- This homework is due at 11:59pm on February 18, 2021. Please submit by email to `natalies@cs.unc.edu+comp790`.
- There are 3 data files provided for the following questions, including, `Levine_matrix.csv`, `cell_graph.edgelist`, and `population_names_Levine_13dim.txt`. Instructions for how to use these data will be provided in each homework problem.
- Unless explicitly asked to write code from scratch, you can use publicly available code. Please reference any code or libraries that you use.
- You are welcome to consult with other colleagues, but please write up your own independent solution.
- You are welcome to use Python, Julia, or R here, though all hints are given for Python, and we will use Node2Vec (Python)
- You are welcome to write up your assignment using the `HW1_790-166.tex` template, or write up the solutions in the method of your choice.
- This homework is worth 82 points total.

Problem 1

(23 Points Total) **(Adjacency Matrix Math)**
Consider an undirected, unweighted graph with N nodes and its corresponding adjacency matrix, $\mathbf{A}$.

- (5 points) Let $\mathbf{1}$ be the column vector of only 1s, and in this case, N 1s. Using $\mathbf{A}$ and $\mathbf{1}$, write an expression for a vector of node degrees, $\mathbf{k}$, such that the ith entry of $\mathbf{k}$, k_i, represents the number of edges incident on node i.
- (3 points) Again, using $\mathbf{A}$ and $\mathbf{1}$ write an expression for the total number of edges in the graph.
- (1 point) Verify the two expressions that you just defined for the following matrix, $\mathbf{A}$, and show that you get $\mathbf{k} = [1, 2, 1]$ and that the number of edges in the graph is 2. You can show this by drawing the graph.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

- (10 points) (**Graph Laplacian**) Recall that the the un-normalized Graph Laplacian, $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Use the provided graph (encoded as an edgelist) `cell_graph.edgelist` and write a function that computes the Graph Laplacian. As a reminder, $\mathbf{D}$ is an $N \times N$ matrix with node degrees on the diagonal, and $\mathbf{A}$ is the adjacency matrix. You can copy and paste your code or take a screen shot, since it should only be a few lines of code.
  - (3 points) Use the function you just wrote to calculate the Graph Laplacian of the Graph stored in `cell_graph.edgelist`. Make a histogram to visualize the distribution of eigenvalues of $\mathbf{L}$. What is the smallest eigenvalue of $\mathbf{L}$? (hint: you can use `https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html#numpy.linalg.eig`, or equivalent.)
  - (1 point) How does the observation of this smallest eigenvalue relate to the structure of the graph?

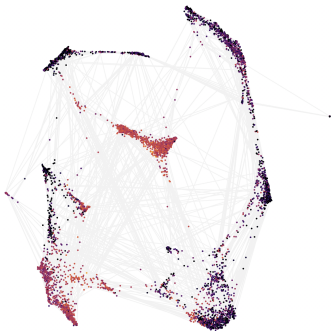

Figure 1: A visualization of the graph in homework problem 2.

Problem 2

(29 Points Total) **Playing with Single Cell Data**
We will consider data from a mass cytometry experiment obtained from `http://flowrepository.org/id/FR-FCM-ZZPH`. Here, we are considering the expression of 13 different protein markers across a set of cells. It has already been pre-processed for you. From the entire dataset, 5,000 cells were sampled for further analysis. You can use the following accompanying data as follows.

- `Levine_matrix.csv` is the cell × marker matrix. Note that the last column is labels for the cells. Let's call this matrix, **X**. **You should not use this column for any kind of clustering.** Some cells are not labeled (hence are called NaN).
- `population_names_Levine_13dim.txt` maps the cell labels from the last column of **X** (number labels) to biologically-interpretable cell-type names.
- `cell_graph.edgelist` is an edgelist for a between-cell graph. We will call this, **G**. Note that the nodes in **G** correspond to the rows in **X**. So, node i maps to row i of **X**, etc.

1) **Clustering on cell × marker data** (7 points): Use a clustering algorithm of your choice to generate a cell-to-cluster partition for the cells, using the matrix, **X**. Use normalized mutual information (NMI) to compute overlap between the true and predicted cell labels. **Note that because not all cells are labeled, you can compute this only based on the labeled cells.** Feel free to use an available implementation, such as, `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html`.

2) **Graph Partitioning** (7 points): Use a graph clustering algorithm to partition **G** into clusters. Consult course notes from days 2-3 for some ideas about this. Similar to Problem 2 - question (1) compute NMI between the labels obtained in graph partitioning and the true cell labels.

3) (3 points) Comment on any observations you observe between the quality of the partitions obtained clustering on **X** in comparison to partitioning **G**. Which approach do you think works better, using the original data, or the graph?

4) **Rare Cell-types** (5 points) `Plasmacytoid_DC_cells`, or pDCs (label 21) are a popular rare cell type, meaning many clustering algorithms will not be able to reliably find them. Report the number of distinct clusters where you found pDCs in both the clustering of **X** and in the partitioning of **G**.

Problem 2 continued on next page...

5) **Cell Classification** (10 points) Select cells from $\mathbf{X}$ with the following labels, $\{11, 12, 17, 18\}$ and $\{1, 2, 3\}$. In general, cells with labels $\{11, 12, 17, 18\}$ are T-cells and cells with labels $\{1, 2, 3\}$ are monocytes. Convert this to a binary classification problem by labeling T-cells with 0 and monocytes with 1. Use your favorite classifier to predict the labels of these cells. Use an ROC curve to visualize the performance. If the performance was not good, explain what could have gone wrong. If your performance is very good, can you identify features from $\mathbf{X}$ that were helpful in predicting labels?

Problem 3

(30 points total) **node2vec**

We will use the implementation of node2vec available in github, `https://github.com/aditya-grover/node2vec` to create vector representations for each node in $\mathbf{G}$ encoded in `cell_graph.edgelist`.

1) (**Clustering on Node2Vec Features** (10 points)) First, use default parameters and follow the instructions in the README on the graph in `cell_graph.edgelist`. This will create a 128-dimensional vector for each node. Cluster the nodes based on these vectors and compare to the ground truth labels in the last column of `Levine_matrix.csv` using NMI. Compare your results to Problem 2, question 3. Does an embedding of the graph offer any apparent advantages in classifying cells?

2) (**Parameters, part 1** (5 points)) Try a few different values for the number of dimensions `--dimensions`, such that some of them are less than 128, and some of them are more than 128. Cluster cells again with the embeddings obtained in different dimensions. Again, you can compute the NMI between the cluster assignments and the ground truth labels. Comment on some observations, and show a plot of NMI plotted against the number of dimensions used.

3) (**Parameters, part 2** (5 points)) Recall that the parameters p and q control the 'breadth' vs 'depth of the walk'. Choose one of these parameters to vary, and repeat the previous question using the default 128 dimensions, but varying values for either p or q. Comment on some observations, and show a plot of NMI against p or q (whichever one you chose).

4) (**Cell Classification, Part II** (10 points)) Repeat Problem 2, question (5). However, instead of using only $\mathbf{X}$ as the feature matrix, we are going to combine the marker expressions with node2vec features. Let $\mathbf{N}$ be your matrix generated through node2Vec. Create a new matrix called $\mathbf{X} = [\mathbf{X}|\mathbf{N}]$. That is, you will simply concatenate $\mathbf{X}$ and $\mathbf{N}$. Formulate the same classification problem from Problem 2, question (5) to classify T-cells from monocytes. Again, report your ROC curve. Comment on the performance, especially in comparison to the results obtained in Problem 2, question (5).