Signal Processing on Graphs – Classification of Cancer Types

Weiyu Huang, Santiago Segarra, and Alejandro Ribeiro

April 20, 2021

Last week, we studied graph signal processing, which defines the concept of frequency and Fourier transform for signals supported on graphs. We observed that the traditional finite discrete time signal processing can be viewed as a particular case of the graph signal processing when the graph considered is a directed cycle. We also studied that the PCA transform can be understood as a graph Fourier transform when the underlying graph is given by the covariance matrix. In this week's lab, we are going to complete our ESE 224 journey with an application of graph signal processing to improve the classification of cancer types through the use of genetic networks.

1 Review of graph theory and graph filters

Formally, a graph is a triplet $G = (\mathcal{V}, \mathcal{E}, W)$ where $\mathcal{V} = \{1, 2, \dots, N\}$ is a finite set of N nodes or vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges defined as order pairs (n, m) and $W : \mathcal{E} \to \mathbb{R}$ is a map from the set of edges to scalar values w_{nm}. Weights w_{nm} represent the similarity or level of relationship from node n to node m. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ of a graph is defined as

$$A_{mn} = \begin{cases} w_{nm}, & \text{if } (n, m) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In *unweighted* graphs, A_{mn} is either 1—if nodes n and m are connected—or 0. For *undirected* graphs, the adjacency matrix is symmetric, i.e., $w_{nm} = w_{mn}$ for all nodes n and m. When this is not the case, we say that the graph is directed. We will only deal with undirected graphs from now on.

Graph signals are mappings $x : \mathcal{V} \to \mathbb{R}$ from the vertices of the graph into the real numbers. Graph signals can be represented as vectors $\mathbf{x} \in \mathbb{R}^N$ where x_n stores the signal value at the nth vertex in $\mathcal{V}$. Notice that this assumes an indexing of the nodes, which coincides with the indexing used in the adjacency matrix.

The degree of a node is the sum of the weights of the edges incident to that node. Formally, the degree of node i is defined as

$$\deg(i) = \sum_{j \in \mathcal{N}(i)} w_{ij}, \tag{2}$$

where $\mathcal{N}(i)$ stands for the neighborhood of node i, i.e., all other nodes connected to node i. The degree matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix such that $D_{ii} = \deg(i)$. Given an undirected graph G with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$, we define the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{3}$$

For an arbitrary undirected graph $G = (\mathcal{V}, \mathcal{E}, W)$, a graph-shift operator $\mathbf{S} \in \mathbb{R}^{N \times N}$ is a matrix satisfying $S_{ij} = 0$ for $i \neq j$ and $(j, i) \notin \mathcal{E}$. Since the graph is undirected, the graph-shift operator can be decomposed into $\mathbf{S} = \mathbf{V}\Lambda\mathbf{V}^H$. Then, the Graph Fourier Transform (GFT) of $\mathbf{x}$ is defined as

$$\tilde{\mathbf{x}}(k) = \langle \mathbf{x}, \mathbf{v}_k \rangle = \sum_{n=1}^{N} \mathbf{x}(n)\mathbf{v}_k^*(n). \tag{4}$$

Equation (4) can be rewritten in matrix form to obtain

$$\tilde{\mathbf{x}} = \mathbf{V}^H\mathbf{x}. \tag{5}$$

Since the columns of $\mathbf{V}$ are the eigenvectors $\mathbf{v}_k$ of $\mathbf{S}$, $\tilde{\mathbf{x}}(k) = \mathbf{v}_k^H\mathbf{x}$, i.e., the inner product between $\mathbf{v}_k$ and $\mathbf{x}$. We think of the eigenvectors $\mathbf{v}_k$ as oscillation modes associated to the eigenvalues in the same way that, in discrete time signal processing, different complex exponentials are associated to frequency values. In particular, GFT is equivalent to DFT when $\mathbf{V} = \mathbf{F}$, i.e. $\mathbf{v}_k = \mathbf{e}_{kN}$, the complex exponential vector.

In order to measure how much a signal oscillates within a graph, the concept of total variation can be extended from traditional signal processing. Classically, the total variation of a signal is defined as the sum of squared differences in consecutive signal samples, $\sum_n (x_n - x_{n-1})^2$. This concept can be extended to graphs where the notion of neighborhood

replaces that of consecutive nodes to obtain

$$TV_G(\mathbf{x}) = \sum_{n=1}^{N} \sum_{m \in \mathcal{N}(n)} (x_n - x_m)^2 \, w_{mn} = \mathbf{x}^T \mathbf{L} \mathbf{x}. \tag{6}$$

As can be seen from (6) the total variation of a signal in a graph can be written as a quadratic form of the Laplacian of that graph. Total variation allows us to interpret the ordering of the eigenvalues of the Laplacian in terms of frequencies, i.e., larger eigenvalues correspond to higher frequencies (larger total variation). The eigenvectors associated with large eigenvalues oscillate rapidly whereas the eigenvectors associated with small eigenvalues vary slowly.

The inverse graph Fourier transform (iGFT) of a graph signal $\tilde{\mathbf{x}} \in \mathbb{R}^N$ is given by

$$\mathbf{x}(n) = \sum_{k=0}^{N-1} \tilde{\mathbf{x}}(k) \mathbf{v}_k(n), \tag{7}$$

which can be rewritten in matrix form as

$$\mathbf{x} = \mathbf{V} \tilde{\mathbf{x}}. \tag{8}$$

The orthonormality of the $\mathbf{v}_k$, i.e., $\mathbf{V}$ is unitary, ensures that indeed the GFT and iGFT are inverse operations. Orthonormality also allows the extension of other classical results to the graph domain, e.g., Parseval's theorem.

Given a graph signal $\mathbf{x}$ with its GFT $\tilde{\mathbf{x}}$, we can filter the graph signal by passing it through a filter $\mathbf{H}$. In frequency domain, the filtered GFT is the multiplication of the original GFT $\tilde{\mathbf{x}}$ with the frequency response of the filter given by $\tilde{H}$, i.e.,

$$\tilde{\mathbf{y}} = \text{diag}(\tilde{H}) \tilde{\mathbf{x}}. \tag{9}$$

And the filtered signal in the graph domain $\hat{\mathbf{x}}$ can be recovered by performing inverse GFT onto the filtered GFT,

$$\mathbf{y} = \mathbf{V} \tilde{\mathbf{y}}. \tag{10}$$

2 Genetic network

Let us begin by analyzing the genetic network describing gene-to-gene interactions. The network was downloaded from the *NCI Nature database*

(link: http://pid.nci.nih.gov/download.shtml). You can get it from the ESE 224 website. 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.

2.1 Understanding the data. Load the file *geneNetwork_rawPCNCI.mat*. You will see the gene network $\mathbf{A} \in \mathbb{R}^{2458\times 2458}$. This is our adjacency matrix for the following analysis. Does this graph contains self-loops? Is the graph directed or undirected, weighted or unweighted? (Hint: think of what the adjacency matrix of these graphs would look like). For graphs possessing this many nodes, it would be very useful to visualize the graph to get a sense of how does the graph looks like. Load the mat file using the `scipy.io.loadmat()` in Python. Plot the graph using the `matplotlib.pylab.spy()` in Python.

Construct $\mathbf{L}$ the Laplacian of the loaded adjacency matrix $\mathbf{A}$. Suppose we define another adjacency matrix $\hat{\mathbf{A}}$ by removing all self-loops from $\mathbf{A}$ (making the diagonal elements 0). Is $\hat{\mathbf{L}}$, the Laplacian of $\hat{\mathbf{A}}$, different from $\mathbf{L}$? Explain why.

2.2 Total variations. For the graph shift operator $\mathbf{S} = \mathbf{L}$, perform an eigendecomposition and find its eigenvectors. Compute the total variation $TV_G(\mathbf{v}_k)$ for each of its eigenvectors $\mathbf{v}_k$ using (6). Plot the total variations for all eigenvectors versus their corresponding eigenvalues. What can you say about this? Recall that the total variation of a signal quantifies how much a signal oscillates on a graph. What does your finding imply about the ordering of frequencies, i.e. do eigenvectors associated with larger eigenvalues oscillate faster or slower?

3 Genetic profiles

In this section, 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.

3.1 Distinguishing power. Take the graph-shift operator to be the Laplacian $\mathbf{S} = \mathbf{L} = \mathbf{V}\Lambda\mathbf{V}^H$. We want to find the oscillation modes $\mathbf{v}_k$ 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 $\tilde{\mathbf{x}}_i = \mathbf{V}^H\mathbf{x}_i$ for all patients. Then for each mode k, define the distinguishing power of $\mathbf{v}_k$ as

$$DP(\mathbf{v}_k) = \left| \frac{\sum_{i:y_i=1} \tilde{\mathbf{x}}_i(k)}{\sum_i \mathbf{1}\{y_i = 1\}} - \frac{\sum_{i:y_i=2} \tilde{\mathbf{x}}_i(k)}{\sum_i \mathbf{1}\{y_i = 2\}} \right| / \sum_i |\tilde{\mathbf{x}}_i(k)|, \tag{11}$$

where $\mathbf{1}$ is the indicator function defined as

$$\mathbf{1}\{A\} = \begin{cases} 1, & \text{if } A \text{ is true;} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

In words, $DP(\mathbf{v}_k)$ 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 $DP(\mathbf{v}_k)$ versus k for all frequency indices k.

3.2 Interpretation. To have a better sense about the distribution of distinguishing powers, generate a boxplot of $DP(\mathbf{v}_k)$ for all k using the command `matplotlib.pyplot.boxplot()`. In the boxplot, the central mark represents the median, the edges of the box are the 25th and 75th percentiles, and the whiskers extend to the most extreme data points that Python considers not to be outliers. The data points Python considers as outliers are plotted individually. Combining this analysis with the plot you generated, what can you say about the distribution of distinguishing power? Oscillation modes with high DP contain distinguishing features

of the two subtypes and oscillation modes with low DP hardly contain useful information. In the following section, we will design a graph filter in order to improve classification accuracy of cancer subtypes. Following the study of ESE224 this semester, what's your best guess about the characteristics of reasonable graph filters?

4 Improving classifications using graph filter

We have applied k-Nearest Neighbors (k-NN) classifications a couple of times so far in the course and we will utilize it again in the cancer subtype classification. We will describe the particular k-NN algorithm we are going to use in this problem and introduce a few vectorized operation functions in Section 4.1 to accelerate the running time for the k-NN algorithm. Then we will design a graph filter to improve classification accuracy for cancer subtypes in Section 4.2.

4.1 k-NN for leave-one-out cross validation

We design the following procedure to test the improvement of filtered graph signals $\mathbf{X}_{\text{f}}$ as compared to the original signals $\mathbf{X}$. In short, we perform leave-one-out cross validation for a k nearest neighbors (k-NN) classifier. More precisely,

(1) compute the pairwise Euclidean distance between all pairs of patients using the original graph signals $\mathbf{X}$

(2) for each patient $p = 1, \ldots, 240$, 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 comparisons in (2)

We will also repeat this process for the filtered graph signals $\mathbf{X}_{\text{f}}$.

Write a Python class that given a matrix of graph signals (either $\mathbf{X}$ or $\mathbf{X}_{\text{f}}$), the vector representing subtypes of patients $\mathbf{y}$, and the number of neighbors to be considered k, computes the global classification accuracy. Run this class for the original graph signals and $k = 3, 5, 7$, report your accuracies.

Here are some **tips** to write a fast vectorized leave-one-out cross validation algorithm, given arbitrary graph signals $\mathbf{Z}$. In computing the pairwise distance between pairs of patients, we can use

```
import scipy.spatial.distance as ssd
d = ssd.squareform(ssd.pdist(Z, 'euclidean'))
```

where `ssd.pdist(·, 'euclidean')` outputs a vector containing the distances between each pair of observations in $\mathbf{Z}$ and `ssd.squareform()` converts this vector into the matrix form.

For leave-one-out cross validation, given the matrix `d` representing the pairwise distance, we can select the nearest neighbors using the following trick. Start with `nn = d.argsort(axis=0)`, whose output is a matrix with the i-th column representing the patient indices ordered from the most similar patient to patient i to most dissimilar. Since the distance from a patient to themselves is always 0, the closest patient to any patient is themselves. Hence, the command `nn_label = y[nn[1:(k+1), :]]` gives us the labels of the k nearest neighbors for all patients. In `nn_label`, the i-th column represents the labels of the k closest patients to patient i. Finally, using `statistics.mode()` along the right dimension yields the prediction results from the k-NN classifier. Comparing the results with the actual label $\mathbf{y}$ gives the accuracy.

4.2 Graph filters

We consider the following two graph filters to improve the classification accuracy results we computed in the previous section. The first graph filter keeps only the information conveyed in the oscillation mode that is most distinguishable for two subtypes. To be more specific, for the genetic profile $\mathbf{x}_i$ of patient i, its GFT $\tilde{\mathbf{x}}_i$ is fed into a graph filter $\tilde{H}_1$ with the following frequency response,

$$\tilde{H}_1(k) = \begin{cases} 1, & \text{if } k = \text{argmax}_k \, DP(\mathbf{v}_k) \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Denote the filtered GFT as $\tilde{\mathbf{x}}_i^f$ and its inverse GFT as $\mathbf{x}_i^f$. We can then form a filtered graph signal matrix with each of its rows representing the filtered genetic profile of each patient. Run the k-NN classifier you wrote in 4.1 with this filtered graph signal matrix and report your accuracies for $k = 3, 5, 7$. Compare the results with those obtained for the original graph signals. What do you observe?

We can consider a more general graph filter $\tilde{H}_p$ with the following frequency response,

$$\tilde{H}_p(k) = \begin{cases} 1, & \text{if } DP(\mathbf{v}_k) \geq p\text{-th percentile of the distribution of } DP \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

where p is any real number between 0 and 100. In words, this family of graph filters keep the information conveyed in oscillation modes that are distinguishable for two subtypes to some extent. The p here is chosen to select the number of oscillation modes. Run the k-NN classifier you wrote in 4.1 with the graph signal fed into this graph filter with your choice of p such that the classification error is much smaller than the classification error using the original graph signals. Report your accuracies for $k = 3, 5, 7$ and $p = 0.75, 0.8, 0.85, 0.9, 0.95$.