ELEC 573: Network Science and Analytics **Fall 2021**

Homework 2

Hand-out Date: 10/04/21 **Due Date: 10/19/21**

This homework focuses on two applications of networks to very different domains: genetics and epidemics. I co-developed the first exercise with Weiyu Huang and Alejandro Ribeiro and the second one with Michael Schaub and Ali Jadbabaie.
Please hand in all of your work, including executable code. The grading is over 100 points distributed as indicated throughout the document.

2.1) Classification of cancer types [35 points]. Let us begin by analyzing a genetic network describing gene-to-gene interactions. The network was downloaded from the NCI Nature database and it is included in the data folder along with this homework. You might want to use the function `scipy.io.loadmat()` to load the data `geneNetwork_rawPCNCI.mat` in Python. The network consists of 2458 genes and, loosely speaking, two genes are connected if the proteins encoded by them participate in the same metabolism process.

We are going to study the genetic profiles of 240 patients diagnosed with different subtypes of ovarian cancer. We will see that by interpreting these genetic profiles as graph signals defined on the genetic networks, we will be able to clearly distinguish patients from different subtypes. Load the file `signal_mutation.mat`. You will see the aggregated graph signals $\mathbf{X} \in \mathbb{R}^{240\times 2458}$. The i-th row of this matrix represents the genetic profile $\mathbf{x}_i$ for the i-th patient. The n-th gene for this patient is mutated if the n-th entry of $\mathbf{x}_i$ is 1 and is not mutated (normal) if the n-th entry is 0. Patients diagnosed with the same disease may exhibit different phenotypes, i.e., different variations of the same disease. The most effective therapies for different phenotypes may differ a lot and, for this reason, it is very beneficial if we can distinguish phenotypes based on the genetic profiles. Load the file `histology_subtype.mat` and you will see a vector $\mathbf{y} \in \mathbb{R}^{240}$ that describes the patients phenotypes. The i-th element is 1 if patient i has serous subtype ovarian cancer and 2 if the patient has endometrioid subtype ovarian cancer. Our goal is to better differentiate between patients with these two subtypes based on graph Fourier analysis.

a) *Distinguishing power [10 points].* Take the graph-shift operator to be the Laplacian $\mathbf{S} = \mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$. We want to find the frequencies $\mathbf{v}_k$ such that the corresponding Graph Fourier Transform coefficient $\tilde{\mathbf{x}}(k)$ differs the most between patients with serous subtype and endometrioid subtype. There are many ways to do this, and we consider the following simple heuristic. First compute the GFTs $\tilde{\mathbf{x}}_i = \mathbf{V}^\top \mathbf{x}_i$ for all patients. Then for each frequency k, define the distinguishing power of $\mathbf{v}_k$ as

$$\text{DP}(\mathbf{v}_k) = \left| \frac{\sum_{i:y_i=1} \tilde{\mathbf{x}}_i(k)}{\sum_i \mathbb{1}\{y_i = 1\}} - \frac{\sum_{i:y_i=2} \tilde{\mathbf{x}}_i(k)}{\sum_i \mathbb{1}\{y_i = 2\}} \right| \Big/ \sum_i |\tilde{\mathbf{x}}_i(k)|, \tag{2.1}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function. In words, $\text{DP}(\mathbf{v}_k)$ computes the normalized difference between the mean GFT coefficient for $\mathbf{v}_k$ among patients with serous subtype and the mean GFT coefficient among patients with endometrioid subtype. Generate a plot of $\text{DP}(\mathbf{v}_k)$ versus k for all frequency indices k. Also, generate a boxplot of all the values taken by $\text{DP}(\mathbf{v}_k)$.

b) *k-NN for classification [8 points].* Implement a k nearest neighbors (k-NN) classifier and perform leave-one-out cross validation. More precisely, implement the following steps:
1) Compute the pairwise Euclidean distance between all pairs of patients using the original graph signals $\mathbf{X}$.
2) For each patient, compare the most common histology of its k nearest neighbors to its actual diagnosis obtained from $\mathbf{y}$.
3) Compute the accuracy of the classifier by aggregating all the comparisons of the previous step.
Report the accuracies for $k \in \{3, 5, 7\}$.

c) *Filtering almost all frequencies [7 points].* Consider a graph filter whose frequency response is to only let pass the frequency with the highest distinguishing power, i.e.

$$\tilde{h}_1(k) = \begin{cases} 1, & \text{if } k = \text{argmax}_k \, \text{DP}(\mathbf{v}_k), \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Denote the filtered GFT as $\tilde{\mathbf{x}}_i^f$ and its inverse GFT as $\mathbf{x}_i^f$. Form a filtered graph signal matrix $\mathbf{X}^f$ with each of its rows representing the filtered genetic profile of each patient. Run the k-NN classifier of the previous point but using $\mathbf{X}^f$ as your data. Report your accuracies for $k \in \{3, 5, 7\}$. What do you observe?

d) *Filtering most frequencies [10 points].* Consider now the more general filter

$$\tilde{h}_p(k) = \begin{cases} 1, & \text{if DP}(\mathbf{v}_k) \geq p\text{-th percentile of the distribution of DP,} \\ 0, & \text{otherwise,} \end{cases} \tag{2.3}$$

where p is any real number between 0 and 100. This family of graph filters keep the information conveyed in frequencies that are distinguishable for two subtypes to some extent. The p here is chosen to select the number of frequencies to keep. Run the k-NN classifier on the filtered signals for different values of p. Report your accuracies for $k \in \{3, 5, 7\}$ and $p \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$.

2.2) Epidemics across the world [65 points]. In this problem we will consider the issue of epidemic spreading on networks and how to prevent their outbreaks. We will consider theoretical and computational aspects of an epidemic model while focusing on a real-world network of global flight connections as our main dataset.

a) *Construct a network from the flight data [10 points].* You have been provided with 2 `csv` files (named `airport_Nodes_GC.csv` and `airport_Edges_GC.csv`) that contain all the information of all the airports (nodes), and flights (links)[1]. The edges are directed and weighted, and are proportional to the number of daily flights from one airport to another. You should construct a network G from this data as follows.

- Read in the edge data and create an initial (directed) network.
- We will work with undirected graphs, so form the undirected graph of this network by symmetrizing its adjacency matrix as follows
$$\mathbf{A}_{sym} = (\mathbf{A} + \mathbf{A}^\top)/2.$$
- If the undirected graph obtained is not connected, keep its giant component as your new graph and work with it from here onwards.

Hint: Note that command `networkx.DiGraph.to_undirected` returns an undirected graph with the same name and nodes and with edge $(u, v, data)$ if either $(u, v, data)$ or $(v, u, data)$ is in the digraph. If both edges exist in the digraph and their edge data is different, **only one edge is created with an arbitrary choice of which edge data** to use. You must check and correct for this manually if desired, or use other method to obtain the symmetry operation required above.

b) *Plot the airport network [10 points].* Plot the flight network, using some of the extra information for the node files: use the longitude and latitude coordinates provided when drawing the nodes, and scale the node size according to their eigenvector centrality. You should take into account the weights of the edges when computing this centrality. You may find it useful as well to adjust the transparency of the edges for visual clarity. **Hint:** As a sanity check, the two nodes with highest eigenvector centrality (eigenvector normalized to have unit norm) should have the values approximately 0.179 and 0.170.

We will now study an epidemic model on this network, where every node is in one of two states: it is either infected, or susceptible to being infected. We denote the set of the vertices by $V = \{v_1, \ldots, v_n\}$ and the adjacency matrix by $\mathbf{A} = [a_{ij}]$. The state of node v_i at time $t \geq 0$ is a binary random variable $X_i(t) \in \{0, 1\}$. The state $X_i(t) = 0$ (resp., $X_i(t) = 1$) indicates that node v_i is in the susceptible (resp., infected) state. We define the vector of states as $X(t) = [X_1(t), \ldots, X_n(t)]^\top$. The state of a node can experience two possible stochastic transitions:

i) Assume node v_i is in the susceptible state at time t. This node can switch to the infected state during the time interval $[t, t + \Delta)$ with a probability that depends on: (a) an infection rate β, (b) the probability of

[1] The original dataset was made available by Tore Opsahl and is discussed in the blog post *Why Anchorage is not (that) important: Binary ties and Sample selection.* Available at `http://wp.me/poFcY-Vw`.

contact with its neighboring nodes $N_i = \{j \mid a_{ij} \neq 0\}$ and their states. We can write the probability of this transition as

$$\text{Prob}[X_i(t+\Delta) = 1 | X_i(t) = 0, X(t)] = 1 - \Pi_{j \in N_i \wedge X_j(t)=1}(1 - \beta a_{ij}). \tag{2.4}$$

We usually have $\beta \ll 1$. Therefore, instead of (2.4), we use the following first-order approximation:

$$\text{Prob}[X_i(t+\Delta) = 1 | X_i(t) = 0, X(t)] = \sum_{j \in N_i} \beta a_{ij} X_j(t). \tag{2.5}$$

ii) Assuming that node v_i is infected, the probability of v_i recovering back to the susceptible state in the time interval $[t, t+\Delta)$ is given by

$$\text{Prob}[X_i(t+\Delta) = 0 | X_i(t) = 1] = \gamma, \tag{2.6}$$

where $0 \leq \gamma \leq 1$ is the curing rate.
c) *[5 points]* Show that the first-order approximation in (2.5) is in fact an upper bound for the original transition probabilities in (2.4).

The spread model characterized by (2.5) and (2.6) may be hard to analyze for large-scale networks. One standard approach is to use a *mean-field approximation* of the model: define $p_i(t) = \text{Prob}[X_i(t) = 1] = \mathbb{E}[X_i(t)]$, i.e., the marginal probability of node v_i being infected at time t. We can use (2.5) and (2.6) to approximate the dynamics of $p_i(t)$:

$$\frac{dp_i(t)}{dt} = \beta(1 - p_i(t)) \sum_{j=1}^{n} a_{ij} p_j(t) - \gamma p_i(t). \tag{2.7}$$

This approximation is widely used in the field of epidemic analysis and control, since it performs numerically well for many realistic network topologies.
d) *Simulating the spread dynamics using the mean-field approximation [20 points].* Suppose that the first 20 nodes (according to their ids in the dataset) are initially infected. Use (2.7) to simulate the evolution of the expected size of the infection defined as $\bar{p}(t) = \sum_{i=1}^{n} p_i(t)$ over time, assuming a recovery rate $\gamma = 0.1$ and an infection rate $\beta = 0.1$. (Hint: you can use $\frac{dp_i(t)}{dt} \approx \frac{p_i(t+\Delta) - p_i(t)}{\Delta}$ where $\Delta = 0.05$ and time horizon $[0, 5]$). Repeat the experiment for $\beta = 0.05$ and $\beta = 0.01$. Plot three curves (one for each value of β) showing the evolution of the expected size of the infection over time.

It can be shown that a sufficient condition for virus extinction, that is to ensure that the virus will eventually die out, is to have

$$\lambda_{\max}(\mathbf{A}) < \frac{\gamma}{\beta}. \tag{2.8}$$

An immunization strategy is to choose a subset of nodes $I \subseteq V$ and immunizing them against the virus. An immunized node can neither get infected nor pass the infection. We call an immunization strategy "effective" if it results in the eventual extinction of the virus (almost surely), *no matter how widespread the initial infection is*. A potential idea for designing an effective immunization strategy is to rank the nodes based on some centrality measure and immunize them in order of their centralities until the condition (2.8) is satisfied. Note that immunizing node v_i is equivalent to removing the i-th row and column from $\mathbf{A}$.
e) *[20 points]* Assume that $\beta = 0.01$ and $\gamma = 0.4$. (i) What is the minimum number of immunizations required to satisfy condition (2.8) if you use (weighted) degree centrality to sort the nodes? (ii) What is this minimum number when you use (weighted) eigenvector centrality instead?

In solving this last point, compute a single ranking for the nodes for the complete network and delete nodes one by one until you satisfy the condition (2.8). No need to recompute the centralities after deleting nodes (although that could be another reasonable heuristic).