t$ unfrozen nodes (those with $f_i = 0$) and $c-1$ possible single-node moves for each of those nodes and compute a log-likelihood of each combination.
- Over that loop, it should track the single best log-likelihood $\mathcal{L}_*$ and the corresponding "best move," i.e., a pair (i, z_i) where z_i is i's new group label r.
- Output: the final $\mathcal{L}_*$ and the pair (i, z_i) the represent the change to z_t that would yield $\mathcal{L}_*$.
- Apply `makeAMove()` to the simple graph G in question 3, with $c = 3$, a random input partition z_t, and $f_i = 0$ for all i except $f_2 = 1$. Provide a visualization of (i) the input graph G with its node labels z_t and its log-likelihood under the DC-SBM, and (ii) the graph G with node labels after applying the best move found and its DC-SBM log-likelihood.

Hint: after you move a node to its $c-1$ alternative group assignments, don't forget to put i back in group s before you go to the next unfrozen node.

(b) (15 pts) Write a function `runOnePhase()` that runs a single phase from start to finish, starting with an input partition z_0 and and evaluates the algorithm's convergence criteria:

- Input: a graph G, an initial partition z_0, and the number of groups c.
- Output: the best partition of the phase z_* and its corresponding DC-SBM log-likelihood $\mathcal{L}_*$, a binary value h that indicates whether the halting criteria was met or not, and a list ℓ that stores the $n+1$ log-likelihood values for the $n+1$ partitions considered in the phase (this part is purely so that we can make a nice visualization of the algorithm's progression).
- Remember that when the phase begins, f is all zeros (all nodes unfrozen), and we call `makeAMove()` to decide which node to "freeze" in a given step of the phase.
- Apply `runOnePhase()` to the simple graph G in question 3, with $c = 3$, a random input partition z_0. Provide a visualization of (i) the input graph G with its node labels z_0 and its log-likelihood under the DC-SBM, (ii) the graph G with node labels z_* at the end of the phase and its DC-SBM log-likelihood $\mathcal{L}_*$, and (iii) a plot of ℓ_t showing how the log-likelihood evolved over the course of the phase.

Hint: if you use python, you may need to use the `copy.deepcopy()` function to "save off" a separate copy of z_t when you update z_*, rather than a "shallow" copy (a pointer).

(c) (15 pts) Write a function `fitDCSBM()` that runs the full locally gready heuristic.

- Input: a graph G, a number of groups c, and the maximum of phases allowed T.
- Output: the best partition z_* and its log-likelihood $\mathcal{L}_*$ found across all phases, an integer p_c that counts the number of phases used, and a list ℓ that stores the log-likelihood values for the $p_c(n+1)$ steps taken across all phases (also purely to make a nice visualization of the algorithm's progression).

- Apply `fitDCSBM()` to the simple graph G in question 3 with $c = 3$ and $T = 30$ phases at most. Provide a visualization of (i) the graph G with final node labels z_* and its DC-SBM log-likelihood $\mathcal{L}_*$, (ii) a plot of ℓ_t showing how the log-likelihood evolved over the course of all the phases, and (iii) report the κ vector and ω matrix for the final partition.
- Comment briefly on what you see in both the quality of the final partition and in the behavior of ℓ_t.

Hint: How good a partition can you get? I got a log-likelihood of -48.14 or so.

(d) (15 pts) Write a function that repeatedly applies `fitDCSBM()` to the Zachary karate club network, and tracks the *best* output over repetitions. (Although the algorithm itself is deterministic, it begins at a random initial partition, and so re-running the algorithm many times allows us to sample different local optima, looking for a good one.) Tinker with the number of repetitions needed to obtain the *optimal* $c = 2$ group partition described in the lecture notes. Comment briefly on how well your algorithm worked at this task, how many runs it took (a lot? a little? more than you expected? fewer?) Bask in the glory of having implemented and applied this amazing algorithm to reproduce the results of Karrer and Newman (2011).

(e) (15 pts *extra credit*) Coarse-grain some more real data. Try out your DC-SBM algorithm on a few of the networks we've used in previous problem sets, or other networks you can find on ICON. Use a low integer choice of c. For each, if it's small enough, make a visualization of G with the final partition z_*, and inspect the final κ and ω. Comment on the kind of stylized community structure you find, if any (none, assortative, disassortative, ordered, or core-periphery).

Warning: depending on your implementation and choice of language, the algorithm may not scale well, so be careful with networks $n > 100$.

5. (10 pts *extra credit*) Reading the literature.

Choose a paper from the Supplemental Reading list on the external course webpage . Read the whole paper. Think about what it says and what it finds. Read it again, if it's not clear. Then, write a few sentences for each of the following questions in a way that clearly summarizes the work, and its context.

- What was the research question?
- What was the approach the authors took to answer that question?
- What did they do well?
- What could they have done better?
- What extensions can you envision?

Do not copy text from the paper itself; write your own summary, in your own words. (Using terminology from the paper is okay, of course.) Be sure to answer each of the five questions. The amount of extra credit will depend on the accuracy and thoughtfulness of your answers.