Readings

Newman, 14.1-14.6.

Problem 1.

Note. This problem is not extremely complex, but it is necessary in order to complete several other problems in this assignment.

Part (a)

In 14.6.3, Newman describes the *Rand Index* as a method of comparing learned clusters from an algorithm to ground-truth clusters. Write a function that accepts two vectors (or lists, or arrays) of node labels and returns the Rand Index. Please carefully organize and comment your code before submitting it.

Part (b)

Demonstrate that your function returns 1 when in an example in which the two label vectors are exactly the same. Submit both your code and output.

Note. Without loss of generality, you may assume that the label vectors contain *integers* describing each group. So, one of the label vectors could look something like this: $\mathbf{g} = [1, 2, 3, 3, 3, 2, 1, 2, 2, 2]$. Here, g_i gives the cluster label of node i.

Note. This isn't a programming class, and we're mostly not assessing you on code quality. That said, an optimal solution to this problem will not use any for-loops.

Problem 2.

In 14.2.3, Newman describes *binary spectral modularity maximization* for partitioning a network into two communities. Using your programming language of choice:

Part (a)

Implement binary spectral modularity maximization as a function which accepts a graph and returns a list or array of node labels. Please carefully organize and comment your code before submitting it.

Part (b)

Using your implementation from the previous problem, compute the Rand Index of your clustering against the labels in a small network data set. The Zachary Karate Club network that comes packaged with NetworkX is a good choice. Please show both your code and the output.

Part (c)

Visualize the network using your clustering, coloring nodes according to the cluster labels.

Please show both your code and your output.

Note. This set of notes gives some demonstrations on how to access and visualize the Zachary Karate Club network and the node attributes giving the "true" communities.

Note. This isn't a programming class, and we're mostly not assessing you on code quality. That said, an optimal solution to this problem will not use any for-loops.

Problem 3.

Generate an Erdős-Rényi random graph on 5,000 nodes with mean degree 3, and extract the largest component of this graph. Call the resulting (connected) graph C.

Part (a)

Use your binary spectral modularity maximization algorithm from the previous problem to find a label vector $\mathbf{g}$ for C.

Part (b)

Compute the modularity of the label vector $\mathbf{g}$ in C. Please show both your code and your output.

Part (c)

Comment on the reliability of modularity maximization as a technique for determining whether a network contains meaningful communities.

Note. This experiment is inspired by Fig. 1 in this paper.

Problem 4.

Let $\mathbf{g}$ be a label vector that partitions the nodes of a graph G into two communities, which we'll label 1 and 2.

The *cut* value of $\mathbf{g}$ is the number of edges in G that have nodes in different communities according to $\mathbf{g}$. It can be computed as:

$$\mathrm{Cut}(\mathbf{g}) = \frac{1}{2} \sum_{i,j \in N} a_{ij}(1 - \delta(g_i, g_j)) \ .$$

The *volume* of a community is the sum of the degrees of nodes contained in that community. For example, the volume of community 1 according to the label vector $\mathbf{g}$ is

$$\mathrm{Vol}(1, \mathbf{g}) = \sum_{i \in N} k_i \delta(1, g_i) \ , \tag{1}$$

where k_i is the degree of i. The volume of the entire graph, which we'll call $\mathrm{Vol}(G)$, is simply $\sum_{i \in N} k_i = 2m$.

Part (a)

Prove using a direct calculation that the modularity objective can also be written

$$Q = 1 - \frac{1}{\mathrm{Vol}(G)} \left[2\mathrm{Cut}(\mathbf{g}) + \frac{\mathrm{Vol}(1, \mathbf{g})^2 + \mathrm{Vol}(2, \mathbf{g})^2}{\mathrm{Vol}(G)} \right] \ . \tag{2}$$

Part (b)

Using (2), argue that maximizing the modularity objective can be interpreted as attempting to minimize the $\mathrm{Cut}(\mathbf{g})$ term while also keeping the two communities of approximately equal size.

Hint. While the notation is different, this calculation is quite similar to one we did in class, outlined here.

Problem 5.

In the previous problem, we saw that the modularity objective can be interpreted as balancing two competing objectives: find communities such that

- There are relatively few edges between the two communities (small Cut).
- Don't let either of the two communities be too large or small (as measured by the Vol).

In fact, there are other ways to combine these two ideas. The *normalized cut* (or NormCut) objective is

$$\mathrm{NC}(\mathbf{g}) = \mathrm{Cut}(\mathbf{g}) \left(\frac{1}{\mathrm{Vol}(1, \mathbf{g})} + \frac{1}{\mathrm{Vol}(2, \mathbf{g})} \right) . \tag{3}$$

This time, we want to find $\mathbf{g}$ to *minimize* $\mathrm{NC}(\mathbf{g})$. The idea is again to keep $\mathrm{Cut}(\mathbf{g})$ small, while also not letting either of the $\mathrm{Vol}(i, \mathbf{g})$ terms get too much larger than the other. This would result in one of the denominators being small, giving an undesirably large value of $\mathrm{NC}(\mathbf{g})$.

"As it turns out," the NormCut minimization problem has a close mathematical relationship to *Normalized Laplacian Spectral Clustering* (NLSC), which you will implement in this problem. NLSC is a bit like modularity spectral clustering, but it uses a different matrix. The *normalized Laplacian matrix* is $\bar{\mathbf{L}} = \mathbf{K}^{-1} [\mathbf{K} - \mathbf{A}]$, where $\mathbf{K}$ is the diagonal matrix of node degrees. To perform binary NLSC:

- Form $\bar{\mathbf{L}}$.
- Compute the eigenvector of $\bar{\mathbf{L}}$ corresponding to the *second-smallest eigenvalue*. Call this eigenvector $\mathbf{v}$.
- Assign node i to group 1 if $v_i > 0$ and to group 2 if $v_i \leq 0$.

Part (a)

Write a function to perform NLSC. Your function should accept a graph argument and return a vector, array, or list of labels. Please carefully organize and comment your code before submitting it.

Part (b)

Using your function, perform community detection in the Karate Club graph. Compute the Rand Index (from Problem 1) of your community labels. Please show both your code and your output.

Part (c)

Visualize the network using your clustering, coloring nodes according to the cluster labels. Please show both your code and your output.

Note. This isn't a programming class, and we're mostly not assessing you on code quality. That said, an optimal solution to this problem will not use any for-loops.

Problem 6.

Now let's do *Multiway Normalized Laplacian Spectral Clustering* (MNLSP). This is an extension to Laplacian spectral clustering in which we aim to extract more than two communities from the network. Fortunately, the extension is relatively simple.

To extract a total of $\ell \geq 2$ communities using MNLSP:

i. Form $\bar{\mathbf{L}}$ as in the previous problem.

ii. Choose the $\ell - 1$ eigenvectors $\{\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_\ell\}$ of $\bar{\mathbf{L}}$ corresponding to the 2nd, 3rd, ..., ℓth smallest eigenvalues of $\bar{\mathbf{L}}$.

iii. Form these eigenvectors into a matrix

$$\mathbf{V} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{v}_2 & \mathbf{v}_3 & \cdots & \mathbf{v}_\ell \\ | & | & \cdots & | \end{bmatrix} .$$

We interpret the ith row of the matrix $\mathbf{V}$ as giving the coordinates of node i in a high-dimensional *embedding space*.

iv. Perform k-means clustering on the matrix $\mathbf{V}$, using ℓ centroids. This results in a label assigned to each node. Return this label vector.

Part (a)

Write a function to implement MNLSP, allowing the user to specify the input graph and the desired number of communities ℓ. Please carefully organize and comment your code before submitting it.

Part (b)

Obtain a graph *other* than the Karate Club graph. I recommend a network with no more than 500 nodes or so. Here is one good choice.

Perform MNLSP with at least three choices of the number of communities ℓ. Visualize each one via a network diagram with node colors corresponding to communities. You don't need to compute a Rand Index in this problem, or do anything else related to true communities in the actual data.

Note. For more detail than you want on Laplacian spectral clustering, see this paper, which has been cited more than 10,000 times!

Problem 7.

Note. This question was inspired by an excellent question in class.

In this problem, you will explore a slightly more detailed case for the *resolution limit* of modularity maximization. This is discussed in the lecture notes, as well as in Newman 14.2.6.

Consider a graph similar to the one shown in Newman Figure 14.5. We impose the following assumptions.

- The bulk of the graph (corresponding to "Remainder of network" in Fig. 14.5) is a c-regular graph with n nodes. (A graph is c-regular if every node has degree c.)
- Each of Group 1 and Group 2 contain k nodes and are c'-regular, *except* for the extra edge that joins one node in Group 1 to one node in Group 2. So, one node in each group has degree $c' + 1$, while all other nodes have degree c. The k-clique example we discussed in class and the lecture notes corresponds to $c' = k$.

Determine, in terms of c, c', n, and k, the conditions under which modularity maximization is able to separate Group 1 from Group 2.

You may use the following approximations in order to simplify your findings:

- Technically, $2m = nc + kc' + 2$, where m is the number of edges. You assume that $kc' \ll nc$, and therefore that $2m \approx nc$.
- You may assume that $1 \ll k \ll kc'$, so you can do things like this: $kc' + 1 \approx kc'$.

Hint. Follow the argument in the the lecture notes, adjusting the computation of e_ℓ and f_ℓ accordingly.